

LNCS 2686

José Mira
José R. Álvarez (Eds.)

Computational Methods in Neural Modeling

7th International Work-Conference on
Artificial and Natural Neural Networks, IWANN 2003
Maó, Menorca, Spain, June 2003
Proceedings, Part I

I
Part I



Springer



Lecture Notes in Computer Science 2686

Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer

Berlin

Heidelberg

New York

Barcelona

Hong Kong

London

Milan

Paris

Tokyo

José Mira José R. Álvarez (Eds.)

Computational Methods in Neural Modeling

7th International Work-Conference on
Artificial and Natural Neural Networks, IWANN 2003
Maó, Menorca, Spain, June 3-6, 2003
Proceedings, Part I



Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editors

José Mira
José R. Álvarez
Universidad Nacional de Educación a Distancia
E.T.S. de Ingeniería Informática
Departamento de Inteligencia Artificial
Juan del Rosal, 16, 28040 Madrid, Spain
E-mail: {jmira,jras}@dia.uned.es

Cataloging-in-Publication Data applied for

A catalog record for this book is available from the Library of Congress

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliographie;
detailed bibliographic data is available in the Internet at <http://dnb.ddb.de>.

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

ISSN 0302-9743

ISBN 3-540-40210-1 Springer-Verlag Berlin Heidelberg New York

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

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

<http://www.springer.de>

© Springer-Verlag Berlin Heidelberg 2003
Printed in Germany

Typesetting: Camera-ready by author, data conversion by DA-TeX Gerd Blumenstein
Printed on acid-free paper SPIN: 10927823 06/3142 5 4 3 2 1 0

Preface

The global purpose of IWANN conferences has been to provide a broad and interdisciplinary forum for the interplay between neuroscience and computation. Our dream has been, and still is: (1) find ways to understand the physiological, symbolic and cognitive nature of the nervous system (NS) with the help of computational and engineering tools; and (2) find rich and insightful sources of inspiration in biology, to develop new materials, mechanisms and problem-solving methods (PSM) of value in engineering and computation. As all of us know well, this dream started with the Ancient Greeks, reappeared in the foundational stage of neurocybernetics and bionics, and is now broadly accepted in the scientific community under different labels such as computational neuroscience (CN) and artificial neural nets (ANN), or genetic algorithms and hybrid neuro-fuzzy systems.

We have also to recognize that there is a considerable lack of credibility associated with CN and ANN among some researchers, both from biology and from the engineering and computation area. Potential causes of this scepticism could be the lack of methodology, formal tools, and real-world applications, in the engineering area, and the lack also of formal tools for cognitive process modeling. There is also the possibility of the computational paradigm being inappropriate to explain cognition, because of the “representational” character of any computational model. Some “more situated” approaches are looking back to the “neurophysiological epistemology” of the 1960’s (mind in a body) to search directly for the mechanisms that could embody the cognitive process, and this means some fresh air for our old dream of connectionism. The tremendous difficulties of these fields (CN and ANN) and of the questions addressed (How does the NS work, and how can we mimic this work in a non-trivial manner?) justify the delay in getting proper answers.

We hope that the papers in these two volumes and the discussions during this seventh working conference will help us to create some new critique and a constructive atmosphere. The neural modeling community and all the people engaged in the connectionist perspective of knowledge engineering would certainly benefit from this new atmosphere.

IWANN 2003, the 7th International Work-Conference in Artificial and Natural Neural Networks took place in Maó, Menorca, Spain, during June 3–6, 2003, addressing the following topics:

Mathematical and computational methods in neural modeling. Levels of analysis. Brain theory. Neural coding. Mathematical biophysics. Population dynamics and statistical modeling. Diffusion processes. Dynamical binding. Synchronization. Resonance. Regulatory mechanisms. Cellular automata.

Neurophysiological data analysis and modeling. Ionic channels. Synapses. Neurons. Circuits. Biophysical simulations.

Structural and functional models of neurons. Analogue, nonlinear, recurrent, RBF, PCA, digital, probabilistic, Bayesian, fuzzy and object-oriented formulations.

Learning and other plasticity phenomena. Supervised, non-supervised, reinforcement and statistical algorithms. Hybrid formulations. Incremental and decremental architectures. Biological mechanisms of adaptation and plasticity. Development and maturing.

Complex systems dynamics. Statistical mechanics. Attractors. Optimization, self-organization and cooperative-competitive networks. Evolutionary and genetic algorithms.

Cognitive Processes and Artificial Intelligence. Perception (visual, auditory, tactile, proprioceptive). Multi-sensory integration. Natural language. Memory. Decision making. Planning. Motor control. Neuroethology. Knowledge modeling. Multi-agent systems. Distributed AI. Social systems.

Methodology for nets design, simulation and implementation. Data analysis, task identification and recursive design. Development environments and editing tools. Implementation. Evolving hardware.

Bio-inspired systems and engineering. Bio-cybernetics and bionics. Signal processing, neural prostheses, retinomorphic systems, and other neural adaptive prosthetic devices. Molecular computing.

Applications. Artificial vision, speech recognition, spatio-temporal planning and scheduling. Data mining. Sources separation. Applications of ANNs in robotics, astrophysics, economics, the Internet, medicine, education and industry.

IWANN 2003 was organized by the Universidad Nacional de Educación a Distancia, UNED, Madrid, in cooperation with IFIP (Working Group in Neural Computer Systems, WG10.6) and the Spanish RIG IEEE Neural Networks Council.

Sponsorship was obtained from the Spanish Ministerio de Ciencia y Tecnología under project TIC2002-10752-E and the organizing university (UNED).

The papers presented here correspond to talks delivered at the conference. After the evaluation process, 197 papers were accepted for oral or poster presentation, according to the recommendations of referees and the author's preferences. We have organized these papers into two volumes arranged basically following the topics list included in the call for papers. The first volume, entitled "Computational Methods in Neural Modeling," is divided into six main parts and includes the contributions on: biological models, functional models, learning, self-organizing systems, artificial intelligence and cognition, and bioinspired developments.

In the second volume, "Artificial Neural Nets Problem-Solving Methods," we have included the contributions dealing with nets design, simulations, implementations, and application developments.

We would like to express our sincere gratitude to the members of the organizing and program committees, in particular to Félix de la Paz López, to the referees, and to the organizers of pre-organized sessions for their invaluable effort

in helping with the preparation of this conference. Thanks also to the invited speakers for their effort in preparing the plenary lectures.

Last, but not least, the editors would like to thank Springer-Verlag, in particular Alfred Hofmann, for the continuous and excellent cooperative collaboration, from the first IWANN in Granada (1991, LNCS 540), to the successive meetings in Sitges (1993, LNCS 686), Torremolinos (1995, LNCS 930), Lanzarote (1997, LNCS 1240), Alicante (1999, LNCS 1606 and 1607), again in Granada (2001, LNCS 2084 and 2085), and now in Maó.

June 2003

José Mira
José R. Álvarez

Congress Board

Alberto Prieto Espinosa, Universidad de Granada (Spain)
Joan Cabestany i Moncusi, Universitat Politecnica de Catalunya (Spain)
Francisco Sandoval Hernández, Universidad de Málaga (Spain)

Local Organizing Committee

Oscar Herreras, Hospital Ramón y Cajal (Spain)
Miguel Angel Vázquez Segura, UNED (Spain)
José Ramón Álvarez Sánchez, UNED (Spain)

Organization Staff

Félix de la Paz López, UNED (Spain)

Invited Speakers

Erik de Schutter, University of Antwerp (Belgium)
Luigi M. Ricciardi, Università di Napoli Federico II (Italy)
Marley Velasco, Pontifícia Universidade Católica do Rio de Janeiro (Brazil)

Field Editors

Miguel Atencia, Universidad de Málaga (Spain)
Emilia I. Barakova, RIKEN (Japan)
Valeriu Beiu, Washington State University (USA)
Carlos Cotta, University of Málaga (Spain)
Marcos Faúndez-Zanuy, Universitat Politecnica de Catalunya (Spain)
Miguel Angel Fernández, Universidad de Castilla-La Mancha (Spain)
Antonio Fernández-Caballero, Universidad de Castilla-La Mancha (Spain)
Carlos G. Puntonet, Universidad de Granada (Spain)
Gonzalo Joya, Universidad de Málaga (Spain)
Dario Maravall, Universidad Politécnica de Madrid (Spain)
Luiza de Macedo Mourelle, State University of Rio de Janeiro (Brazil)
Nadia Nedjah, State University of Rio de Janeiro (Brazil)
Ignacio Rojas, Universidad de Granada (Spain)
Eduardo Ros, Univesidad de Granada (Spain)
Javier Ruiz-del-Solar, Universidad de Chile (Chile)
Moisés Salmerón Campos, Universidad de Granada (Spain)
Jordi Solé-Casals, Universitat de Vic (Spain)
Ramiro Varela Arias, Universidad de Oviedo (Spain)
Changjiu Zhou, Singapore Polytechnic (Singapore)

Scientific Committee (Referees)

- Ajith Abraham**, Oklahoma State University (USA)
Igor Aizenberg, Neural Networks Technologies Ltd. (Israel)
Igor Aleksander, Imperial College of Science, Technology and Medicine (UK)
Amparo Alonso Betanzos, Universidade da Coruña (Spain)
José Ramón Álvarez Sánchez, UNED (Spain)
Miguel Atencia, Universidad de Málaga (Spain)
Antonio Bahamonde, Universidad de Oviedo en Gijón (Spain)
Emilia I. Barakova, RIKEN (Japan)
Senen Barro Ameneiro, Universidade de Santiago de Compostela (Spain)
Valeriu Beiu, Washington State University (USA)
Gustavo A. Camps i Valls, Universitat de València (Spain)
Andreu Català Mallofré, Universitat Politècnica de Catalunya (Spain)
Carlos Cotta, Universidad de Málaga (Spain)
Félix de la Paz López, UNED (Spain)
Angel P. del Pobil, Universitat Jaume-I (Spain)
Ana E. Delgado García, UNED (Spain)
Jose Dorronsoro, Universidad Autónoma de Madrid (Spain)
Richard Duro, Universidade da Coruña (Spain)
Reinhard Eckhorn, Philips University (Germany)
Marcos Faúndez-Zanuy, Universitat Politècnica de Catalunya (Spain)
Jianfeng Feng, Sussex University (UK)
Antonio Fernández-Caballero, Universidad de Castilla-La Mancha (Spain)
Jose Manuel Ferrández, Univ. Politécnica de Cartagena (Spain)
Kunihiro Fukushima, Tokyo University of Technology (Japan)
Carlos G. Puntonet, Universidad de Granada (Spain)
Manuel Graña Romay, Universidad País Vasco (Spain)
Oscar Herreras, Hospital Ramón y Cajal (Spain)
Gonzalo Joya, Universidad de Málaga (Spain)
Elka Korutcheva, UNED (Spain)
Dario Maravall, Universidad Politécnica de Madrid (Spain)
José Mira, UNED (Spain)
Javier Molina Vilaplana, Universidad Politécnica de Cartagena (Spain)
Juan M. Moreno Aróstegui, Universidad Politécnica de Cataluña (Spain)
Luiza de Macedo Mourelle, State University of Rio de Janeiro (Brazil)
Nadia Nedjah, State University of Rio de Janeiro (Brazil)
Julio Ortega Lopera, Universidad de Granada (Spain)
Alberto Prieto Espinosa, Universidad de Granada (Spain)
John Rinzel, New York University (USA)
Eduardo Ros, Universidad de Granada (Spain)
Javier Ruiz-del-Solar, Universidad de Chile (Chile)
Moisés Salmerón Campos, Universidad de Granada (Spain)
Eduardo Sánchez Vila, Universidade de Santiago de Compostela (Spain)
Francisco Sandoval, Universidad de Málaga (Spain)
José Santos Reyes, Universidade da Coruña (Spain)

Luis Manuel Sarro Baro, UNED (Spain)

Jordi Solé-Casals, Universitat de Vic (Spain)

Emilio Soria Olivas, Universitat de València (Spain)

Ramiro Varela Arias, Universidad de Oviedo (Spain)

Marley Vellasco, Pontificia Universidade Catolica (Brazil)

Michel Verleysen, Université Catholique de Louvain (Belgium)

Changjiu Zhou, Singapore Polytechnic (Singapore)

Table of Contents, Part I

Biological Models

Modeling Neuronal Firing in the Presence of Refractoriness	1
<i>L.M. Ricciardi, G. Esposito, V. Giorno, and C. Valerio</i>	
A Study of the Action Potential Initiation Site Along the Axosomatodendritic Axis of Neurons Using Compartmental Models	9
<i>J.M. Ibarz and O. Herreras</i>	
Real Neurons and Neural Computation: A Summary of Thoughts	16
<i>José Mira</i>	
Synchronous Firing in a Population of Inhibitory Interneurons Coupled by Electrical and Chemical Synapses	24
<i>Santi Chillemi, Angelo Di Garbo, and Alessandro Panarese</i>	
Stochastic Networks with Subthreshold Oscillations and Spiking Activity	32
<i>Nazareth P. Castellanos, Francisco B. Rodríguez, and Pablo Varona</i>	
Selective Inactivation of Neuronal Dendritic Domains: Computational Approach to Steady Potential Gradients	40
<i>I. Makarova, J.M. Ibarz, L. López-Aguado, and O. Herreras</i>	
Intermittent Burst Synchronization in Neural Networks	46
<i>Christian Hauptmann, Annette Gail, and Fotios Giannakopoulos</i>	
Diffusion Associative Network: Diffusive Hybrid Neuromodulation and Volume Learning	54
<i>P. Fernandez Lopez, C.P. Suarez Araujo, P. Garcia Baez, and G. Sanchez Martin</i>	
The Minimum-Variance Theory Revisited	62
<i>Jianfeng Feng</i>	
Sleep and Wakefulness in the Cuneate Nucleus: A Computational Study	70
<i>Eduardo Sánchez, Senén Barro, Jorge Mariño, and Antonio Canedo</i>	
Effects of Different Connectivity Patterns in a Model of Cortical Circuits	78
<i>Carlos Aguirre, Doris Campos, Pedro Pascual, and Eduardo Serrano</i>	
A Digital Neural Model of Visual Deficits in Parkinson's Disease	86
<i>Igor Aleksander and Helen Morton</i>	

Intersensorial Summation as a Nonlinear Contribution to Cerebral Excitation	94
<i>Isabel Gonzalo and Miguel A. Porras</i>	
Interacting Modalities through Functional Brain Modeling	102
<i>Tino Lourens, Emilia Barakova, and Hiroshi Tsujino</i>	
Life-Long Learning: Consolidation of Novel Events into Dynamic Memory Representations	110
<i>Emilia I. Barakova, Tino Lourens, and Yoko Yamaguchi</i>	

Functional Models

Real-time Sound Source Localization and Separation Based on Active Audio-Visual Integration	118
<i>Hiroshi G. Okuno and Kazuhiro Nakadai</i>	
Hierarchical Neuro-Fuzzy Systems	126
<i>M. Vellasco, M. Pacheco, and K. Figueiredo</i>	
A Functional Spiking Neuron Hardware Oriented Model	136
<i>Andres Upequi, Carlos Andrés Peña-Reyes, and Eduardo Sanchez</i>	
SO(2)-Networks as Neural Oscillators	144
<i>Frank Pasemann, Manfred Hild, and Keyan Zahedi</i>	
A Rotated Kernel Probabilistic Neural Network (RKPNN) for Multi-class Classification	152
<i>Ingo Galleske and Juan Castellanos</i>	
Independent Residual Analysis for Temporally Correlated Signals	158
<i>L.-Q. Zhang and A. Cichocki</i>	
MLP+H: A Hybrid Neural Architecture Formed by the Interaction of Hopfield and Multi-Layer Perceptron Neural Networks	166
<i>Clayton Silva Oliveira and Emilio Del Moral Hernandez</i>	
Linear Unit Relevance in Multiclass NLDA Networks	174
<i>José R. Dorronsoro, Ana González, and Eduardo Serrano</i>	
Bootstrap for Model Selection: Linear Approximation of the Optimism	182
<i>G. Simon, A. Lendasse, and M. Verleysen</i>	

Learning

A Learning Rule to Model the Development of Orientation Selectivity in Visual Cortex	190
<i>Jose M. Jerez, Miguel Atencia, Francisco J. Vico, and Enrique Dominguez</i>	

Sequence Learning Using the Neural Coding	198
<i>Sorin Moga, Philippe Gaussier, and Jean-Paul Banquet</i>	
Improved Kernel Learning Using Smoothing Parameter	
Based Linear Kernel	206
<i>Shawkat Ali and Ajith Abraham</i>	
Automatic Car Parking: A Reinforcement Learning Approach	214
<i>Darío Maravall, Miguel Ángel Patricio, and Javier de Lope</i>	
Discriminative Training of the Scanning N-Tuple Classifier	222
<i>Simon M. Lucas</i>	
A Wrapper Approach with Support Vector Machines	
for Text Categorization	230
<i>E. Montañés, J.R. Quevedo, and I. Díaz</i>	
Robust Expectation Maximization Learning Algorithm	
for Mixture of Experts	238
<i>Romina Torres, Rodrigo Salas, Héctor Allende, and Claudio Moraga</i>	
Choosing among Algorithms to Improve Accuracy	246
<i>José Ramón Quevedo, Elías F. Combarro, and Antonio Bahamonde</i>	
Evolutionary Approach to Overcome Initialization Parameters	
in Classification Problems	254
<i>P. Isasi and F. Fernandez</i>	
Text Categorisation Using a Partial-Matching Strategy	262
<i>J. Ranilla, I. Díaz, and J. Fernández</i>	
A New Learning Method for Single Layer Neural Networks	
Based on a Regularized Cost Function	270
<i>Juan A. Suárez-Romero, Oscar Fontenla-Romero,</i>	
<i>Bertha Guijarro-Berdiñas, and Amparo Alonso-Betanzos</i>	
A Better Selection of Patterns	
in Lazy Learning Radial Basis Neural Networks	278
<i>P. Isasi, J.M. Valls, and I.M. Galván</i>	
An Iterative Fuzzy Prototype Induction Algorithm	286
<i>Inés González Rodríguez, Jonathan Lawry, and Jim F. Baldwin</i>	
A Recurrent Multivalued Neural Network for Codebook Generation	
in Vector Quantization	294
<i>R. Benítez-Rochel, J. Muñoz-Pérez, and E. Mérida-Casermeiro</i>	
The Recurrent IML-Network	302
<i>Joern Fischer</i>	
Estimation of Multidimensional Regression Model	
with Multilayer Perceptrons	310
<i>Joseph Rynkiewicz</i>	
Principal Components Analysis Competitive Learning	318
<i>Ezequiel López-Rubio, José Muñoz-Pérez,</i>	
<i>and José Antonio Gómez-Ruiz</i>	

Self-Organizing Systems

Progressive Concept Formation in Self-Organising Maps	326
<i>Emilio Corchado and Colin Fyfe</i>	
Supervised Classification with Associative SOM	334
<i>Rafael del-Hoyo, David Buldain, and Alvaro Marco</i>	
Neural Implementation of Dijkstra's Algorithm	342
<i>E. Merida-Casermeiro, J. Muñoz-Pérez, and R. Benítez-Rochel</i>	
Spurious Minima and Basins of Attraction in Higher-Order Hopfield Networks	350
<i>M. Atencia, G. Joya, and F. Sandoval</i>	
Cooperative Co-evolution of Multilayer Perceptrons	358
<i>P.A. Castillo Valdivieso, M.G. Arenas, J.J. Merelo, and G. Romero</i>	
A Statistical Model of Pollution-Caused Pulmonary Crises	366
<i>Daniel Rodríguez-Pérez, Jose L. Castillo, and J.C. Antoranz</i>	
A New Penalty-Based Criterion for Model Selection in Regularized Nonlinear Models	374
<i>Elisa Guerrero, Joaquín Pizarro, Andrés Yáñez and Pedro Galindo</i>	
Data Driven Multiple Neural Network Models Generator Based on a Tree-Like Scheduler	382
<i>Kurosh Madani, Abdennasser Chebira, and Mariusz Rybnik</i>	
Performance-Enhancing Bifurcations in a Self-Organising Neural Network	390
<i>Terence Kwok and Kate A. Smith</i>	
A Competitive Neural Network Based on Dipoles	398
<i>M.A. García-Bernal, J. Muñoz-Pérez, J.A. Gómez-Ruiz, and I. Ladrón de Guevara-López</i>	
An N-parallel Multivalued Network: Applications to the Travelling Salesman Problem	406
<i>E. Mérida-Casermeiro, J. Muñoz-Pérez, and E. Domínguez-Merino</i>	
Parallel ACS for Weighted MAX-SAT	414
<i>Habiba Drias and Sarah Iibri</i>	
Learning to Generate Combinatorial Action Sequences Utilizing the Initial Sensitivity of Deterministic Dynamical Systems	422
<i>Ryunosuke Nishimoto and Jun Tani</i>	
BICONN: A Binary Competitive Neural Network	430
<i>J. Muñoz-Pérez, M.A. García-Bernal, I. Ladrón de Guevara-López, and J.A. Gómez-Ruiz</i>	

SASEGASA: An Evolutionary Algorithm for Retarding Premature Convergence by Self-Adaptive Selection Pressure Steering	438
<i>Michael Affenzeller and Stefan Wagner</i>	
Cooperative Ant Colonies	
for Solving the Maximum Weighted Satisfiability Problem	446
<i>Habiba Drias, Amine Taibi, and Sofiane Zekour</i>	
Designing a Phenotypic Distance Index	
for Radial Basis Function Neural Networks	454
<i>Jesús González, Ignacio Rojas, Héctor Pomares, and Julio Ortega</i>	
The Antiquadrupolar Phase of the Biquadratic Neural Network	462
<i>David R.C. Dominguez</i>	
Co-evolutionary Algorithm for RBF	
by Self-Organizing Population of Neurons	470
<i>A.J. Rivera, J. Ortega, I. Rojas, and M.J. del Jesús</i>	
Studying the Capacity of Grammatical Encoding	
to Generate FNN Architectures	478
<i>Germán Gutiérrez, Beatriz García, José M. Molina, and Araceli Sanchis</i>	
Optimal Phasor Measurement Unit Placement	
Using Genetic Algorithms	486
<i>F.J. Marín, F. García-Lagos, G. Joya, and F. Sandoval</i>	
On the Evolutionary Inference of Temporal Boolean Networks	494
<i>Carlos Cotta</i>	
Specifying Evolutionary Algorithms in XML	502
<i>Juan Julián Merelo Guervós, Pedro Ángel Castillo Valdívieso, Gustavo Romero López, and Maribel García Arenas</i>	
Analysis of the Univariate Marginal Distribution Algorithm Modeled	
by Markov Chains	510
<i>C. González, J.D. Rodríguez, J.A. Lozano, and P. Larrañaga</i>	
Node Level Crossover Applied to Neural Network Evolution	518
<i>E. Sanz-Tapia, N. García-Pedrajas, D. Ortiz-Boyer, and C. Hervás-Martínez</i>	
Genetic Search of Block-Based Structures	
of Dynamical Process Models	526
<i>A. López and L. Sánchez</i>	
Visualization of Neural Net Evolution	534
<i>G. Romero, M.G. Arenas, P.A. Castillo Valdívieso, and J.J. Merelo</i>	
Separable Recurrent Neural Networks Treated	
with Stochastic Velocities	542
<i>A. Castellanos-Moreno</i>	

Studying the Convergence of the CFA Algorithm	550
<i>Jesús González, Ignacio Rojas, Héctor Pomares, and Julio Ortega</i>	
Auto-adaptive Neural Network Tree Structure	
Based on Complexity Estimator	558
<i>Mariusz Rybnik, Abdennasser Chebira, and Kurosh Madani</i>	

Artificial Intelligence and Cognition

Intelligence and Computation: A View from Physiology	566
<i>Juan Vicente Sanchez-Andres</i>	
Image Understanding Analysis at the Knowledge Level as a Design Task	574
<i>M. Rincón, M. Bachiller, J. Mira, and R. Martínez</i>	
Morphological Clustering of the SOM for Multi-dimensional Image Segmentation	582
<i>Aureli Soria-Frisch and Mario Köppen</i>	
Introducing Long Term Memory in an ANN Based Multilevel Darwinist Brain	590
<i>F. Bellas and R.J. Duro</i>	
New Directions in Connectionist Language Modeling	598
<i>María José Castro and Federico Prat</i>	
Rules and Generalization Capacity Extraction from ANN with GP	606
<i>Juan R. Rabuñal, Julián Dorado, Alejandro Pazos, and Daniel Rivero</i>	
Numerosity and the Consolidation of Episodic Memory	614
<i>J.G. Wallace and K. Bluff</i>	
Rule Extraction from a Multilayer Feedforward Trained Network via Interval Arithmetic Inversion	622
<i>Carlos Hernández-Espinosa, Mercedes Fernández-Redondo, and Mamen Ortiz-Gómez</i>	
Necessary First-Person Axioms of Neuroconsciousness	630
<i>Igor Aleksander and Barry Dunmall</i>	
An Agent Based Approach of Collective Foraging	638
<i>Marian Gheorghe, Carlos Martín-Vide, Victor Mitrana, and Mario J. Pérez Jiménez</i>	
Hybrid Architecture Based on Support Vector Machines	646
<i>Haydemar Núñez, Cecilio Angulo, and Andreu Catalá</i>	
A Neural Approach to Extended Logic Programs	654
<i>Jesús Medina, Enrique Mérida-Casermeiro, and Manuel Ojeda-Aciego</i>	

Bioinspired Developments

Solving SAT in Linear Time with a Neural-Like Membrane System	662
<i>Juan Pazos, Alfonso Rodríguez-Patón, and Andrés Silva</i>	
An Exponential-Decay Synapse Integrated Circuit for Bio-inspired Neural Networks	670
<i>Ludovic Alvado, Sylvain Saïghi, Jean Tomas, and Sylvie Renaud</i>	
A High Level Synthesis of an Auditory Mechanical to Neural Transduction Circuit	678
<i>J.M. Ferrández, M.A. Sacristán, V. Rodellar, and P. Gómez</i>	
Restriction Enzyme Computation	686
<i>Olgierd Unold and Maciej Troć</i>	
Neurally Inspired Mechanisms for the Dynamic Visual Attention Map Generation Task	694
<i>Maria T. López, Miguel A. Fernández, Antonio Fernández-Caballero, and Ana E. Delgado</i>	
A Model of Dynamic Visual Attention for Object Tracking in Natural Image Sequences	702
<i>Nabil Ouerhani and Heinz Hügli</i>	
Neural Competitive Structures for Segmentation Based on Motion Features	710
<i>Javier Díaz, Sonia Mota, Eduardo Ros, and Guillermo Botella</i>	
Background Pixel Classification for Motion Detection in Video Image Sequences	718
<i>P. Gil-Jiménez, S. Maldonado-Bascón, R. Gil-Pita, and H. Gómez-Moreno</i>	
COBRA: An Evolved Online Tool for Mammography Interpretation	726
<i>Carlos-Andrés Peña-Reyes, Rosa Villa, Luis Prieto, and Eduardo Sanchez</i>	
CBA Generated Receptive Fields Implemented in a Facial Expression Recognition Task	734
<i>Jose M. Jerez, Leonardo Franco, and Ignacio Molina</i>	
A Hybrid Face Detector Based on an Asymmetrical Adaboost Cascade Detector and a Wavelet-Bayesian-Detector	742
<i>Rodrigo Verschae and Javier Ruiz del Solar</i>	
Neural Net Generation of Facial Displays in Talking Heads	750
<i>Max H. Garzon and Kiran Rajaya</i>	
Author Index	759

Modeling Neuronal Firing in the Presence of Refractoriness

L.M. Ricciardi¹, G. Esposito¹, V. Giorno², and C. Valerio¹

¹ Dipartimento di Matematica e Applicazioni, Università di Napoli Federico II, Via Cintia, Napoli, Italy

luigi.ricciardi@unina.it,

² Dipartimento di Matematica e Informatica, Università di Salerno, Via Allende, Baronissi (SA), Italy

{giorno}@unisa.it

Abstract. A mathematical characterization of the membrane potential as an instantaneous return process in the presence of refractoriness is investigated for diffusion models of single neuron's activity. The statistical features of the random variable modeling the number of neuronal firings is analyzed by including the additional assumption of the existence of neuronal refractoriness. Asymptotic exact formulas for the multiple firing probabilities and for the expected number of produced firings are finally given.

1 Introduction

Since the classical paper by Gerstein and Mandelbrot [4], numerous attempts have been made to formulate stochastic models for single neuron's activity that would reproduce some of the essential features of the behavior exhibited by living neural cells under spontaneous or stimulated conditions (see, for instance, [9], [12] [13] and the bibliography quoted therein). In particular, a quantitative description of the behavior of the neuron's membrane potential as an instantaneous return process has also been the object of various investigations [5], [7], [8], [11], under the assumption that after each firing the neuron's membrane potential is either reset to a unique fixed value, or that the reset value is characterized by an assigned probability density function (pdf). Recently, in [1] and [2] the presence of refractoriness has been included in the mathematical characterization of the membrane potential as an instantaneous return process within diffusion models of single neuron's activity, assuming that the firing threshold acts as an elastic barrier, that is 'partially transparent', in the sense that its behavior is intermediate between total absorption and total reflection.

In the present paper, the return process paradigm for the description of the time course of the membrane potential is instead analyzed by assuming that the neuronal refractoriness period is described by means of a random variable with a preassigned probability density function (pdf). In Section 2 the statistical features of the random process modeling the number of firings released by the neuron will be analyzed by means of the afore mentioned return process. Finally,

in Section 3 an asymptotic analysis of the random process modeling the number of neuronal firings will be provided, assuming that the neuronal refractoriness period is described by a deterministic pdf.

It must be underlined that the present approach towards inclusion of refractoriness in models for neuron's activity differs in an essential way from the first approach attempted in the sixties by one of the present authors [10] and from that successively pursued in [15]. Indeed, in there refractoriness was treated within the context of point processes, and not by using continuous processes frameworks.

2 Probability Distribution of the Firing Frequency

Let $\{X(t), t \geq 0\}$ be a regular, time-homogeneous diffusion process, defined over the interval $I = (r_1, r_2)$, characterized by drift and infinitesimal variance $A_1(x)$ and $A_2(x)$, respectively, that we assume to satisfy Feller conditions [3]. Let $h(x)$ and $k(x)$ denote scale function and speed density of $X(t)$:

$$h(x) = \exp \left\{ -2 \int^x \frac{A_1(z)}{A_2(z)} dz \right\}, \quad k(x) = \frac{2}{A_2(x) h(x)}$$

and by

$$H(r_1, y] = \int_{r_1}^y h(z) dz, \quad K(r_1, y] = \int_{r_1}^y k(z) dz$$

scale and the speed measures, respectively.

We define the random variable "first passage time" (FPT) of $X(t)$ through S ($S \in I$) with $X(0) = x < S$ as follows:

$$T_x = \inf_{t \geq 0} \{t : X(t) \geq S\}, \quad X(0) = x < S. \quad (1)$$

Then,

$$g(S, t|x) = \frac{\partial}{\partial t} P(T < t), \quad x < S \quad (2)$$

is the FPT pdf of $X(t)$ through S conditional upon $X(0) = x$.

In the neuronal modeling context the state S represents the neuron's firing threshold, FPT through S the firing time and $g(S, t|x)$ the firing pdf. In the sequel we assume that one of the following cases holds:

- (i) r_1 is a natural nonattracting boundary and $K(r_1, y] < +\infty$
- (ii) r_1 is a reflecting boundary or it is an entrance boundary.

Under such assumptions the first passage probability $P(S|x)$ from x to S is unity and the mean FPT is given by (cf. [14]):

$$\ell_1(S|x) := \int_0^\infty t g(S, t|x) dt = \int_x^S h(z) dz \int_{r_1}^z k(u) du \quad (x < S). \quad (3)$$

We assume that after each firing a period of refractoriness of random duration occurs, during which either the neuron is completely unable to respond, or it only partially responds to the received stimulations.

We now construct the return process $\{Z(t), t \geq 0\}$ in (r_1, S) as follows. Starting at a point $\eta \in (r_1, S)$ at time zero, a firing takes place when $X(t)$ attains the threshold S for the first time, after which a period of refractoriness of random duration occurs. At the end of the period of refractoriness, $Z(t)$ is instantaneously reset at a certain fixed state η . The subsequent evolution of the process then goes as described by $X(t)$, until the boundary S is again reached. A new firing then occurs, followed by the period of refractoriness, and so on.

The process $\{Z(t), t \geq 0\}$, describing the time course of the membrane potential thus consists of recurrent cycles $\mathcal{F}_0, \mathcal{R}_1, \mathcal{F}_1, \mathcal{R}_2, \mathcal{F}_2, \dots$, each of random duration, where the durations F_i of \mathcal{F}_i ($i = 0, 1, \dots$) and the durations of refractoriness period R_i of \mathcal{R}_i ($i = 1, 2, \dots$) are independently distributed random variables. Here, F_i ($i = 0, 1, \dots$) describes the length of the firing interval, i.e. of the time interval elapsing between the i -th reset at the value η and the $(i+1)$ -th FPT from η to S . Instead, R_i ($i = 1, 2, \dots$) describes the duration of i -th refractoriness period. Since the diffusion process $X(t)$ is time-homogeneous, the random variables F_0, F_1, \dots can be safely assumed to be independent and identically distributed, each with pdf $g(S, t|\eta)$ depending only on the length of the corresponding firing interval. Furthermore, we assume that R_1, R_2, \dots are independent and identically distributed random variables, each with pdf $\varphi(t)$ depending only on the duration of the refractoriness period.

Hereafter, we shall provide a description of the random process $\{M(t), t \geq 0\}$ representing the number of firings released by the neuron up to time t . To this purpose, for all $z \in (r_1, S)$, let

$$p_n(t|z) = P\{M(t) = n | Z(0) = z\} \quad (n = 0, 1, \dots). \quad (4)$$

be the probability of having n firings up to time t . Recalling that the diffusion process $X(t)$ is time-homogeneous and that R_1, R_2, \dots are independent and identically distributed random variables, the following relations can be seen to hold:

$$p_0(t|\eta) = 1 - \int_0^t g(S, \tau|\eta) d\tau \quad (5)$$

$$\begin{aligned} p_k(t|\eta) &= [g(S, t|\eta) * \varphi(t)]^{(k)} * \left[1 - \int_0^t g(S, \tau|\eta) d\tau \right] \\ &\quad + g(S, t|\eta) * [\varphi(t) * g(S, t|\eta)]^{(k-1)} * \left[1 - \int_0^t \varphi(\tau) d\tau \right] \\ &\quad \quad \quad (k = 1, 2, \dots), \end{aligned}$$

where $(*)$ means convolution, exponent (r) indicates (r) -fold convolution, $g(S, t|\eta)$ is the FPT pdf of $X(t)$ through S starting from $X(0) = \eta < S$ and $\varphi(t)$ is the pdf of the refractoriness period. The probabilities $p_k(t|\eta)$ can be used to explore

the statistical characteristics of the random variable that describes the number of firings.

For $k = 0, 1, 2, \dots$, let

$$\pi_k(\lambda|z) = \int_0^{+\infty} e^{-\lambda t} p_k(t|z) dt \quad (\lambda > 0) \quad (6)$$

be the Laplace transform of $p_k(t|z)$. Denoting by $g_\lambda(S|\eta)$ and $\Phi(\lambda)$ the Laplace transforms of $g(S, t|\eta)$ and $\varphi(t)$, respectively, from (5) we have:

$$\begin{aligned} \pi_0(\lambda|\eta) &= \frac{1}{\lambda} [1 - g_\lambda(S|\eta)] \\ \pi_k(\lambda|\eta) &= \frac{1}{\lambda} g_\lambda(S|\eta) [g_\lambda(S|\eta) \Phi(\lambda)]^{k-1} [1 - g_\lambda(S|\eta) \Phi(\lambda)] \\ &\quad (k = 1, 2, \dots). \end{aligned} \quad (7)$$

Further, let

$$\psi_n(\lambda|\eta) = \int_0^{-\infty} e^{-\lambda t} E\{[M(t)]^n|\eta\} dt \quad (n = 1, 2, \dots) \quad (8)$$

be the Laplace transform of the n -th order moment of the number of firings released by the neuron up to time t . From (7) it then follows:

$$\begin{aligned} \psi_n(\lambda|\eta) &= \sum_{k=1}^{+\infty} k^n \pi_k(\lambda|\eta) = \frac{1}{\lambda} g_\lambda(S|\eta) [1 - g_\lambda(S|\eta) \Phi(\lambda)] \\ &\quad \times \sum_{k=1}^{+\infty} k^n [g_\lambda(S|\eta) \Phi(\lambda)]^{k-1} \end{aligned} \quad (9)$$

In particular, since

$$\sum_{k=1}^{-\infty} k x^{k-1} = \frac{1}{(1-x)^2}, \quad \sum_{k=1}^{-\infty} k^2 x^{k-1} = \frac{1+x}{(1-x)^3} \quad (|x| < 1),$$

from (9) for $\lambda > 0$ one obtains:

$$\begin{aligned} \psi_1(\lambda|\eta) &= \frac{g_\lambda(S|\eta)}{\lambda [1 - g_\lambda(S|\eta) \Phi(\lambda)]}, \\ \psi_2(\lambda|\eta) &= \frac{g_\lambda(S|\eta) [1 + g_\lambda(S|\eta) \Phi(\lambda)]}{\lambda [1 - g_\lambda(S|\eta) \Phi(\lambda)]^2}. \end{aligned} \quad (10)$$

Let now I_1, I_2, \dots denote the random variables describing the interspike intervals and $\gamma_k(t)$ the pdf's of I_k ($k = 1, 2, \dots$). We note that I_1 identifies with the

FPT through the threshold S starting at initial state $X(0) = \eta < S$, so that $\gamma_1(t) \equiv g(S, t|\eta)$. Instead, I_k ($k = 2, 3, \dots$) describes the duration of the interval elapsing between the $(k-1)$ -th spike and the k -th spike. Since $X(t)$ is time-homogeneous and R_1, R_2, \dots are independent and identically distributed, there holds:

$$\begin{aligned} P(I_2 > t | I_1 = \tau) &= 1 - \int_0^t \varphi(\vartheta) d\vartheta + \int_{\tau}^{t+\tau} \varphi(\vartheta - \tau) P\{M(t + \tau - \vartheta) = 0 | \eta\} d\vartheta \\ &= 1 - \int_0^t \varphi(\vartheta) d\vartheta + \int_0^t \varphi(x) \left[1 - \int_0^{t-x} g(S, u|\eta) du \right] dx \\ &= 1 - \int_0^t \varphi(x) dx \int_0^{t-x} g(S, u|\eta) du. \end{aligned} \quad (11)$$

By virtue of the independence of $P(I_2 > t | I_1 = \tau)$ on τ , it follows that I_2 is independent of I_1 . The iteration of this argument implies that the interspike intervals I_2, I_3, \dots are independent and identically distributed random variables having pdf

$$\gamma_k(t) = \int_0^t \varphi(\vartheta) g(S, t - \vartheta|\eta) d\vartheta \quad (k = 2, 3, \dots). \quad (12)$$

3 Constant Refractoriness Periods

In this Section an asymptotic analysis of the random process modeling the number of neuronal firings is provided, assuming that the neuronal refractoriness period is unique and non-random. Hence, we assume that R_1, R_2, \dots are independent and identically distributed random variables, each with pdf

$$\varphi(t) = \delta\left(t - \frac{1}{\xi}\right),$$

with $\xi > 0$. Hence, all the above refractoriness periods have duration $1/\xi$, so that from (12) the interspike interval pdf is expressed as:

$$\gamma_k(t) = \begin{cases} g\left(S, t - \frac{1}{\xi} \middle| \eta\right), & t > \frac{1}{\xi} \\ 0, & t \leq \frac{1}{\xi} \end{cases} \quad (k = 2, 3, \dots).$$

Being $\Phi(\lambda) = e^{-\lambda/\xi}$ the Laplace transform of $\varphi(t)$, from (7) one obtains:

$$\sum_{i=k}^{\infty} \pi_i(\lambda|\eta) = \frac{1}{\lambda} [g_\lambda(S|\eta)]^k \exp\left\{-\frac{\lambda(k-1)}{\xi}\right\} \quad (k = 1, 2, \dots). \quad (13)$$

If $\{X(t), t \geq 0\}$ possesses a steady state distribution, we expect that for large firing thresholds an exponential behavior of the firing pdf $g(S, t|\eta)$ takes place

(cf. [6]), i.e. that

$$g_\lambda(S|\eta) \sim \frac{1}{1 + \lambda t_1(S|\eta)} \quad (14)$$

with t_1 defined by (3). Hereafter, we shall assume throughout that this is indeed the case, so that use of (14) can be made. Then, (13) reads:

$$\sum_{i=k}^{\infty} \pi_i(\lambda|\eta) \sim \frac{1}{\lambda} \frac{1}{[1 + \lambda t_1(S|\eta)]^k} \exp\left\{-\frac{\lambda(k-1)}{\xi}\right\} \quad (k = 1, 2, \dots). \quad (15)$$

The right-hand side of (15) identifies with the Laplace transform of

$$\left[\frac{1}{t_1(S|\eta)} \int_0^{t-(k-1)/\xi} q_{k-1}(\tau) d\tau \right] u\left(t - \frac{k-1}{\xi}\right), \quad (16)$$

where

$$u(t) = \begin{cases} 0, & t < 0 \\ 1/2, & t = 0 \\ 1, & t > 0. \end{cases} \quad (17)$$

denotes the unit step function and

$$q_j(t) = \frac{1}{j!} \left[\frac{t}{t_1(S|\eta)} \right]^j \exp\left\{-\frac{t}{t_1(S|\eta)}\right\} \quad (k = 0, 1, \dots). \quad (18)$$

is a Poisson distribution of parameter $[t_1(S|\eta)]^{-1}$. Since from (18) one has

$$\begin{aligned} \int_0^{t-(k-1)/\xi} q_{k-1}(\tau) d\tau &= t_1(S|\eta) \left[1 - \exp\left\{-\frac{1}{t_1(S|\eta)} \left(t - \frac{k-1}{\xi}\right)\right\} \right] \\ &\times \sum_{m=0}^{k-1} \frac{1}{[t_1(S|\eta)]^m m!} \left(t - \frac{k-1}{\xi}\right)^m, \end{aligned} \quad (19)$$

making use of (15), (16) and (19), it follows:

$$\begin{aligned} \sum_{i=k}^{\infty} p_i(t|\eta) &\sim u\left(t - \frac{k-1}{\xi}\right) \left[1 - \exp\left\{-\frac{1}{t_1(S|\eta)} \left(t - \frac{k-1}{\xi}\right)\right\} \right] \\ &\times \sum_{m=0}^{k-1} \frac{1}{[t_1(S|\eta)]^m m!} \left(t - \frac{k-1}{\xi}\right)^m \quad (k = 1, 2, \dots). \end{aligned} \quad (20)$$

Hence, for large firing thresholds one has:

$$p_0(t|\eta) \sim \exp\left\{-\frac{t}{t_1(S|\eta)}\right\} \quad (21)$$

and

$$p_k(t|\eta) \sim u\left(t - \frac{k-1}{\xi}\right) \left[1 - \exp\left\{-\frac{1}{t_1(S|\eta)} \left(t - \frac{k-1}{\xi}\right)\right\} \right]$$

$$\begin{aligned}
& \times \sum_{m=0}^{k-1} \frac{1}{[t_1(S|\eta)]^m m!} \left(t - \frac{k-1}{\xi} \right)^m \Big] \\
& - u \left(t - \frac{k}{\xi} \right) \left[1 - \exp \left\{ - \frac{1}{t_1(S|\eta)} \left(t - \frac{k}{\xi} \right) \right\} \right. \\
& \quad \left. \times \sum_{m=0}^k \frac{1}{[t_1(S|\eta)]^m m!} \left(t - \frac{k}{\xi} \right)^m \right] \quad (k = 1, 2, \dots). \quad (22)
\end{aligned}$$

Furthermore, for large firing thresholds, the expected value of the number of firings released by the neuron up to time t is approximately given by

$$\begin{aligned}
E[M(t)|\eta] &= \sum_{j=1}^{+\infty} j p_j(t|\eta) = \sum_{k=1}^{+\infty} P[M(t) \geq k|\eta] \\
&\sim \sum_{k=0}^{\lceil t\xi \rceil} \left[1 - \exp \left\{ - \frac{1}{t_1(S|\eta)} \left(t - \frac{k}{\xi} \right) \right\} \sum_{m=0}^k \frac{1}{[t_1(S|\eta)]^m m!} \left(t - \frac{k}{\xi} \right)^m \right], \quad (23)
\end{aligned}$$

where $\lceil x \rceil$ denotes the greater integer less or equal to x .

4 Concluding Remarks

The inclusion of refractoriness in a model for single neuron's activity has been the object of the present paper. Differently from previous approaches, in which neuronal input was modeled by means of point processes, a diffusion process has been invoked here to describe the time course of the membrane potential, and refractoriness has been viewed as described by a sequence of independent and identically distributed random variables. A return process has then been constructed, by means of which the probabilities for the neuron to elicit any assigned number of firings up to any assigned time and the interspike probability density have been determined in closed forms for any given probability density for the refractoriness period. Furthermore, in the case of constant duration of the refractoriness period, asymptotic simple formulas have been obtained under the assumption of exponentially distributed firing times.

References

1. Buonocore, A., Giorno, V., Nobile, A.G., Ricciardi, L.M.: Towards modeling refractoriness for single neuron's activity. In *Cybernetics and Systems 2002* Vol. 1 (Trappl, R., ed.). Austrian Society for Cybernetics Studies, Vienna. (2002) 319-324
2. Buonocore, A., Giorno, V., Nobile, A.G., Ricciardi, L.M.: A neuronal modeling paradigm in the presence of refractoriness. *BioSystems* **67** (2002) 35-43.
3. Feller, W.: The parabolic differential equations and the associated semi-groups of transformations. *Ann. Math.* **55** (1952) 468-518.
4. Gerstein, G.L., Mandelbrot, B.: Random walk models for the spike activity of a single neuron. *Biophys. J.* **4** (1964) 41-68.

5. Giorno, V., Lánský, P., Nobile, A.G., Ricciardi, L.M.: Diffusion approximation and first-passage-time problem for a model neuron. III. A birth-and-death process approach. *Biol. Cybern.* **58** (1988) 387–404.
6. Giorno, V., Nobile, A.G., Ricciardi, L.M.: On the asymptotic behavior of first-passage-time densities for one-dimensional diffusion processes and varying boundaries. *Adv. Appl. Prob.* **22** (1990) 883–914.
7. Giorno, V., Nobile, A.G., Ricciardi, L.M.: Instantaneous return process and neuronal firings. In *Cybernetics and Systems Research 1992* (Trappl, R., ed.). World Scientific (1992) 829–836.
8. Giorno, V., Nobile, A.G., Ricciardi, L.M.: On asymptotic behaviors of stochastic models for single neuron's activity. In *Cybernetics and System 1996* (Trappl, R., ed.). Austrian Society for Cybernetic Studies (1996) 524–529.
9. Lánský, P., Smith, C.E.: The effect of a random initial value in neuronal first-passage-time models. *Math. Biosci.* **93** (1989) 191–215.
10. Ricciardi, L.M., Esposito, F.: On some distribution functions for non-linear switching elements with finite dead time. *Kybernetik* **3** (1966) 148–152.
11. Ricciardi, L.M., Di Crescenzo, A., Giorno, V., Nobile, A.G.: On the instantaneous return process for neuronal diffusion models. In *Structure: from Physics to General Systems* (Marinaro, M., Scarpetta, G., eds.). World Scientific (1992) 78–94.
12. Ricciardi, L.M., Di Crescenzo, A., Giorno, V., Nobile, A.G.: An outline of theoretical and algorithmic approaches to first passage time problems with applications to biological modeling. *Math. Japonica* **50** No. 2 (1999) 247–322.
13. Ricciardi, L.M., Lánský, P.: Diffusion models of neuron activity. In *The Handbook of Brain Theory and Neural Networks* (Arbib, M.A., ed.). The MIT Press, Cambridge (2002) 343–348.
14. Siegert, A.J.F.: On the first passage time probability problem. *Phys. Rev.* **81** (1951) 617–623.
15. Teich, M.C., Matin, L., Cantor, B.I.: Refractoriness in the maintained discharge of the cat's retinal ganglion cell. *J. Opt. Soc. Am.* **68**(3) (1978) 386–402.

A Study of the Action Potential Initiation Site Along the Axosomatodendritic Axis of Neurons Using Compartmental Models

J.M. Ibarz and O. Herreras

Dept. Investigación, Hospital Ramón y Cajal, 28034-Madrid, Spain
{jose.m.ibarz, oscar.herreras}@hrc.es

Abstract. The classic view establishing a unique site for AP initiation at the axon initial segment is not longer accepted. Different subregions of the neuron morphology may initiate and conduct APs, widening enormously the computational capabilities of individual neurons. The experimental investigation of non-linear dendritic properties is subject to error because of they are very sensitive to recording conditions. We used an experimentally-fitted compartmental model to examine the factors modulating the AP initiation site of hippocampal pyramidal neurons. Our results indicate that synchronous spatially clustered inputs initiate dendritic APs more effectively than equivalent disperse inputs. The average electrical distance from the dendritic input to the axon trigger zone also determines the initiation site, but it may be strongly modulated by a number of factors, such as the relative excitability of the two trigger zones. The computational meaning of the shifts in the AP initiation locus is discussed.

1 Introduction

Communication between neurons is coded in temporal series of action potentials (APs). A single AP is, however, the common answer to a variety of different processing modes performed by individual neurons. A number of neuron subtypes initiate action potentials (APs) in dendritic loci that may or may not propagate to the axon. Initiation of dendritic APs or spikes enables parallel processing in separate dendrites of individual branches or trees [1], so that cell output may be decided not just in the axon but also in dendritic subunits. This feature is common in neurons with complex dendritic trees such as neocortical and hippocampal pyramidal cells that display a variety of dendritic voltage-dependent activities. The standing question is whether the excitability of dendrites is high enough to initiate APs or simply to conduct backpropagated spikes initiated in the axon initial segment (AIS). The experimental study of active properties is complicated by their intrinsic nonlinear behavior that makes them highly sensitive to recording conditions and tissue preparation. It is only recently that the delicate and complex intrinsic and external modulations are beginning to be studied [2-6].

AP backpropagation in pyramidal cells of the hippocampal regions CA1, is well documented [7-9], but the reports on synaptic initiation of dendritic APs and forward

propagation are variable, ranging from very high probability of initiation to only backpropagation [10-14]. This variability prompted us to investigate on the factors that might influence the site of AP initiation using realistic compartmental models of the hippocampal CA1 pyramidal cells.

2 The model

The model reproduced in detail the average pyramidal cell morphology [9,15]. This was simulated using 265 compartments, distributed in soma, apical and basal dendritic trees and an axon consisting of myelinated portions, Ranvier nodes, initial segment (AIS) and axon hillock (AH) (a 2-D projection of the model neuron is shown in Fig. 1). Compartment length was always >0.01 and $<0.2 \lambda$. A detailed description of the cell morphology can be found in the <http://navier.ucsd.edu/ca1ps> address. Total effective area of the neuron was $66,800 \mu\text{m}^2$ (including spine area). The electrotonic parameters for soma and dendrites were $R_m=70,000 \Omega \text{ cm}^2$, $R_i=75 \Omega \text{ cm}$ and $C_m=0.75 \mu\text{F/cm}^2$, while R_m was 100 and $0.5 \cdot 10^6 \Omega \text{ cm}^2$ for non-myelinated and myelinated axon compartments, respectively. The later had a C_m of $0.04 \mu\text{F/cm}^2$. The input resistance measured at the soma was $140 \text{ M}\Omega$, and τ was 25 ms.

We used seven types of ionic channels to simulate the active properties of the somatodendritic membrane: fast sodium (Na^+), calcium (Ca^{++}), and five potassium currents: delayed rectifier (DR), small persistent muscarinic (M), A-type transient (A), short-duration [Ca]- and voltage-dependent (C) and long duration [Ca]-dependent (AHP). Conductance variables were described with Hodgkin-Huxley type formalism. Due to lengthy description of kinetics, we only provide here the sources and modifications (see *http* address above for details). Except when specified, the kinetics of all these channels were obtained from Warman et al. [16] with some modifications. An A-type K^+ channel was modified from Hoffman et al. [17] as follows: $I_A = \bar{g}_A a^4 b(V_m - E_K)$, $\tau_a = 0.15 \text{ ms}$, $m_{a\infty} = 1/\{1+\exp[(-5-V_m)/10]\}$ for proximal dendrites (up to $150 \mu\text{m}$ from the soma), and $m_{a\infty} = 1/\{1+\exp[(-15-V_m)/8]\}$ for distal dendrites, $\tau_b = 5 \text{ ms}$ if $V_m < -30 \text{ mV}$; $\tau_b = 5+0.26(V_m+30)$ if $V_m \geq -30 \text{ mV}$, $m_{b\infty} = 1/\{1+\exp[(-56-V_m)/8]\}$. In the axon, Na^+ channels were identical as for the somatodendritic membrane, while the DR-type K^+ channel was obtained from Traub et al. [18]. Equilibrium potentials were +45 and -85 mV for Na^+ and K^+ , respectively.

Since the objective of this work requires the reproduction of APs along a somatodendritic axis of heterogeneous geometry and active properties, the channel distributions along the cell morphology has been tuned in a feed-back manner. We begun by using previously optimized electrotonic constants, and the values for channel density (maximum conductance) and kinetics as found during the computer reproduction of the spatiotemporal field potential map of the population spike, which is extremely sensitive to small variations in unitary parameters [9]. The detailed distributions can be obtained from <http://navier.ucsd.edu/ca1ps>. Novel features include the somatofugally increasing density of the K_A current [17] and the slow recovery from inactivation of the dendritic Na^+ channels [19]. Synaptic activation was simulated using conductances that followed alpha functions: $g_{\text{syn}}(t) = \bar{g}_{\text{syn}}$

$(t/\tau_{\text{syn}})\{\exp(1-t/\tau_{\text{syn}})\}$. The synaptic currents are defined as $I_{\text{syn}} = g_{\text{syn}}(t)(V_m - E_{\text{syn}})$, with τ_{syn} of 7 ms and reversal potential of -75 mV for inhibitory GABA_A-type mediated currents. These were distributed in the soma and proximal apical shaft, and initiated 1.5 ms after excitatory synaptic activation. The AMPA-type excitatory synaptic input was simulated with τ_{syn} of 2 ms and reversal potential at 0 mV. The AMPA currents were distributed along the entire apical tree, and activated according to the experiment requirements. Different spatial distributions, loci and maximum conductances were checked for their ability to trigger APs. To determine the AP initiation site and the direction of propagation, we compared the latency of APs recorded in the axon-soma and the main apical dendritic shaft from the readouts of the simulations.

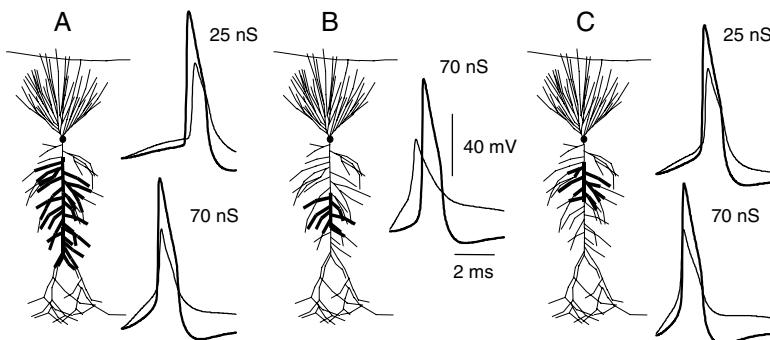


Fig. 1. Effect of varying the spatial pattern and strength of synaptic inputs on the AP initiation site. Drawings correspond to a 2-D projection of the neuron model. Thick-line compartments indicate active excitatory synaptic inputs. A: Scattered patterns of synaptic inputs initiate the AP at the axon-soma (larger AP, thick tracing). Increasing the total input strength accelerated AP backpropagation. B: Clustering of synaptic inputs shift the AP initiation locus toward dendritic locations. C: Moving the synaptic input toward proximal regions again shifted the AP initiation locus toward the axon-soma, but strong inputs (70 nS) still initiated dendritic APs.

3 Results

After optimization of electrotonic parameters and spatial distribution of channel conductances, gross features of excitatory synaptic potentials (EPSPs) and APs, AP initiation and propagation reproduced well experimental findings. APs recorded in the soma are larger (90-100 vs. 40-60 mVs) and shorter than those in the apical shaft. The voltage threshold for AP initiation was about 8-10 mV at the axon-soma and >20 mV at the apical shaft. However, once initiated at any location, the AP propagated without interruptions along the axo-somato-dendritic main axis for most input patterns examined.

We first examined the effect of varying the spatial distributions of excitatory inputs by activating the same number of synapses on dendritic bands of different spatial

extent. Using a scattered input that spanned throughout 300 μm of the apical dendritic tree (about 70% of the total extent), the AP initiated typically in the axon-soma region. Within scattered input patterns, the latency of the AP decreased with increasing number of activated synapses, and most notably, the apparent speed of backpropagation (axon-soma-to-dendrite conduction) increased (compare 25 nS and 70 nS in Fig. 1A). When the same number of activated synapses were clustered within narrower dendritic bands (with the same average electrical distance to the axon-soma), the initial AP locus shifted from the axon toward dendrites (see Fig. 1B). This synaptic clustering caused higher local voltages that reached threshold before the soma-axon region, turning the dendrite into a decision-making structure. This result has a notable physiological meaning as several inputs from afferent regions contact discrete bands within the dendritic tree. The location of the activated zone is still important. The closer is the activated band to the soma region the higher is the probability of axonal initiation (Fig. 1C), although again strong inputs may be effective to discharge dendrites first (25 vs. 70 nS, in Fig. 1C).

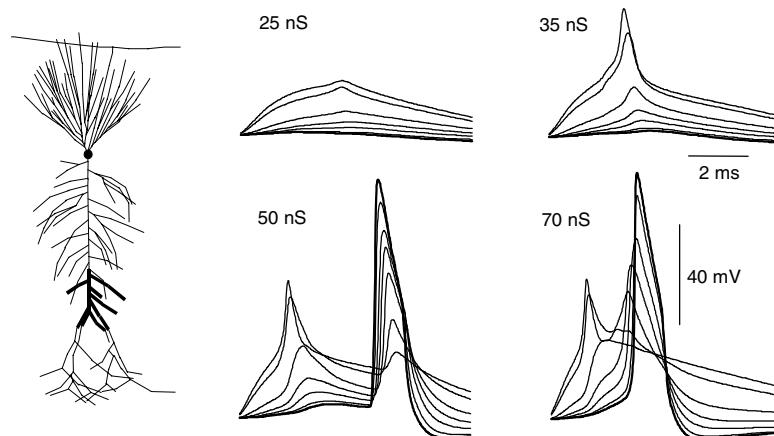


Fig. 2. At specific input locations, the AP initiation and conduction may show different patterns with varying synaptic strength. Thick tracings correspond to the soma and successive tracings to distal dendritic loci.

Increasing the strength of spatially clustered synapses at a distal location may cause a variety of spike patterns (Fig. 2). Small inputs caused no AP initiation (25 nS) or distal dendritic APs that failed to propagate into the apical shaft and soma-axon (35 nS). Moderate inputs initiated distal spikes that failed to conduct along the apical shaft but regenerated an axonal spike (saltatory-like conduction) that backpropagated again into dendrites (50 nS). Larger inputs initiated distal abortive spikes that regenerated proximal dendritic spikes and forward conduction (70 nS).

An important factor to determine the initial locus of AP is the excitability of the trigger zone at the axon initial segment (AIS), classically considered the primary trigger zone. The distance of the synaptic input to the AIS, and the relative excitability of dendritic and axonal trigger zones define the initial AP locus. Figure 3

shows parametric diagrams for clustered (top) and scattered (bottom) inputs. Overall, the higher the axonal excitability the higher is the probability of axonal AP initiation, regardless of the input strength. At low axonal excitability rates, dendritic AP initiation or failure becomes more likely.

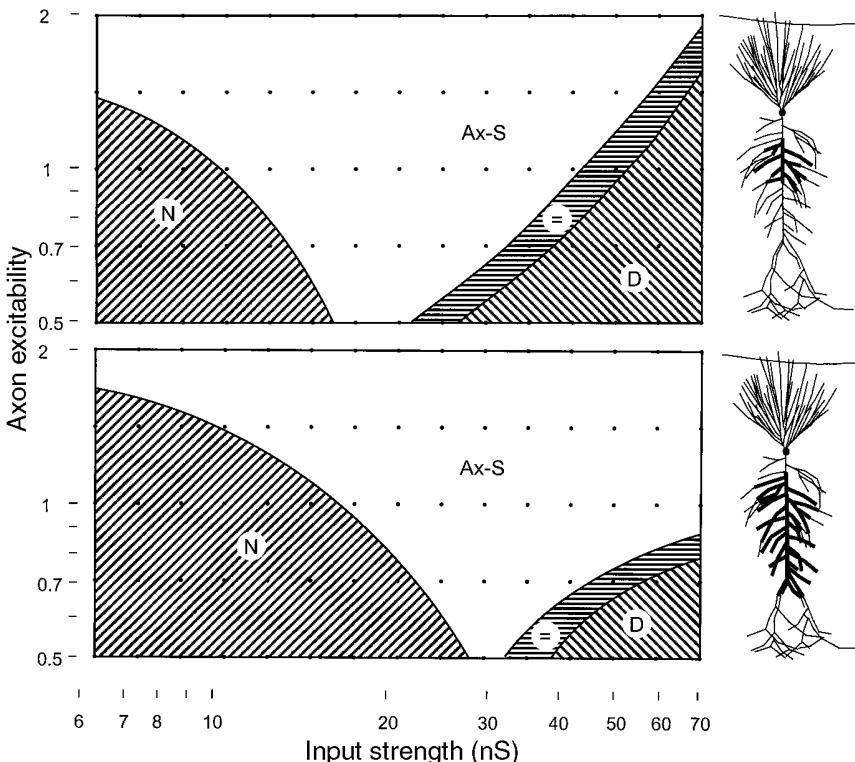


Fig. 3. Parametric diagrams showing the different AP initiation loci for increasing input strengths (abscissa) and increasing excitability of the AIS (ordinate) for clustered (top) and scattered (bottom) inputs. D: dendritic initiation; Ax-S: axonal initiation; =: simultaneous firing; N: No firing.

5 Discussion: The computational meaning of shifting AP initiation loci

The role played by the AP machinery in the thick apical shaft is crucial since synaptic inputs converge there before reaching the output zone at the soma/axon. Whether the AP is first initiated in the apical shaft or the axon is not a trivial question as it defines when a neuron operates as a multiplexing device (multiple decision-making zones) or as a simple integrator. Somatodendritic APs constitute a major element of dialog between input (dendrites) and output regions (axon), with different functional implications when travelling in one direction or another. Backward conduction

(backpropagation) of APs into the dendritic tree is being extensively studied as a possible mechanism to modulate synaptic inputs, participating in different plastic phenomena. A different role for dendritic APs is their secondary triggering following slow calcium spikes rised by a backpropagated axo-somatic fast sodium spike. Those may in turn be conducted forward again to the soma-axon, building up complex patterns of output signals that were in fact tailored by spike activity in dendrites.

Knowing whether the apical shaft is geometrically and electrically designed to conduct or to fade APs out is also crucial, since it will define the degree of final decision for cell output endowed to dendrites. In neocortical pyramidal cells, Larkum et al. [20] have proposed that the proximal portion of the apical shaft is a strongly interactive coupling zone between two other triggering regions located in the distal dendrites and the AIS. Not only subtle voltage modulations in this zone may control the passage of distally initiated APs, but these may also be initiated there [10,14,20,21]. In general, a higher reliability of AP backpropagation over forward conduction of dendritic APs is assumed based on the asymmetric geometry and/or the weaker excitability of soma and dendrites. From a functional point of view, the heterogeneous geometry of the activated cell subregions (AIS, soma and apical dendrite) implies that the main cell axis behaves as a totally different conducting cable when the AP propagates in one direction or another [22]. Contrary to current views, a moderately flaring apical shaft is a suitable structure that facilitates AP forward conduction.

As it concerns to the shift of AP locus between two main triggering zones in the AIS and the main apical shaft of CA1 pyramidal cells, since dendritic firing imposes cell output, it actually means successful transmission of highly specific information that impinged on discrete strips of the apical tree. The apical dendrite may be envisaged in these cases as a collector zone from a multiplexing device performing parallel processing in different dendritic subregions, each capable by itself or in combination with other types of synaptic input to decide cell output. Axonal triggering of APs makes specific inputs to become less relevant and a strong cooperative behavior between different types of incoming information would be necessary to set the output. It is as yet unknown whether the two modes of neuron operation coexist or else they are characteristic of different behavioral states.

References

- 1 Shepherd GM and Brayton RK Logic operations are properties of computer-simulated interactions between excitable dendritic spines. *Neuroscience* (1987) 21:151-65.
- 2 Cantrell AR, Scheuer T and Catterall WA Voltage-dependent neuromodulation of Na^+ channels by D1-like dopamine receptors in rat hippocampal neurons. *J Neurosci* (1999) 19: 5301-5310.
- 3 Tsubokawa H and Ross WN IPSPs modulate spike backpropagation and associated $[\text{Ca}^{2+}]_i$ changes in the dendrites of hippocampal CA1 pyramidal neurons. *J Neurophysiol* (1996) 76: 2896-2906.
- 4 Tsubokawa H and Ross WN Muscarinic modulation of spike backpropagation in the apical dendrites of hippocampal CA1 pyramidal neurons. *J Neurosci* (1997) 17:5782-5791.

- 5 Hoffman DA and Johnston D Downregulation of transient K⁺ channels in dendrites of hippocampal CA1 pyramidal neurons by activation of PKA and PKC. *J Neurosci* (1998) 18:3521-3528.
- 6 Colbert CM and Johnston D Protein kinase C activation decreases activity-dependent attenuation of dendritic Na⁺ current in hippocampal CA1 pyramidal neurons. *J Neurophysiol* (1998) 79:491-495.
- 7 Sperti L, Gessi T, Volta F Extracellular potential field of antidromically activated CA1 pyramidal neurons. *Brain Res* (1966/67) 3:343-361.
- 8 Leung LS Potentials evoked by alvear tract in hippocampal CA1 region in rats. II. Spatial field analysis. *J Neurophysiol* (1979) 42: 1571-1589.
- 9 Varona P, Ibarz JM, López-Aguado L and Herreras O. Macroscopic and subcellular factors shaping CA1 population spikes. *J. Neurophysiol.* 83:2192-2208, 2000.
- 10 Regehr W, Kehoe J, Ascher P and Armstrong C Synaptically triggered action potentials in dendrites. *Neuron* (1993) 11:145-151.
- 11 Stuart GJ and Sakmann B Active propagation of somatic action potentials into neocortical pyramidal cell dendrites. *Nature* (1994) 367:69-72.
- 12 Colbert CM and Johnston D Axonal action-potential initiation and Na⁺ channel densities in the soma and axon initial segment of subiculum pyramidal neurons. *J Neurosci* (1996) 16: 6676-6686.
- 13 Stuart G, Schiller J and Sakmann B Action potential initiation and propagation in rat neocortical pyramidal neurons. *J Physiol* (1997) 505:617-63.
- 14 Golding NL and Spruston N Dendritic sodium spikes are variable triggers of axonal action potentials in hippocampal CA1 pyramidal neurons. *Neuron* (1998) 21:1189-1200.
- 15 Varona P, Ibarz JM, Sigüenza JA, and Herreras, O. Current source density analysis as a tool to constrain the parameter space in hippocampal CA1 neuron models. Lecture Notes in Computer Science1240, pp 82-90, 1997.
- 16 Warman EN, Durand DM and Yuen GLF Reconstruction of hippocampal CA1 pyramidal cell electrophysiology by computer simulation. *J Neurophysiol* (1994) 71:2033-2045.
- 17 Hoffman DA, Magee JC, Colbert CM and Johnston D K⁺ channel regulation of signal propagation in dendrites of hippocampal pyramidal neurons. *Nature* (1997) 387:869-875.
- 18 Traub RD, Jefferys JGR, Miles R, Whittington MA and Tóth K A branching dendritic model of a rodent CA3 pyramidal neurone. *J Physiol* (1994) 481:79-95.
- 19 Colbert CM , Magee JC , Hoffman DA and Johnston D Slow recovery from inactivation of Na⁺ channels underlies the activity-dependent attenuation of dendritic action potentials in hippocampal CA1 pyramidal neurons. *J Neurosci* (1997) 17:6512-6521.
- 20 Larkum ME, Zhu JJ and Sakmann B Dendritic mechanisms underlying the coupling of the dendritic with the axonal action potential initiation zone of adult rat layer 5 pyramidal neurons. *J Physiol* (2001) 533.2: 447-466.
- 21 Shen GY, Chen WR, Midtgård J, Shepherd GM and Hines ML Computational analysis of action potential initiation in mitral cell soma and dendrites based on dual patch recordings. *J Neurophysiol* (1999) 82: 3006-3020.
- 22 López-Aguado L, Ibarz JM, Varona P and Herreras O. Structural inhomogeneities differentially modulate action currents and population spikes initiated in the axon or dendrites. *J. Neurophysiol* (2002) 88:2809-2820.

Real Neurons and Neural Computation: A Summary of Thoughts

José Mira Mira

Dpto. de Inteligencia Artificial,
Facultad de Ciencias y ETS Ing. Informática. UNED. Madrid. SPAIN
jmira@dia.uned.es

Abstract. It is our deep feeling that Computational Neuroscience and Connectionist Engineering are in stagnancy and some fresh air is needed. The purpose of this paper is to contribute to the description of the possible causes of this blockage: lack of a new mathematics for plasticity, fault tolerance, cooperative processes, and abstraction mechanisms to link the physical level to the cognitive processes. Also is clear that we have much more data than contributions to a realistic theory of the brain. As engineering we find again the lack of methodology, serious limitations of the formal model underlying all the ANN paradigms and the necessity to end with the old rivalry between the symbolic and connectionist perspectives of AI.

1 Introduction

Many symposia on neural networks, as IWANN [10], and plenty of papers in journals on neural computation or artificial neural nets are based on the old dream of neuocybernetics about the usefulness of the interplay between two fields: (1) Mathematical modeling of real neurons and neural networks and (2) A model of connectionist computation (an adaptive graph) with a large number of “small grain” elements of local calculus (weighted sum followed by sigmoid), where explicit and declarative external programming is partially substituted by “learning” through supervised and/or non-supervised algorithms.

In spite of the efforts made by a relevant part of the scientific community it is our deep feeling that both fields (Computational Neuroscience and Connectionist Engineering) are in stagnancy and some fresh air is needed. The purpose of this paper is to contribute to the description of the possible causes of this blockage.

The content of this work is organized as follows. Section two is devoted to enumerate the bottlenecks in neural modeling which are essentially three: (1) The lack of a new mathematics to short the distance, almost irreversible, of the usual analogical and logical models from the biological phenomenology of real neurons. (2) The lack of new sources of biological inspiration, including abstraction mechanisms to link the physical level to the cognitive processes. (3) The need of more contributions in search of a realistic theory of the brain. This new brain theory is needed to undergo a qualitative leap

forward from the old logical theory of McCulloch and Pitts, but still the unique complete.

Section three deals with a potential solution to the problem of abstraction and the use of inferential rules to model electrophysiological data without the formal restrictions of analytical tools.

Section four constitute the second part of this paper and is devoted to enumerate the equivalent problems in the engineering counterpart of neural modeling, known as “artificial neural nets” (ANN). These problems are essentially the lack of methodology, the limitations of the computational model underlying all the ANN paradigms (“simple” cooperativity) and the old rivalry between symbolic and connectionist problem solving methods (PSMs). Presently there is a tendency to accept that ANNs are “simply” a set of PSMs in order to be used alternatively or concurrently with other “symbolic” PSMs to decompose generic classification tasks in those situations where we have more data than knowledge. That is to say, inferential rules and ANNs are different ways of operationalization of the primitive inferences that constitute the different reasoning steps by means of which a neurosymbolic PSM solves a task.

2 Neural Modeling and the Mathematics of Physics

The phenomenology addressed in neural modeling covers the different levels of integration from ionic channels and membrane potentials to dendro-dendritic architectures, neurons and neural networks and uses three types of formal tools [1,8]:

- (1) Linear and non-linear *integro-differential equations* in the Biophysical models that take carefully into account the morphology of neurons relating the ionic currents with the spatio-temporal evolution of the membrane potential.
- (2) *Convolution like functionals* in the “punctual” (morpho-less) models, only concerned with the functional relationships between the input and output spaces. The output functional of a neuron at a point and a specific time is the result of an operation on the data sampled from the receptive field. To this group of neural models belong all the models exported to the field of ANN, including multilayer perceptrons, radial-basis function networks and self-organizing circuits [3].
- (3) Probability theory and statistical mechanics in the stochastic models of single cells and of a mass of cells.

A distinctive characteristic of all these models (biophysical, punctual and stochastic) is that they are built using only the mathematics of physics. In fact in most of the cases it would suffice to exchange the words “electron” or “atom” by “synapse” or “neuron” to find the possibility of rewriting in neuronal terms a chapter of a book on physics [2]. This is so because the observables used to build these models correspond to measured electrical magnitudes as function of time. Then, the mathematical operators currently used in physics are proper to describe the relationships between the input and output observables of a neural circuit.

This physical approach to the computational modeling of the Nervous System is valid when the number of neurons is reduced and they are close to the external environment, because under these circumstances we can measure the electrophysiological

data (ganglion cells, motoneurons). However, this approach tends to be inadequate when the number of neurons is very large and we move towards central areas, far away from both sensors and effectors. This is just the case when we try to model the cerebral dynamics processes of cortex [7]. Lesion based experimental results also point in this direction. The high reliability and functional stability of neural tissues after local lesions demands computational models of cooperative processes between complex entities, almost social in nature.

The information associated to the signals measured in cortex is the result of: (1) A considerable number of convergent and divergent processes in the neural wiring (2) multiple, and nested, feedback loops (3) several synaptic recoding processes (4) adaptive changes of wiring and synaptic functionality during operation. Consequently, it is really difficult, when not impossible, to keep the spatio-temporal track of the multiple causal links of this unfinished architecture contributing to the final measured value.

Usual models consider a structure like that of figure 1 where each new layer of “neurons” is generating an internal representation of the external world. Each axon (“axis”) represents a measure of a linear or nonlinear property of the outputs of the previous layer. Let us call these functionals $\{\Phi_j(\bar{x})\}$. The output representation space is then constituted by the axons $\{z_k\}$ of this layer. And this scheme is repeated again and again. In the simplest case it is assumed that the neurons of the output layer compute a convolution, $y_j = \Phi_j(x)$, with some specific kernel, $k(y, \beta)$. In more complex cases it is considered that the interaction between neurons is non-linear and some sort of Wiener-Volterra functional series expansion is used. High order kernels represent now the physical interactions between different components of the past of the input, and between different inputs.

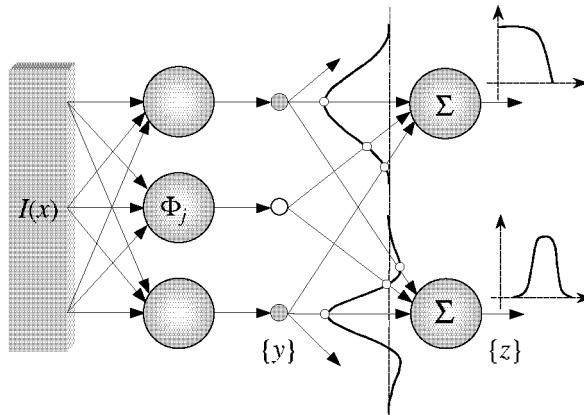


Fig. 1. General structure of many neural models. The first layer is used to create a functional expansion of the input space (“internal representation”). Then, the elements of the second layer compute an adaptive combination of the outputs of this new basis.

How can we escape from this blockage? We think that the mathematical nature of neural nets is still an open question and we need new formal tools to cope with plasticity, information, neural assemblies, neurophysiological symbols and cooperative proc-

esses. In the meantime it could be valuable to complement the mathematics of physics with the formal tools developed in Artificial Intelligence (AI) to model non-analytical human knowledge and reasoning processes in problem solving tasks. This could start by using inferential rules to model neurophysiological data. Then, the functionals Φ^* can be algorithmic and the “totalizers” (convolution) can be formulated as computational ways of combining evidence. Each node (neuron, synaptic contact) is functionally modeled using an inference rule (“if condition then action”) where the left side of the rule can contain any logic or relational expression capable of being evaluated by a compiler. In the same way, the right part of the rule codifies the output action, which is a labeled line with an output selected from a library of potential response pattern generators. Given that the condition part of the rule can include any combination of analog, logic or relational operators, the usual analog models (linear and nonlinear) are particular cases of the inferential model. The same happens with the action part of the rule that can accommodate the usual decision function (linear, sigmoid, gaussian, ...) [13].

This extension of the functionals Φ_j to inferential rules enables us to include more functionalities of real neurons such us: non linear transducers, pre-synaptic multiplicative interaction, absolute facilitation and inhibition and different functional modes (oscillatory, tuned, tonically activated). The inferential model can also accommodate a great diversity of response firing patterns such us single output and slow potential mode, single output and spike train mode, pacemaker mode or coordinated modes (synchronization, cooperation, group selection, ...).

3 From Anatomical Circuits to Inferential Schemes: An Abstraction Process

The usual way to envisage the neural circuits for both, modeling the causal links between physiological signals and looking after biological inspiration for ANN computation, has been and still is to look at the anatomical level in an attempt to reproduce the anatomical “wiring” assuming as a fact that we can duplicate the neural function and the plasticity of synapses.

But there is another way to look at the anatomical circuits: By means of an abstraction process specified according to the following rules (see figure 2):

1. All the physiological entities and relations are abstracted into inferential entities (nouns and roles) and relations (inferential verbs).
2. Input and output spaces of physical signals are changed to representation spaces to constitute the “domain ontology”.
3. Local functions are abstracted as the inferential verbs “evaluate” and “select”.
4. Passive anatomical connections are now considered static roles of the neural inference and active connections are changed to dynamic roles shared by causally connected inferences.
5. Finally, the complete anatomical circuit under the abstraction process becomes an inferential scheme that represents the “symbolic” reasoning steps carried out by the neural net.

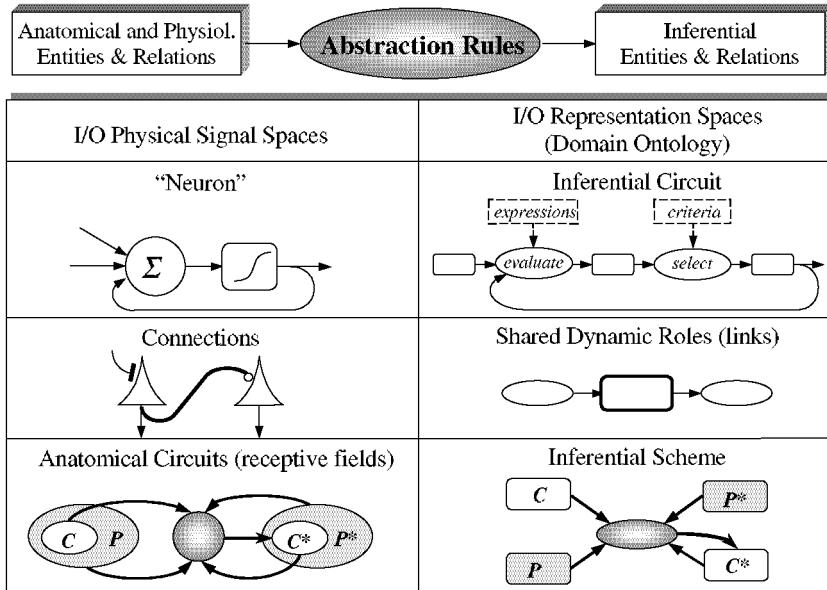


Fig. 2. Summary of abstraction rules to go from anatomical circuits to inferential schemes at the cognitive level. A way of interplay between neural and symbolic modeling

This abstraction process could enable us to interpret neural function at cognitive level and also provides a link with the inferential models of ANN previously mentioned. In this way symbolic and connectionist models of neural networks and knowledge-based-systems (the computational perspective of cognitive processes) would be integrated. The outstanding difference only emerges when we select the specific mathematical operators used to make these inferences computable. In some cases these operators are proper for the modeled phenomenology. In most other cases, these operators only exist in the language but we lack of computable (mathematical) versions. And this is the bottle cork we need to remove to solve the stagnancy in neural modeling and ANN.

4 Some Problems in ANN: The Lack of Methodology and the Need of Neurosymbolic Integration

Three fundamental problems in ANN are (1) the lack of methodology (2) the limitations of the formal model underlying all the ANN paradigms and (3) the necessity to end with the old rivalry between the symbolic and connectionist perspectives of AI.

Concerning methodology it seems crystal clear to us that the development of applications using ANN has to follow, at least, the same steps followed by software engineering and knowledge engineering [12]. Instead of start with "blind nets" we have to start with a "theory of the calculus" or conceptual model of the task as stated by Marr

[5] and Newell [11]. This “theory of calculus” constitutes the additional knowledge used in the skeletal design of the net architecture. Then, we have to develop libraries of reusable components of modeling (abstraction, classification and refinement subtasks, functional expansion methods, learning methods, primitive inferences and control). We also need to develop and use connectionist ontologies for the domain knowledge. In the build-up and use of the *ontology*, a great effort is needed, first in data analysis and codification (labeled input and output lines) and later in the semantic interpretation of the ANN numerical results. Finally, we need clear rules for the operationalization of the primitive inferences according to the balance between data (labeled and unlabeled) and knowledge available for each one of these inferences.

The second problem of ANN is concerned with the computational limitations of the mathematical model underlying most of the usual paradigms (MLP, RBF, ...), including Wiener-Volterra series and Kolmogorov theorem. We have commented on this point in the second and third sections of this paper and we have no other alternatives that to look after new mathematical developments and, in the meantime, use the modal probabilistic and fuzzy *logico-relational* operators usual in symbolic AI. Obviously we have to take into account the need of “small grain” to cope with the eventual necessity of real time learning (adjusting the values of parameters in the left and/or right parts of rules).

This problem of the limitations of local function leads us to the third problem: the need of neuro-symbolic integration [4,14]. The past rivalry between symbolic and connectionist PSMs was due essentially to ignorance on the implications of the knowledge level, as introduced by Newell [11] and Marr [5]. Now is generally accepted that they are different and complementary forms of modeling and operationalizing the inferences in terms of which a problem solving method (PSM) decomposes a task.

There are three basic strategies for neuro-symbolic integration (NSI): (1) System level, (2) architecture and data structure level and (3) the unified approach that uses “inferential neurons” as the unique functional module for both sort of systems, ANN and KBSSs [4,9,14].

The system level approach includes paradigms such as neuro-fuzzy systems (NFS) and adaptive resonance theory architectures (ART) that mix together neural, fuzzy and symbolic elements in the different phases of the development of an application. In NFS, for instance, we can use fuzzy techniques for knowledge acquisition and neural techniques for learning. Also we can use numerical inputs (parameters that define a membership function) and neural calculus (adders, products and threshold functions). In some others cases ANN are used to obtain symbolic rules.

The “architecture and data structure” strategies use symbolic and neuronal modules integrated in a hybrid architecture where the connectionist component (N) play the role of pre-processor, post-processor or sub-processor of a main symbolic processor (S). When the integration is faced close to the implementation phase, the N and S components have to share the data structures and take into account the exchange of information and the S control of the N subordinated components. There is another way of understanding the hybrid approach which consists on facing the NC integration very early, in the conceptual modeling phase. That is, at the knowledge level, in the domain of the external observer, and as soon as we have a clear idea on the data and knowl-

edge available to solve a task, a subtask or an elemental inference, as illustrated in figure 3 [6,9]. In this way we can consider that all the PSM are hybrids in principle, because we are only forced to select between N or S formal operators at the operationalization phase. Then, all the PSMs inferences and formal tools belong to a unique library of reusable components for knowledge modeling.

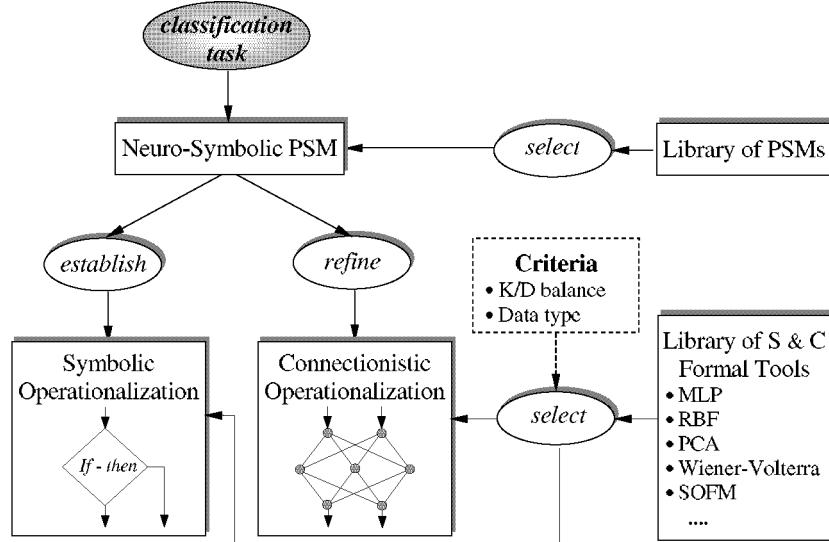


Fig. 3. The knowledge level approach to neurosymbolic integration

Finally, in the unified approach, it is supported that we only need ANN with extended computational facilities to cope with a broad set of tasks. A possible generic PSM that we propose here to solve these tasks is a modification of the “heuristic classification” proposed by Clancey where we have changed “*match*” for “*classify*”. The connectionist PSM for the unified approach could be “*abstract-classify-refine*”, as shown in figure 4. The “*abstract*” inference includes all the data analysis, sensitivity studies and selection of labeled lines. The inference “*classify*” takes care of the numerical associations and “*refine*” is in charge of the interpretation of the results, including the injection of semantics.

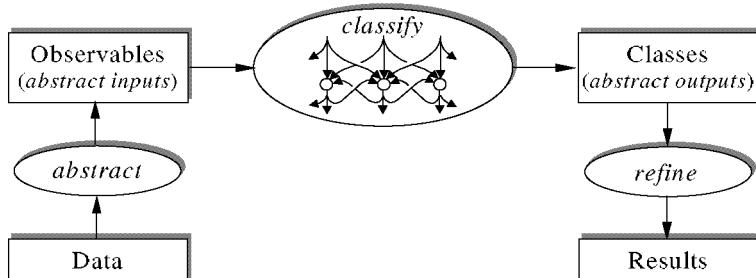


Fig. 4. A generic method for the unified approach

Of the three integration strategies (unified, hybrid and system level) we consider as more comprehensive the knowledge level approach of the hybrid one, because this include as particular cases the other two.

Conclusions

We believe the selected bottlenecks are representative of the current state in the area but there are still many other conceptual, formal and methodological items which need to be addressed to unblock the field.

References

1. Arbib, M. (ed.): *The Handbook of Brain Theory and Neural Networks*. The MIT Press, Cambridge, MA (1995)
2. Feynman, R.P., Leighton, R.B., Sands, M.: *The Feynman Lectures on Physics*. Addison-Wesley, Reading, Mass (1966)
3. Haykin, S.: *Neural Networks: A Comprehensive Foundation*, Prentice-Hall. (1999)
4. Hillario, M., M., Llamement, Y. and Alexandre, F.: Neurosymbolic integration: Unified versus hybrid approaches. *The European Symposium on Artificial Neural Networks*. Brussels, Belgium (1995)
5. Marr, D.: (1982) *Vision*. Freeman, New York
6. Mira, J. and Delgado, A.E.: Where is knowledge in robotics? Some methodological issues on symbolic and connectionist perspectives of AI. In Ch. Zhou, D. Maravall and Da Rua (eds.) *Autonomous Robotic Systems*. Ch. I. Physica-Verlag. Springer-Verlag (2003) 3-34
7. Mira, J., Delgado, A.E.: A Logical Model of Co-Operative Processes in Cerebral Dynamics. *Cybernetics and Systems, An International Journal*, 18: (1987) 319-349
8. Mira, J., Delgado, A.E.: Neural Modeling in Cerebral Dynamics. *Mathematical Biosciences*. Special number ed. by L. Ricciardi, 2003 (pend. pub.)
9. Mira, J., Delgado, A.E., Taboada, M.J.: Neurosymbolic Integration: The Knowledge Level Approach. R. Moreno-Díaz et al (eds) *Cast and Tools for Complexity in Biological, Physical and Engineering Systems*. Extended Abstracts of EUROCAST (2003)
10. Mira, J., Prieto, A. (eds.): *Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence*. LNCS 2084, Springer Verlag (2001)
11. Newell, A.: The knowledge level. *AI Magazine*, summer 1-2 (1981)
12. Schreiber, et al.: *Engineering and managing knowledge: The CommonKADS methodology*. The MIT Press (1999)
13. Schwartz, E.L. (ed.): *Computational Neuroscience*. MIT Press (1990)
14. Sun, R., Alexandre, F. (eds.): *Connectionist-Symbolic Integration*. Lea. London (1997)

Synchronous firing in a population of inhibitory interneurons coupled by electrical and chemical synapses

Santi Chillemi, Angelo Di Garbo, Alessandro Panarese

Istituto di Biofisica CNR, Area della Ricerca di Pisa, G. Moruzzi 1, 56124 Pisa (Italy)
chillemi@ib.pi.cnr.it, digarbo@ib.pi.cnr.it, a.panarese@ib.pi.cnr.it

Abstract. The synchronization phenomena occurring in a population of Fast Spiking interneurons, coupled by electrical and inhibitory synapses, are investigated. It is shown numerically that there are regions of the coupling parameters space where synchronous regimes occur. The corresponding theoretical results agree with the numerical ones. It is shown that the presence of the electrical coupling in a heterogeneous population of inhibitory FS cells promote their synchronous discharge.

1 Introduction

At present a relevant and open question is to understand how the coupling features among interneurons lead to the emergence of their synchronous firing [1]. Recent experiments have shown that networks of GABAergic inhibitory interneurons in neocortex are able to generate synchronous discharges in the gamma frequency range (30 – 80 Hz) [1, 2]. Besides, both experimental and theoretical evidences relate the synchronous discharge of a population of inhibitory interneurons to some features of their coupling, like the duration and intensity of the synaptic current [2-7].

Recently two classes of neocortical inhibitory interneurons – Fast Spiking (FS) and Low Threshold Spiking cells (LTS) – were found to interact by electrical synapses too [8, 9]. FS cells exhibit a higher level of inhibitory connections, among them, while a lower level is found for the LTS group. Instead, electrical synapses between two members of different interneuron classes were found to be rare. Also, paired recordings from interneurons of the same group revealed their tendency to discharge synchronously. These findings suggest that FS and LTS cells could be two independent networks having different roles for the neural information coding [8, 9].

The above results raise important questions concerning the role played by both the chemical and electrical synapses for generating coherent firing activity in networks of inhibitory interneurons. In this paper the synchronization properties in a population of FS interneurons interacting by inhibitory and electrical synapses are investigated.

2 Model description and analysis method

The basic properties of an FS cell are: a) the action potentials are shorter than those of pyramidal neurons and exhibit afterhyperpolarization; b) the decay time constant of the synaptic currents is $\sim 10.5 \text{ msec}$ [8, 9]. A single compartment model of an FS cell, well accounting for those features, is the following [10]:

$$\begin{aligned} C dV/dt &= I_E - g_{Na} m^3 h(V-V_{Na}) - g_K n^4 (V-V_K) - g_L (V-V_L) \\ dx/dt &= (x_\infty - x) / \tau_x, \quad x_\infty = \alpha_x / (\alpha_x + \beta_x), \quad \tau_x = 1 / (\alpha_x + \beta_x), \quad (x \equiv m, h, n) \end{aligned} \quad (1)$$

where $C = 1 \mu\text{F}/\text{cm}^2$ and I_E is the external stimulation current, and $\alpha_m = 4.2 \exp[(V+34.5)/11.57]$, $\beta_m = 4.2 \exp[-(V+34.5)/27]$, $\alpha_h = 0.09 \exp[-(V+45)/33]$, $\beta_h = 0.09 \exp[(V+45)/12.2]$, $\alpha_n = 0.3 \exp[(V+35)/10.67]$, $\beta_n = 0.3 \exp[-(V+35)/42.68]$. The maximal specific conductances and the reversal potentials are, respectively: $g_{Na} = 100 \text{ mS/cm}^2$, $g_K = 40 \text{ mS/cm}^2$, $g_L = 0.1 \text{ mS/cm}^2$ and $V_{Na} = 55 \text{ mV}$, $V_K = -90 \text{ mV}$ and $V_L = -68 \text{ mV}$. In this paper, in order to simplify the computation, a reduction of this model will be employed. Firstly, assuming that the fast gating variable m is set to its equilibrium value leads to a reduction of the previous model to one described by three equations. Moreover, simulations of the complete model show that the sum of the gating variables n and h is approximately constant. Then, after estimating the average value of this quantity, the number of equations is further decreased by one by setting $h = 0.927-n$. So the final two dimensional model of each FS cell reads

$$\begin{aligned} C dV/dt &= I_E - g_{Na} (m_\infty)^3 (0.927-n)(V-V_{Na}) - g_K n^4 (V-V_K) - g_L (V-V_L) \\ dn/dt &= (n_\infty - n) / \tau_n. \end{aligned} \quad (2)$$

This model retains many of the dynamical properties characterizing the complete one. For instance, for the complete model, it was shown in [7] that the transition from rest to periodic firing occurs by a saddle-node bifurcation. Thus, the excitability properties are those of class I neural models (i.e. the periodic solution originates with arbitrarily low frequency values). Now, for the reduced model there is a critical value of the stimulation current, $I^* \cong 0.254 \mu\text{A}/\text{cm}^2$, above which the model generates periodic firing of arbitrarily low frequency. For $I_E < I^*$ there are three fixed points which can be characterized by inspection of the eigenvalues of the Jacobian matrix, M : two of them are unstable (and correspond to a source and a saddle) and the remaining one is stable (corresponding to a sink). Moreover all eigenvalues are real. Let $\text{Tr } M$ and $\det M$ denote the trace and the determinant of M , respectively. Then for a fixed value of I_E the sink and the saddle are located in the region $V < -40 \text{ mV}$ and the sink is a stable node ($\text{Tr } M < 0$, $\det M > 0$). The source falls in the region $V > -40 \text{ mV}$ and it is an unstable node ($\text{Tr } M > 0$, $\det M > 0$). As I_E approaches I^* from the left the stable node and the saddle stationary points get closer up to collide (for $I_E = I^*$) and disappear. For $I_E > I^*$ only one fixed point remains (the unstable node) and a periodic orbit or limit cycle appears. The periodic orbit arises through a saddle node bifurcation and the corresponding frequency is plotted vs. I_E in the left panel of figure 1; inspection of this plot shows that also the reduced model is class I. Close to a saddle-node

bifurcation point the discharge frequency scales as $v = A(I_E - I^*)^{1/2}$ where A is a constant, and the corresponding fit for $I_E > I^*$ is reported in the same panel ($A = 42$). The above results imply that the reduced model preserves the main qualitative and quantitative features of the complete one and so it will be used here to model the FS interneuron. The main advantage of this choice is a significant reduction of the computational effort weight.

Experimental studies have shown that the average conductance values for electrical and inhibitory synapses are $G_{El} = 0.66 \text{ nS}$ and $G_{Sy} = 1.7 \text{ nS}$, respectively [8,9]. The electrical and chemical synapses were modelled as follows. The postsynaptic current for the GABAergic synapse is given by: $I_{Sy}(t) = g_{Sy} S_{Pre}(t)(V(t) - V_{Rev})$, where g_{Sy} is the maximal specific conductance for the inhibitory coupling and $V_{Rev} = -75 \text{ mV}$ is the reversal potential for the neurotransmitter. The fraction of receptors in the open state on the postsynaptic neuron $S_{Pre}(t)$ obeys the first order kinetics $dS_{Pre}(t)/dt = T(V_{Pre})(1 - S_{Pre}(t)) - \tau^{-1} S_{Pre}(t)$, where α is the channel opening rate, $T(V_{Pre}) = 1/[1 + \exp(-V_{Pre})]$ represents the dependence of transmitter concentration on presynaptic voltage and τ is the time constant of the synaptic current. To derive physiological values of g_{Sy} we assume a soma radius $R_{Soma} = 7.5 \mu\text{m}$ and use the experimental value $G_{Sy} = 1.7 \text{ nS}$, whence the estimate $g_{Sy} = G_{Sy}/4\pi(R_{Soma})^2 \cong 0.24 \text{ mS/cm}^2$ follows. The electrical synapse generates the current $I_{Sy}(t) = g_{El} (V(t) - V_{Pre})$, where g_{El} is the maximal specific conductance. Now, from $G_{El} = 0.66 \text{ nS}$, we obtain $g_{El} = G_{El}/4\pi(R_{Soma})^2 \cong 0.1 \text{ mS/cm}^2$. In the simulations, values close to the physiological ones were used for both maximal conductances.

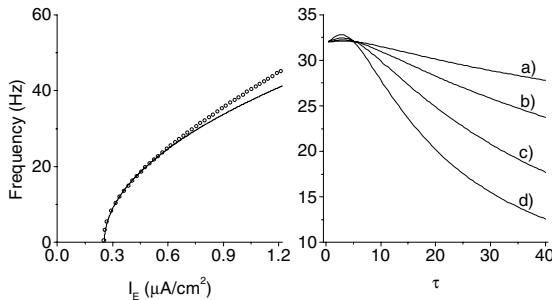


Figure 1. Left panel: discharge frequency of the reduced model obtained by simulations (open circles) and theoretically (solid line). Right panel: discharge frequency of a self-inhibited cell for $I_E = 0.8 \mu\text{A}/\text{cm}^2$ and with a) $g_{Sy} = 0.05 \text{ mS/cm}^2$; b) $g_{Sy} = 0.1 \text{ mS/cm}^2$; c) $g_{Sy} = 0.2 \text{ mS/cm}^2$; d) $g_{Sy} = 0.4 \text{ mS/cm}^2$.

The right panel of figure 1 illustrates the behaviour of a self-inhibited FS cell as the decay constant of the synaptic current is varied and for different values of g_{Sy} . An unexpected result is that for τ values lower than $\sim 5 \text{ msec}$ the GABAergic synapses behave as an excitatory one (i.e. the discharge frequency increases as g_{Sy} increases). We found that this behaviour depends on the value of the reversal potential V_{Rev} of the neurotransmitter: as V_{Rev} gets more negative values the region of τ values where that occurs reduces. It is worth noticing that recently it was shown experimentally that

GABA synapses can have depolarizing effects [11]. In the region $\tau > 5 \text{ msec}$ the FS cell firing frequency decreases as g_{sy} increases. This behaviour is the effect of a longer duration of the repolarization phase arising from the stronger self-inhibition. Moreover, according to the experimental results [1, 2], for fixed values of g_{sy} the firing frequency decreases as τ increases. All the numerical integrations presented in this report were done by using a 4th-order Runge-Kutta method with integration step $\Delta t = 0.002$.

Now let us outline the tools that were used to study analytically, by following [12], the phase locking states of a pair of coupled FS cells. When the inhibitory coupling is symmetric, the time evolution of the phase difference between the two oscillators reads $d\phi/dt = \varepsilon [H_1(-\phi) - H_1(\phi)] = \varepsilon D(\phi)$, where ε is the intensity of the coupling and $H_1(\phi) = T^{-1} \int_0^T Y_1(t) s_0(t+\phi)(V_{rev} - V_0(t))dt$. The function $s_0(t)$ is the solution of the equation $dS_{Pre}(t)/dt = T(V_{Pre})(1 - S_{Pre}(t)) - \tau^{-1} S_{Pre}(t)$ where $V_{Pre}(t) = V_0(t)$ represents the voltage component of the unperturbed T -periodic solution of equation 2. Similarly, for the electrical synapse it is $H_1(\phi) = T^{-1} \int_0^T Y_1(t)(V_0(t+\phi) - V_0(t))dt$. In both cases $Y_1(t)$ is the first component of the solution of the adjoint equation $\dot{\vec{Y}} = -J^T(\vec{X}_0(t))\vec{Y}$, J^T being the transpose of the Jacobian matrix; moreover it is normalized ($\int_0^T \vec{Y}(t) \cdot \dot{\vec{X}}_0(t) dt = 1$). Let ϕ_S be a fixed point of the last equation, then it will be stable (unstable) for $\dot{D}(\phi_S) < 0$ ($\dot{D}(\phi_S) > 0$). When inhibitory and electrical synapses are present $H_1(\phi)$ is the sum of the corresponding terms. It is easy to show that for both electrical and inhibitory synapses $\phi = 0$ and $\phi = T/2$ are always fixed points of $\dot{\phi} = \varepsilon D(\phi)$.

3 Results

The degree of coherence of two spike trains is quantified by the normalized cross-correlation $M = [\sum_j x(j)y(j)] / [\sum_j x(j)\sum_j y(j)]^{1/2}$, the sums being over all the bins, each of 1 msec duration. Moreover, for each spike train, the variables $x(j)$ and $y(j)$ are defined to be 1 or 0 according as a spike is fired or not in the j -th bin. Firstly, the synchronization properties of two identical mutually inhibiting FS cells are investigated numerically. The initial values of membrane voltage are randomly chosen within a neighbourhood of the resting potential of the unperturbed neuron. For each value of I_E the two cells synchronize in a well defined region of the parameter space (τ, g_{sy}) , the so-called synchronization manifold (SM), and a typical case is reported in the left panel of figure 2. The transition to regimes of synchronous discharge (as well as the reverse one) occurs sharply as those parameters are varied. Moreover, in the synchronous state the average frequency of the two interneurons is in the gamma range and, in agreement with the results in experimental results [1,2], it decreases as τ increases (data not shown).

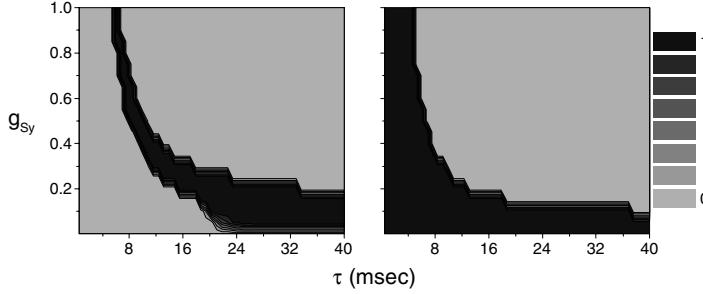


Figure 2. Left panel: contour plot of the values of the synchronization measure M for two mutually inhibiting FS cells with $I_E = 1 \mu\text{A}/\text{cm}^2$ for both cells and $g_{EI} = 0 \text{ mS}/\text{cm}^2$; 51 values for τ and 21 for g_{Sy} were uniformly chosen in the intervals $[1, 40 \text{ msec}]$ and $[0.001, 1 \text{ mS}/\text{cm}^2]$, respectively. Right Panel: as for the left panel, but with $g_{EI} = 0.08 \text{ mS}/\text{cm}^2$

Next, adding the electrical coupling to the pair of mutually inhibitory FS interneurons yields the result reported in the right panel of figure 2. So, the main effect of the electrical coupling is to widen the SM in the (τ, g_{Sy}) space in agreement with the results obtained from the complete model [7]. Similar results were found for other values of I_E (data not shown).

Now let us discuss the theoretical results, obtained in the weak coupling limit, for a pair of coupled FS interneurons. The stationary values of the phase difference between mutually inhibiting FS cells are reported in the left panel of figure 3 vs. τ and with $g_{Sy} = 0$. For τ values lower than $\tau_l = 5.68 \text{ msec}$ there are only stable antiphase states (corresponding to $M = 0$), while for $\tau > \tau_l$ there is a region of τ values characterized by a bistable regime (synchronous and antisynchronous).

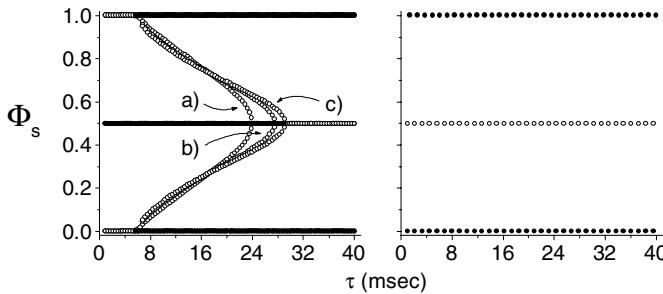


Figure 3. Stationary values of the phase difference against the time constant of the inhibitory synapses. Left panel: $g_{EI} = 0$ and a) $I_E = 0.5 \mu\text{A}/\text{cm}^2$, b) $I_E = 0.8 \mu\text{A}/\text{cm}^2$, c) $I_E = 1 \mu\text{A}/\text{cm}^2$. Right panel: $g_{EI} / g_{Sy} = 0.4$ and $I_E = 0.8 \mu\text{A}/\text{cm}^2$. For both panels the open (solid) circles denote unstable (stable) states.

For a fixed I_E let $\tau^*(I_E)$ be the highest τ value where bistability occurs. We can see from figure 3 that for $\tau_l < \tau < \tau^*(I_E)$ there are unstable phase locking states that

bifurcate from $\tau^*(I_E)$ (through a subcritical pitchfork bifurcation). These unstable states are the separatrices between inphase and antiphase regimes, and their position can be used to estimate the probability of occurrence of the regime of complete synchrony between the two cells. For instance, for τ values just to the right of τ_l , the unstable branches are closer to the synchronous ones and this means that for arbitrary initial conditions the probability to get an antisynchronous regime is about unity. Let us remark that $\tau^*(I_E)$ is a monotonically increasing function of I_E . For $\tau > \tau^*(I_E)$ the remaining stable states are only the synchronous ones. The results obtained in the presence of both types of synapse are reported in the right panel of figure 3 for $g_{EI} / g_{Sy} = 0.4$ (near to the ratio between the corresponding experimental values). These data shows that the presence of electrical coupling in a pair of inhibitorily coupled FS cells promotes synchrony. Regions of τ values where bistability occurs can be found for values of g_{EI} / g_{Sy} lower than 0.4 (for instance when it is $g_{EI} / g_{Sy} = 0.05$). As the value of g_{EI} / g_{Sy} increases, starting from 0, there is a systematic reduction of the size of the bistability region (data not shown). The comparison of these theoretical results with the numerical ones in figure 2 shows that they are in good agreement.

Next we study the dependence of the phase locking regimes on the stimulation current in a pair of cells coupled by electrical and chemical synapses. The corresponding results, for a fixed value of the ratio g_{EI} / g_{Sy} , are shown in the left panel of figure 4. There is a critical value I^* of the stimulation current from which two unstable phase locking states originate through a subcritical pitchfork bifurcation, and these results are qualitatively similar to those reported in the left panel of figure 3. As the intensity of the electrical coupling increases, the position of the bifurcation point moves to the left, until it disappears completely. For $I_E < I^*$ there is a bistable behaviour where stable antiphase and inphase states coexist, while for $I_E > I^*$ there is only stable synchrony. These findings show that synchrony is promoted when the stimulation current increases, in agreement with the numerical simulations (data not shown).

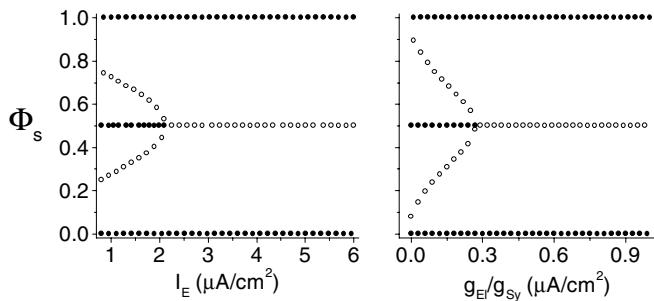


Figure 4. Left panel: stationary values of phase difference against I_E for $g_{EI} / g_{Sy} = 0.1$. Right panel: stationary values of phase difference against g_{EI} / g_{Sy} for $I_E = 0.8 \mu\text{A}/\text{cm}^2$. For both panels it is $\tau = 8 \text{ msec}$ and the open (solid) circles denote unstable (stable) states.

Finally, the right panel of figure 4 shows that, for a fixed value of g_{Sy} , synchrony is promoted as g_{EI} increases.

The last issue addressed concerns the effect of the electrical coupling on the synchronization properties of an heterogeneous network of larger size ($N=50$). The investigation is done numerically, as the theoretical approach should now become very hard. It is assumed that the coupling is all to all and the heterogeneities are introduced by using different stimulation currents: $I_E(j) = I_0 + \xi(j)$, ξ being a gaussian random variable of zero mean and standard deviation σ . The inhibitory and electrical synaptic inputs to the j -th cell are $I_{Sy}(j) = g_{Sy} (N-I)^{-1} \sum_{k \neq j} S_{Pre}(k)(V_j - V_{Rev})$ and $I_{El}(j) = g_{El} (N-I)^{-1} \sum_{k \neq j} (V_j - V_{Rev})$, respectively. The corresponding results for $g_{Sy} = 0.2 \text{ mS/cm}^2$ and $I_0 = 1 \mu\text{A/cm}^2$ are reported in figure 5. For $g_{El} = 0$ and $\sigma = 0$ a complete synchronous state is reached (panel a), while the introduction of heterogeneities ($\sigma = 0.1$) breaks the synchrony (panel b). When the electrical coupling is set on, $g_{El} = 0.05 \text{ mS/cm}^2$, no substantial differences with the case $g_{El} = 0$ and $\sigma = 0.1$ are observed (panel c). As the value of the electrical coupling increases, $g_{El} = 0.15 \text{ mS/cm}^2$, a partial synchrony is restored (panel d). The effect of the electrical coupling to raise the coherence level becomes relevant as g_{El} overcomes some threshold; similar findings were obtained for networks of different size (data not shown).

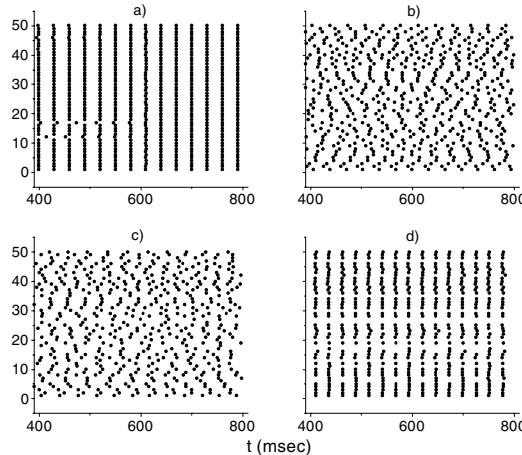


Figure 5. – Synchronization of an heterogeneous population ($N=50$) of FS cells interacting via inhibitory and electrical synapses: a) $g_{El} = 0$ and $\sigma = 0$; b) $g_{El} = 0$ and $\sigma = 0.1$; c) $g_{El} = 0.05 \text{ mS/cm}^2$ and $\sigma = 0.1$; d) $g_{El} = 0.15 \text{ mS/cm}^2$ and $\sigma = 0.1$. For all panel is $g_{Sy} = 0.2 \text{ mS/cm}^2$ and $I_0 = 1 \mu\text{A/cm}^2$.

4 Conclusions

Firstly we showed that a two variables biophysical model for an FS cell mimics accurately all the basic properties of a four variable one used elsewhere [7]. Then the experimental findings concerning the coupling between FS cells were used to model the electrical and chemical coupling among interneurons. With these tools at our disposal we considered two identical mutually inhibiting model cells and showed that

their discharges synchronize in a well defined region of the parameters space (τ , g_{sy}). The introduction of the electrical coupling yields a widening of this region.

In the weak coupling limit the theoretical results agree with the numerical ones. When $g_{EI} = 0$ there are regions of τ values where bistable dynamical regimes occur (antiphase and inphase); moreover, the increase of τ favours the emergence of synchrony. For $g_{EI} \neq 0$ and $g_{sy} \neq 0$ we found that increasing either the stimulation current or the intensity of the electrical coupling leads to the emergence of synchronous regimes between FS cells.

The investigation of larger network of interacting cells in presence of heterogeneities was done numerically showing that: a) the electrical coupling prevents the decoherence effects due to heterogeneity; b) there is a threshold - like mechanism leading to the emergence of synchrony when the electrical coupling intensity is increased.

References

1. C. J. McBain, A. Fisahn, "Interneurons unbound", *Nature Rev. Neurosci.*, vol. 2, pp. 11-23, 2001.
2. M. A. Whittington, R. D. Traub, J. G. R. Jefferys, "Synchronised oscillations in interneuron networks driven by metabotropic glutamate receptor activation", *Nature*, vol. 373, pp. 612-615, 1995.
3. X. J. Wang, J. Rinzel, "Alternating and synchronous rhythms in reciprocally inhibitory model neurons", *Neural Comput.*, vol. 4, pp. 84-97, 1992.
4. X. Wang and G. Buzsaki, "Gamma oscillations by synaptic inhibition in an interneuronal network model", *J. Neurosci.*, vol. 16, 6402-6413, 1996.
5. C. van Vreeswijk, L. F. Abbott, B. Ermentrout, "When inhibition not excitation synchronizes neural firing", *J. Comput. Neurosci.*, vol. 1, 313-321, 1994.
6. M. A. Whittington, R. D. Traub, N. Kopell, B. Ermentrout, E. H. Buhl, "Inhibition-based rhythms: experimental and mathematical observations on network dynamics", *International J. of Psychophysio.*, vol. 38, 315-336, 2001.
7. A. Di Garbo, M. Barbi, S. Chillemi, "Synchronization in a network of fast-spiking interneurons", *BioSystems*, vol. 67, 45-53, 2002.
8. M. Galarreta and S. Hestrin, "A network of fast-spiking cells in the cortex connected by electrical synapses", *Nature*, vol. 402, pp. 72-75, 1999.
9. J. R. Gibson, M. Belerlein, B. W. Connors, "Two networks of electrically coupled inhibitory neurons in neocortex", *Nature*, vol. 402, pp. 75-79, 1999.
10. D. Durstewitz, J. K. Seamans, T. J. Sejnowski, "Dopamine-mediated stabilization of delay-period activity in a network model of prefrontal cortex", *J. Neurophysiol.* vol. 83, pp. 1733-1750, 2000.
11. R. Kohling, "GABA becomes exciting", *Science*, vol. 298, pp. 1350-1351, 2002.
12. B. Ermentrout, "Neural networks as spatio-temporal pattern-forming systems", *Rep. Prog. Phys.*, vol. 61, 353-430, 1998.

Stochastic Networks with Subthreshold Oscillations and Spiking Activity

Nazareth P. Castellanos, Francisco B. Rodríguez, Pablo Varona

Grupo de Neurocomputación Biológica (GNB).

Dpto. de Ingeniería Informática, Escuela Politécnica Superior,

Universidad Autónoma de Madrid,

Ctra. Colmenar Viejo, Km. 15, 28049 Madrid, Spain.

{nazareth.perales, francisco.rodriguez, pablo.varona}@ii.uam.es

Abstract. Subthreshold oscillations actively participate in information coding and processing of living neural systems. In this paper we present a new model of stochastic neural networks in which the neurons display subthreshold oscillations and spiking activity. The network is built with diffusive coupling among close neighbors. We show that these stochastic networks are able to generate a wide variety of coherent spatio-temporal patterns. Emerging phenomena in this networks can be of great interest in applications related to the coding and control of rhythms.

1 Introduction

Several neural systems of living animals exhibit subthreshold oscillations in addition to spiking activity. One example is the ampullae of Lorenzini of the dogfish where the subthreshold oscillations determine the basic rhythm of the impulse generation [1]. Another example is the inferior olive (IO) of mammals which is composed of networks of electrically coupled neurons. The IO cells generate subthreshold oscillations and spiking activity [2]. *In vitro* experimental recordings using slices of IO neurons have shown the presence of characteristic spatio-temporal patterns of activity [3, 4]. The cerebellum and the IO have been studied extensively, but the role of this system remains unclear. Several hypothesis relate the IO to the control and learning of motor rhythm coordination [5, 6, 2, 7]. Computer models of the IO can provide useful information to disclose its function. In [8] the authors studied the characteristic spatio-temporal patterns generated by a network of Hodgkin-Huxley type neurons showing that these patterns of network activity can easily encode several coexisting rhythms.

In this paper we propose a stochastic model to study the generality of emerging properties of systems with electrically coupled neurons displaying subthreshold oscillations and spiking activity. Several interesting phenomena related to this particular behavior have been observed in living neural systems and in conductance-based models. The stochastic model can reproduce qualitatively these phenomena but considerably reducing the time of computation needed to implement large networks with more realistic approaches [9, 8]. The model can

be useful not only to test hypothesis about the role of the IO and other neural systems, but also to investigate the computational abilities of such networks. The activity of the isolated neuron is implemented using a random walk with absorbent barriers [10, 11]. The interaction between single units is introduced through a diffusive coupling among close neighbors. We show that networks of stochastic neurons with subthreshold oscillations can exhibit coherent spatio-temporal patterns which are similar to those obtained with detailed conductance-based models [8].

2 The Stochastic Model

We have built a stochastic neural model with subthreshold oscillations and spiking activity. The spontaneous evolution of the activity of an isolated neuron follows a random walk. The neuron activity is considered as a discrete variable and characterized in time by $a(t)$. The stochastic dynamics of a single unit i is given by:

$$a_i(t+1) = \begin{cases} a_i(t) + C & \text{with probability } p \\ a_i(t) & \text{otherwise,} \end{cases} \quad (1)$$

where p is the transit probability of its internal state per time step. Therefore $1 - p$ is the probability of remaining in the current state. C is a parameter that depends on the temporal evolution of the unit activity. This parameter can have three different values. The neuron starts to increase its activity from an initial state, with probability p , using $C = 1$. If the neuron reaches its activation threshold, L , it produces a spike according to a firing probability p_f . The activity is incremented in $3L$ in order to model the spike event. After the spike, the neuron activity begins to decrease to the initial state following equation 1 with $C = -L$ until it reaches the lower activity. When the unit reaches the threshold and it does not fire, with probability $1 - p_f$, its activation begins to decrease again, but now $C = -L/5$. A schematic representation of this model is shown in Figure 1. The evolution of an isolated neuron is shown in Figure 2 for different values of p_f . For all simulations, the initial activity of each neuron is chosen randomly in the $[1, L]$ interval.

To construct the networks we use difussive coupling among neighbor units. We build bidimensional networks with periodic boundary conditions to avoid border effects. The neurons are connected emulating electrical coupling, and the interchange rule between unit i and its neighbors j is defined by:

$$a_i(t) = a_i(t) + g \sum_{j=\text{neighbors}} [a_j(t-1) - a_i(t-1)], \quad (2)$$

where g is the electrical coupling conductance and $a_j(t)$ is the activity of its neighbors. The value of the conductance g is the same in every connection. The electrical coupling induces a local synchronization in the network as a function of g .

The evolution of each neuron activity in the network is given by two contributions: the spontaneous random walk described by equation 1 and the

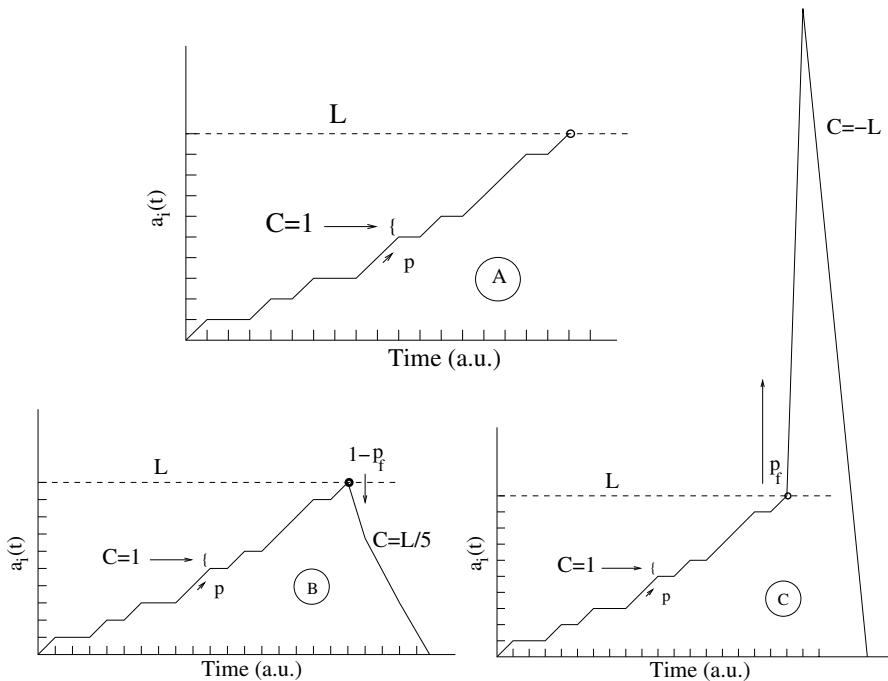


Fig. 1. Schematic representation of the stochastic single neuron model. (A) Spontaneous evolution of the subthreshold activity as determined by the parameters of the model. (B) Subthreshold oscillation after failure to produce a spike. (C) Emission of a spike after reaching the threshold.

interaction among neighbor neurons (diffusive electrical coupling described by equation 2).

3 Results

We have studied the generation of coherent spatio-temporal patterns of sub-threshold and spiking activity in bidimensional networks of 50×50 identical stochastic neurons. In all networks described in this paper, each neuron was connected to its four nearest neighbors through the diffusive coupling described in the previous section. The appearance of spatio-temporal patterns of activity in these networks strongly depends on g , the value of the electrical coupling conductance.

We first tested the model in two extreme cases. For very small coupling, $g = 0.001$, the neurons behave similarly to the non-coupled case (see time series at the top of Figure 3 and the temporal sequence at the top of Figure 4). In Figure 4 dark dots correspond to neurons with subthreshold activity, while white dots mean spiking activity. Neurons with the same color have the same value of

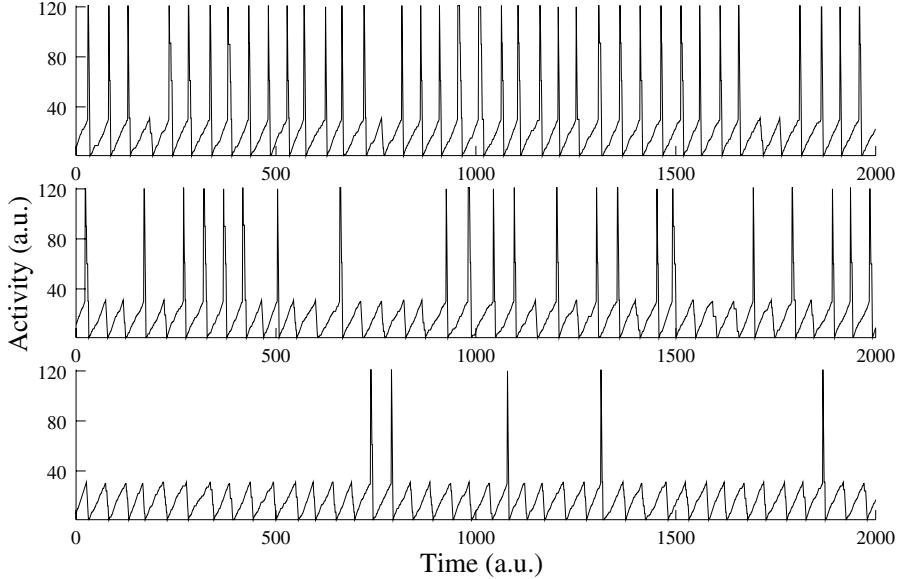


Fig. 2. Activity of an isolated neuron for three different values of p_f . From top to bottom: $p_f = 0.9, 0.5, 0.1$ ($p=0.7$ and $L=30$ in all simulations).

the activity. Neighbor neurons are not synchronized due to the different initial activities and the stochastic nature of their evolution including the spike generation. Conversely, for strong coupling $g = 0.17$ the neuronal activity is nearly synchronized as we can observe in the third row of Figure 3 and Figure 4. Coherent spatio-temporal patterns with well-defined wave fronts can not be found in the activity of the network for these two extreme values of the coupling.

The spatio-temporal patterns emerge for a moderate value of the electrical conductance. When a neuron fires, it increases the probability of spiking in its neighbors. Close neighbors tend to fire with a small phase shift. This generates a propagating wave front in the network that creates the pattern. An example of intermediate electrical coupling, $g = 0.06$, is shown in the second row of Figures 3 and 4. Note that the spike width depends on the value of the electrical coupling. An example can be seen in Figure 3 by comparing the activity for $g = 0.001$ and $g = 0.17$. We must recall that the stochastic model makes $C = -L$ (see equation 1) to simulate the activity decay after firing. When the electrical coupling is high, the activity of neighbor units accelerate the descent, and as a consequence, there is an appreciable spike focusing phenomenon in the network.

We are interested in the study of the ability of these networks to encode different rhythms that can coexist in the network. Thus, we introduced stimuli in two separate clusters within the 2D network. Each cluster is formed by a matrix of 6×6 neighbor neurons. We show the exact position of the stimulus clusters $S1$ and $S2$ in the network at the right panels of Figure 3 together with

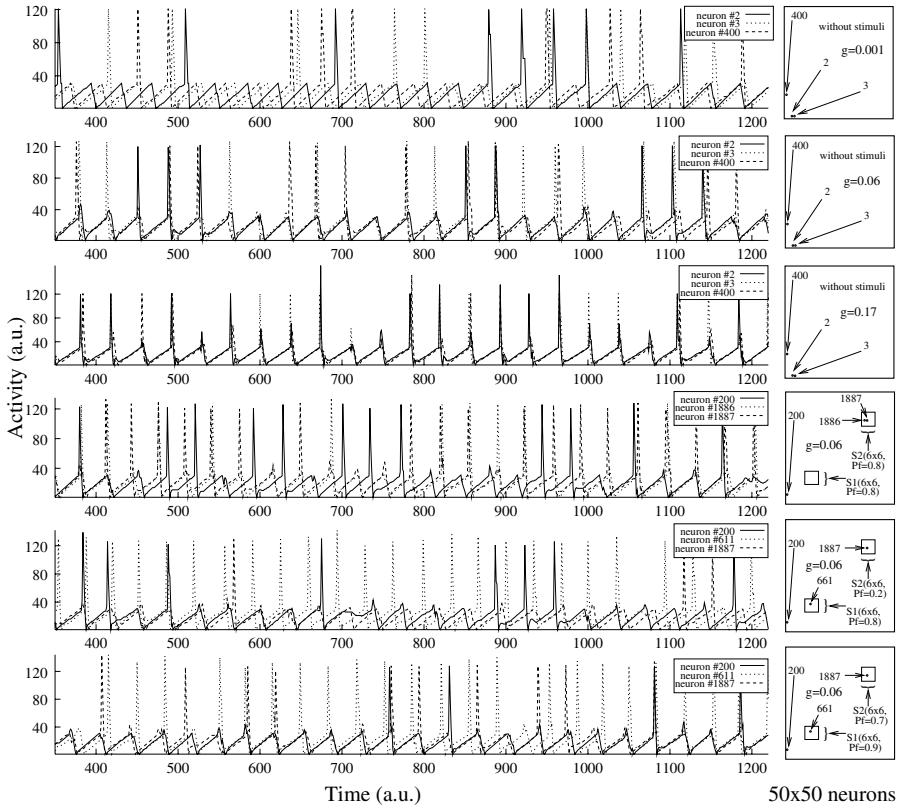


Fig. 3. Subthreshold oscillation and spiking activity for three neurons in a population of 50×50 units. In all simulations $L = 30$, $p = 0.9$ and $p_f = 0.4$. The first three rows show time series for three different values of the conductance $g = 0.001$, 0.06 , 0.17 . The last three rows show time series in networks with two 6×6 stimulus clusters, $S1$ and $S2$, in which the neurons had a different p_f than the value used in the rest of the neurons in the ensemble. The firing probability of the neurons not belonging to the stimulus cluster was $p_f = 0.4$. In 4th plot the firing probability for stimulus clusters $S1$ and $S2$ is the same, $p_f = 0.8$. In 5th plot the firing probability for stimulus clusters $S1$ and $S2$ are respectively $p_f = 0.8$, 0.2 . Lastly, in 6th plot the firing probability for stimulus clusters are $p_f = 0.9$ for $S1$ and $p_f = 0.7$ for $S2$.

the unit number in which we measure the activity (see 4th, 5th and 6th time series in Figure 3).

Fourth row in Figure 3 and 4 correspond to a network in which the neurons that belong to the stimulus clusters had a high spiking probability, $p_f = 0.8$. In this case, the propagating wave always emerged from the stimulus clusters (see 4th temporal sequence in Figure 4). When one cluster had a low spiking probability, $p_f = 0.2$, and the other high probability, $p_f = 0.8$, the low frequency cluster acted as a sink for the activity propagated from the other one (see fifth

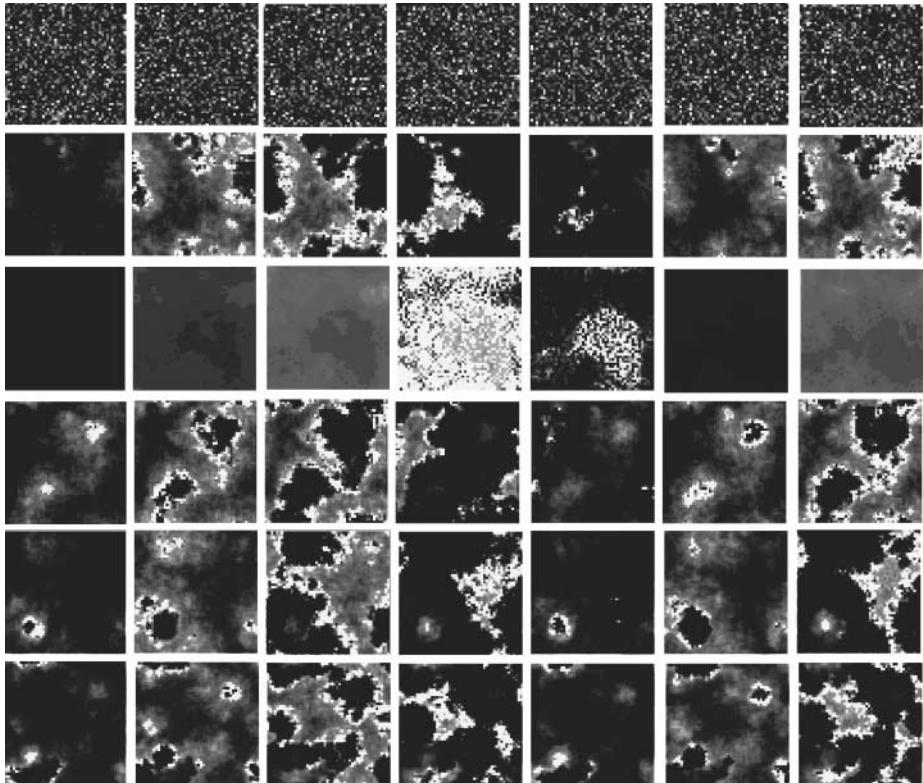


Fig. 4. Spatio-temporal patterns displayed by the networks of 50×50 neurons described in figure 2. Time sequences go from left to right. Neurons with the same color are synchronized. The first three rows correspond to networks without stimuli. In the last three rows two clusters of stimuli are introduced in the network. See detailed explanation in the text.

temporal sequence in Figure 4, and fifth plot in Figure 3). Lastly, when both stimulus clusters had a high spiking probability but different p_f (0.7 and 0.9, respectively), the propagating wave emerged from the clusters in different temporal phases. In the last two cases, there was coexistence of regions with different spiking frequencies in the network.

4 Discussion

The dynamics of spiking neurons synaptically coupled in a bidimensional array has been extensively studied [12–16]. We propose a new stochastic neural model with the ability to generate subthreshold oscillations and spiking activity. In two-dimensional networks with electrical connections, the subthreshold activity is crucial to establish an overall synchronization of under-threshold activity

while keeping a high degree of independence in the spiking behavior of the population. This allows the coding of different rhythms induced by stimuli in the spiking activity under the modulation of a nearly homogeneous subthreshold oscillations. This simple model reproduces the essential phenomena found in more complex realistic networks of inferior olive neurons, showing the generality of the phenomena and also the ability of the model to be used as a tool to study hypothesis about the role of this neural system. The model does not include a refractory period in the neurons after the occurrence of a spike. In consequence, the frontiers of the spatio-temporal patterns are not as sharp and well defined as those observed with more realistic approaches. This aspect of the model will be a subject for future work.

While often neglected in the analysis of living systems and the design of artificial neural networks, subthreshold oscillations can actively participate in the coding and processing of information. In living neural systems such as the IO, the properties given by the diffusive coupling and the subthreshold oscillations could be used to encode and control different rhythms of motor behavior.

5 Acknowledgments

This work has been supported partially by CICYT TIC98-0247-C02-02, TIC2001-0572-C02-02 grants , and MCT BFI2000-0157 grant.

References

1. Braun H.A., Wissing H., Schäfer k. and Hirsch, M.C. 1994. Oscillation and noise determine signal transduction in shark multimodal sensory cells. *Nature* **367**, 270-273.
2. De Zeeuw C.I., Simpson J.I., Hoogenraad C.C., Galjart N., Koekkoek S.K.E., and Ruigrok T.J.H. 1998. Microcircuitry and function of the inferior olive. *Trends Neurosci* **21**: 391-400.
3. Leznik E., Makarenko V. and Llinas R. 2002. Electrottonically Mediated Oscillatory Patterns in Neuronal Ensembles: An In Vitro Voltage-Dependent Dye-Imaging Study in the Inferior Olive. *Journal of Neuroscience* **22**(7):2804-2815.
4. P. Varona, J.J. Torres, H.D.I. Abarbanel,V.I. Makarenko R. Llinás and M.I. Rabinovich. 1999. Modeling collective oscillations in the inferior olive. *Soc. for Neurosci. Abs.* **25**, (368.8).
5. Llinás R. and Welsh J.P. 1993. On the cerebellum and motor learning. *Curr. Opin. Neurobiol.* **3**, 958.
6. Welsh J.P., Lang E.J., Sugihara I., Llinás R. 1995. Dynamic organization of motor control within the olivocerebellar system. *Nature*. **374**, 453-456.
7. Ito M. 1982. Cerebellar control of the vestibulo-ocular reflex-around the flocculus hypothesis. *Annual Review of Neuroscience* **5**: 275-96.
8. Varona P., Aguirre C., Torres J.J., Rabinovich M.I., Abarbanel H.D.I.. 2002. Spatio-temporal patterns of network activity in the inferior olive. *Neurocomputing* **44-46**: 685-690.
9. Schweighofer N. , Doya K., Kawato M.. 1999. Electrophysiological properties of inferior olive neurons: A compartmental model. *J. Neurophysiol.* **82**(2): 804-817.

10. Gerstein G.L. and Mandelbrot B. 1964. Random Walk Models for the Spike Activity of Single Neuron. *Biophys. J.* **4**, 41–68.
11. Rodríguez F.B., Suárez A. and López V. 2001 Period Focusing Induced By Network Feedback in Populations of Noisy Integrate-and-Fire Neurons. *Neural Computation* **13**, 2495–2516.
12. Usher M., Stemmler M. 1995. Dynamic Pattern Formation Leads to 1/f Noise in Neural Populations. *Physical Review Letters*, **74**(2): 326–329.
13. Fohlmeister, C, Gerstner W, Ritz, R. van Hemmen J.L. 1995. Spontaneous Excitations in the Visual Cortex: Stripes, Spirals, Rings, and Collective Bursts. *Neural Computation* **7**, 905–914.
14. Lin J.K., Pawelzik K., Ernst U., and Sejnowski T.J. 1998. Irregular synchronous activity in stochastically-coupled networks of integrate-and-fire neurons. *Network* **9**(3), 333–344.
15. Rabinovich M.I, Torres J.J., Varona P., Huerta R., Weidman P. 1999. Origin of Coherent Structures in a Discrete Chaotic Medium. *Physical Review E*, **60**(2), R1130-R1133.
16. Rabinovich M., Ezersky A.B., Weidman P., *The Dynamics of Patterns*. World Scientific, Singapore, 2000.

Selective Inactivation of Neuronal Dendritic Domains: Computational Approach to Steady Potential Gradients

I. Makarova^{1,2}, J.M.Ibarz¹, L. López-Aguado¹ and O. Herreras¹

¹Dept. Investigación, Hospital Ramón y Cajal, 28034-Madrid, Spain

²Dept. Matemática Aplicada, Fac. Biol., Univ. Complutense, 28040-Madrid, Spain

¹{jose.m.ibarz, laura.lopez, oscar.herreras}@hrc.es

²julia@fluidos.pluri.ucm.es

Abstract. The complex and heterogeneous morphology of neurons ensures that different domains are not isopotential, enabling the use of local voltage transients to code information. During certain pathophysiological events such as anoxia or spreading depression (SD) waves, neurons lose electrogenesis and membrane potential collapse, presumably becoming isopotential. However, in some cases inactivation is not complete, and some dendritic domains may still develop normal electrogenesis. Using a compartmental pyramidal neuron, we simulated the electrical status of the membrane during SD waves and examined the voltage changes along the neuron morphology. Strong axial gradients of potential arise that “isolate” the seized dendritic domains while leaving near regions capable to develop normal electrical activity.

1 Introduction

Some neuron models in neural network simulations use single compartments to integrate the incoming information. Real neurons are complex morphological structures with heterogeneous geometry and variable electrogenic machinery in different subdomains. These two features ensure that neurons will never be isopotential. As a result, real neurons use (1) local voltage transients to code incoming information, and (2) voltage propagation along cable-like structures as an efficient mechanism for signal modulation and integration. When a neuron die, its membrane potential (V_m) dissipates, becomes isopotential and becomes unable to generate currents. A similar membrane breakdown is thought to occur during anoxia and spreading depression (SD) waves of Leao, since neurons lose completely their electrogenic capabilities and transmembrane ion gradients are known to collapse [1]. However, some well-known concomitants are hard to reconcile with a complete membrane breakdown, such as the characteristic extracellular negative DC voltage associated to the intracellular massive depolarization [2]. Some authors had proposed that this negative DC could be generated by cellular elements other than neurons (e.g. glia), but our team refuted this possibility [3,4]. Thus, neurons are left as the only possible generators of extracellular voltages in a moment in which their intracellular compartment is presumed isopotential at 0 mV [5,6]. In order to find a biophysical explanation to account for this apparent disagreement we have used a compartmental model of the CA1 pyramidal cell to simulate the electrical membrane situation during

SD waves, and intra and extracellular DC recordings and evoked field potentials during experimental SD waves.

2 CA1 Pyramidal Cell Model of Spreading Depression Waves

The compartmental model reproduced in detail the average pyramidal cell morphology [7-9]. This was simulated using 265 compartments, distributed in soma, apical and basal dendritic trees and axon (a 2-D projection of the model neuron is shown in Fig. 1). Compartment length was always >0.01 and $<0.2 \lambda$. A detailed description of the cell morphology can be found in the <http://navier.ucsd.edu/ca1ps> address. Total effective area of the neuron was $66,800 \mu\text{m}^2$ (including spine area). The electrotonic parameters for soma and dendrites were $R_m=70,000 \Omega \text{ cm}^2$, $R_i=75 \Omega \text{ cm}$ and $C_m=0.75 \mu\text{F/cm}^2$, while R_m was 100 and $0.5 \cdot 10^6 \Omega \text{ cm}^2$ for non-myelinated and myelinated axon compartments, respectively. The later had a C_m of $0.04 \mu\text{F/cm}^2$. The input resistance measured at the soma was $140 \text{ M}\Omega$, and τ was 25 ms.

The ion channels, their kinetics and distribution along the morphology of the model neuron are as described in Ibarz and Herreras [9]. Conductance variables were described with Hodgkin-Huxley type formalism. Equilibrium potentials were +45 and -85 mV for Na^+ and K^+ , respectively. The details can be obtained from <http://navier.ucsd.edu/ca1ps>. Spreading depression was simulated by introducing a leak current of 700 S/m^2 maximum conductance in certain domains ($E_{\text{Leak}} = +10 \text{ mV}$).

3 Experimental recordings

Anesthetized female rats were used. Hippocampal tissue slices were obtained following standard protocols. Glass micropipettes filled with the appropriate electrolytes were used for intra and extracellular recordings (DC and evoked potentials), and tungsten monopolar electrodes used for electrical stimulation of afferent and efferent fibers. SD waves were elicited by ejection of KCl microdrops into either the stratum radiatum (apical dendritic tree) or the stratum oriens (basal tree).

4 Totally inactivated and normal dendritic subdomains coexist.

Normally, SD waves sweep across the CA1 layer and seize both basal and apical dendritic trees almost simultaneously [3]. This standard SD wave causes a large negative DC potential all along the dorso-ventral extension of the CA1 pyramidal cells, spanning the basal and apical dendritic trees and the soma layer, and all evoked activity is abolished, suggesting total inactivation of the whole CA1 pyramidal cell. However, when the SD is forced to initiate in the basal tree, it usually runs well ahead of that in the apical tree that lags many seconds behind (Fig.1, arrows *b*). The arrival of the SD to the basal tree (arrow *a*) can be seen as a partial depolarization in the

intracellular somatic electrode (e_1), while no potential is still recorded in the extracellular field electrode (e_2). Before the arrival of the SD to the recording electrodes, the activation of the synapses causes an extracellular negative potential (field excitatory postsynaptic potential, fEPSP) in the apical tree, and fires an action potential (AP) as recorded intracellularly (traces labeled 1). Surprisingly, after the SD wave has seized the basal but not yet the apical tree (period between arrows a and b) the evoked fEPSP showed a waveform and amplitude similar to control, but the somata did not longer fire APs (traces labeled 2). This result indicates that the basal and apical trees can be inactivated independently from each other, and that the non-seized dendritic tree must maintain a V_m close enough to resting as to develop normal synaptic responses.

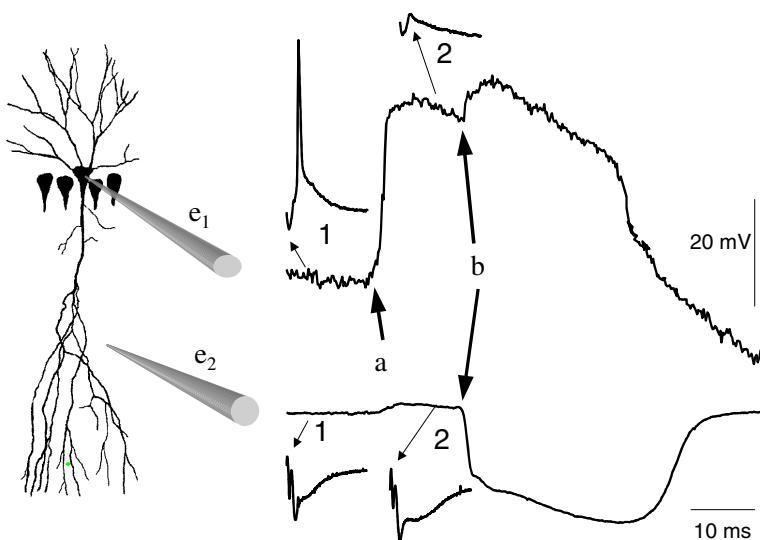


Fig. 1. Experimental recording of an SD wave that runs in the basal tree seconds ahead of the apical dendrites.

5 Model results for the intracellular membrane potential during SD

In order to know what the V_m spatial distribution could be within one dendritic tree while only the other is undergoing SD, we first set SD conditions in the main apical shaft, sparing lateral smaller dendrites and the soma. After a few tens milliseconds, the V_m reached a steady state with different values at different loci (see Fig. 2), creating stationary potential intracellular gradients. The highest depolarized values

were in the apical shaft, as expected. However, the distal non-seized apical dendrites (at11a) and the basal dendritic tree (bp7, bx9b) were only slightly depolarized. The soma and adjacent compartments developed a repetitive spikelet activity that corresponded to full APs generated far in the axon.

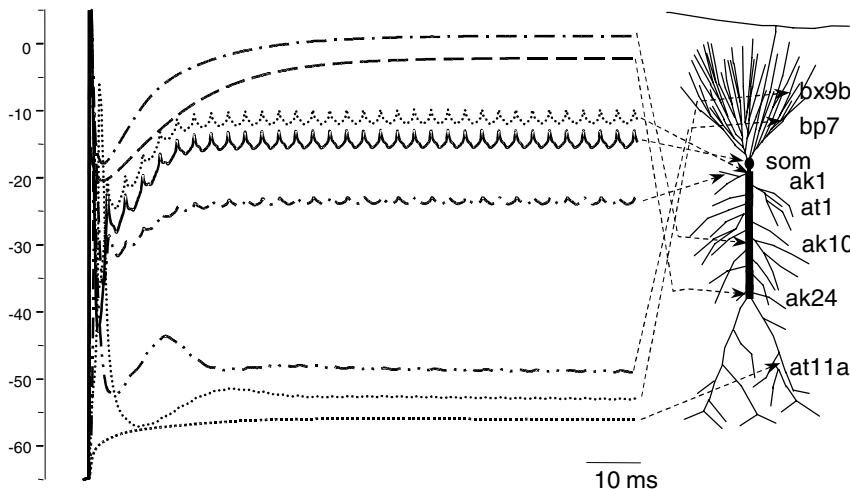


Fig. 2. Steady state V_m reaches different values during simulated apical SD, barely affecting distal apical dendrites and the basal tree.

The axial gradients of potential are better observed in the spatial plots for the V_m at steady state shown in Fig. 3 along a continuous cable from the basal to the distal-most dendrite. The experiment of Fig. 2 (apical-only SD) is represented by black dots. It can be appreciated how the seized cable (black bar) is not isopotential, the V_m decreasing at both ends, more markedly in the somatic side. This drop occurs because the larger amount of current drained by the larger soma/basal tree surface than the distal apical dendrites. The soma, which is electrically contiguous to a SD-seized compartment (the first apical compartment, ak1), gets a V_m about 18 mV lower than the core of the seized region (ak10), and about 30 mV below the E_{leak} set for the SD.

The contribution of the basal tree is shown in the superimposed empty-circle plot. In this case, both the apical shaft and a wide strip in the middle of the basal tree were seized by SD. It is evident that the basal depolarization barely affects the apical tree. Most relevant is that the soma itself is only 2 mV more depolarized than when the apical shaft is seized alone. In the basal tree, a strong gradient of potential also developed.

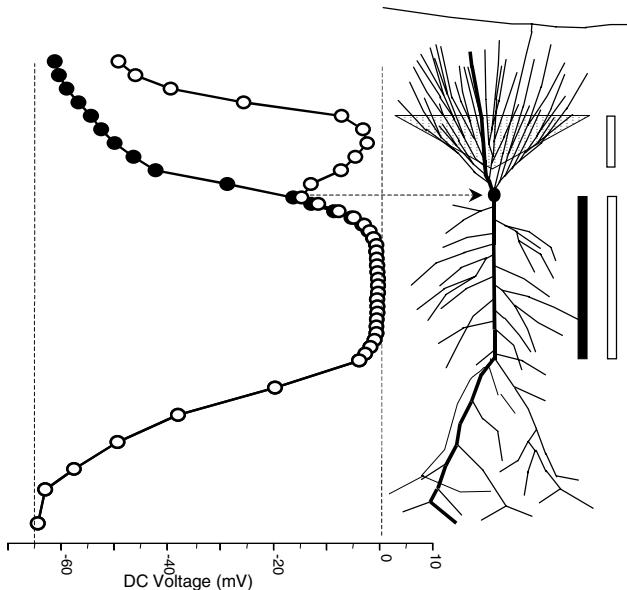


Fig. 3 Simulated steady state profile along the pyramidal morphology during only apical (black circles) or combined apical and basal (empty circles) SD. Vertical bars on the right indicate the seized regions. Thick dendrites indicate the compartments plotted on the left.

6 Discussion

The present results indicate that one dendritic tree may lose completely its membrane potential while the other maintains V_m levels close to rest as to maintain normal electrogenesis. Both, the experimental and model results agree on this view. When the basal tree is inactivated, the apical one may still develop normal synaptic currents. However, these will not be effective because the soma is too depolarized and, in practice, its electrogenic machinery has also been inactivated. Although the model indicates that the axon is still firing APs in a bursting manner, the dendritic information will hardly have an impact on cell output.

The physiological relevance of the present results is that during simulated SD cells need not to be totally inactivated. Inactivation of discrete regions of the cell morphology may account for the experimental results and also provide an explanation for the extracellular potentials associated to SD. If cells are totally inactivated or isopotential along their entire morphology, no current would be established between any two points of the cell, hence there is no flow of transmembrane current nor extracellular DC potential [7]. However, partial inactivation allows strong gradients and set axial currents from the core of the seized to adjacent regions [10]. The circuit

completes with transmembrane outward currents in the later regions and across the extracellular space, back to the depolarized region where currents are inwards.

The present model requires refinement. For instance, we have not used the extracellular DC potential to calculate V_m . Also, we have not include the well known change of ion gradients across the membrane [1] that may affect both, the equilibrium potential for similar conductances at different compartments, and also give rise to a flow of axial current because of the massive intra- and extracellular ion migration. Still, the present results, though preliminary, are promising and encourage the use of computer models to analyze some physiological situations when electrodes face major practical difficulties.

References

- 1 Hansen AJ, and Zeuthen T. Extracellular ion concentrations during spreading depression and ischemia in the rat brain cortex. *Acta Physiol Scand* (1981) 113:437-445.
- 2 Herreras, O. and Somjen, G.G. Analysis of potentials shifts associated with recurrent spreading depression and prolonged unstable SD induced by microdialysis of elevated K⁺ in hippocampus of anesthetized rats. *Brain Res.*, (1993) 610:283-294.
- 3 Largo C, Ibarz, JM, Herreras O. Effects of the gliotoxin fluorocitrate on spreading depression and glial membrane potential in rat brain in situ. *J. Neurophysiol.* (1997) 78:295-307.
- 4 Largo, C., Cuevas, P., Somjen, G.G., Martín del Río, R. and Herreras, O. The effect of depressing glial function on rat brain in situ on ion homeostasis, synaptic transmission and neuronal survival. *J. Neuroscience*, (1996) 16:1219-1229.
- 5 Herreras, O., Largo, C., Ibarz, J.M., Somjen, G.G. and Martín del Río, R. Role of neuronal synchronizing mechanisms in the propagation of spreading depression in the in vivo hippocampus. *J. Neuroscience*, (1994) 14:7087-7098.
- 6 Herreras O and Largo C. Tracking along the path toward ischemic neuronal death. *Rev Neurol* (2002), 35:838-45.
- 7 Varona P, Ibarz JM, López-Aguado L and Herreras O. Macroscopic and subcellular factors shaping CA1 population spikes. *J. Neurophysiol.* (2000) 83:2192-2208.
- 8 Varona P, Ibarz JM, Sigüenza JA, and Herreras, O. Current source density analysis as a tool to constrain the parameter space in hippocampal CA1 neuron models. *Lecture Notes in Computer Science* (1997) 1240:82-90.
- 9 Ibarz J.M. and Herreras O. A study of the action potential initiation site along the axosomatodendritic axis of neurons using compartmental models. *Lecture Notes in Computer Science* (2003), this issue.
- 10 López-Aguado, L., Ibarz J.M., Varona, P. and Herreras, O. Structural inhomogeneities differentially modulate action currents and population spikes initiated in the axon or dendrites. *J. Neurophysiol.* (2002) 88:2809-2820.

Intermittent burst synchronization in neural networks

Christian Hauptmann^{1,4}, Annette Gail², and Fotios Giannakopoulos³

¹ Centre for Nonlinear Dynamics in Physiology and Medicine, Department of Physiology, McGill University, 3655 Drummond Street, Montreal, Quebec, Canada H3G 1Y6

² Mathematical Institute, University of Cologne, Weyertal 86-90, 50931 Cologne, Germany agail@mi.uni-koeln.de

³ Fraunhofer Institute for Autonomous intelligent Systems - AiS, Schloss Birlinghoven, 53754 Sankt Augustin, Germany
fotios.giannakopoulos@ais.fraunhofer.de

⁴ Research Center Jülich, 52425 Jülich, Germany c.hauptmann@fz-juelich.de

Abstract. The dynamics of a network built out of excitatory and inhibitory neurons is investigated in terms of burst synchronization. The inherent activity of the neurons is controlled by a set of three delay differential equations. We take into account time delays due to propagational and synaptical delays, nonlinearities due to the synaptic transmission process and the spike generation. Intermittent synchronized network activity is observed. A mechanism for this self organized activity is proposed and bases on the occurrence and propagation of bursting activity. The results are discussed in the context of epilepsy research.

1 Introduction

Since synchronization is a phenomenon which seems to be responsible for the beneficial functions of the brain, cognition [16] etc., as well as malfunctions, epilepsy [12], Parkinson disease [18] etc., the study of synchronization and desynchronization mechanisms is an important field in science, for instance see [13]. In our simulational study we observed a mechanism for intermittent synchronization which bases on the intrinsic properties of the spiking neuron model and the time delay in the system. This mechanism results in self organized intermittent synchronization observed in homogeneous networks of excitatory and inhibitory neurons and indicates a mechanism for the initiation and coordination of spontaneously synchronized discharges as proposed by Connors [1].

Numerous studies of synchronization in neural networks investigate the synchronized or desynchronized behavior of the network as a result of parameter variation or stimulation from outside, for examples see [11, 14, 15, 19] and for systems including delays see [3, 8].

The neuron model used in this study is simple but includes the properties which seem to be important for the proposed synchronization mechanism, the intrinsic properties of bistability and the time delayed feedback. We are able to elucidate a possible mechanism for intermittent synchronization.

2 The mathematical model

The investigation of burst synchronization as related to the onset of an epileptic seizure calls for a model that displays the spiking as well as the bursting behavior on the single neuron level. While small phase shifts have strong influences on the synchronization properties the model has to take into account the delays induced by the synaptical delay and the finite propagation speed of the neural signals. Another requirement balancing the above ones is the necessity of simplicity to allow insight and guaranty benefit from the modeling study.

Following these requirements our neural network consists of m pairs of an excitatory and an inhibitory neuron, where the mathematical model of a single neuron consists of a network equation and the nonlinear oscillator representing the spike generator, a set of three ordinary differential equations with time delay results. As a simple spike generator we choose the well known FitzHugh-Nagumo oscillator, for details see [5, 6]. Consequently the network is controlled by a set of $6 \cdot m$ delay differential equations. For the excitatory neurons we have:

$$\begin{aligned} \tau_{exc} \dot{u}_{exc,i}(t) &= -u_{exc,i}(t) - p^- c_{exc/inh} g(v_{inh,i}(t - T^\sigma)) \\ &\quad + p^+ \sum_k c_{ik} g(v_{exc,k}(t - T^\sigma - T_{ik})) + e_{exc,i}(t) \\ \dot{v}_{exc,i}(t) &= c \left(w_{exc,i}(t) + v_{exc,i}(t) - \frac{1}{3} v_{exc,i}^3(t) \right) + u_{exc,i}(t) \quad (1) \\ \dot{w}_{exc,i}(t) &= (a - v_{exc,i}(t) - b w_{exc,i}(t)) / c. \end{aligned}$$

and for the inhibitory neurons we have:

$$\begin{aligned} \tau_{inh} \dot{u}_{inh,i}(t) &= -u_{inh,i}(t) + p^+ c_{inh/exc} g(v_{exc,i}(t - T^\sigma)) \\ &\quad - p^- c_{inh/inh} g(v_{inh,i}(t - T^\sigma)) + e_{inh,i}(t) \\ \dot{v}_{inh,i}(t) &= c \left(w_{inh,i}(t) + v_{inh,i}(t) - \frac{1}{3} v_{inh,i}^3(t) \right) + u_{inh,i}(t) \quad (2) \\ \dot{w}_{inh,i}(t) &= (a - v_{inh,i}(t) - b w_{inh,i}(t)) / c. \end{aligned}$$

where the variable $u_{exc/inh,i}$ denotes the total postsynaptic potential of the i th excitatory/inhibitory neuron resulting from the delayed incoupling (time delay T^σ , T_{ik}) of the incoming membrane potentials $v_{exc/inh,i}$ of the surrounding excitatory/inhibitory neurons, $i = 1, \dots, m$. The coupling constants are $c_{exc/inh}$, $c_{inh/exc}$, $c_{inh/inh}$ and c_{ik} , while the latter coupling strength is a function of the distance between the considered neurons i and k . $w_{exc/inh,i}$ is a recovery variable. The parameters p^+ and p^- allow us to change the excitation and inhibition simultaneously. g is a monotonically increasing, nonnegative and bounded function

$$g(x) = \frac{1}{1 + e^{-4x}}. \quad (3)$$

$\tau_{exc/inh}$ denotes the synaptic time constant. The parameters a , b , and c specify the dynamical properties of the FitzHugh-Nagumo oscillator. For a more

detailed description of the spiking neuron model see [6]. A constant external current is indicated by $e_{exc,i}$ and $e_{inh,i}$.

The pairs of excitatory and inhibitory neurons are arranged in a network while we take into account local connections between the excitatory and inhibitory neurons and global, distance dependent connections between the excitatory neurons, see figure 1.

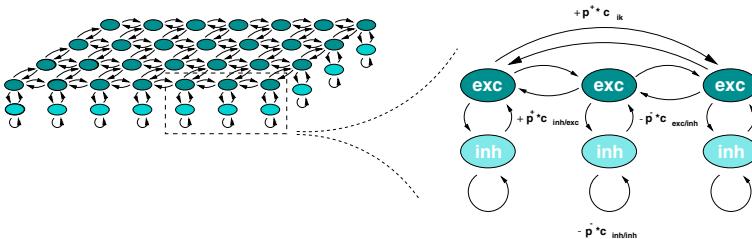


Fig. 1. Geometry of the simulated network. There are two layers of $m = 400$ neurons (left figure), the neurons in the upper layer (dark shaded) are denoted as excitatory neurons, the neurons in the lower layer (light shaded) are denoted as inhibitory neurons. The distance between neighboring excitatory neurons is $l = 0.2$ mm.

3 Numerical Results

Using numerical methods we investigate the dynamical properties of a neural network consisting of 400 pairs of neurons. In the simulations we use periodic boundary conditions to avoid boundary effects. The simulations are initialized with the stable stationary solution of the system. A single short stimulation affecting a neuron from the middle of the network is applied. The neural activity propagates through the network, see figure 2. The dominant dynamics of the network, desynchronized, partial synchronized and synchronized bursting activity can be found for different parameter regimes [6]. The parameter regimes are separated by a parameter domain where an interesting behavior of the neurons is observed and on which we will focus: Intermittent synchronized (in-phase) network dynamics occurs out of a mostly desynchronized (out-of-phase) dynamics.

Intermittent synchrony is found in the large network for a wide parameter range and proves itself as a robust phenomenon. In particular the size of the network and most important parameters controlling the slow dynamics of the total post-synaptic potentials τ_{exc} and τ_{inh} can be changed over a wide range without destroying the occurrence of intermittent synchrony. The effect of τ_{exc} as a control parameter is elucidated in more detail in the discussion. The investigation of time courses of long duration (90 seconds) indicates that there is no periodic recurrence of the synchronized domains. The duration of the syn-

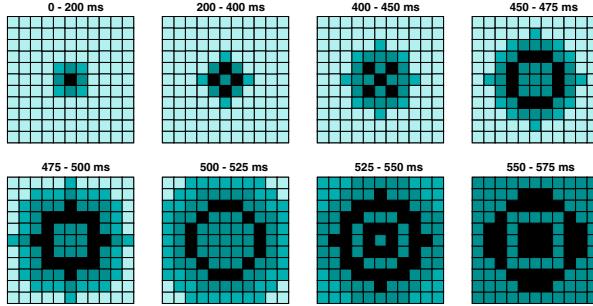


Fig. 2. Propagation of bursting activity. The neurons are initialized in the stationary solution (light grey squares) and a middle neuron is stimulated. Bursting is indicated by a black square. The bursting activity starts in the center of the network and spreads to the edges. Parameters: $m = 400$, $l = 0.2$ mm, $\tau_{exc} = 21.0$ ms, $\tau_{inh} = 12.0$ ms, $a = 0.9$, $b = 0.9$, $c = 2.0$, $e_{exc,i} = -2.6$, $e_{inh,i} = -2.7$, $v_{signal} = 0.5$ m/s, $T_{ik} = d_{ik}/v_{signal}$ (d_{ik} : distance between the excitatory neurons), $T^\sigma = 1.0$ ms, $p^+ = 0.25$, $p^- = 2.4$, $\Delta t = 0.1$ ms, $c_{exc/inh} = 1.0$, $c_{inh/exc} = 1.2$, $c_{inh/inh} = 0.125$. $c_{ik} = s_{exc} - d_{ik} * r_{exc}$ for $d_{ik} \leq 0.5$ mm, $c_{ik} = 0.0$ otherwise, $s_{exc} = 0.5$, $r_{exc} = 1$.

chronized domains range from 500 ms up to 3 seconds. The detailed network dynamics during the intermittent synchronized activity is shown in figure 3.

To quantify the synchrony of the network dynamics, e. g. figure 3, a synchronization measure defined by Kuramoto [10] and Strogatz and Mirollo [17] which was applied to spiking neuron models by Pinsky and Rinzel [15] is modified. The instantaneous phase $\Theta_i(t)$ of each neuron is detected using the starting time of the burst as a trigger. If $t_{i,k}$ is the time the i th neuron starts its k th burst $\Theta_i(t)$ is defined by

$$\Theta_i(t) = \frac{t - t_{i,k}}{t_{i,k+1} - t_{i,k}}, \quad (4)$$

where $t_{i,k} < t < t_{i,k+1}$. With $\Theta_1, \dots, \Theta_n$ and $\Theta_i \in [0, 1]$, a complex order parameter is defined by

$$r(t) = \frac{1}{n} \sum_{j=1}^n \exp(2\pi i \Theta_j(t)), \quad (5)$$

$r(t)$ measures the phase coherence or synchrony at time t of n neurons. The absolute value of r is a useful measure for the synchrony of the network dynamics. Figure 3, lower plot, shows the synchrony measure corresponding to the above displayed network dynamics. The synchrony measure shows small values $|r(t)|$ during the desynchronized and large values $|r(t)|$ during the synchronized domains.

During the synchronized activity the silent period between the bursts is enlarged which affects a decreased frequency of the population dynamics during

synchronized activity compared with the frequency of the population dynamics during desynchronized activity. This behavior is caused by the loss of synaptic input during the silent but excitable period. These results might be a possible explanation for the widely observed decrease of the outstanding frequency in the EEG diagram during an epileptic seizure (for a review see [12] and references therein).

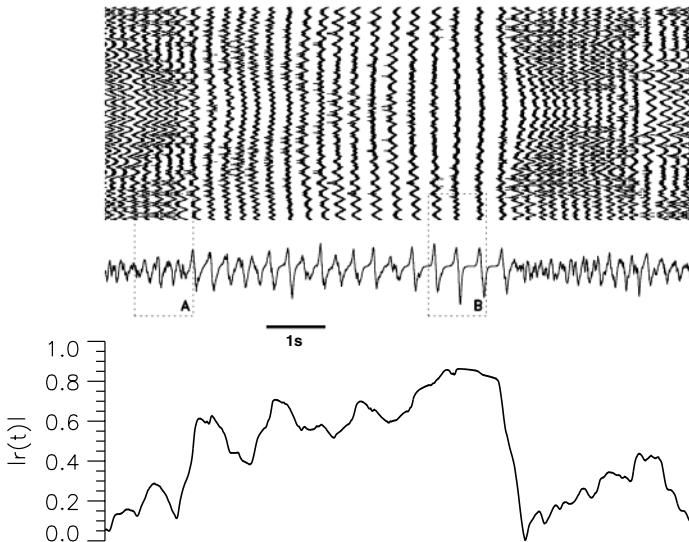


Fig. 3. Intermittent synchronized network dynamics (upper graph) and synchrony measure (lower graph). In the upper part the pattern of neural activity is plotted. From bottom to top the time courses of the 400 excitatory neurons are shown. Each spike is indicated by a dot. The time scale is indicated by the bar. The mean total postsynaptic potential $\langle u(t) \rangle$ is plotted below the activity pattern. During synchronized network activity (B), corresponding to high amplitude and low frequency oscillations of $\langle u(t) \rangle$, we find synchrony measures $|r(t)| \geq 0.5$, and if the pattern of the neural activity gets unstructured (A) the synchrony measure decreases. High frequency and low amplitude fluctuations of $\langle u(t) \rangle$ correspond to this regime. Parameter: $\tau_{exc} = 18.0$ ms, for other parameters see figure 2.

4 Discussion

Despite of the fact that the mechanisms of seizure termination are more complex than represented in the used model the observed intermittent synchronous activity might be a part of a possible mechanism for the initiation of synchronized

activity associated with epileptic seizures. Two observations might suggest this hypothesis: bursting activity seems to be a necessary requirement for the occurrence of the intermittent synchronization and a frequency shift towards low frequency oscillations for synchronized activity and high frequency fluctuations for desynchronized activity is observed. These mechanisms do not require large networks but also occur in small networks of pairs of neuron models [7].

Furthermore the investigations of the network dynamics as well as the dynamics of isolated sub populations of two pairs of excitatory and inhibitory neurons showed that the synchronization and desynchronization mechanism bases on the occurrence of a secondary activation of several excitatory neurons. The neurons show a secondary activation during the highly refractory period after the termination of the burst - this special type of activity pattern might be called a *burst doublet*, see figure 4. The secondary activation of the advanced neuron (with re-

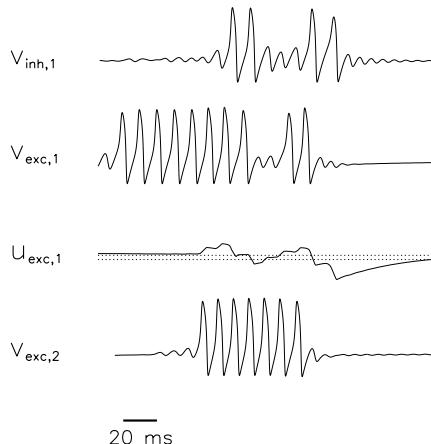


Fig. 4. Mechanism for the burst doublet generation and synchronization. Example of the detailed dynamics of two isolated pairs of neurons. $v_{exc,1}, v_{exc,2}, v_{inh,1}$ and the total postsynaptic potential of the first excitatory neuron $u_{exc,1}$ are shown. The delayed incoupling membrane potentials $v_{exc,2}$ and $v_{inh,1}$ are shifted on the time axis by the delay time $T_{12} + T^\sigma$ and T^σ , respectively. The boundaries of the bistable domain where a stable stationary and a stable periodic solution of the spike generator coexist are indicated by dashed lines. The membrane potentials $v_{exc,2}$ and $v_{inh,1}$ drive the total postsynaptic potential of the first neuron $u_{exc,1}$, see equation 1, 2. Induced by the bistability of the spike generator a secondary activation of $v_{exc,1}$ is possible.

spect to the surrounding neurons) results in a tardy termination of the activity in this neuron, hence the relative phase lag between the neurons is reduced due to the burst doublet - the burst doublets initialize and stabilize the synchronized

activity. Similar activity patterns were also observed by Pinsky & Rinzel [15] but they did not identify it as a possible mechanism for synchronization.

Recent experimental observations [9] support our conjecture. Hippocampal slices were investigated by using microelectrode arrays to detect spatio-temporal activity. They focussed on the initiation of population spikes as observed in the initiation pattern of epileptic seizures. They present the prevalently observed double- or multi-peaked waveforms in the spike time distribution with a second peak leading into the first peak of the next channel as a possible mechanism of synchronization. The double-peaked waveform was interpreted as double-peaked bursts. These findings support our interpretation that the secondary activation of several neurons might take part in the initiation of epileptic seizure.

In agreement with experimental observations [4] which showed that fast gap junctions characterized by fast interaction cause synchronized high frequency oscillations in *in vitro* preparations [2] small $\tau_{exc} < 17.0$ ms result in synchronized and large $\tau_{exc} > 25.0$ ms in desynchronized activity (for intermediate synaptic time constants intermittent synchronization is observed [7]) and mostly synchronized activity is found for vanishing time delay. In contrast strong time delay T_{ik} and T^σ support the occurrence of desynchronized activity. The time delay permanently results in small phase shifts in the coaction of the connected neurons.

Even though the presented model of an isotropic and homogeneous cortical network does not replicate the real *in vitro* situation the simulations are in good agreement with experimental observations. The results of this simulational study lead to the hypothesis that burst doublets and burst prolongation are part of a network-intrinsic mechanism for neural synchronization.

References

- Connors, B. W. (1984). Initiation of synchronized neuronal bursting in neocortex. *Nature*, 310:685–687.
- Draguhn, A., Traub, R. D., Schmitz, D., and Jefferys, J. G. R. (1998). Electrical coupling underlies high-frequency oscillations in the hippocampus *in vitro*. *Nature*, 394:189–192.
- Ernst, U., Pawelzik, K., and Geisel, T. (1998). Delay-induced multistable synchronization of biological oscillators. *Physical Review E*, 57:2150–2162.
- Fisahn, A., Pike, F. G., Buhl, E. H., and Paulsen, O. (1998). Cholinergic induction of network oscillations at 40 Hz in the hippocampus *in vitro*. *Nature*, 394:186–189.
- FitzHugh, R. (1961). Impulses and physiological states in theoretical models of nerve membrane. *Biophysical Journal*, 1:445–466.
- Giannakopoulos, F., Bihler, U., Hauptmann, C., and Luhmann, H. J. (2001). Epileptiform activity in a neocortical network: a mathematical model. *Biological Cybernetics*, 85:257–268.
- Hauptmann, C. (2000). Epileptiform activity in differential equation models of neuronal networks. PhD thesis, University of Cologne ISBN 3-8265-7974-7, Shaker-Verlag Aachen, Germany.
- Karbowski, J. and Kopell, N. (2000). Multispikes and synchronization in a large neural network with temporal delays. *Neural Computation*, 12:1573–1606.

9. Knott, T. (2001). *Population synchronization during propagation of epileptiform activity in organotypic hippocampal slices - a microelectrode array study*. Ph. D. thesis, Fakultät für Biologie der Eberhard Karls Universität Tübingen, Der Andere Verlag, Osnabrück.
10. Kuramoto, Y. (1984). *Chemical oscillations, waves and turbulence*. Springer, Berlin.
11. Lewis, T. J. and Rinzel, J. (2000). Self-organized synchronous oscillations in a network of excitable cells coupled by gap junctions. *Network: Computational Neural Systems*, 11:299–320.
12. McCormick, D. A. and Contreras, D. (2001). On the cellular and network bases of epileptic seizures. *Annual Review of Physiology*, 63:815–846.
13. Special issues on Synchronization (2000). *International Journal of Bifurcation and Chaos*, 10(10 & 11).
14. Pasemann, F. (1999). Synchronized chaos and other coherent states for two coupled neurons. *Physica D*, 128:236–249.
15. Pinsky, P. F. and Rinzel, J. (1995). Synchrony measures for biological neural networks. *Biological Cybernetics*, 73:129–137.
16. Singer, W. (1993). Synchronization of cortical activity and its putative role in information processing and learning. *Annu. Rev. Physiol.*, 55:349–374.
17. Strogatz, S. and Mirollo, R. (1991). Stability of incoherence in a population of coupled oscillators. *J. Stat. Phys.*, 63:613–636.
18. Tass, P. A. (1999). *Phase Resetting in Medicine and Biology*. Springer, Berlin.
19. Varona, P., Torres, J. J., Abarbanel, H. D., Rabinovich, M. I., and Elson, R. C. (2001). Dynamics of two electrically coupled chaotic neurons: Experimental observations and model analysis. *Biological Cybernetics*, 84:91–101.

Diffusion Associative Network: Diffusive Hybrid Neuromodulation and Volume Learning

Fernandez Lopez², P., Suarez Araujo¹, C.P., Garcia Baez³, P., and Sanchez Martin², G.

¹ University Institute for Cibernetic Sciences and Technologies. University of Las Palmas de Gran Canaria, 35017 Las Palmas de Gran Canaria, Canary Islands, Spain.
cpsuarez@dis.ulpgc.es

² Department of Computer Sciences and Systems. University of Las Palmas de Gran Canaria, 35017 Las Palmas de Gran Canaria, Canary Islands, Spain.
pfernandez@dis.ulpgc.es

³ Department of Statistics, Operating Research and Computation. University of La Laguna, 38071 La Laguna, Canary Islands, Spain.
pgarcia@ull.es

Abstract. In this paper, we show how the diffusion of the Nitric Oxide retrograde neuromessenger (NO) in the neural tissue produces Diffusive Hybrid Neuromodulation (DHN), as well as positively influencing the learning process in the artificial and biological neural networks. It also considers whether the DHN, together with the correlational character that helps Hebb's law, is the best schema to represent the resulting learning process. We conducted the entire study by means of analysing the behaviour of the Diffusion Associative Network (DAN) proposed by our group [1].

In addition, and in order to identify the way in which the diffusion of the NO may coincidentally affect the learning process, such as those supported by Hebb's Law, we created and studied recursive schemas of the calculation of the optimal weight matrix in the lineal association, which were then used to try to identify possible ways to express the diffusion. From this last study, we concluded that the recursive schemas are not sufficient in the calculation of the weight matrix in order for the expression identification of the diffusion phenomena in the learning process. These results pointed us towards the search for new schemas for the Hebb law, based on the effect of the NO and that it is different to the modulated correlation, as well as for the search for a more appropriate diffusion model for the NO, such as the compartmental model.

1 Introducción

During these last years, a new type of process for signalling between cells seems to be emerging. This process is the *volume transmission*. The volume transmission is performed by means of a gas diffusion process, whose underlying mechanism is a diffusion signalling molecule (NO). NO is considered an *atypical cellular messenger* working as a retrograde cellular messenger. From studies performed

by us about the functional and organisational role of NO in learning we could conclude about the significance to define a global study framework of diffusion messenger NO (GSFNO) [2]. The GSFNO is composed of different four theoretical and experimental frameworks. One of them is the neurocomputational framework (NCF). This frame might be able to provide possible biological support for many aspects of artificial neural networks (ANNs) and to generate new concepts to improve the structure and operation of them. It is the responsible to obtain more realistic and powerful neural computational structures.

The body of this work is characteristic of the NCF and it is a scientific paper belonging to NO and ANN Series [1].

Finally, and with main goal to discover the way affecting the diffusion of NO to the coincidence learning processes, we study several recursive schemes for computing the optimal matrix weights for the lineal association.

2 Diffusion Associative Network (DNA).

In this section, we describe the behaviour of Diffusion Associative Network (DNA)[1], both in the learning mode and functioning mode, using a non-orthogonal set of patterns, *chines characters*, Fig. (2). We have developed the diffusion version of a classic architecture, the Associative Network (AN) [3], using the analytical model for volumetric transmission based on Bessel functions [1].

The Associative Network is based on correlation matrix model [3]. The learning process to memorise data consists of increasing the value of every associator element M_{ji} by the amount directly proportional to $a_i u_j$.

For a set of patterns $(\mathbf{a}^1, \mathbf{u}^1), (\mathbf{a}^2, \mathbf{u}^2), (\mathbf{a}^3, \mathbf{u}^3), \dots, (\mathbf{a}^p, \mathbf{u}^p)$,

$$M_{ji} = \rho \sum_p a_i^p u_j^p . \quad (1)$$

where ρ is the normalising constant.

The recall of a particular u_i^r associated with a key vector \mathbf{a}^r is made by transformation

$$\hat{u}_j^r = \sum_i M_{ji} a_i^r . \quad (2)$$

The structure of the Diffusion Associative Network (DAN) is shown in the Fig. (1). The associator elements are organised in neural assemblies. The number of neural assemblies in a DAN is equal to the dimension of the key field, n . Each assembly is made up by a number of associator elements equal to the size of the data field of the DNA. Each associator belonging to an i th neural assembly receives two connections from the receptor field. One comes from the i th receptor of the key field, which is the one that defines the neural assembly, and the other connection is received from the receptor belonging to the data field, of the same order as the associator. An associator element of the DNA is defined by M_{ji} , where the subscript i indicates the i th neural assembly, and the subscript j the

jth associator element of the assembly. Thus, a jth associator element of the ith assembly receives the input of the u_j and a_i elements. Fig. (1).

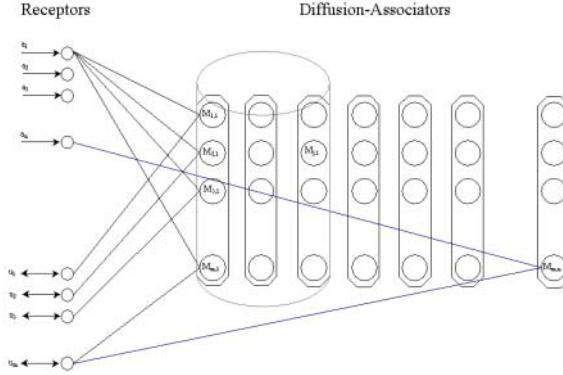


Fig. 1. Structure of a Diffusion Associative Network (DNA).

This artificial neural network involves the influence of neural structures in each of its neurons, not only due to the synaptic transmission by means of the specified synaptic connections, but also to the volumetric transmission by means of the diffuse neighbourhoods (DNB). In this way, from the emergence of DNB and from the range of NO spread, the NO can alter neuronal circuits by means of the diffusive hybrid neuromodulation (DHN). In this monodimensional structure of neural assemblies, we have selected influences such as the ith assembly, which can be influenced by the neural assemblies (i-1)th, ith and (i+1)th, during the diffusion period. This involves the existence of an extrinsic neuromodulation produced by NO effect. Our model includes the inter-assembly volumetric transmission and therefore the existence of the volume learning is considered. The volume learning will be composed by classical neurotransmission learning and by fast diffuse neural learning (FDNL) [2]. This is expressed in equations (3) and (4), where the DNA learning process is formalised. This learning law will specifically reflect its most simple abstraction.

$$M_{ji} = \rho \sum_p c_i^p a_i^p u_j^p . \quad (3)$$

In this case the correlation is weighed by c_i^p which has a behaviour based in the diffusion

$$c_i^p = (1 - 2D)f(a_i^p) + D(c_{i-1}^{p-1} - c_{i+1}^{p-1}) . \quad (4)$$

and where

$$f(a_i^p) = \begin{cases} \eta & \text{if } a_i^p = \max(a_i^p) \\ c_i^{p-1} & \text{otherwise} \end{cases} \quad (5)$$

where η is the strength of the NO neurons.

2.1 Results and discussion

Once the DAN was described, we analysed the behaviour of the two networks, the Classic Associative Network and the Diffusion Associative Network, in the auto-association of the set of highly non-orthogonal patterns shown in Fig. (2).



Fig. 2. Set of non-orthogonal patterns, *Chinese Characters*, used to train the classic associative network and the diffusion associative network.

In the right column of Fig. (3(a)) we can see a notable improvement in the capacity of the network to recover patterns can also be observed with respect to the classic associator, the average column in Fig. (3(a)).

These improvements in the auto-associative capacity of the DAN, confirm the hypothesis of the existence of the DHN and of the influence of the NO retrograde neuro-transmitter in the learning process by means of the FDNL, which infers the existence of volume learning in the neural networks. However, we cannot say that this improvement effect transgressed by the NO in the DAN is total, as we can also observe in Fig. (3(b)) that the improvement in the recovery does not cover all the patterns. This allows us to state that the fast diffusion neural learning cannot only be reflected in the schema of the emerging modulation due to the NO effect. It also requires a formal non-correlational framework for Hebb's law. The value of these results is in the experimental medium which provide some physiological evidence of the discrepancies between the physiology of the LTP and the Hebb correlation requirement, as well as the need to define a new formal framework that represents Hebb's law [2].

We are thus continuing to find the answer to a key question within the effect of the NO on the learning processes, both in the biological and artificial neural networks: How does the diffusion of No influence the learning processes?



Fig. 3. (a) Set of patterns where the DAN network notably improves its auto-associative capacity compared to the classic associator. (b) Set where the DAN does not improve its capacity.

Our objective in the different sections was therefore, and following the analytical model of the diffusion of the NO based on Bessel functions [1], to determine the way in which the diffusion can affect the learning processes by coincidence, (Hebb Law).

3 Lineal association and optimal lineal association: Recursive schemas to calculate W

The main function of the lineal association, typical of the classic associator, [3] is reflected in the expression (6),

$$\mathbf{y}^i = W \mathbf{x}^i, \forall i = 1..l \quad (6)$$

where $\mathbf{y}^i \in \mathbb{R}^n$, and $\mathbf{x}^i \in \mathbb{R}^m$.

We take optimal lineal association to be, when we are working with sets of non-orthogonal model, the one that allows us to calculate a W capable of minimising the average quadratic error (7).

$$F(W) = \frac{1}{l} \sum_{k=1}^l |\mathbf{y}_k - W \mathbf{x}_k|^2 \quad (7)$$

A W calculated according to the expression (8), minimises (7).

$$W = Y X^+ \quad (8)$$

where $Y = (\mathbf{y}^1 \mathbf{y}^2 \dots \mathbf{y}^l)$ and X^+ is the generalised inverse matrix (pseudo-inverse) [4] of $X = (\mathbf{x}^1 \mathbf{x}^2 \dots \mathbf{x}^l)$.

We centred on the calculation of W in order to auto-associate $X = (\mathbf{x}^1 \dots \mathbf{x}^l)$ where $\mathbf{x}^i \in \mathbb{R}^m$.

Various schemas have been proposed to calculate a W based on the pseudo-inverse X^+ matrix, [4]. Using the expressions resulting from the (9) and (10) equations, which represent the dynamics of a V variable, which converges to X^+ for the cases in which $m > l$ and $m \leq l$ respectively, [5],

$$\frac{dV}{dt} = -\mu X^t X V + \mu X^t \quad (9)$$

$$\frac{dV}{dt} = -\mu V X X^t + \mu X^t \quad (10)$$

where μ is a positive constant.

We applied the finite differences method to these expressions to discredit the V variable and using (8), we can obtain the recursive schemas of W .

$$W^{k+1} = (1 - \gamma)W^k + \gamma X V^k - \gamma \eta X X^t X V^k + \eta X X^t \quad (11)$$

$$W^{k+1} = (1 - \gamma)W^k + \gamma X V^k - \gamma \eta X V^k X^t X + \eta X X^t \quad (12)$$

where γ and η are positive constants.

3.1 Results and discussion

We tested the new network with the set of patterns shown in Fig. (2), and we observed how their behaviour, with respect to the recovery of the patterns, improved considerably in relation to the classic associative network.

With respect to the Diffusion Associative Network, we saw that the results in the model recovery is comparable to the behaviour of the associative network when we used the weight matrix obtained with the recursive schemas, and with a high number of iterations. Figure (4) shows us the different recoveries of the associative network for patterns belonging to the set in question, and different number of iterations of the recursive schema expressed by the (11) equation, where we observed that for $k = 40$ in the patterns shown, the recovery is comparable to that obtained by the diffusion associative network.

4 Diffusion schemas

It will be in the schemas represented in (11) and (12) where we will try to identify diffusion phenomena. It will also be very useful to be able to detect possible dependencies that would indicate those operations that could be linked to such diffusion phenomena, as well as to modulation.

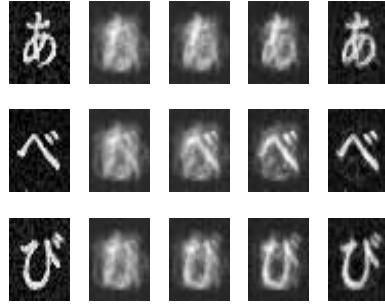


Fig. 4. Example of the behaviour for three patterns (left) of the schema represented by (11) for the values of $k = 5, 20, 40$ and 200 (from left to right).

The calculation of the individual elements of the V matrix, which converges to X^+ when $k \rightarrow \infty$, is expressed by (13) and (14) in function of whether $m > l$ or $m \leq l$.

The schemas expressed in (11) and (12) converge towards the $w_{ij}^\infty = x_{i1}v_{1j}^\infty + x_{i2}v_{2j}^\infty + \dots + x_{il}v_{lj}^\infty$ value when $k \rightarrow \infty$, where the v_{ij}^{k+1} value is given by the following expressions:

$$v_{ij}^{k+1} = (\delta_{i,1} - (\mathbf{x}^i)^t \mathbf{x}^1 \dots \delta_{i,i} - (\mathbf{x}^i)^t \mathbf{x}^i \dots \delta_{i,l} - (\mathbf{x}^i)^t \mathbf{x}^l) \cdot \begin{pmatrix} v_{1j}^k \\ \vdots \\ v_{ij}^k \\ \vdots \\ v_{lj}^k \end{pmatrix} + \eta x_{ji} \quad (13)$$

$$v_{ij}^{k+1} = (v_{i1}^k \dots v_{ij}^k \dots v_{im}^k) \cdot \begin{pmatrix} \delta_{1,j} - \eta h_{1j} \\ \vdots \\ \delta_{i,j} - \eta h_{ij} \\ \vdots \\ \delta_{m,j} - \eta h_{mj} \end{pmatrix} + \eta x_{ji} \quad (14)$$

where $\delta(i, j) = 1$ if and only if $i = j$, and $h_{ij} = \sum_{k=1}^m x_{ik}x_{jk}$.

These two schemas have a non-local character in time and space. The (13) equation represents the calculation of v_{ij}^{k+1} , when $m > l$, few patterns to learn with great resolution. The (14) equation represents the v_{ij}^{k+1} calculation, when $m \leq l$, many patterns to learn with little resolution. The arguments identified in (13) and (14) are not sufficient to establish a diffusion schema in the Diffusion Associative Network. These schemas seem to have a similar shape to the diffusion-reaction schemas. The greatest difference between both of them is that the one represented by (13) seems to have a time effect, calculates the v_{ij}^{k+1} value with all the j components of the \mathbf{v}^1 until \mathbf{v}^l . And to the contrary, the calculation of the (14) in the v_{ij}^{k+1} is performed with all the components of the \mathbf{v}_i vector.

5 Conclusions

The in-depth analysis of the Diffusion Associative Network, with a rough and complex set of highly non-orthogonal patterns, together with the comparative study with the classic associator and with the classic associator with optimum W that reacts to recursive schemas, has shown us that extrinsic and intrinsic diffusion neuromodulation exists. This confirms our hypothesis of the emergence of the DHN and of the influence of the NO retrograde neurotransmitter in the learning process by means of the FDNL, from which the "volume learning" is inferred as an improvement to its auto-association activity relating to the associative networks studied.

Another interesting conclusion has been to establish that the effect of the improvement caused by the NO in the DAN is not total, as the improved recovery does not cover all the patterns. We therefore consider that the fast diffusion neural learning cannot only be reflected in the schema of the emerging modulation due to the NO effect. A formal non-correlation framework is also required for the Hebb law.

We also concluded that the recursive schemas of the calculation of the optimal W has a temporal dynamic character, in the case of few high-resolution learning patterns, and a spatial dynamic character, in the case of many low-resolution learning patterns. We have showed that the diffusion analytical model of the NO, based on Bessel functions, does not fully represent the behaviour and effect of the NO in the neural information and learning procedure. This leads us to propose the development of a new diffusion model in the discrete, based on compartments, from which we can study the essence of the diffusion processes and their relationship with the dynamics of learning processes, such as Hebb's. This will allow us to progress in the development of the theoretical framework to study the diffusion of NO in neural tissue.

References

1. Suárez Araujo, C.P., Fernández López, P., García Báez, P.: Towards a Model of Volume Transmission in Biological Neural Networks: a Cast Approach. *Lectures Notes in Computer Science*, Springer Verlag (2001); Vol. 2178; pp: 328 - 342.
2. Suarez Araujo, C.P.: Study and Reflections on the Functional and Organisational Role of Neuromessenger Nitric Oxide in Learning: An Artificial and Biological Approach. *Computer Anticipatory Systems* (2000); AIP 517; pp: 296 - 307.
3. Kohonen, T.: Correlation matrix memories. *IEEE Transactions on Computers* (1972); C-21; pp: 353 - 359.
4. Ben-Israel, A. and Greville, T.N.E.: *Generalized Inverses: Theory and Applications*, Wiley, New York (1974).
5. Wan, J.: Recurrent neural networks for computing pseudoinverses of rank-deficient matrices. *SLAM J. Sci. Comput.* (1997); Vol 18, num. 5; pp: 1479 - 1493.

The Minimum-Variance Theory Revisited

Jianfeng Feng

COGS, Sussex University, Brighton BN1 9QH, UK
<http://www.cogs.susx.ac.uk/users/jianfeng>

Abstract. The minimum-variance theory [12] was proposed to account for the eye and arm movement. However, we point out here that i) the input signals used in the simulations are not Poisson processes; ii) when the input signal is a Poisson process, the solution of the minimum-variance is degenerate.

1 Introduction

One of the most puzzling issues in neuroscience is why neurons employ stochastic rather than deterministic signals to process information (for example, see review [1, 2, 5, 6, 9, 14, 18]. There are many different approaches to tackle the problem such as stochastic resonance [8], correlated computation [16] and others [10, 11]. Recently Harris and Wolpert[12] (also see recent review [21]) proposed a theory, the minimum-variance theory as an example of optimal control models of movement, aiming to provide a plausible explanation of 'noise calculation' in the cortex. The idea of the theory is quite straightforward: to find a noise signal which minimizes the variance of certain quantities, say the arm movement trajectory.

For concreteness, let us first formulate their theory. The discrete time version used in [12, 17] is

$$x_{t+1} = Ax_t + C(u_t + ku_t^\alpha \xi_t) \quad (1)$$

and $x_0 = 0$ where A is a $d \times d$ matrix and C a $d \times 1$ matrix, k, α are positive constants¹, ξ_t is a normal random variable and $u(t) \in I\!\!R$ is the input signal. $x(t) \in I\!\!R^d$ could be the movement trajectory, velocity etc. Analytically it is not easy to deal with a discrete time system, so we rewrite Eq. (1) as the following continuous time version

$$\begin{cases} dx(t) = (A - I)x(t)dt + C(u(t)dt + ku^\alpha(t)dB_t) \\ x(0) = 0 \end{cases}$$

where B_t is the Brownian motion and $I = (\delta_{ij})_{d \times d}$. We therefore have

$$x(t) = \int_0^t \exp((A - I)(t-s))Cu(s)ds + \int_0^t \exp((A - I)(t-s))Cku^\alpha(s)dB_s \quad (2)$$

¹ $\alpha = 1$ in [12], see also [7].

For a vector $D \in I\!\!R^d$ it is required that

$$Ex(T) = \int_0^T \exp((A - I)(T - s))C \cdot u(s)ds = D \in I\!\!R^d \quad (3)$$

i.e. at time T the mean trajectory ends at the position D . Eq. (3) is the constraint for all problems considered below.

In [12] the authors then assumed that the variance of $x(t)$ in time window $[T, T + R]$ is minimized, i.e. post movement minimization of variance. More specifically, their minimum-variance theory can be summarized as:

- To find an input signal $u(t), t \in [0, T + R]$ such that the covariance $x(t)$ is minimized for $t \in [T, T + R], R > 0$.

Via solely numerical simulations, they then compared their results with experimental data and concluded that their results based upon the minimum-variance theory (MVT) fit well with arm and eye movements. Currently the theory becomes quite influential, opens up many interesting problems for further investigations, both in theory and in experiments, and might provide a clue for 'noise calculation'.

Despite the success of their model, there are quite a few issues which remain elusive. For example, we have pointed in [7] that the signal used in the minimum-variance theory is not Poisson process, in contrast with the claim in [17]. Furthermore, how the model behaviour depends on specific parameters used in the simulations is not clear. Here we first develop a theoretical approach to the minimum-variance theory with post movement minimization, and analytical solutions of the theory are obtained. Unfortunately, at moment it is impossible to directly compare our theoretical solutions with their numerical results in [12, 20].

The theory of minimizing the variance of post movement is interesting, but a biological system might never care about its behaviour after its motion has been stopped. We then propose a new version of the minimum-variance theory: minimizing the variance during movement rather than after it. In other words we intend to minimize the variance during time window $[T - R, T]$ for $T \geq R > 0$. Again in the case, analytical solutions are obtained. Numerically we find that the larger the R is, the smaller the variance.

2 Input Signals

Suppose that a cell receives EPSPs (excitatory postsynaptic potentials) at N_E excitatory synapses and IPSPs (inhibitory postsynaptic potentials) at N_I inhibitory synapses. When the membrane potential Z_t is between the resting potential V_{rest} and the threshold V_{thre} , it is given by

$$dZ_t = -\frac{1}{\gamma}(Z_t - V_{rest})dt + a \sum_{i=1}^{N_E} dE_i(t) - b \sum_{j=1}^{N_I} dI_j(t) \quad (4)$$

where $1/\gamma$ is the decay rate, $E_i(t), I_i(t)$ are point processes and a, b are magnitudes of each EPSP and IPSP[4]. Once Z_t crosses V_{thre} from below a spike is generated and Z_t is reset to V_{rest} . This model is termed as the integrate-and-fire model. For the facilitation of theoretical treatment, we further assume that $N^E(t) = \sum_{i=1}^{N_E} E_i(t)$ and $N^I(t) = \sum_{j=1}^{N_I} I_j(t)$ are renewal processes. Denote t_i^E as the i -th inter-EPSP interval, and t_i^I as the i -th inter-IPSP interval. As in the literature we suppose $1/\langle t_i^E \rangle = r/\langle t_i^E \rangle$, $1/\langle (t_i^E - \langle t_i^E \rangle)^2 \rangle = r/\langle (t_i^E - \langle t_i^E \rangle)^2 \rangle$ with $r \in [0, 1]$, the ratio between inhibitory and excitatory inputs. Hence when $r = 0$ there is no inhibitory inputs and when $r = 1$ inhibitory and excitatory inputs are exactly balanced.

According to a basic result of the renewal process, we have $N^E(t) \rightarrow \text{Nor}(t/\langle t_i^E \rangle, t\langle (t_i^E - \langle t_i^E \rangle)^2 \rangle / (\langle t_i^E \rangle)^3)$. Hence the process $N^E(t)$ can be approximated by $dN^E(t) = \frac{1}{\langle t_i^E \rangle} dt + \frac{\sqrt{\langle (t_i^E - \langle t_i^E \rangle)^2 \rangle}}{(\langle t_i^E \rangle)^{3/2}} dB_t^E$ where B_t^E is the standard Brownian motion. Similarly for IPSP input we have $dN^I(t) = \frac{r}{\langle t_i^E \rangle} dt + \frac{\sqrt{r\langle (t_i^E - \langle t_i^E \rangle)^2 \rangle}}{(\langle t_i^E \rangle)^{3/2}} dB_t^I$ where B_t^I is a standard Brownian motion, independent of B_t^E .

From now on we assume that $a = b, V_{rest} = 0$ mV. Hence the diffusion approximation of the original jump (point) process Z_t is given by

$$dV_t = -\frac{1}{\gamma} V_t dt + dI(t) \quad (5)$$

where synaptic input $dI(t) = \mu dt + \sigma dB_t$ with

$$\mu = a \frac{1-r}{\langle t_i^E \rangle} \quad \sigma^2 = a^2(1+r) \frac{\langle (t_i^E - \langle t_i^E \rangle)^2 \rangle}{(\langle t_i^E \rangle)^3} \quad (6)$$

and B_t is the standard Brownian motion. The interspike interval of efferent spikes (the firing time) is $T = \inf\{t : V_t \geq V_{thre}\}$

Therefore we arrive at the first conclusion.

Theorem 1. *For the membrane potential defined by Eq. (4), i.e. the integrate-and-fire model with renewal processes input, its diffusion approximation is given by Eq. (5).*

Later on we want to check the accuracy of the diffusion approximation. This has been extensively investigated for Poisson process inputs. It is, however, not clear for general renewal process inputs. For carrying out numerical simulations, we confine ourselves further to the case that inter-EPSP of inter-IPSP is distributed according to Gamma distributions.

A Gamma distribution with positive parameters (α, ν) is defined by $f_{\alpha,\nu}(t) = \frac{1}{\Gamma(\nu)} \alpha^\nu t^{\nu-1} \exp(-\alpha t)$ with mean $\langle t_i^E \rangle = \nu/\alpha$ and $\langle (t_i^E - \langle t_i^E \rangle)^2 \rangle = \nu/\alpha^2$, i.e. its coefficient of variation of interspike interval is $CV_G(I) = \frac{1}{\sqrt{\nu}}$

According to Theorem 1 the diffusion approximation now takes the following form $dv_t = -\frac{v_t}{\gamma} + a\frac{\alpha}{\nu}(1-r)dt + a\sqrt{1+r}\frac{\sqrt{\alpha}}{\nu}dB_t$. Define $\lambda = \alpha/\nu$, we have

$$dv_t = -\frac{v_t}{\gamma} + a\lambda(1-r)dt + a\sqrt{1+r}\frac{\sqrt{\lambda}}{\sqrt{\nu}}dB_t \quad (7)$$

When $\nu = 1$, a renewal process input is exactly Poisson process with the coefficient of variation (CV) of firing rates $CV_P(R) = \frac{1}{\sqrt{\lambda}}$ and the CV of interspike intervals (ISIs) $CV_P(I) = 1$

3 MVT of Post Movement

In stead of minimizing $\text{cov}(\int_T^{T+R} x(s)ds)$, we simply take into account to minimize the variance [12] $\|\int_T^{T+R} x(s)ds\|^2$. For $R > 0$, let us define

$$\begin{aligned} Y &= \int_T^{T+R} x(t)dt \\ &= \int_T^{T+R} \int_0^t \exp((A-I)(t-s))Cu(s)ds dt \\ &\quad + \int_T^{T+R} \int_0^t \exp((A-I)(t-s))Cku^\alpha(s)dB_s dt \end{aligned} \quad (8)$$

Hence Y is the average trajectory over time window $[T, T+R]$. Remember that the trajectory stops at time T , therefore in time interval $[T, T+R]$ the movement has stopped (post movement).

We have the following conclusion.

Theorem 2.

$$EY^2 - (EY)^2 = 2 \int_T^{T+R} (T+R-t) \int_0^t \|\exp((A-I)(t-s))C\|^2 k^2 \cdot u^{2\alpha}(s) ds dt$$

In terms of Theorem 2, the original optimization problem can be stated as to find a function $u(t)$ satisfying

$$\min_u I(u) = \min_u \left[2k^2 \int_T^{T+R} (T+R-t) \int_0^t \|\exp((A-I)(t-s))C\|^2 |u(s)|^{2\alpha} ds dt \right] \quad (9)$$

subject to the constraint

$$\int_0^T \exp((A-I)(T-s))Cu(s)ds = D \in I\mathbb{R}^d \quad (10)$$

This is an infinity dimension optimization problem since we have to find a function in $L^{2\alpha}[0, T+R]$ satisfying Eq. (9) and (10).

Theorem 3. Denote $F(t - s) = \exp((A - I)(t - s))C$, then the unique solution of Eq. (9) and (10) $u(t)$ is the solution of the following finite dimension optimization problem. For $t \in [0, T]$ $u(t)$ satisfies

$$\inf_{\lambda \in \mathbb{R}^d} \left[\int_0^T \frac{||\lambda F(T - s)||^{2\alpha/(2\alpha-1)}}{H^{1/(2\alpha-1)}(s)} ds \right] \quad (11)$$

with the constraint

$$\int_0^T F(T - s) \frac{\operatorname{sgn}(\lambda F(T - s)) ||\lambda F(T - s)||^{1/(2\alpha)}}{H^{1/(2\alpha-1)}(s)} ds = D \quad (12)$$

where $H(s) = \int_T^{T+R} (T + R - t) ||F(t - s)||^2 dt$, for $s \in [0, T]$; for $t \in [T, T + R]$, $u(t) = 0$.

For general cases, from Theorem 3, we have to finally resort to numerical simulations to obtain the solution $u(t)$ of the optimal problem defined in Theorem 3. However, for two special cases $\alpha = 1$ and $d = 1$ we can carry out further calculations, which enable us to grasp the property of $u(t)$ of the optimization problem. We first consider the case of $\alpha = 1$.

3.1 Case $\alpha = 1$

For this special case it is readily seen that u is the solution of

$$\inf_{\lambda \in \mathbb{R}^d} \left[\int_0^T \frac{||\lambda F(T - s)||^2}{H(s)} ds \right] \quad (13)$$

with the constraint

$$\int_0^T F(T - s) \frac{\operatorname{sgn}(\lambda F(T - s)) ||\lambda F(T - s)||}{H(s)} ds = D \quad (14)$$

Define $F(T - s) = (v_i(s))$ and

$$W = \left(\int_0^T \frac{v_i(s)v_j(s)}{H(s)} ds \right)_{i,j}$$

and if W^{-1} exists, we have

$$u(s) = \begin{cases} \frac{D^T (W^{-1})^T F(T - s)}{H(s)} & \text{if } 0 \leq s \leq T \\ 0 & \text{otherwise} \end{cases} \quad (15)$$

Basically in [12] the authors carried out simulations of results presented in current subsection, i.e. $\alpha = 1$.

3.2 Case $d = 1$

When $d = 1$ a straightforward calculation tells us that (under the assumption that $F(t) = \exp(\sigma t)$)

$$u(t) = \begin{cases} \frac{D}{T} \exp[-\sigma(T-t)] & \text{if } \alpha = 1 \\ \frac{D\sigma \left(1 - \frac{1}{2\alpha-1}\right)}{\exp\left[\sigma T\left(1 - \frac{1}{2\alpha-1}\right)\right] - 1} \exp\left[-\frac{\sigma}{2\alpha-1}(T-t)\right] & \text{otherwise} \end{cases} \quad (16)$$

for $t \in [0, T]$ and $u(t) = 0$ for $t \in [T, T+R]$. Furthermore the solution is independent of $R > 0$. This implies that no matter how short the time window of $[T, T+R]$ will be, the solution of the optimization problem of Eq. (9) and Eq. (10) is identical and so we could find the solution with a time window of post movement as short as possible. Note that it is essential for the MVT since if the window $[T, T+R]$ is too large, a fast or an instantaneous control becomes impossible.

It is interesting to consider the case of Poisson inputs, i.e. $\alpha = 1/2$ ([19]). However, this is the singular case of the results above, i.e. $u(t) = c_1\delta_0(t)$ when $\sigma < 0$ and $u(t) = c_1\delta_T(t)$ for $\sigma > 0$, where c_1 is an appropriate constant. In Fig. 1 we plot $u(t)$ vs. time t , $\text{var}(t) = \int_0^t \int_T^{T+R} (T+R-v) \exp(2\sigma(v-s)) dv u^{2\alpha}(s) ds$ vs. t and $\bar{x}(t) = \int_0^t \exp(\sigma(T-s)) u(s) ds$ vs. time t for $t \in [0, T]$. For $t \in [T, T+R]$ we have $u(t) = 0$.

Furthermore when $d > 1$ but

$$F(t) = \sum_{i=1}^d c_{i+1} \exp(\sigma_i t)$$

$u(t)$ can be explicitly calculated as well.

4 Discussion

We have presented a simple analysis of the minimum-variance and pointed out that a. the signal used in the minimum-variance is not a Poisson process; b. when the input is a Poisson process, its solution is degenerate. All results obtained here are relied on the linear system defined in Section 1. Certainly in reality a biological system is highly nonlinear. How to generalize our results here to a nonlinear system is an intriguing topic.

Acknowledgement

Partially supported by a grant from EPSRC(GR/R54569), a grant of the Royal Society and an exchange grant between UK and China of the Royal Society.

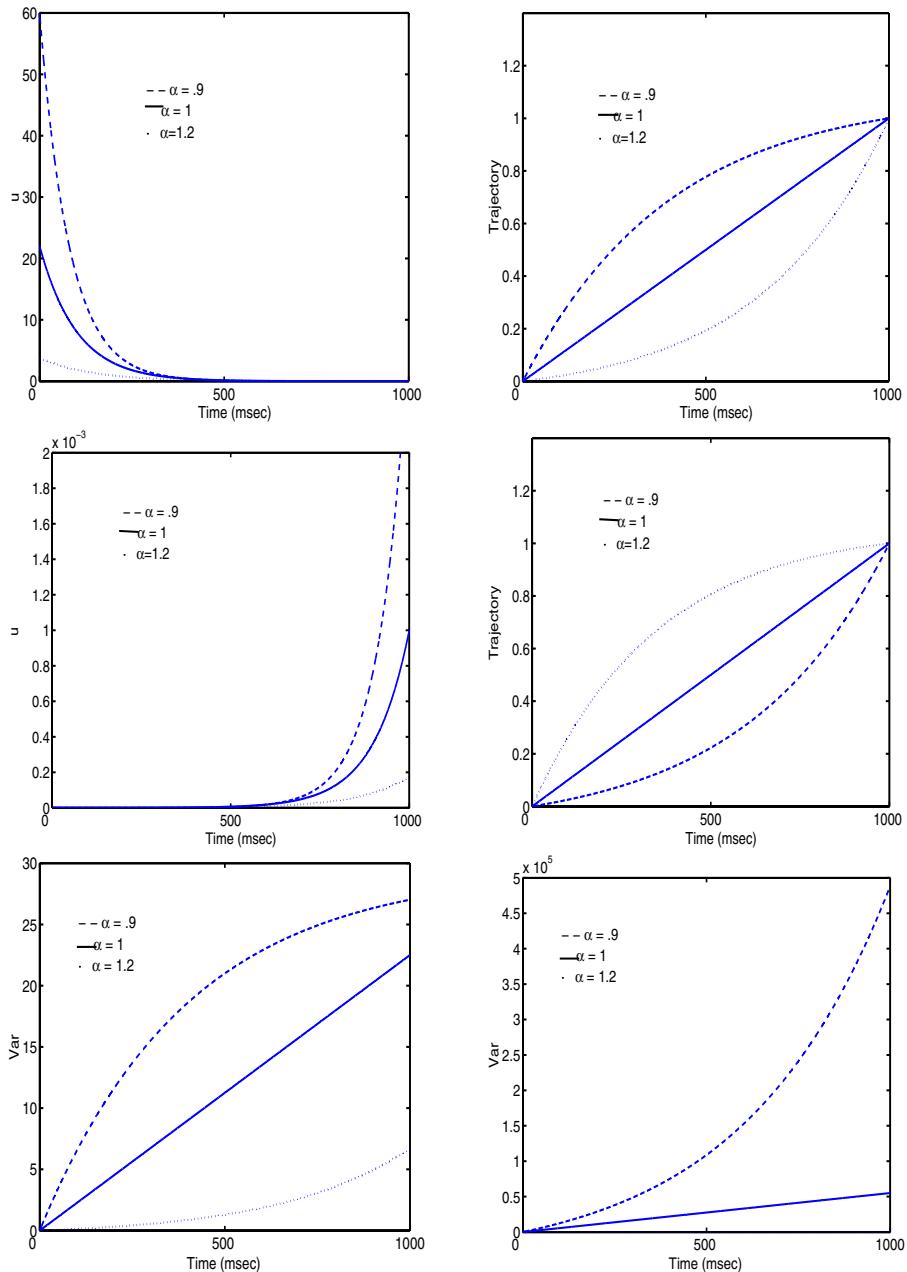


Fig. 1. $\sigma = -0.01, R = 500, T = 1000, d = 1$ (upper panel and bottom panel (left)). $\sigma = 0.01, R = 500, T = 1000, d = 1$ (middle panel and bottom panel(right)). Trajectory is $\bar{x}(t) = \int_0^t \exp(\sigma(T-s))u(s)ds$

References

- [1] Albright T. D., Jessell T. M., Kandel E. R., and Posner M. I. (2000). Neural science: A century of progress and the mysteries that remain. *Cell* **100**, s1-s55. **62**
- [2] de Vries G., and Sherman A. (2000). Channel sharing in pancreatic beta-cells revisited: Enhancement of emergent bursting by noise. *J. Theor. Biol.*, **207**:513-530 **62**
- [3] Doya K., Kimura H., Kawato M. (2001). Neural mechanisms of learning and control. *IEEE Control Systems Magazine*, **21**, 42-54.
- [4] Brown D., Feng J., and Feerick, S. (1999) Variability of firing of Hodgkin-Huxley and FitzHugh-Nagumo neurons with stochastic synaptic input. *Phys. Rev. Letts.* **82**, 4731-4734. **64**
- [5] Feng J. (2001). Is the integrate-and-fire model good enough? –a review. *Neural Networks* **14** 955-975. **62**
- [6] Feng, J. (2002). *Frontiers In Computational Neuroscience* Feng J. (Ed.) CRC Publisher: Boca Raton. **62**
- [7] Feng, J., Tartaglia, G.G., and Tirozzi B. (2001). A note on minimum-variance theory and beyond (submitted). **62, 63**
- [8] Gammaioni L., Hägggi P., Jung P. and Marchesoni F. (1998). Stochastic resonance. *Reviews of Modern Physics* **70** 224-287. **62**
- [9] Gerstner W., Kreiter A.K., Markram H., and Herz A. V. M. (1997). Neural codes: firing rates and beyond. *Proc. Natl. Acad. Sci. USA* **94**, 12740-12741. **62**
- [10] Hopfield J. J., and Brody C. D. (2000). What is a moment? 'Cortical' sensory integration over a brief interval. *Proc. Natl. Acad. Sci. USA* **97** 13919-13924. **62**
- [11] Hopfield J. J., and Brody C. D. (2001). What is a moment? Transient synchrony as a collective mechanism for spatiotemporal integration. *Proc. Natl. Acad. Sci. USA* **98** 1282-1287. **62**
- [12] Harris C. M., and Wolpert D. M. (1998). Signal-dependent noise determines motor planning *Nature* **394**: 780-784. **62, 63, 65, 66, 69**
- [13] Kawato M. (1999). Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology* **9**,718-727.
- [14] Koch, C. (1999). *Biophysics of Computation*, Oxford University Press: Oxford. **62**
- [15] Ricciardi, L. M., and Sato, S. (1990). Diffusion process and first-passage-times problems. *Lectures in Applied Mathematics and Informatics* Ricciardi, L. M. (ed.), Manchester: Manchester University Press.
- [16] Salinas E., and Sejnowski T. (2001). Correlated neuronal activity and the flow of neural information. *Nature Reviews Neuroscience* **2**, 539-550. **62**
- [17] Sejnowski T. J. (1998). Making smooth moves *Nature* **394** 725-726. **62, 63**
- [18] Shadlen M. N., and Newsome W. T.(1994). Noise, neural codes and cortical organization, *Curr. Opin. Neurobiol.* **4**, 569-579. **62**
- [19] Tuckwell H. C. (1988). *Stochastic processes in the neurosciences*. Philadelphia: SIAM. **67**
- [20] Wolpert D. M. (2001). The code for [12] is not available (private communication). **63**
- [21] Wolpert D. M., and Ghahramani Z. (2000). Computational principles of movement neuroscience *Nature Neuroscience* **3**:1212-1217 **62**

Sleep and wakefulness in the cuneate nucleus: a computational study

Eduardo Sánchez¹, Senén Barro¹, Jorge Mariño³, and Antonio Canedo²

¹ Grupo de Sistemas Intelixentes (GSI)

Departamento de Electrónica e Computación, Facultade de Físicas,

Universidade de Santiago de Compostela,

15782 Santiago de Compostela, Spain

{eduardos, elsenen}@usc.es

<http://www-gsi.usc.es/index.html>

² Departamento de Fisiología, Facultade de Medicina,

Universidade de Santiago de Compostela,

15782 Santiago de Compostela, Spain

f.sancale@usc.es

³ Departamento de Medicina, Escola Universitaria de Fisioterapia,

Universidade da Coruña,

15006 A Coruña, Spain

xurxo@mit.edu

Abstract. We present a computational study about the influence of the sensorimotor cortex on the processing of the cuneate nucleus during sleep as well as wakefulness. Realistic computational models were developed supported by experimental data obtained from intact-brain preparations in cat. Furthermore, a physiologically plausible circuit is proposed and predictions under both different cortical stimulation and synaptic configurations are suggested. The computer simulations show that the CN circuitry (1) under sleep conditions can block the transmission of afferent sensory information, and (2) under awaking conditions can perform operations such as filtering and facilitation.

1 Introduction

The cuneate nucleus (CN) is located within the brain stem in the most rostral area of the spinal chord, between the Obex and 4 mm caudal to it. This zone is composed of a core or central region, and a shell or peripheral region. The core is basically made up of clusters of cuneothalamic neurones, also referred to as projection or relay neurones, which are surrounded by interneurons, also known as local neurons or non-thalamic projection neurones. The shell, comprising the ventral and marginal regions, is made up of interneurons and other projecting neurons. Cuneothalamic cells project their axons to the VPL nucleus whereas interneurons exert their influence over the cuneothalamic neurons as well as on other interneurons.

The CN receives four main types of afferent fibres: primary afferent fibres from the cutaneous and deep receptors, fibres coming from the brainstem, direct

corticocuneate fibres, and collaterals of the corticospinal tract originated in the cerebral cortex. The cutaneous afferent fibres are confined basically to the core or central region of the medial cuneate and mostly establish axodendritic synapses [4]. The descending direct and collaterals fibres can be grouped based on their origin within the sensorimotor cortex (SMC): those originating in Brodmann's areas 4 (motor cortex) and 3a (proprioceptive somatosensory cortex) contact the shell [3]; those from area 3b (cutaneous somatosensory cortex) project directly to the core; and those from areas 1 and 2 (cutaneous somatosensory cortex) project directly into the shell as well as into the core but within zones surrounding the cluster of cuneothalamic cells.

With regard to the effects of the descending fibres, several studies [7] found that the sensorimotor cortex either excites, blocks or does not affect the cell activity. Gordon and Jukes [5] studied the target cells of the descending fibres and found that cortical stimulation inhibits neurons in the medial zone. This inhibition, classically explained by means of a presynaptic mechanism, is still an open issue after recent experiments reporting evidences of postsynaptic rather than presynaptic inhibition [1]. On the other hand, it has been demonstrated that the sensorimotor cortex could directly excite cuneothalamic cells of the CN [2].

Our interest is focused on the role of the corticocuneate circuitry in sleep and wakefulness. About wakefulness, it was proposed that the SMC could control the information flow through the CN in two complementary ways [11]: (1) facilitating the transmission of signals originated in those peripheral areas which either processes a high intensity stimulus or they are inside the cortical attentional window; and (2) filtering irrelevant information. About sleep, recent findings show that cortical descending fibres play an important role by imposing oscillatory rhythms in the CN [9]. These rhythms, intensively studied in the thalamus [10], are supposed to be in charge of blocking the ascending transmission at subcortical level.

In this paper we propose a plausible corticocuneate circuitry and study how it might work in both sleep and awaking conditions. The presentation is organized as follows: (1) a brief description of experimental and computational methods, (2) the introduction of the corticocuneate circuitry, (3) computer experiments under sleep conditions, (4) computer experiments under awaking conditions, and (4) a final discussion.

2 Methods

2.1 Experimental

All experiments conformed to Spanish guidelines (BOE 67/1998) and European Communities council directive (86/609/EEC). Moreover, all efforts were made to minimize the animals used. Data were obtained from cats (2.3-4.5 kg), which were anesthetized (α -cloralosa, 60 mg/Kg), i.p.), paralyzed (Pavulon, 1 mg/ kg/h, i.v) and placed under artificial respiration. A set of six bipolar stimulating electrodes

was mounted in a tower and lowered to 1-1.5 mm deep in the pericruciate cortex to stimulate corticocuneate neurons. The cortical electrodes were disposed so that primary motor and primary somatosensory cortices could be stimulated. A bipolar concentric electrode and sharp electrodes were used to record the electrocorticogram (EcoG) and to record intracellularly from cuneate neurons, respectively. All cuneolemniscal cells satisfied the collision test with spontaneous action potentials.

2.2 Computational

Neuron level The multi-compartmental approach was used to model the neurons. A detailed description of the compartments and the ionic currents used for each neuron can be found in Sánchez et al. [13].

Membrane level All ionic currents are described by following the Hodgkin-Huxley model. The mathematical description of all this currents can be found in Sánchez et al. [14].

Synaptic level To describe the conductance associated with the synaptic current we have used the *alpha* function [8]. The synaptic current is determined by $I_{syn}(t) = g_{syn}(t)(V - V_{syn})$, where V_{syn} denotes the resting potential of the postsynaptic membrane. In our model we introduce both excitatory and inhibitory connections. We assume the later being glutamatergic and the former being gabaergic. The excitatory connections are modeled by setting V_{syn} to zero and the inhibitory connections to a negative value.

Simulation tools We have used *Neuron* as it provides an optimised integration method developed by Hines [6]. The machine used for the simulations was a PC with a 600 Mhz AMD Athlon processor.

3 Results

3.1 Corticocuneate circuitry

Experiments were performed in intact-brain preparations in anaesthetized cats. Simultaneous recordings in the sensorimotor cortex (SMC) and the cuneate nucleus (CN) were obtained while the SMC was stimulated with 1-10 Hz electrical pulses, whereas intracellular recordings were prepared to study the effects of SMC stimulation over CN cells. Figure 1(A) shows selected recordings providing experimental evidence of excitation and inhibition of CN cells. According to these data, a circuit is proposed (figure 1(B)) and a computational model is developed in order to explain the observed effects. For the CN neurons we have used realistic computational models [12, 14] that were constructed based on data from current injection experiments and validated against neuronal activity recorded *in vivo* in anaesthetized cats.

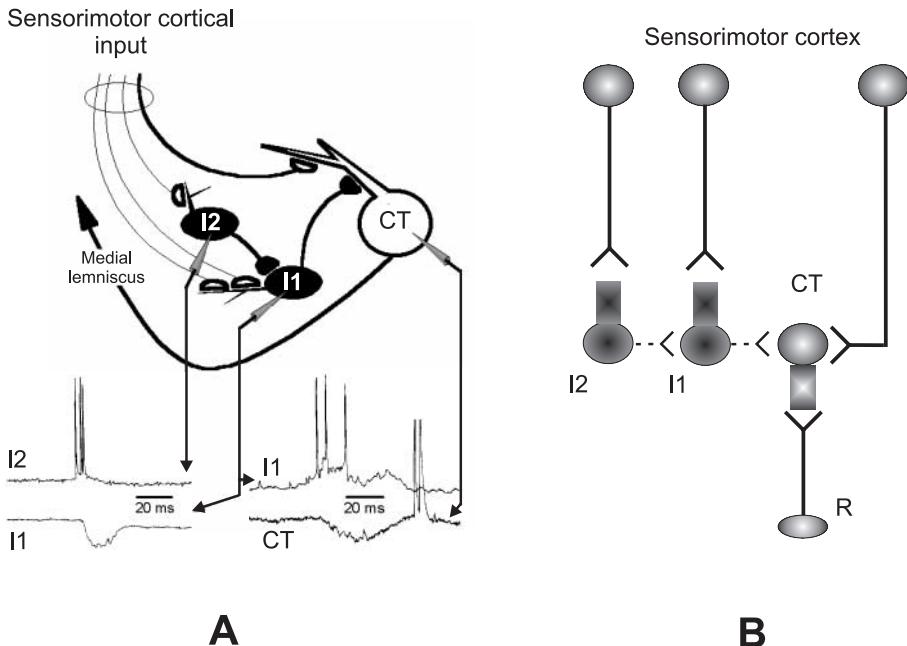


Fig. 1. The corticocuneate circuitry. (A) Intracellular recordings show excitation and inhibition effects over both cuneothalamic (CT) neurons and interneurons (I1 and I2). (B) The circuitry considered for simulation: cortical excitation and inhibition over interneurons, cortical inhibition over cuneothalamic neurons, and cortical excitation over cuneothalamic neurons.

3.2 Sleep state

The synapses of the computational model were adjusted by comparing simulated outcomes with experimental neuronal activity recorded in anaesthetized cats. The model is therefore tuned to sleep state conditions, and ready to test whether the corticocuneate circuitry can accomplish the hypothesized blockade of sensory information. Figure 2 shows a cuneothalamic neuron (CT) being inhibited after cortical stimulation. Initially, the inhibition hyperpolarizes the neuron and activates I_h and I_T currents, which leads to a burst of spikes and the beginning of a spindle-like rhythm. Next, we activate a sensory receptor to generate afferent signals. At the hyperpolarization subcycle, the firing threshold is so high that sensory signals can only drive a small membrane depolarisation. At the depolarised subcycle, the intrinsic-generated spikes completely mask the processing of afferent signals. Therefore, the sensory information do not alter the oscillatory pattern, and the signal is completely blocked.

But, how does the complete circuitry work? To analyze this point we have set a simulation with the whole circuitry depicted in figure 1. We have stimulated the cortical neurons that elicit inhibition (CI) and excitation (CE) over

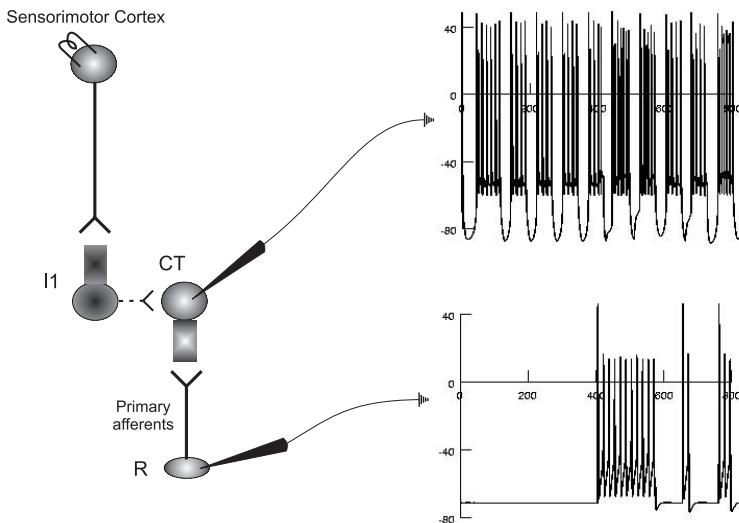


Fig. 2. Oscillatory activity and blockade of afferent information. The inhibition originated from the sensorimotor cortex elicits an oscillatory bursting response over the CT neuron (upper right). From 400 to 800 ms, cutaneous stimulation is generated with different temporal durations (lower right), but the oscillatory activity blocks the transmission of this information.

CT neurons (figure 3). We have also assumed an interconnection between CE and cortical neurons that inhibit interneurons (CD) [2]. The simulations indicate that the duration of both depolarized and hyperpolarized subcycles periods is longer if comparing them with the spontaneous activity in figure 3. The hyperpolarized subcycle duration is explained because we have simulated a low rhythm in the SMC, therefore the CT neurons are inhibited during a longer period of time. The depolarized subcycle is explained because of the excitatory cortical descending fibres providing excitation. In fact, we have found that this is the unique mechanism to accomplish an activity pattern similar to that observed *in vivo* [9]. So, we predict that the excitatory connections play a fundamental role in the shaping of the slow rhythms at the CN determining the duration of the spike subcycle.

3.3 Awaking state

And now we question the functioning of the corticocuneate circuitry in wakefulness. Considering the Mountcastle approach, the SMC would influence the CN to perform two main tasks: filtering of irrelevant or unwanted information, and facilitation of relevant stimuli. To model this behaviour we first need to consider what it means to be at wakefulness. In other words, what should we change in our model to satisfy the awaking conditions?. In the cuneate nucleus, brain stem

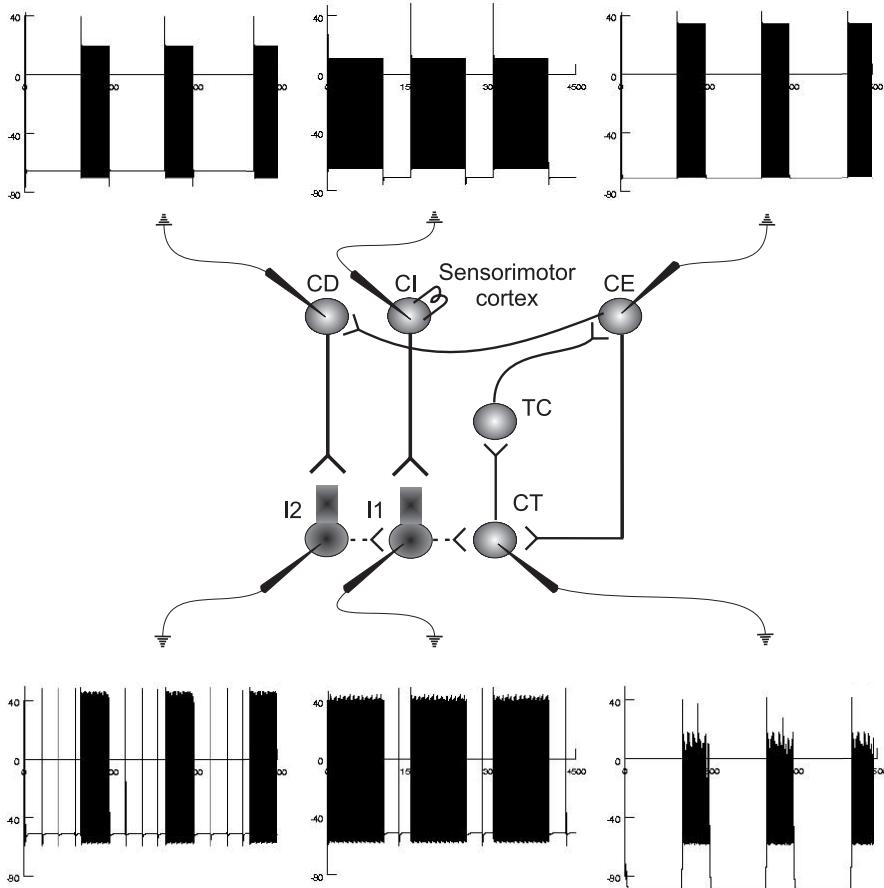


Fig. 3. Slow rhythms and the participation of excitatory synapses on their activity pattern. A sensorimotor cortical neuron is stimulated to induce a slow rhythm ($< 1\text{Hz}$) that spreads throughout the entire corticocuneate circuitry. As a result, the CT neuron shows an oscillatory activity very similar to that observed *in vivo*. The hyperpolarized subcycle is determined by the duration of inhibition carried out by I₁. The depolarised bursting subcycle is determined by the duration of the cortical excitatory effects maintained through the CT-TC loop.

or cortical afferents could induce similar effects to those observed in the thalamus [15] and be in charge of inducing the transition from sleep to wakefulness. We have simulated these effects by changing V_{syn} from -90 mV to -75 mV. In this way, we avoid the activation of I_h and I_T currents.

In figure 4 we show a circuitry that integrates the three descending pathways in charge of inhibition, excitation and disinhibition of cuneothalamic neurons. We have also connected, as proposed by Canedo et al. [2], the cortical neuron

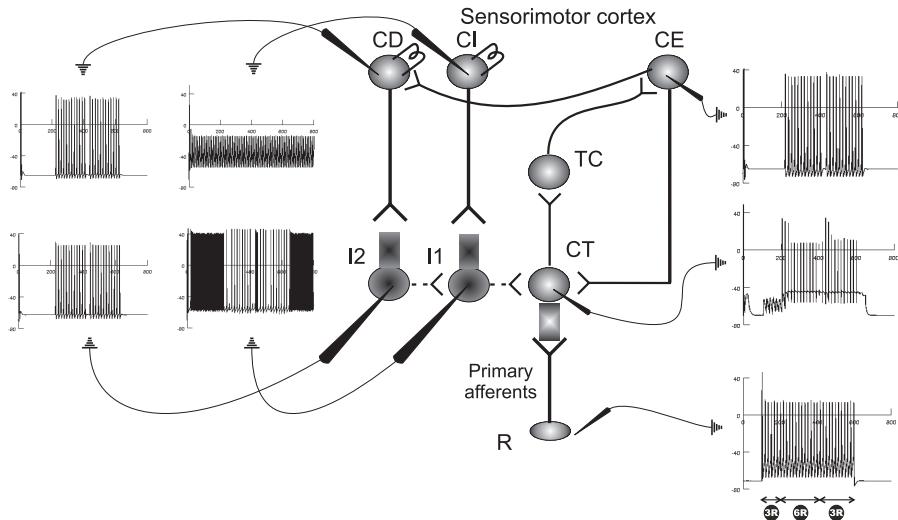


Fig. 4. Filtering and facilitation. Cutaneous stimulation of 3 receptors (R) cannot overcome the filtering originated from the sensorimotor cortex. We need 6 receptors (from 200 to 400 ms) to completely transmit the information. When we set back the number of activated receptors to 3, both cortical excitation and disinhibition facilitates the transmission of cutaneous information.

in charge of cuneate excitation with the other one in charge of cuneate disinhibition. The synapse between interneurons was set with $g_{syn}^{max} = 0.1$ and $\tau = 4$. Under activation of 3 receptors (0.005 nA) during 100-200 ms, there is no response in the cuneothalamic neuron because of the cortical inhibition imposed by CI. After increasing the activation to 6 receptors in the interval 200-400 ms, the cuneothalamic neuron overcomes the cortical filtering generating action potentials, and thus activating, through the thalamocortical (TC) neuron, the excitatory cortical neuron. This one activates the disinhibitory cortical neuron, which, in turn, activates the corresponding cuneate interneuron. The final effect is a partial inhibition of the interneuron I2, which decreases substantially its activity. The most salient aspect can be found in the interval from 400 to 600 ms, in which the activated receptors were reduced to 3. The cortical action facilitates the transmission of cutaneous information by maintaining a sustained depolarisation over the cuneothalamic neuron. The simulation finalizes, after 600 ms, when the stimulation is removed and the cuneothalamic neuron ceases its activity.

4 Discussion

The main contribution of this work is to show how the same circuitry can perform completely different tasks during sleep and wakefulness. These states impose gen-

eral conditions that determine not only the activity of the cuneate nucleus, but also the activity of the overall brain. We hope that the study presented here will encourage further research concerning how the circuits studied in anaesthetized or sleep conditions can perform tasks during wakefulness.

Acknowledgments

We would like to thank to **Laboratorios de Neurociencia y Computación neuronal (LANCON)**, the environment in which this work has been developed, and the support from the Xunta de Galicia through grant PGIDIT-02-TIC20601PR.

References

1. Canedo, A.: Primary motor cortex influences on the descending and ascending systems. *Progress in Neurobiology*. Vol. 51 (1997) 287–335
2. Canedo A., Aguilar J., Mariño J.: Lemniscal recurrent and transcortical influences on cuneate neurons. *Neuroscience*. Vol. 97 **2** (2000) 317–334
3. Cheema SS, Fyffe RE, Light A, Rustioni A: Arborizations of single corticofugal axons in the feline cuneate nucleus stained by iontophoretic injection of horseradish peroxidase. *Brain research* Vol. 290. (1984) 158–164
4. Fyffe Robert E., Cheema Surindar S., Rustioni A.: Intracellular Staining Study of the Feline Cuneate Nucleus. I. Terminal Patterns of Primary Afferent Fibers. *Journal of Neurophysiology*. Vol. 56. **5** (1986) 1268–1283
5. Gordon G, Jukes MGM: Descending influences on the exteroceptive organizations of the cat's gracile nucleus. *J. Physiol. (London)* Vol. 173 (1964) 291-319
6. Hines M.: A program for simulation of nerve equations with branching geometries. *International Journal of Biomedical Computation*. Vol. 24 (1989) 55–68
7. Jabbur SJ, Towe AL: Cortical excitation of neurons in dorsal column nuclei of cat. *J. Neurophysiology* Vol. 24 499–509
8. Jack J. J. B., Redman S. J.: The propagation of transient potentials in some linear cable structures. *J. of Physiology (London)*. Vol. 215 (1971) 283–320
9. Mariño J, Canedo A, Aguilar J: Sensorimotor cortical influences on cuneate nucleus rhythmic activity in the anesthetized cat. *Neuroscience* Vol. 95 **3** (2000) 657–673
10. McCormick DA, Bal T: Sleep and arousal: Thalamocortical mechanisms *Annual Review Neuroscience* Vol. 20 (1997) 185–215
11. Mountcastle VB, Darian-Smith I: Neural mechanisms in somesthesia In *Medical Physiology*. Mosby, St. Louis. Mountcastle VB (Ed). (1968) 1372–1423
12. Sánchez E., Barro, S., Canedo, A., Martínez, L., Mariño, J.: A computational model of cuneate nucleus interneurons. *Eur. J. Neurosci.* Vol. 10. **10** (1998) 402
13. Sánchez E, Barro S, Mariño J, Canedo A: A realistic computational model of the local circuitry of the cuneate nucleus In *Lecture Notes in Computer Science*. Volume I. Springer Verlag. Mira J. and Prieto A. (Eds). (2001) 21–29.
14. Sánchez E, Barro S, Mariño J, Canedo A: A computational model of cuneothalamic projection neurones. *Network: Computation in neural systems*. Vol. 14. (2003) 211-231
15. Steriade M, McCormick DA, Sejnowski TJ: Thalamocortical oscillations in the sleeping and aroused brain. *Science* Vol. 262. (1993) 679-685

Effects of different connectivity patterns in a model of cortical circuits

Carlos Aguirre, Doris Campos, Pedro Pascual, and Eduardo Serrano

GNB, Escuela Politécnica Superior, Universidad Autonoma de Madrid,
28049 Madrid, Spain

{Carlos.Aguirre, Doris.Campos, Pedro.Pascual, Eduardo.Serrano}@ii.uam.es

Abstract. Cortical circuits are usually modeled as a network of excitatory and inhibitory neurons with a completely regular or a random connectivity pattern. However, neuroanatomy of the macaque and the cat cortex shows that cortical neurons are organized into densely linked groups that are sparsely and reciprocally interconnected. Interesting properties arise in the average activity of an ensemble of cortical neurons when the topology of the network itself is an intrinsic parameter of the model that can vary with a given set of rules. In this work we show that both the temporal activity and the encoded rhythms in an ensemble of cortical neurons depend on the topology of the network.

1 Introduction

Graph theory [1] provides the most adequate theoretical framework in order to characterize the anatomical connectivity of a biological neural network. The representation of neural networks as graphs allows a complete structural description of the network and the comparison with different known connection patterns. The application of graph theory to modeling neural networks appears in theoretical neuroanatomy for the analysis of the functional connectivity in the cerebral cortex. In [2] it is shown that the connection matrices based on neuroanatomical data that describes the macaque visual cortex and the cat cortex, present structural characteristics that coincide best with graphs whose units are organized in densely linked groups that were sparsely but reciprocally interconnected. These kind of networks also provide the best support for the dynamics and high complexity measures that characterize functional connectivity.

There are some well known biological neural networks [3, 4] that present a clear clustering in their neurons but have small distances between each pair of neurons. These kind of highly clusterized, highly interconnected sparse networks are known as Small-World (SW) networks. SW topologies appear in many real life networks [6, 7], as a result of natural evolution [4] or a learning process [8]. In [5] it is shown that on SW networks coherent oscillations and temporal coding can coexist in synergy in a fast time scale on a set of coupled neurons.

Cortical circuits are usually modeled as a network of excitatory and inhibitory neurons [9] with a completely regular or a random connectivity pattern [10, 11].

However, as shown in [13], the connection substrate can have a major effect on the dynamical behavior of the elements in a network, even when these elements are highly synchronized. This means that it is necessary to consider the topology of the network as an additional parameter that can be modified. In this paper we study the different behavior of the average activity in a model of cortical circuits as a function of the network topology.

2 Network Models

We explore three topological models: a regular grid, a random network and a SW network.

In [15] a method to study the dynamic behavior of networks when a network is shifted from a regular, ordered network to a random one is proposed. The method is based on the random rewiring with a fixed probability p for every edge in the graph. We obtain the original regular graph for $p = 0$, and a random graph for $p = 1$. This method shows that the characteristic path length L (the average distance, measured as the minimal path length between them, between nodes) decreases with the increasing value of p much more rapidly than the clustering coefficient C (the average number of neighbors of each node that are neighbors between them) does. There is a range of values of p where paths are short but the graph is highly clustered, this range of values of p is known as the small-world area.

In this paper we take the architecture of the regular model as a directed and weighted grid where nodes connect to the two closest neighbors that are in each of the four possible directions in the grid. We take $p = 0.06$ for the generation of the SW network as this probability ensures the maximum distance between the value of C and L when the original substrate is a regular grid [14]. For the random model we have followed the same rewiring procedure with $p = 1$. In Fig. 1 we show the connectivity pattern of the cells in the regular, SW and random networks.

The values of L and C for the three models of graphs can be seen in Table 1. Random graphs present a low L and C while the regular grid present a high L and a high C . The SW model have L values similar to the values of random networks, but C values closer to those of regular networks.

	L	C
REGULAR GRID	13.005	0.214
SW	5.498	0.180
RANDOM	4.021	0.002

Table 1. Values of L and C for different graphs models. In the three models the number of nodes is 2500 and the average number of neighbors per node is 8.

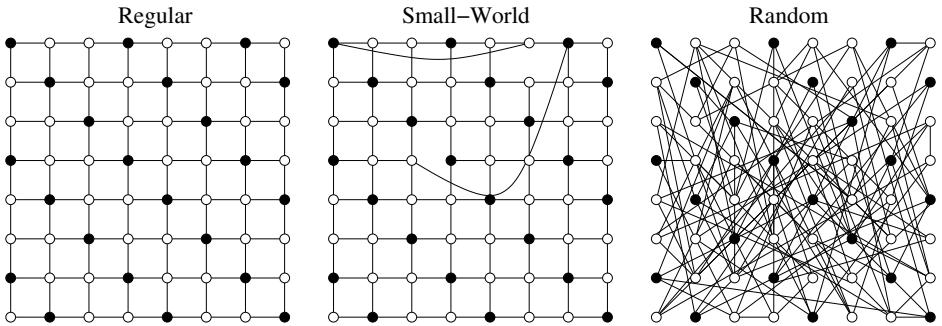


Fig. 1. Regular, SW, and random networks. Empty circles represent excitatory neurons. Filled circles represent inhibitory neurons. The three networks are composed of 64 nodes and each node is connected in average to 4 neighbors.

As shown in [15] synchronization in a network of automata is favored by low values of L due to the efficiency in the transmission of information when short paths are present in the network. On the other hand, high values of C tend to favor local effects. This makes us expect that in the SW model, encoded rhythms that come both from global synchronization and local effects [18] can coexist .

3 Neuron Model

In this work we will consider a network of simple integrate and fire (IF) neurons. This model of neuron is often utilized in simulation studies. Simple IF neurons have been shown to provide good approximations to the dynamics of more complex model neurons [12]. We consider a network of N neurons, $N^1 = 0.75N$ of them excitatory and $N^2 = 0.25N$ of them inhibitory. We name the population of excitatory neurons as population 1 and the population of inhibitory neurons as population 2. Each neuron is connected in average to 8 neurons; this value ensures the sparsity of the network. Each neuron also receives an input from excitatory neurons outside of the network.

The connection between the i -th presynaptic neuron of population k and the j -th postsynaptic neuron of the l population will be denoted W_{ij}^{kl} where $k, l = 1, 2$. The connections provinient from neurons of the first population (excitatory) W_{ij}^{1l} , are positive meanwhile the connections provinient from neurons of the second population W_{ij}^{2l} are negative (see fig 2).

The state of each neuron is a binary variable σ . Each neuron can present two different states, 0 meaning a quiescent state or refractory state and 1 representing a fire state. The state of neuron j of population l at time $t+1$ comes determined by the following update rule.

$$\sigma_j^l(t+1) = \Theta(u_j^l(t)) \quad (1)$$

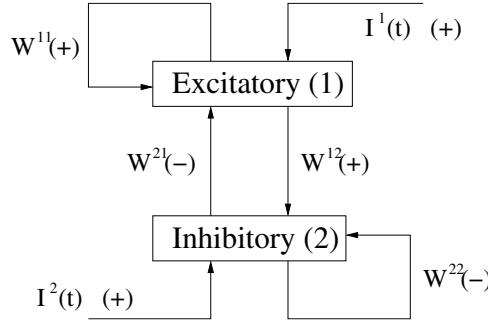


Fig. 2. Scheme of the network, the sign of each connection is noted between parentheses.

Where Θ is the usual Heaviside function

$$\Theta(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{otherwise} \end{cases} \quad (2)$$

The total synaptic input to neuron j of population l at time t , $u_j^l(t)$ comes determined by the following expression:

$$u_j^l(t) = \left(\sum_{k=1}^2 \sum_{i=1}^{N^k} W_{ij}^{kl} \sigma_i^k(t) \right) + I_j^l(t) + \theta^l \quad (3)$$

Where $I(t)$ represents the external input and θ represents a bias. The state of each neuron is calculated synchronously at fixed time intervals τ^l for each population l . In this work we assume $\tau^1 = \tau^2 = 1$. Each time the neurons fires (i.e., $\sigma_j^l(t) = 1$) the neurons enters in a refractory state r^l . During the refractory state the neuron can't fire and its status σ is assumed as 0. In this work we will use $r^1 = 3$ and $r^2 = 11$ for excitatory and inhibitory neurons respectively, this values correspond with anatomical estimates for neocortex.

This model present a large parameter space, we choose $W_{ij}^{11} = 1$, $W_{ij}^{12} = 1$, $W_{ij}^{21} = -2$ and $W_{ij}^{22} = -1.8$. The bias will be taken as $\theta^1 = -1$ $\theta^2 = -0.7$.

The expression for the external input is:

$$I_i^k(t) = \begin{cases} 2.26E^k & \text{if } i \in S \text{ and } t < 100 \\ 0.56E^k & \text{otherwise} \end{cases} \quad (4)$$

where S is a set of 400 neurons (300 excitatory and 100 inhibitory) and $E^1 = 1$ and $E^2 = 0.8$; i.e., there is an initial overexcitation of a subset of the network. These values are selected as they correspond to the numerical values presented in [10] and [11].

4 Experimental results

In order to detect the oscillatory activity of the network, we study for each population of neurons, both the activity of the network at time t , $a^k(t) = \left(\sum_{i=1}^{N^k} \sigma_i^k(t) \right) / N$ where N is the total number of cells in the network and the modulus of the Fourier transform of the activity $|\hat{a}^k|^{1/2}(f)$ $k = 1, 2$, where f represents the frequency.

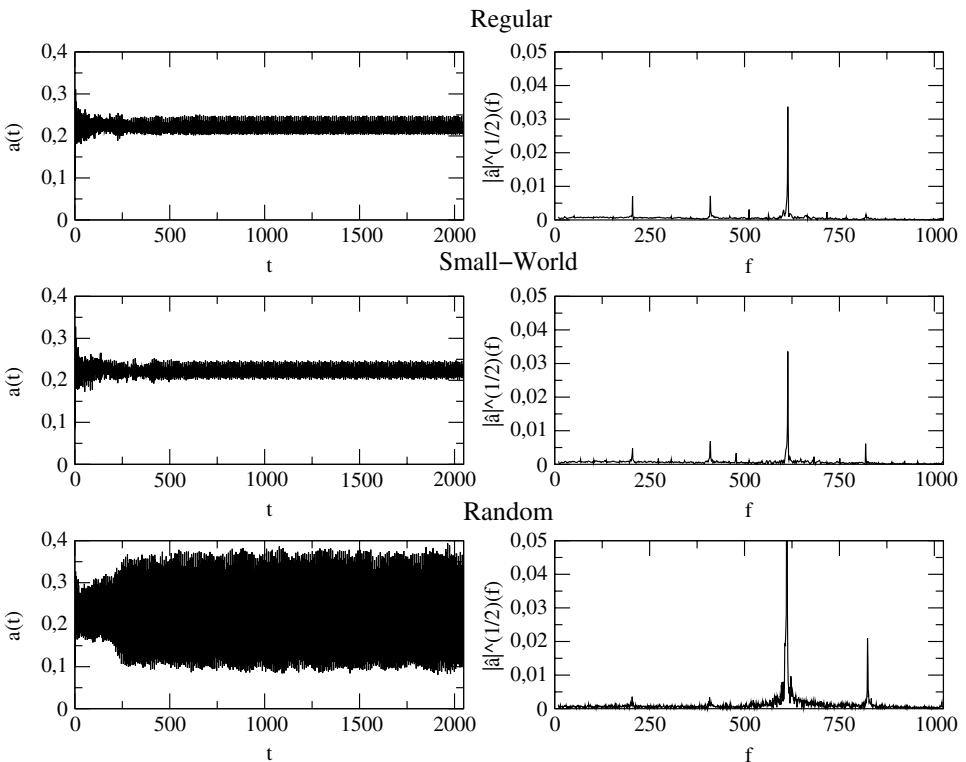


Fig. 3. Temporal activity (left) and power spectrum (right) of the set of excitatory neurons for the regular (top), SW (middle) and random (bottom) models

In figure 3 the network activity of the set of excitatory neurons and its Fourier transform are plotted. In the three models the temporal average activity of the network $\sum_{t=1}^T a^i(t)/T$ has the same value 0.22, where T is the simulation time limit; this average value coincides with the one obtained in [10]. However, the standard deviation of the average activity varies sensibly in the case of the random network. Regular and Small-World network present an standard deviation of 0.01 indicating a low level of overall synchronization meanwhile in the random model the standard deviation is 0.09.

The temporal average activity in the random model present a value of .1, indicating a higher synchronization in the network. This can be clearly seen when we consider the Fourier transform of the average activity. The random network has the strongest components of the Fourier transform values in the high frequency area, loosing the low frequency components. The regular network, on the other side, has the strongest components of the Fourier transform only in the low frequency area. The SW model has frequency components both in the high and low frequencies area. This behavior is due to the fact that the low frequency oscillations come from both local interactions between neurons [18] and the frequency associated to the refractory time of the inhibitory neurons, while high frequency components are favored by the overall synchronization of the excitatory neurons in the network. The SW network allows both local and global interaction. The highest component in frequency for the three models corresponds to a frequency over 612Hz that best corresponds to the firing period of the excitatory neurons, which is closely related to their refractory time.

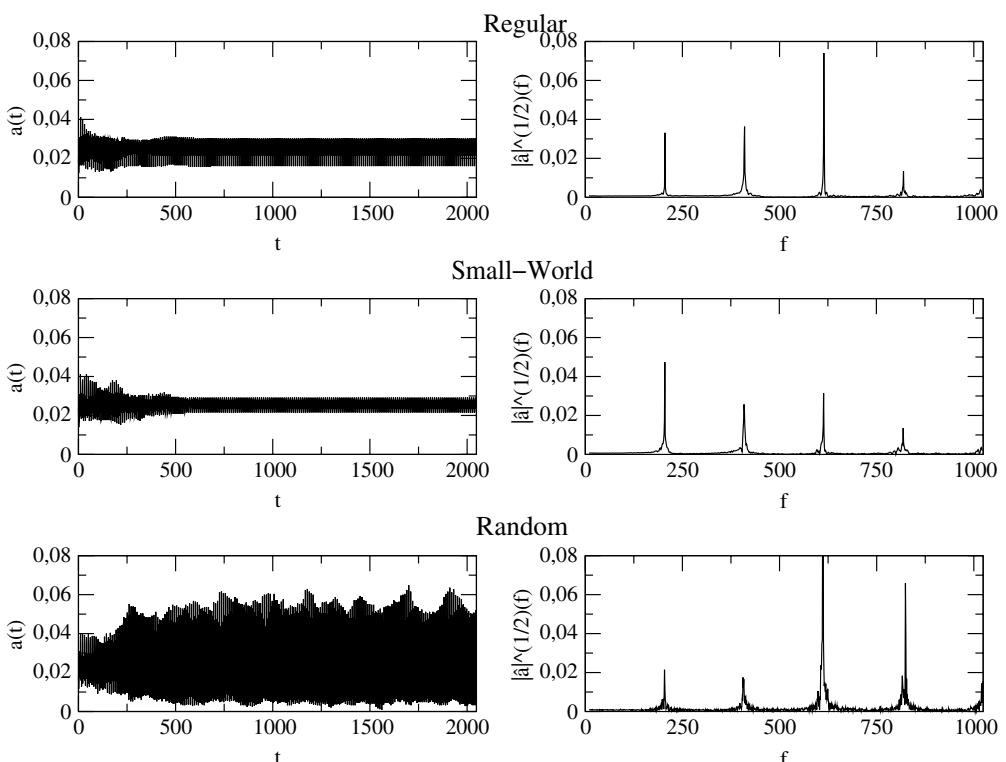


Fig. 4. Average activity (left) and power spectrum (right) of the set of inhibitory neurons for the regular (top), SW (middle) and random (bottom) models

In figure 4 the activity of the set of inhibitory neurons and its Fourier transform is plotted. The three models present again the same overall average activity 0.02 for inhibitory neurons, and the highest standard deviation 0.02 corresponds to the random model being very close to 0 in the case of the regular and random networks.

In the case of inhibitory neurons, both high and low frequencies are present in the three topological models. In the random model the larger components of the Fourier transform correspond again to high frequencies, with the maximum corresponding to the period of the excitatory neurons. In the regular case, frequencies are concentrated in the low frequencies area, and the maximum value of the Fourier transform corresponds to the frequency associated to the excitatory neurons. In the SW there exist similar values for the high and low frequencies; however, the maximum frequency component corresponds to the firing period of the inhibitory neurons.

5 Conclusions

The previous results allow us to establish the following conclusions.

- The selected topology affects the dynamics of an ensemble of excitatory-inhibitory neurons.
- The overall activity of the network does not depend on the selected topology, while the temporal behavior does.
- The regular model tends to favor low frequency oscillations in the network.
- The random model tends to synchronize the network in a frequency corresponding to the period of the excitatory neurons.
- The SW model presents a good balance between low and high frequency oscillations.

We thank the Ministerio de Ciencia y Tecnología (BFI 2000-015). (PP) and (CA) are partially supported by BFM2002-02359. (PP) and (CA) also receive a partial support by POCTI/MAT/40706/2001. (ES) receive a partial support by (TIC 2002-572-C02-02).

References

1. Chartrand, G.: *Introductory Graph Theory*. Dover Publications, Mineola New York (1985).
2. Sporns, O., Tononi, G., Edelman, G.M.: Relating Anatomical and Functional Connectivity in Graphs and Cortical Connection Matrices. *Cerebral Cortex*, Vol. 10. Oxford University Press, New York (2000) 127–141
3. White, J.G., Southgate, E., Thompson, J.N., Brenner,S.: The structure of the nervous system of the nematode *Caenorhabditis elegans*. *Philosophical Transactions of the Royal Society of London. Series B* **314** (1986) 1–340
4. Achacoso, T.B., Yamamoto, W.S.: AY's Neuroanatomy of *C.elegans* for Computation. CRC Press, Boca Raton Florida (1992).

5. Lago L.F. Huerta R. Corbacho F. and Siguenza J.A. Fast Response and Temporal Coding on Coherent Oscillations in Small-world Networks, *Physical Review Letters*, **84** (12) (2000), 2758-2761.
6. A. G. Phadke A.G. and Thorp J.S.: Computer Relaying for Power systems Wiley, New York, (1988).
7. Milgram S.: The Small World Problem, *Psychology today*, **2** (1967), 60-67.
8. Araújo T. and Vilela Mendes R.: Function and Form in Networks of Interacting Agents, *Complex Systems* **12** (2000) 357-378.
9. Adini Y., Sagi D. and Tsodyks M. Excitatory-Inhibitory Network in the Visual Cortex: Psychophysical Evidence *Proceedings of the National Academy of Sciences USA*, **94** (1997) 10426-10431.
10. van Vreeswijk C. and Sompolinsky H., Chaotic Balanced State in a Model of Cortical Circuits *Neural Comp.* **10** (1998) 1321-1372.
11. Brunel, N., Dynamics of Sparsely Connected Networks of Excitatory and Inhibitory Spiking Neurons *Jour. of Computational Neuroscience* **8** (2000) 183-208.
12. Bernarder O, Koch C, Usher M, Synaptic background activity determines spatio-temporal integration in single pyramidal cells. *Proceedings of the National Academy of Sciences USA*, **88** (1991) 11569-11573.
13. C. Aguirre, F. Corbacho, R. Huerta, A realistic substrate for Small-world networks modeling: *Proceedings of the 12th International Workshop on Database and Expert Systems Applications*, IEEE Computer Society (2001).
14. C. Aguirre, R. Huerta, F. Corbacho P. Pascual, Analysis of biologically inspired Small-World networks: *Artificial Networks-ICANN 2002*, Lecture Notes in Computer Science, Springer (2002) 27-32.
15. Watts, D.J.: *Small Worlds: The dynamic of Networks between Order and Randomness*, Princeton University Press, Princeton, New Jersey (1999).
16. Bollobas, B.: *Random Graphs*. Harcourt Brace Jovanovich, Orlando Florida (1985).
17. Watts, D.J., Strogatz, S. H. Collective dynamics of small-world networks, *Nature*. **393** (1998) 440.
18. Compte A., Sanchez-Vives M. V., McCormick D. A. and Wang X. J., Cellular and network mechanism of slow oscillatory activity in a cortical network model. *Journal of Neurophysiology*, In press (2003)

A Digital Neural Model of Visual Deficits in Parkinson's Disease

Igor Aleksander¹ and Helen Morton²

¹ Intelligent and Interactive Systems Group, Imperial College, London SW7 2BT, UK
 {i.aleksander@imperial.ac.uk}
Human Sciences Department, Brunel University, London UB8 3PH, UK
 {helen.morton@brunel.ac.uk}

Abstract. Visual deficits have been discovered in sufferers of Parkinson's disease [1] and the cause of this is not clearly understood. This paper reports on a digital neuromodel that investigates a hypothesis that the deficit may be due to a projection from the Basal Ganglia to the Superior Colliculus where a shortage of Dopamine introduces noise in the oculo-motor loop. New experiments were done with Parkinson's patients to track the deficit and the neuromodel predicts performance against oculo-motor noise. It is seen that a group Parkinson's sufferers in whom the deficit is pronounced follow the predicted law, while controls with poor performance do not follow the law. This helps to uphold the noise hypothesis.

1 Introduction

Much of the work that leads to the conclusion that Parkinson's sufferers have a visual deficit [1] is based on a visual planning problem (Tower of London or TOL) where the participant has to visualise the rearrangement of coloured spheres from the way they are drawn in a workspace to the way they are rearranged in a given target. Parkinson's patients have demonstrated a significant depression in their performance over controls. These tests have also shown, through measurement with an eye tracker, that Parkinson's Disease (PD) sufferers move their eyes in a much less purposeful way when they perform this experiment. The authors of this paper, under a project funded by the Wellcome Trust of the UK, set out to ask whether the visual memory deficit is likely to be caused by a direct influence of the basal ganglia on the oculo-motor system through a reduction of inhibition at the superior colliculus. The challenge present in this work is whether the neuromodel could display on a screen 'what it is like to have this visual deficit?' The reduction of dopamine has the effect of increasing noise in the oculo-motor loop. To answer the question we first designed a test that was directly apposite to the linking of visual memory performance to eye movement (without the complication of planning as in TOL). This led to the discovery that visual memory seemed defective in a very specific group of PD sufferers. Here we briefly present the results of these tests, and concentrate on the neuromodel that incorporates the effect of the basal ganglia. We show that the introduction of noise in the system creates a locus of performance which is more closely followed by the PD sufferers and not as much by badly performing controls.

2 Visual Memory Test

The type of patterns used in the test is shown in fig. 1.

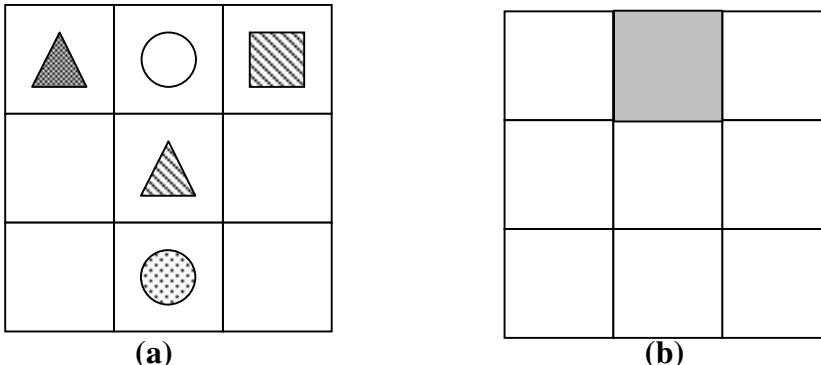


Fig. 1. (a) Typical grid of objects shown to the participant (the different shadings represent one of four primary colours). (b) Typical ‘question’ pattern that requires the participant to remember what was the shape and colour of the object in that position.

The object was shown to participants under a full regime of exposure times and strategies for variation. After allowing observation of the pattern for a prescribed time, the ‘question’ pattern is presented and the participant is asked to recall the colour and shape that were present in that position. The participant is then shown yet another layout where the entire set of objects is present and he/she is required to pick out which object has been presented. A performance figure was obtained that incorporated both the accuracy of recall and the time required to find the desired match in the final part of the test.

A large portfolio of results (20 sufferers and 20 controls) has been normalised for age, Intelligence Quotient, disease severity and duration. Comparisons between controls and sufferers showed that 20% of PD participants (4 subjects) had performance deficits that were entirely outside of the range of control performances. The neural modelling described in this paper was designed to explain the mechanisms behind the deficits of these four sufferers.

3 A Computational Neuromodel

The history of the style of digital neuromodelling used in this work may be traced in the literature relating to our work [2], [3], [4]. The prime principle is to capture the *architectural* details of the brain that are thought to be responsible for the cognitive phenomenon that is being studied. The neurons are G-RAM types the history of which has been summarised recently [5].

3.1 The model of visual reconstruction

Figure 2 shows the essential parts of the model

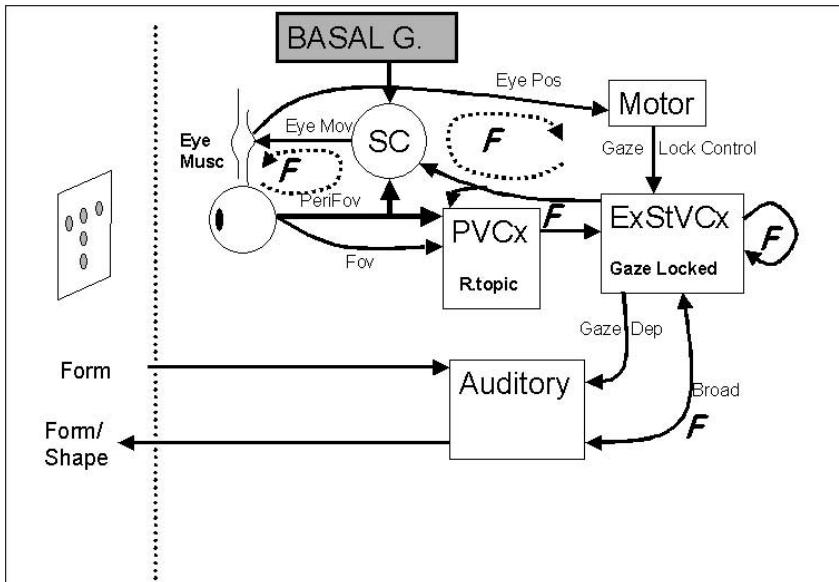


Fig. 2: Broad sketch of the neuromodel adopted showing a Superior Colliculus (SC) the Primary Visual Cortex (PVCx), the Extrastriate Visual Cortex (ExStVCx), Motor and Auditory areas. The point at which the Basal Ganglia have an inhibitory input is shown.

As the central hypothesis of this work involves the effect of dopamine deficit on eye movement, testing of this system involves the same sequence as the test on human participants including the eye movements needed in every phase of the look-recall-select cycle. The process starts with a presentation of the pattern to be remembered as shown fig. 3. The model possesses a foveal region and a perifoveal region as shown. The superior colliculus uses the perifoveal information to drive the fovea to areas of interest in the image. The driving muscles of the eye feed signals to the motor areas which project into the extra-striate where gaze-locked cells are known to exist [6]. This enables us to display on the screen the reconstructed sensation that is an integration of the foveal retinotopic images that reside in the model of the primary visual cortex. Important feedback loops are marked **F** and it is clear that the loop that links the eye muscle to the motor area, then to the extrastriate and back to the superior colliculus and eye movement relies for its stability on the inhibitory effect of the basal ganglia. This is the loop that we studied to understand the effect of noise on performance.

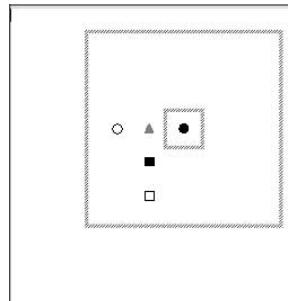


Fig. 3: The overall input field showing the inner foveal area centering on one object .

It is known that the role of the superior colliculus in the model is to drive the eye to areas of high spatial activity and the process of doing this repeatedly causes a reconstruction of the pattern in the extrastriate area [9]. Short term memory effects are thought to be at work here which, in the model are implemented as a noise-driven fade to black. So at any point in the process of looking at the pattern, the reconstruction is typically as shown in fig. 4 (a). This includes the effect of fading memory and shows that some objects are forgotten.



Fig. 4 : (a) shows a normal reconstruction while (b) shows a reconstruction disrupted by noise in the superior colliculus.

At an arbitrary point in time, the input image is removed and replaced by a target image that draws the fovea to a spatial position previously occupied by one of the coloured shapes. This causes the system to attend internally to the memory of one of the areas of fig. 4 (a). The final step is to show a picture of all possible shapes and colours on the screen and pick out one (by random shape-centering saccades) among the shapes that best compares with the memory of the shape resulting from the previous phase of the test. The correctness of the choice is logged. The fading parameters were adjusted to obtain performance distributions that most closely match those of the control group. As the simulation refers to the behaviour of a single individual, it has been tuned to and compared to the results of a member of the control group whose individual score is closest to the average of the control group. What follows is a description of the way that the lack of dopamine is thought to introduce noise that causes a lack of synchronisation between the reconstructive properties of the extrastriate region and world events.

3.2 Disruption of the model.

The key hypothesis here is that a possible cause of visual difficulties in PD is the disruption in the behaviour of the superior colliculus (SC) due to a faulty action of the projection from the basal ganglia. Normally such a projection is thought to be inhibitory which is conducive to stable behaviour in the ocular-motor loop in which the SC is active. A dopaminergic deficit, due to a reduction of inhibition, is thought to allow the loop to become less stable, which would allow random, uncontrolled action to take place. In the model this is introduced by means of injecting various amounts of noise to the function of the SC. Such noise was introduced into the SC model. This influences the mapping of the next attractor in the spatial map of the SC that determines a target for eye movement. Figure 4(b) shows the effects of disruption at the SC as can be seen in the model. But the disruption is not present only at the disruption phase of the experiment, but also when the model tries to match the attended memory to a full set of shapes. Fig. 5 shows how the superior colliculus mis-foveates on shapes, causing an increase in ‘finding’ time in this phase of the test.

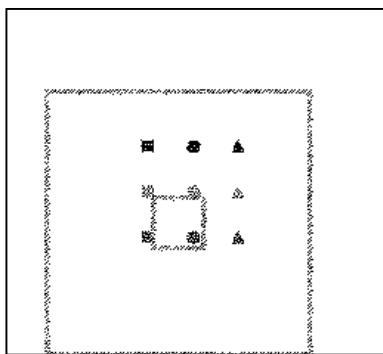


Fig. 5: Noise in the superior colliculus disrupts the foveating process increasing the finding time for a match.

Clearly, the failed reconstruction in visual memory leads to a depression of scored performance and the foveating failures during attending to the target and finding lead to an increase in the time required to find a target. A measurement of these two disruptions (normalised to be asymptotic to chance) is given below.

% Noise	Score	Finding time in secs.
0	24.3	40.5
2	18.5	50
4	15.6	65.4
10	13.9	95

Note, the time readings are included only so that they can be related to one another: they are not meant to compare dimensionally with human results. It is noticed that even small amounts of noise have a significant effect on performance.

3.3. Comparison with performance of PD participants

We first normalise the noise model scores obtained above

%Noise	Norm Score	Rel Time
0	2.4	1.0
2	1.9	1.2
4	1.6	1.6
10	1.4	2.4

On score, we take chance as having a value of 1 and any higher score as the actual score divided by the chance score. On time, we take 40.5 as a measure of the norm of value 1 and any higher value (divided by 40.5) as a measure of deficit. Figure 6 shows a plot of these modelled results We call this 'the SC Noise Law'.

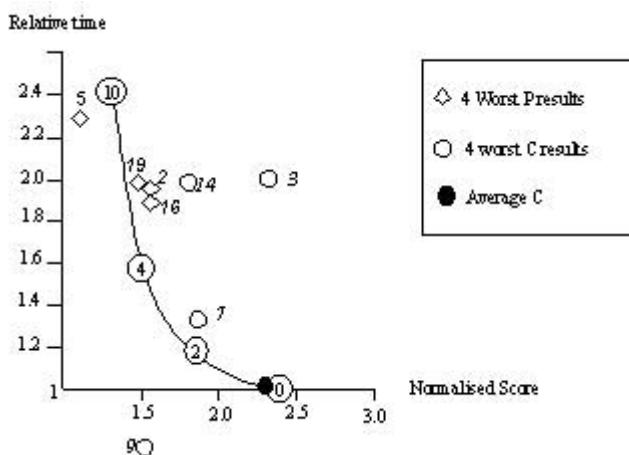


Fig. 6 : The Superior Colliculus Noise Law is indicated by the theoretical noise percentages 0,2,4 and 10. Superimposed on this are the normalised results for the worst affected PD results (P) and the 4 worst controls (C).

It is now possible to look at the results obtained with human participants as superimposed on this graph (Figure 6). We first note the position of the four worst P participants and their Time, Score coordinates ranked from participant no.5 (2.3,1.2) (worst) to participant no. 16 (1.9,1.6) (best of the four). On the same graph we superimpose the four worst Control participants in order from worst to best of four: 14 (2.0,1.8), 7 (1.3,1.8), 3 (2.0,2.4) 9 (0.4,1.5). Also we show an average Control participant. It is seen that the P results appear roughly to follow the upper reaches of

the SC Noise law, whereas the poor C measurements seem to have a greater scatter across the space.

4 Summary and Discussion

We recall that one of the headline questions in this project is “What is it like to have a visual deficit?” Also, this project is considered as a feasibility study to pronounce on whether digital neuromodelling helps to answer the question. We have reported on the design and deployment of a visual memory task which was used with PD participants and controls. The salient result was that among the PD population there were individuals whose response to tasks requiring visual memory was clearly affected in ways that were not found in the control population. We call these VAPD participants (Visually Affected PD). Within our sample 20% of the PD population were found to be VAPD.

Discussions with both PD participants and controls indicate that the task is conducted not entirely under full conscious control. That is, participants largely underestimate their ability to perform the task ‘feeling’ that they were guessing while their performance was actually significantly above chance. This is an interesting effect which merits further investigation. However, should the process be impaired as it appears to be in VAPD participants it means that this will not be ‘felt’ as a direct visual phenomenon, and will result in a lower performance of acting on visual clues without the participant necessarily being aware of this happening.

The modelling side of the work concentrated in simulating a hypothesised source of noise in the eye-movement control circuitry of this disadvantaged group. This corresponds to measured eye-movement disruptions on PD sufferers which show hesitation and inaccuracy in saccades which were also the results of noise injection in the model (Fig. 4, above). The model resulted in a locus (the SC Noise Law) of points in a Time/Score space. It was noted that the VAPD group fell within reach of this locus, while badly performing control participants were scattered across the space. We conclude that the noise hypothesis has some merit and should be investigated further both clinically and through modelling.

The ‘what it is like’ question may be discussed by looking at the difference between figure 4(a) and figure 4(b). If the hypothesis that the content of the artificial extrastriate region relates to what the subject may be visually aware of, then a normal level of awareness indicates a visual memory of the observed pattern has an acuity which is greater for recently foveated shapes in the overall pattern than for earlier ones. Interestingly, the mis-reconstruction in (b) is clearly responsible for a mismatch between the stimulus in figure 1(b) and the memory content of the extrastriate.

However, broadly speaking, the major result of this type of neural modelling is that it addresses those architectural issues that link brain function to inner sensation and the cognition that follows from this. This has instigated further work on basic issues such

as the hypothesis that eye movement is central to attentional processes in memory activity in the extra-striate cortex.

References

1. Hodgson TL, Bajwa A, Owen AM, Kennard C: *Cognitive Neuroscience*. **12**, 5, 894-907, 2000
2. Aleksander, I., Dunmall, B., Del Frate, V., Neurocomputational Models of Visualisation: A preliminary report, Proc. IWANN 99, Alicante – Spain (1999)
3. Aleksander, I., Dunmall, B., Seeing is Believing. Proc. IWANN 99, Granada – Spain (2001)
4. Aleksander, I. and Dunmall, B.: An extention to the hypothesis of the asynchrony of visual consciousness. Proc R Soc Lond B **267**, (2000) 197-200
5. Lockwood, G., and Aleksander, I., Predicting the behaviour of G-RAM Networks. Neural Networks, Volume 16, Issue 1, January 2003, Pages 91-100
6. Galletti, C. and Battaglini, P. P., Gaze-Dependent Visual Neurons in Area V3A of Monkey Prestriate Cortex. Journal of Neuroscience, **6**, (1989), 1112-1125

Intersensorial Summation as a Nonlinear Contribution to Cerebral Excitation

Isabel Gonzalo¹ and Miguel A. Porras²

¹ Departamento de Optica. Facultad de Ciencias Físicas.

Universidad Complutense de Madrid. Ciudad Universitaria s/n. 28040-Madrid. Spain

E-mail: igonzal@fis.ucm.es

² Departamento de Física Aplicada. ETSIM. Universidad Politécnica de Madrid.

Rios Rosas 21. 28003-Madrid. Spain

Abstract. Certain aspects of the J. Gonzalo's research on inverted vision and intersensorial summation (facilitation or reinforcement) in patients with brain damage, are formulated and interpreted from a macroscopic time-dispersive model of cerebral dynamics. We suggest that cerebral excitation from intersensorial summation is essentially nonlinear with stimuli.

1 Introduction

J. Gonzalo characterized the *central syndrome* associated to a unilateral lesion in the parieto-occipital cortex, equidistant from the visual, tactile and auditory projection areas (central lesion) [8]-[13]. A central lesion produces a deficit in the cerebral excitability, and a diminution in the reaction (response) velocity of the cerebral system. The corresponding central syndrome allows the dynamics of the cerebral cortex to be investigated since the cerebral system keeps the same organization plan and the same physiological laws as in the normal case, but in a smaller excitability scale [9]: *All* sensory systems are involved, in all their functions and with *symmetric* bilaterality, suffering an allometric *dissociation* or *desynchronization* of sensory qualities (united in normal perception) according to their excitability demands. In the visual system, for instance, when the illumination of a vertical white arrow is diminishing, the perception of the arrow is at first upright and well-defined; next the arrow is perceived to be more and more rotated, becoming at the same time smaller, and losing its form and colors in a well defined order. The sensorial perception thus splits into components. One of them is the direction function, which gives place to the striking phenomenon of the inverted vision: about 160 degrees in patient "M" under low stimulation [8, 9].

The investigations performed by J. Gonzalo were connected with those of other authors [7], [14]-[17], [22, 23, 25], taken into account in other works [1]-[4], [19]-[21] and can be related to other approaches in cerebral dynamics [5, 18, 24].

In a previous work [13], we introduced a linear time-dispersive model that describes some manifestations of the central syndrome, including temporal summation. Basic macroscopic concepts as the excitability and reaction velocity were

there introduced to characterize the cerebral system and a sensory function. In the present paper we deal with *intersensorial summation* (facilitation or reinforcement). This phenomenon occurs when the perception of a sensory function, stimulated by a certain stimulus S , is improved by other type of stimulus. Intersensorial summation is very noticeable in central syndrome. For example, strong muscular contraction improves significantly visual perception. Unlike summation by iteration, intersensorial summation modifies the cerebral system essentially, becoming more rapid and excitable, i.e., it supplies in part the neural mass lost in the central lesion [8, 9]. This effect is greater as the deficit (the central lesion) is greater, being null in a normal case. It is the aim of this work to show that intersensorial summation can be described as a nonlinear perturbation to the linear time-dispersion model introduced in [13].

2 The Model

We recall [13] that a system is said to be time-dispersive if its response at a time t depends not only on the stimulus at that time, but also at previous times, $t' \leq t$. If we consider an stimulus $S(t')$ acting on the cerebral system, the excitation $E(t)$ produced at time t in the cerebral system is

$$E(t) = \int_0^\infty \chi(\tau)S(t - \tau)d\tau , \quad (1)$$

where $\tau = t - t'$, and the excitation permeability χ is related to the capability of the system to be excited. An approximate form is $\chi(\tau) = \chi^{(0)}e^{-a\tau}$, which is the response to an impulse stimulus (a delta function stimulus). The constant a characterizes the response velocity of the system. Introducing the expression of $\chi(\tau)$ into (1), and assuming a constant stimulus S during the time interval $[0, t]$, (1) yields

$$E(t) = \chi^{(0)}S \int_0^t e^{-a\tau}d\tau = (\chi^{(0)}/a)S(1 - e^{-at}) . \quad (2)$$

Let us assume [13] that the threshold of the cerebral excitation E necessary to perceive the minimum sensation of a particular sensorial function is the same for the normal man and for a patient with central lesion. It was then showed [13] that the excitability $\chi^{(0)}$, the reaction velocity a and the quotient $\chi^{(0)}/a$ are smaller in a central lesion patient than in normal man, but the necessary stimulus S is higher [as can also be seen from (2)].

It can be said that the cerebral system of the normal man works like a saturated system, in the sense that a very low stimulus induces cerebral excitation enough to perceive not only the simplest sensorial functions but also the most complex ones in a synchronized way. A luminous sensation (simple function) of a white arrow, for instance, is perceived together with its color, localization, direction and recognition (more complex functions). In the case of central lesion, however, a low stimulus produces a cerebral excitation in deficit with respect to the normal man. The most complex, excitation-demanding sensorial functions, are then lost or retarded, leading to the dissociation phenomena.

We consider now intersensorial summation. In contrast to temporal summation, several stimuli act at one time on different receptors of the cortex. It was found [8, 9] that patients with central syndrome are very sensitive to intersensorial summation or facilitation. For example, strong muscular contraction is very efficient at improving the perception of all sensory systems, for example, it straightens instantly a test arrow perceived inclined, dilates the visual field, etc. Other types of facilitation come from binocular summation, tactile and acoustic stimuli. Facilitation phenomena are more noticeable as the deficit in the cerebral excitation is greater, and are null in a normal case[8, 9].

In more formal terms, the perception of a sensory function, say F_i , is not only determined by the stimulus S_i on the cerebral receptor i , but also by other stimuli S_j , $j \neq i$, acting on other cerebral receptors j . In our simple dispersive model, the total cerebral excitation E_i for sensorial function F_i is assumed to be the sum of different excitations of the type (1) originated by the different stimuli,

$$E_i(t) = \sum_{j=1,2,\dots} \int_0^\infty \chi_{i,j}(\tau) S_j(t-\tau) d\tau , \quad (3)$$

where, as above, $\chi_{i,j}(\tau) = \chi_{i,j}^{(0)} e^{-a_{i,j}\tau}$, and where $\chi_{i,j}^{(0)}$ is the excitability of the cerebral system associated to the sensory function F_i when the stimulus S_j is acting, and $a_{i,j}$ is the corresponding reaction velocity. In the case of constant stimuli S_j during the time interval $[0, t]$, (3) becomes

$$E_i(t) = \sum_{j=1,2,\dots} (\chi_{i,j}^{(0)} / a_{i,j}) S_j (1 - e^{-a_{i,j}t}) , \quad (4)$$

and in the stationary situation, in which all stimuli are acting for large time enough ($t \rightarrow \infty$) so that there is no changes in perception, (4) further simplifies to $E_i = \sum_{j=1,2,\dots} (\chi_{i,j}^{(0)} / a_{i,j}) S_j$, or in matrix form,

$$\begin{pmatrix} E_1 \\ E_2 \\ \vdots \\ \vdots \end{pmatrix} = \begin{pmatrix} \chi_{1,1}^{(0)} / a_{1,1} & \chi_{1,2}^{(0)} / a_{1,2} & \cdots \\ \chi_{2,1}^{(0)} / a_{2,1} & \chi_{2,2}^{(0)} / a_{2,2} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} S_1 \\ S_2 \\ \vdots \\ \vdots \end{pmatrix} . \quad (5)$$

The simplest possible case of intersensorial summation takes place when, in addition to stimulus S_1 , a secondary one S_2 supplies excitation enough to improve the perception of sensory function F_1 . (5) then reduces to

$$E_1 = (\chi_{1,1}^{(0)} / a_{1,1}) S_1 + (\chi_{1,2}^{(0)} / a_{1,2}) S_2 . \quad (6)$$

Given the peculiarities of the facilitation phenomenon, it is difficult to make hypotheses on the nature of the constants $\chi_{1,2}^{(0)}$ and $a_{1,2}$ that characterize this phenomenon. However, as in similar situations of several interacting subsystems, the usual way to proceed is to consider the excitation to be given by a single term of the form

$$E_1 = (\chi_{\text{eff}}^{(0)} / a_{\text{eff}}) S_1 , \quad (7)$$

where $\chi_{\text{eff}}^{(0)}$ and a_{eff} are effective excitability and reaction velocity parameters, respectively, that take into account the effects of the other stimuli. In fact, the intersensorial summation or facilitation appears to modify in some essential way the cerebral system, which becomes more excitable (characterized by large $\chi_{\text{eff}}^{(0)}$) and faster (large a_{eff}), as was shown in several experiences [8, 9, 13]. To write down an expression of $\chi_{\text{eff}}^{(0)}/a_{\text{eff}}$ that reflects the experimental observations [8, 9], we first note, from (7), that $\chi_{\text{eff}}^{(0)}/a_{\text{eff}}$ is the total excitation per unit stimulus S_1 . Second, this quotient has been shown in Ref. [13] to increase with the facilitation provided by the stimulus S_2 , that is, to approach that of a normal man. Thus, assuming a growth-type law for the restoration of the deficit of excitation, we can write, as a first approximation,

$$\chi_{\text{eff}}^{(0)}/a_{\text{eff}} = (\chi_{1,1}^{(0)}/a_{1,1}) + D(1 - e^{-\rho_{1,2}S_2}) , \quad (8)$$

where D is the excitation deficit per unit stimulus S_1 , due to the lesion, and $\rho_{1,2}$ is a constant. The functional form $1 - e^{-\rho S_2}$ accounts for saturation of $\chi_{\text{eff}}^{(0)}/a_{\text{eff}}$ at the highest value $\chi_{1,1}^{(0)}/a_{1,1} + D$ equal to the value $\chi_{1,1}^{(0)}/a_{1,1}$ of normal man, if facilitation is able to restore the excitation deficit asymptotically. On the other hand, proportionality to D in (8) accounts for the experimental fact [8] that restoration capability is greater as the deficit D is greater. Introducing (8) into (7) we obtain for the total stationary excitation

$$E_1 = S_1 \left[(\chi_{1,1}^{(0)}/a_{1,1}) + D(1 - e^{-\rho_{1,2}S_2}) \right] \simeq (\chi_{1,1}^{(0)}/a_{1,1})S_1 + D\rho_{1,2}S_1S_2 \quad (9)$$

due to stimuli S_1 and S_2 . The last linear approximation $[(1 - e^{-\rho_{1,2}S_2}) \simeq \rho S_2]$ holds for small restoration of excitation by stimulus S_2 , as is the case in the experiments [8, 9]. We note that the last equality in (9) is independent of the exact choice of the growth curve in (8) in the small restoration approximation. Finally, comparison of (9) and (6) leads to the approximate functional form for the intersensorial coefficient

$$\chi_{1,2}^{(0)}/a_{1,2} = D\rho_{1,2}S_1 . \quad (10)$$

The previous expressions fit to what is known from experimentation: 1) in normal man ($D = 0$) there is no facilitation ($\chi_{1,2}^{(0)}/a_{1,2} = 0$); 2) in absence of primary stimulus ($S_1 = 0$), facilitation cannot act ($\chi_{1,2}^{(0)}/a_{1,2} = 0$). As a third consequence, we see in (9) that facilitation is essentially a nonlinear phenomenon. From certain experiences [8], it seems that more complex nonlinearities could appear for high excitation, as a decrease of D and presumably of $\chi_{1,1}^{(0)}/a_{1,1}$ with increasing S_1 .

Finally we must consider the perception P_1 of the sensory function F_1 . Sensory perception is assumed in many cases to depend on the external physical stimulus I_1 according to a logarithmic-type law (Fechner law) in a certain range of values. This law has been much-discussed and other functional dependences have been proposed, but there is no a definitely accepted one [6, 22]. We will assume for simplicity the validity of this law, at least in a certain range. Since

the cerebral excitation E_1 is, as a first approximation, proportional to the cerebral stimulus S_1 , and this one can be considered proportional to the external physical stimulus I_1 , we shall consider, in the framework of the Fechner law, that the perception is related to the logarithm of the cerebral excitation E_1 in the form

$$P_1 = Z \log E_1, \quad (Z \text{ a constant}) . \quad (11)$$

3 Perception Curves

Let us analyze within the previous model some of the data of the visual system obtained by Gonzalo [8, 9] from patient M (very pronounced central syndrome case), right eye.

First we consider the case of no intersensorial summation (inactive case), analyzed in Ref. [13], to be compared with cases of intersensorial summation. An upright white 10 cm size arrow on black background, at distance of 40 cm from the patient was illuminated with light intensity I . The corresponding stimulus S_1 on the cerebral system is assumed to be proportional to the intensity $S_1 = KI$. For very low intensity the perception of the arrow was almost inverted and constricted, while for higher intensities the perception improves until the arrow is perceived almost upright. The reinversion of the arrow made by the cerebral system was called direction function, which reaches its maximum value (180 degrees) when the image is seen upright. Each value of the stimulus S_1 is maintained constant during long time enough until the perception of the arrow is stationary. The function fitted to the experimental data is, from (11),

$$P_1 = Z \log \left[(\chi_{1,1}^{(0)} / a_{1,1}) S_1 \right] = Z \log(YI) , \quad (12)$$

where $Z = 54$ and $Y \equiv (\chi_{1,1}^{(0)} / a_{1,1})K = 90$. The experimental data and the fitted curve (i) are shown in Fig. 1.

In the following case, we consider the same conditions as in the precedent case but with reinforcement by means of a second constant stimulus S_2 consisting on strong muscular contraction of the patient (40 kg held in each hand). Intersensorial summation improves now the perception of the arrow significantly. The function fitted to the experimental data is now

$$P_1 = Z \log \left[(\chi_{\text{eff}}^{(0)} / a_{\text{eff}}) S_1 \right] = Z \log(Y'I) , \quad (13)$$

with $Z = 40$ and $Y' \equiv (\chi_{\text{eff}}^{(0)} / a_{\text{eff}})K = 9000$, which is shown in Fig. 1[curve (r)] together with the data. Discrepancies between curves and data can be due to the controversial logarithmic Fechner law, to measurement errors (more pronounced for high excitation under reinforcement because the image becomes very unstable [8]), and to the assumption that $(\chi_{\text{eff}}^{(0)} / a_{\text{eff}})$ and $(\chi_{1,1}^{(0)} / a_{1,1})$ are independent of S_1 (only valid for not very high excitation values).

From comparison of cases (i) and (r) in Fig. 1, we see that facilitation supplied by S_2 (strong muscular contraction) allows the patient to perceive the arrow almost upright with much lower stimulus S_1 (lower I) than in the inactive case.

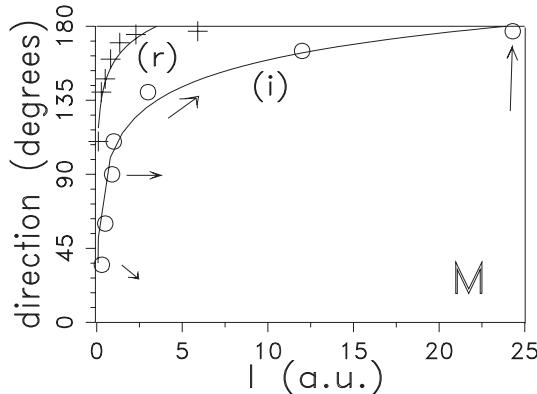


Fig. 1. Perceived direction of the upright test arrow versus light intensity for inactive case (i), and reinforced case (r) by muscular contraction (80 kg).

We analyze now the direction function versus facilitation for given stimulus S_1 . In this case the light intensity illuminating the arrow is constant (I and $S_1 = KI$ constants), while the stimulus S_2 (muscular contraction) varies. As in the preceding cases, the stimuli were acting during long time enough to reach an stationary state for each measurement. The cerebral excitation E_1 is given by (6), where $S_2 = CR$ (C constant) is now the variable and the remaining quantities take fixed values. The variable R is expressed in kg. For the sensorial perception of the direction of the arrow we then have, according to (11),

$$P_1 = Z \log(A + BR) , \quad (14)$$

where $A \equiv (\chi_{1,1}^{(0)} / a_{1,1})S_1$ and $B \equiv (\chi_{1,2}^{(0)} / a_{1,2})C = D\rho_{1,2}S_1C$. The fitting of this function to the data yields $Z = 78$, $A = 2.43$ and $B = 1.5$. It is shown in Fig. 2 (a).

The following example is the same as the previous one except that facilitation is not muscular contraction but luminous intensity I_l on the left eye, while the observation of the arrow is made, as above, with the right eye only [8]. In this case $S_2 = K_l I_l$ (K_l constant). The constant stimulus S_1 is now higher than in the previous case. The fitting of $P_1 = Z \log(A + BI_l)$ to the data gives $Z = 41$, $A = 100$ and $B \equiv (\chi_{1,2}^{(0)} / a_{1,2})K_l = D\rho_{1,2}S_1K_l = 80$. It is shown in Fig. 2(b). The values of A and B are higher than in the previous case in agreement with the higher value of S_1 during the experience.

4 Conclusions

We have extended our previous time-dispersion formulation for cerebral excitation to describe the effects of intersensorial summation. On the assumption that

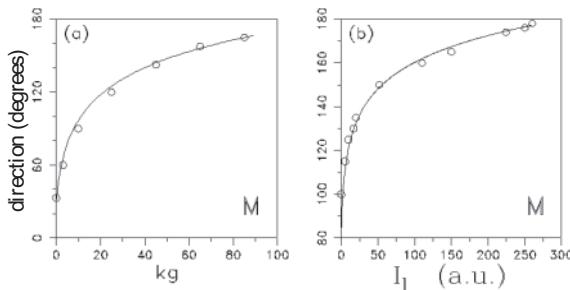


Fig. 2. Perceived direction of the test arrow versus facilitation (a) by muscular contraction , (b) by light intensity I_L on left eye. See the text.

cerebral excitation is additive, the contributions from intersensorial facilitation or reinforcement are intrinsically nonlinear. In fact, in the limit of low excitation, the contribution to the excitation E_i from stimulus S_i is linear, but the contribution from other stimuli take the quadratic form $\alpha S_i S_j$, where $\alpha = 0$ in normal man. This form is in agreement with the observed facts that intersensorial summation is more pronounced as the reinforcing stimulus S_j grows, but necessarily requires a primary stimulus S_i , and only acts on an in-deficit cerebrum due to a central lesion. The reasonably good agreement of the functional forms proposed with the experimental data would appear to support the model. Further observations made by J. Gonzalo indicate, however, that higher excitation levels could involve the onset of more complex, saturation-like nonlinearities with direct and crossed stimuli.

References

1. de Ajuriaguerra J. et Hécaen H.: *Le Cortex Cerebral. Etude Neuro-psychopathologique*, Masson, Paris 1949.
2. Bender M.B. and Teuber H.L.: "Neuro-ophthalmology" in: Progress in Neurology and Psychiatry, Ed. Spiegel E.A., **III**, Chap. 8, 163-182 (1948)
3. Critchley Mc.D.: *The Parietal lobes*, Arnold, London 1953.
4. Delgado García A.G.: *Modelos Neurocibernéticos de Dinámica Cerebral*, Ph.D. Thesis. E.T.S. de Ingenieros de Telecomunicación. U.P.M. Madrid 1978.
5. Engel A.K. et al.: "Temporal coding in the visual cortex: new vistas on integration in the nervous system", *TINS*, **15**, 6, 218-226 (1992).
6. Forgus R.H.: *Perception*, Mc. Graw-Hill, New York, 1966
7. Gelb A. and Goldstein K.: *Psychologische Analysen hirnpathologischer Fälle*. Barth, Leipzig 1920. "Psychologische Analysen hirnpathologischer Fälle auf Grund Untersuchungen Hirnverletzter: VII Ueber Gesichtsfeldbefunde bei abnormer Ermüdbarkeit des Auges (sog. Ringskotome)", Albrecht v. Graefes Arch. Ophthal., **109**, 387-403 (1922).
8. Gonzalo J.: *Investigaciones sobre la nueva dinámica cerebral. La actividad cerebral en función de las condiciones dinámicas de la excitabilidad nerviosa*, Publ. C.S.I.C.,

- Instituto S. Ramón y Cajal. Vol. I , 342 pp., Madrid 1945. Vol. II, 435 pp., Madrid 1950.
9. Gonzalo J.: "Las funciones cerebrales humanas según nuevos datos y bases fisiológicas. Una introducción a los estudios de Dinámica Cerebral", Trabajos del Instituto Cajal de Investigaciones Biológicas, C.S.I.C., Vol. XLIV, 95-157 (1952).
 10. Gonzalo I. and Gonzalo A.: "Functional gradients in cerebral dynamics: The J. Gonzalo theories of the sensorial cortex" in *Brain Processes, theories, and models. An international conference in honor of W.S. McCulloch 25 years after his death*, 78-87, Moreno-Díaz R. and Mira-Mira J. (Eds.), The MIT Press, Cambridge, Massachusetts 1996.
 11. Gonzalo I.: "Allometry in the Justo Gonzalo's Model of the Sensorial Cortex", Lecture Notes in Computer Science, **124**, Proceedings of the International Work-Conference on Artificial and Natural Neural Networks, Lanzarote, Canary Islands, Spain, June 1997, 169-177, Mira J., Moreno-Díaz R. and Cabestany J. (Eds.), Springer-Verlag, Berlin 1997.
 12. Gonzalo I.: "Spatial Inversion and Facilitation in the J. Gonzalo's Research of the Sensorial Cortex. Integrative Aspects", Lect. Not. in Comp. Sci. **1606**, Proc. Int. Work-Conf. on Artificial and Natural Neural Networks, Alicante, Spain, June 1999, Vol. I, 94-103, Mira J., and Sánchez-Andrés J.V. (Eds.), Springer-Verlag, Berlin 1999.
 13. Gonzalo I. and Porras M.A.: "Time-dispersive effects in the J. Gonzalo's research on cerebral dynamics", Lect. Not. in Comp. Sci., **2084**, Proc. Int. Work-Conf. on Artificial and Natural Neural Networks, Granada, Spain, June 2001p, Vol. I, 150-157, Mira J. and Prieto A. (Eds.), Springer, Berlin, 2001.
 14. Köhler W.: *Gestalt Psychology*, Liveright, New York 1929. *Dynamics in Psychology*, Liveright, New York 1940.
 15. Lapique L.: *L'excitabilité en Fonction du Temps*, Hermann et Cie., Paris 1926.
 16. Lashley K.S.: "Integrative functions of the cerebral cortex", Psychol. Rev., **13**, 1-42 (1933). "Functional determinants of cerebral localization", Arch. Neurol. Psychiat. (Chicago), **30**, 371-387 (1937). "The problem of cerebral organization in vision", Biol. Symp., **7**, 301-322 (1942).
 17. Luria A.R.: *Restoration of Function after Brain Injury*, Pergamon, Oxford 1963.
 18. Llinás R.R.: "The intrinsic electrophysiological properties of mammalian neurons: Insights into central nervous system function", Science, **242**, 1654-1664 (1988).
 19. Manjarrés A.: *Modelado Computacional de la Decisión Cooperativa: Perspectivas Simbólica y Conexionalista*, Ph.D. Thesis. Fac. de Ciencias UNED, Madrid 2001.
 20. Mira J., Delgado A.E. and Moreno-Díaz R.: "The fuzzy paradigm for knowledge representation in cerebral dynamics", Fuzzy Sets and Systems, **23**, 315-330 (1987).
 21. Mira J., Delgado A.E., Manjarrés A., Ros S. and Alvarez J.R.: "Cooperative processes at the symbolic level in cerebral dynamics: reliability and fault tolerance" in *Brain Processes, theories, and models. An international conference in honor of W.S. McCulloch 25 years after his death*, 244-255, Moreno-Díaz R. and Mira-Mira J. (Eds.), The MIT Press, Cambridge, Massachusetts 1996.
 22. Piéron H.: *La connaissance sensorielle et les problèmes de la vision*, Hermann, Paris 1936. "Physiologie de la vision" in *Traité d'ophtalmologie*, Masson, Paris 1939.
 23. Pötzl O.: "Die optisch-agnostischen Störungen" in *Handbuch der Psychiatrie*, Aschaffenburg, Wien 1928.
 24. Rakic D. and Singer W. (Eds): *Neurobiology of Neocortex*, Wiley, 1988.
 25. Ramón y Cajal S.: *Histología del sistema nervioso del hombre y de los vertebrados*, Vol. II, Madrid 1899.

Interacting Modalities through Functional Brain Modeling

Tino Lourens¹, Emilia Barakova², and Hiroshi Tsujino¹

¹ Honda Research Institute Japan Co., Ltd.
8-1 Honcho, Wako-shi, Saitama, 351-0114, Japan
{tino, tsujino}@jp.honda-ri.com
² Brain Science Institute, RIKEN
2-1 Hirosawa, Wako-shi, Saitama, 351-0198, Japan
emilia@brain.riken.go.jp

Abstract This paper proposes a concept for modeling modalities and understanding the interaction between modalities through functional brain modeling (FBM). FBM proves to be a powerful method for functional behavior prediction of a group of neuronal cells with equivalent functional behavior. An example of interacting groups of neuronal cells, utilizing FBM, in early vision is given. A broad setup of functional behavior and interaction between different groups of cells in early vision has similar conceptual properties as cells that process other sensory information or multi modal sensory information.

1 Introduction

Currently the chemical, biological, and functional structure and behavior of neurons are well understood. Nowadays desktop computing power also exceeds that of a fly's brain. Nevertheless there is no model or robotics system that is able to completely simulate its behavior. We propose to process sensory or cognitive data by functional brain modeling (FBM) to achieve better functional abilities than with existing simulations.

The aim of FBM is not to make a model of the brain on neural level from biological, electronic, chemical, and physical microscopic (molecular) level, but merely to model the functionality of a group of cells or even a complete area that have equivalent or similar functional behavior, and give the interaction between groups with different functional characteristics on macroscopic level.

FBM is aiming to reveal functional interaction mechanisms between different groups of neurons by simulation and predict behavior of cells in adjacent areas in the brain. Physical recordings of single (or a small number) of neurons, in practice, give a good insight in the functional behavior of a group of cells, but due to physical restrictions not all parts of the brain can be explored that way. Brain activity measurement methods, like EEG, fMRI, and PET, explain a lot about the interconnectivity of different areas in the brain, but are too crude to give detailed functional characteristics of a particular area or group of cells.

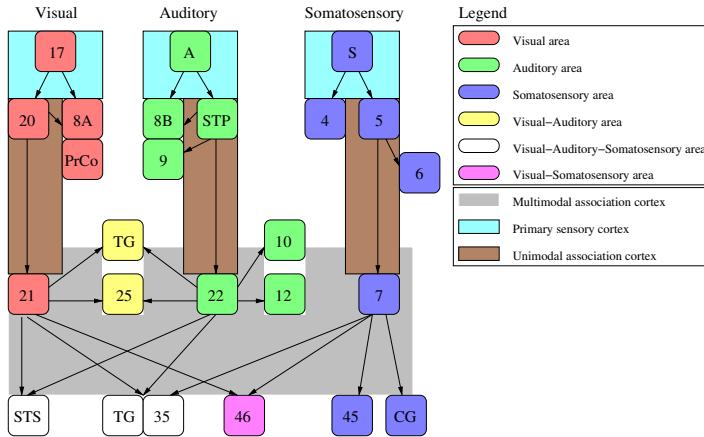


Figure 1. Schematic overview of visual, auditory, and somatosensory pathways. Adapted from [7]. Numbers and initials denote respective brain areas.

Functional brain simulations could be utilized to get a better insight in the behavior of certain brain areas.

We hypothesize that sensory data is processed, as illustrated in Figure 1, in a feedforward manner. Initially, the data in the stream is unimodal, but is progressively getting more abstract and converging on multi modal sensory areas. However, we suggest that the main importance of the hierarchy of the brain stems from bi-directional processing³ based upon a brain architecture that consists of both an *archicortex* and a *neocortex*, which will be outlined in Section 2. Section 3 gives an overview of different cells in early vision from functional perspective. Section 4 shows, by example, how functional behavior of subsequent groups of cells in the feed forward data stream of early vision can be predicted. The paper finalizes with a discussion.

2 Integration of sensory areas

The cerebral cortex is anatomically divided into four (frontal, parietal, temporal, and occipital) lobes, and has functionally distinct regions. Most areas are primarily concerned with processing sensory information or delivering motor commands. Studies of afferent sensory pathways and association areas in the cortex have lead to three important principles of sensory information processing [7]:

1. Sensory data is processed in a series of relays along several parallel pathways from peripheral receptors through primary cortex and unimodal association cortex to the multimodal association cortex, see Figure 1.

³ Bi-direction processing incorporates feedforward and feedback mechanisms that can be executed in parallel in multiple areas in the brain.

2. Sensory data representing different modalities converge upon areas of cortex that integrate the data.
3. The posterior association areas that process sensory data are highly interconnected with the frontal association area responsible for planning motor actions.

Unimodal sensory outputs converge on multimodal association areas in the *prefrontal*, the *parietotemporal*, and *limbic* cortices. Neurons in these areas respond to combinations of signals representing different sensory modalities by constructing an internal representation of the sensory stimulus concerned with a specific aspect of behavior.

However, data does not converge to a single highly specialized area [23], therefore one can state that the *feed forward* sensory streams should be interpreted as preprocessing for the cognitive brain areas, which include the archicortex and neocortex, where data is processed in a bi-directional way. The main reason of the hierarchy of the brain stems from bi-directional processing that consists of both an archicortex and a neocortex. Preprocessed sensory data feeds both cortices in a parallel manner. Where the areas, illustrated at the bottom part of Figure 1 can serve as input for the archicortex.

2.1 Feed forward data streams

Hubel and Wiesel did pioneering work in the cat's striate cortex. They explored various visual cortical areas with micro-electrodes and divided the recorded cells into four distinct classes (center-surround, simple, complex, and hyper complex cells) [5]. This division is based upon a building block architecture where center-surround type of cells form the input for the simple cells, simple cells in turn form the input for complex cells, and so on.

Currently, a subdivision into four classes would not suffice anymore, since many new types of cells have been found. FBM utilizes the concept of the building block architecture since it is very attractive from functional point of view. The building block concept will result in groups of cells that are all responsible for one or few specific features. Feature extraction yields a strong reduction of data (but moderate reduction of relevant information). Feature extraction in turn requires a processing mechanism to combine the features. As data progresses its representation is gradually becoming more abstract.

2.2 Bi-directional data streams

The human brain is evolved from the brain of other mammals having a special brain structure: the neocortex, which provides us strong cognitive abilities. Animals not having a neocortex achieve their highest cognitive processing in the archicortex [3]. The sensory pathways developed in the "old" archicortex are still active in the primate brain. They can process multimodal sensory data and provide basic cognitive skills, e.g., how to survive.

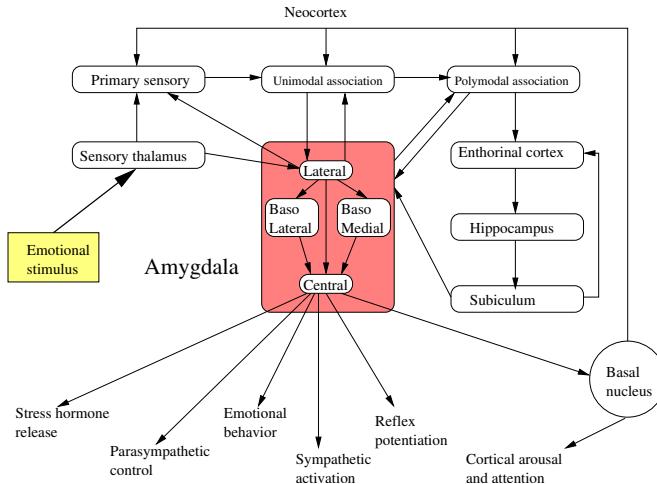


Figure 2. Schematic overview of emotional stimulus processing, from [18].

The *Amygdala*, known as center of many emotional processes, gets its sensory inputs not only from the neocortex but also from hypothalamus, thalamus, and brainstem [9,18], see Figure 2. Although the sensory input from thalamus activates the amygdala faster than input from neocortex [10,13], the quality of sensory processing is coarse because of the archicortex's relatively poor neuronal circuit. The fast and coarse poly-sensor processing in amygdala sending output to both unimodal association cortex and multimodal association cortex could be the top-down hypothesis on the feed-forward sensory stream to combine the relevant processes in the cortex by bi-directional connections [22]. Additionally, prefrontal cortex, anterior cingulated cortex, and hippocampus get input from amygdala [1,2,15,18], and could serve temporally dynamic integration of the neocortical processes.

Our aim is to utilize FBM to encapsulate the detailed neuronal interaction to elucidate the above described complex interaction within the archicortex.

3 Functional behavior of cells in early vision

Complex cells respond vigorously to lines and edges, but also to grating patterns of appropriate frequency and orientation. This response to grating patterns seems an artifact. However, grating cells receive inputs from complex cells, so therefore complex cells do respond to grating patterns too. Grating cells, most likely, play a role in both texture processing and figure-ground segregation [19,20,14]. From functional point of view it is expected that the complex type of cells segregate into a grating type of cell and a *complex type 2* type of cell that responds solely to lines and edges. Latter type of cell will receive excitatory input from the complex cells and inhibitory input from the grating cells.

A similar concept holds for junctions.⁴ Endstopped cells respond well to all junctions (including line ends and corners), except to junctions where four lines end [4]. A typical four-line-end junction is a crossing of two lines. In the brain so-called crossing cells are found that respond to crossings only [16,17]. An important question to be solved is why end-stopped and crossing type of cells are found. It is expected that in one of the subsequent layers a *junction* type of cell is found, i.e., grouping of end-stopped and crossing responses takes place there. The junction type of cell responses is most easily obtained by a combination of end-stopped and crossing type of cells. If junction type of cells are found in one of the former layers, in the feed forward data stream, then it is likely that end-stopped and crossing type of cells play a different role in subsequent layers.

Utilizing the building block principle to subsequent groups of cells in a feed forward manner will at a certain moment lead to groups of cells that respond to highly specific features. It is clear from functional point of view that these cells, at a certain stage in the feed forward stream, will be fused to object like entities. For example, in the inferior temporal cortex cells have been found that strongly responding to face like stimuli [8].

4 Early vision simulation

Simulations of functional behavior of complex, endstopped, and grating cells give clear hints of the existence of other types of cells. Figure 3a illustrates an early vision simulation of complex, endstopped, and grating cells. The parameter settings that are used for the complex and endstopped cells have size $\sigma = 3.5$, wavelength $\lambda = 1$, spatial aspect ration $\gamma = 1$, and number of orientations $N = 8$. For mathematical details of the complex and endstopped cells we refer to [11,21]. For the grating cells the following parameters are used: $\sigma = 3.5$, $\lambda = 1$, $\gamma = 0.25$, and $N = 8$. Functional description and detailed properties can be found in [12].

Figure 3b is a synthetic stimulus that is created with the purpose to illustrate the need of strong interaction between different groups of cells. A grating pattern is used to partly mask a rectangle, but the setting is made such that a conflicting situation is created, i.e, until where the left vertical line belongs to the rectangle. The lines of the upper right corner of the rectangle are extended with the purpose to illustrate the behavior of endstopped cells. An early vision simulation (Figure 3a) using the input image of Figure 3b yield the results illustrated in Figure 4.

Complex cells respond to lines and edges of a specific orientation. In Figure 4a the results of a superposition of all N orientations of complex cells is illustrated. Since a grating pattern is made up of a set of alternating stripes, complex cells respond vigorously. From functional aspect such type of response is not desired because of its redundant character. It would be far more attractive to only have the boundaries of the grating pattern and to mark the interior as grating texture.

The fact that grating cells make up around 4 and 1.6 percent of the number of cells in monkey areas V1 and V2 [20], respectively, indicate that these cells

⁴ A *junction* is a spatial coordinate where one or more lines or edges end.

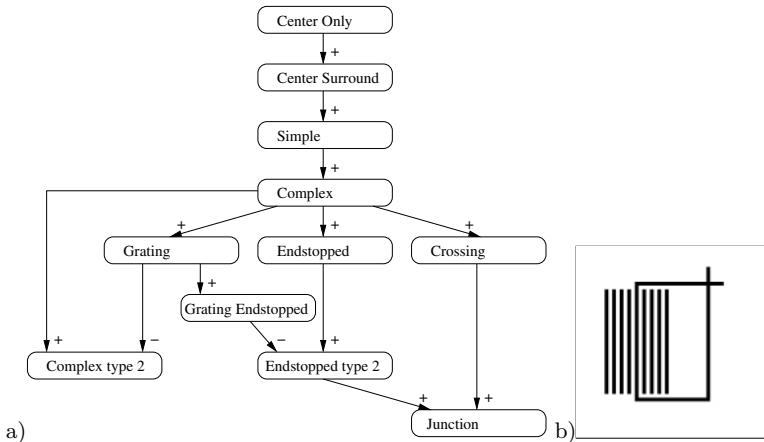


Figure 3. a) Early vision feedforward data processing network. The network includes simple, complex, endstopped, and grating cell responses. Complex type 2 cells respond strongly to edges and lines of a preferred orientation, while proposed junction cells respond strongly to all types of junctions. b) A synthetic input stimulus.

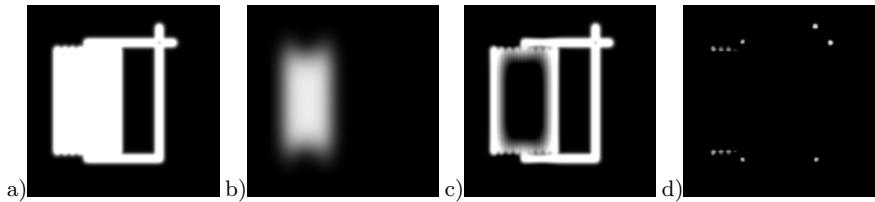


Figure 4. Responses of complex cells, grating cells, complex cells inhibited by grating cells, and endstopped cells, a-d, respectively. White denotes strong response, black no response.

play a prominent role in form segregation. Grating cells are highly orientation and frequency sensitive [19,20] which makes them most likely to be an inhibition mechanism for the complex cell responses rather than a texture processing mechanism [12]. Figure 4c illustrates the results of this inhibition mechanism. It is remarkable that the model for grating cells shows a slight drop in response at positions where the vertical line of the rectangle is masked by the grating pattern.

Endstopped cells respond to line ends, corners, and junctions, but not to crossings [4]. The model for endstopped cells shows that this is the case (Figure 4d). However, from functional perspective it is desirable to have all junction type of responses, as proposed in the feedforward data processing network of Figure 3a. Crossing type or double orientation tuning type of cells have been found in the cat striate cortex [16,17].

Endstopped responses also occur at the ends of the stripes in the grating pattern, this is a non-desired effect to endstopped responses. Endstopped type of grating cells have been found as well in monkey areas V1 and V2 [20]. It is likely that endstopped cells are inhibited by grating-endstopped type of cells, resulting in type 2 endstopped cells, and have excitatory connections with crossing type of cells to yield responses to all types of junctions, which is illustrated in Figure 3a as so-called junction type of cells.

5 Discussion

We proposed to process sensory and cognitive data by FBM. The aim of FBM is not to make a model of the brain on microscopic level, but merely to model, on macroscopic level, the functionality of a group of cells or even a complete area that have equivalent or similar functional behavior, and give the interaction between groups with different functional characteristics. From functional perspective the brain can be crudely divided into two types of data streams: a feed forward (sensory) data stream and a bi-directional (cognitive) data stream.

The disadvantage of FBM is its relatively static nature due to functional description and the assumption of data flow directions. The development and learning capacities are not as flexible as in the brain. Most of the primary sensory areas deal mainly with feature enhancement, extraction, and grouping. It is therefore not expected that primary areas need to adapt quickly or strongly. Hence, its sensory processing apparatus is doing similar work from functional perspective. Experiments performed by Hubel and Wiesel [6] give evidence that cat's primary visual sensory area is developed during the first weeks after birth, but at a later stage it hardly develops.

An early vision simulation utilizing FBM showed that resulting cell responses give strong insight in interaction between different groups of neuronal cells and one can predict the functionality of adjacent groups of neuronal cells.

References

1. D. G. Amaral and J. L. Price. Amygdalo-cortical projections in the monkey. *J. Comp. Neurol.*, 230:465–496, 1984.
2. S. T. Carmichael and J. L. Price. Limbic connections of the orbital and medial prefrontal cortex in macaque monkeys. *J. Comp. Neurol.*, 363:615–641, 1996.
3. J. P. Ewert, H. Buxbaum-Conradi, F. Dreisvoigt, M. Glagow, C. Merkel-Harf, A. Röttingen, E. Schürg-Pfeiffer, and W. W. Schwippert. Neural modulation of visuomotor functions underlying prey-catching behaviour in anurans: perception, attention, motor performance, learning. *Comparative Biochemistry and Physiology*, A 128(3):417–461, 2001.
4. F. Heitger, L. Rosenthaler, R. von der Heydt, E. Peterhans, and O. Kübler. Simulation of neural contour mechanisms: from simple to end-stopped cells. *Vision Research*, 32(5):963–981, 1992.
5. D. Hubel and T. Wiesel. Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex. *Journal of Physiology, London*, 160:106–154, 1962.

6. D. H. Hubel. *Eye, Brain, and Vision*. Scientific American Library, New York, 1988.
7. E. R. Kandel, J. H. Schwartz, and T. M. Jessell. *Principles of neural science*. McGraw Hill, fourth edition, 2000.
8. E. Kobatake and K. Tanaka. Neuronal selectivities to complex object features in the ventral visual pathway of macaque cerebral cortex. *Journal of Neurophysiology*, 70:856–867, 1994.
9. J. E. LeDoux. *Handbook of Physiology*, volume 5, part 1, chapter 1, Emotion, pages 419–459. American Physiological Society, 1987.
10. J. E. LeDoux. *The emotional brain: mysterious underpinnings of emotional life*. Simon & Schuster, New York, 1996.
11. T. Lourens. *A Biologically Plausible Model for Corner-based Object Recognition from Color Images*. Shaker Publishing B.V., Maastricht, The Netherlands, March 1998.
12. T. Lourens, H. G. Okuno, and H. Kitano. Detection of oriented repetitive alternating patterns in color images – a computational model of monkey grating cells. In J. Mira, editor, *Proceedings of the International Workshop on Artificial Neural Networks, IWANN 2001*, volume 1676, Part I of *Lecture Notes in Computer Science*, pages 95–107, Granada, Spain, June 2001. Springer-Verlag.
13. J. S. Morris, A. Ohman, and R. J. Dolan. A subcortical pathway to the right amygdala mediating "unseen" fear. *PNAS*, 96(4):1680–1685, 1999.
14. E. Peterhans. *Cerebral Cortex*, volume 12, chapter 8, Functional Organization of Area V2 in the Awake Monkey, pages 335–357. Plenum Press, New York, 1997.
15. A. Pitkänen, M. Pikkariainen, N. Nurminen, and A. Ylinen. Reciprocal connections between the amygdale and the hippocampal formation, perirhinal cortex, and postrhinal cortex in rat: a review. *Annals New York Academy of Sciences*, 911:369–391, 2000.
16. I. A. Shevelev, N. A. Lazareva, B. V. Novikova, A. S. Tikhomirov, and G. A. Sharaev. Double orientation tuning in the cat visual cortex units. *Neuroscience*, 61(4):965–973, 1994.
17. I. A. Shevelev, R. V. Novikova, N. A. Lazareva, A. S. Tikhomirov, and G. A. Sharaev. Sensitivity to cross-like figures in the cat striate neurons. *Neuroscience*, 69(1):51–57, 1995.
18. L. R. Squire, F. E. Bloom, S. K. McConnell, J. L. Roberts, N. C. Spitzer, and M. J. Zigmond, editors. *Fundamental Neuroscience*. Academic Press, second edition, 2002.
19. R. von der Heydt, E. Peterhans, and M. R. Dürsteler. Grating cells in monkey visual cortex: Coding texture? Visual Cortex. In B. Blum, editor, *Channels in the visual nervous system: neurophysiology, psychophysics and models*, pages 53–73, London, 1991. Freund Publishing House Ltd.
20. R. von der Heydt, E. Peterhans, and M. R. Dürsteler. Periodic-Pattern-selective Cells in Monkey Visual Cortex. *The Journal of Neuroscience*, 12(4):1416–1434, April 1992.
21. R. P. Würtz and T. Lourens. Corner detection in color images through a multiscale combination of end-stopped cortical cells. *Image and Vision Computing*, 18(6–7):531–541, April 2000.
22. M. P. Young, J. W. Scannell, G. Burns, and C. Blakemore. Analysis of connectivity: neural systems in the cerebral cortex. *Reviews in the Neurosciences*, 5:227–250, 1994.
23. S. Zeki. *A Vision of the Brain*. Blackwell science Ltd., London, 1993.

Life-long learning: consolidation of novel events into dynamic memory representations

Emilia I. Barakova, Tino Lourens and Yoko Yamaguchi

RIKEN-BSI, DEI Lab, 2-1, Hirosawa, Wako-shi, Saitama, 351-0198, Japan,
Honda RI, Japan Co., Ltd, 8-1, Honcho, Wako-shi, Saitama, 351-0114, Japan.

Abstract. Life-long learning paradigm accentuates on the continuity of the on-line process of integrating novel information into the existing representational structures, and recategorization or update of these structures. This paper brings up the hypothesis, that memory consolidation is a biological mechanism that resembles the features of life-long learning paradigm. A global model for memory consolidation is proposed on a functional level, after reviewing the empirical studies on the hippocampal formation and neocortex. Instead of considering memory as storage, the proposed model reconsiders the memory process as recategorization. Distinct experiences that share a common element can be consolidated in the memory in a way such that they are substrata for a new solution. The model is applied to an autobiographical robot.

1 Motivation

Life-long learning paradigm accentuates on the continuity and recategorisation in the learning process. The learned novel fact, event or skill is not independent from the background of the learner. In contrast, the knowledge obtained from previous tasks alleviates acquiring of new information by constraining the possible hypothesis space for those tasks. Furthermore, it is a basis for emergence of novel solutions derived by translating the experience from different domains. The process by which the novelty finds its place in the complete system of previously learned structures, updates these structures, or in other words causes reconceptualization.

While living organisms naturally store and use creatively information, experienced even once in a life-span, it is extremely hard to simulate similar phenomena on artificial agents[1][20]. The major difficulty in simulating life-long learning systems stems from the complexity of the integration and categorization of continuously changing sensory and motor information within the existing representational structures and concepts. The newly acquired information in its turn changes the representations and causes reconceptualization. In addition, all the artificial learning systems suffer from catastrophic forgetting, a process by which the new knowledge dominates over or cancels the previously learned ones.

The neurophysiological mechanism, that resembles the main features of the long-life learning, the continuity, novelty integration and reconceptualization of the existing representational structures is the memory consolidation process. In this paper the consoli-

dation process between the new coming sensory-motor signals and the forming representations and categories is modelled.

In neuroscience, the memory consolidation is understood as a process where declarative memories are gradually encoded into the neocortex. A wide array of neuropsychological, neuroanatomical, imaging, and neurophysiological data define the system crucial for declarative long-term memory and memory consolidation as: the hippocampal formation as one-shot memory encoder, and the neighboring cortical areas in the ventro-medial temporal lobe as consolidating and integrating part and the associative neocortex as a long-term storage part, e.g. [2][9][16].

Miller [14] proposes a theory describing how the interaction of the hippocampus, the theta rhythm, and the isocortex allow the brain to represent contexts. He suggests that theta rhythm mediates and synchronizes this process.

In the recent work of Yamaguchi[22], that models the neural dynamics of the hippocampal network, is concluded, that theta phase precession provides an efficient on-line mechanism for memory storage of novel temporal sequences. This mechanism is used as starting point.

Memory consolidation is understood as a two-fold process. Initially, the integration of novel information into the existing representational structures or concepts is performed. On its turn, this integration updates the previously formed representations. Modelling this process is based on novelty encoding. Detailed modelling of the brain mechanisms ensuring the consolidation is not aimed, but rather adopting the main operational principles that will allow to achieve a life-long learning functionality.

Behaviourally-oriented memory functionality, as a part of a life-long learning system, is illustrated in a robotics framework. The ultimate goal is building an autobiographical robot. The current achievement is in learning distinct experiences, that share common elements (overlapping concepts). The test has to find out whether these experiences have been integrated into the memory in a way that enables emergence of solutions for novel problems. The proposed model for concept formation resembles the neocortical functionality and its interplay with the hippocampal formation in the following way: Reactivated concepts initiate a sequence of meaningful events, as previously formed and stored in another network, modelling the hippocampal functioning. The newly experienced events can change the topology of the concept-forming network. A further aim is to extend the experimental and thus the theoretical research to dynamical and different environments, so to create an autobiographical robot. As an autobiographical robot is understood a robot which can recollect its past and use it in an creative way.

2 Consolidation and complementary memory systems

There are two established views of what memory consolidation implies. The first view considers the consolidation as initiation of a cascade of cellular and molecular events by a distinctive experience, that forms a durable form of synaptic alteration [5]. The second view equalizes the consolidation to a post-processing of memory traces, during which the traces may be reactivated, analysed, and gradually incorporated into long-term event memory and general knowledge base [12]. We do not think that the two views are contradictory. The second view explains the recall dynamics of the stored ex-

periences, while the first one describes the concepts of formation. Naturally, we will start with second one, since it has more direct relation to solving the life-long learning problem. Going to the detail of the memorization, the first view is naturally exploited. The view of consolidation as a processes that alter the newly stored and still labile information to the more stable long term storage relies on the functionality of the cortico - hippocampal memory system. The episodic memory consolidation involves a recording process that transfers the episodic memory trace of an event from the hippocampus to the neocortex [11][12][16][14].

The brain areas, involved in memory consolidation include: the hippocampal formation (HF), the medial frontal areas (MTL), and the neocortical association areas (NAA). We will restrict our analysis to HF and NAA (Figure 1).

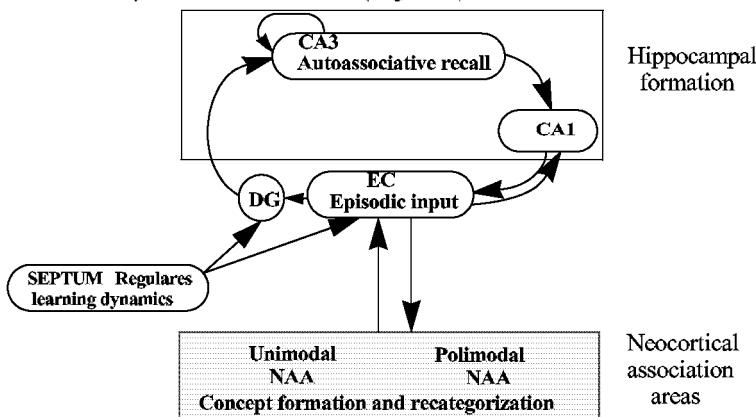


Figure 1 Memory system involved in the consolidation process

There is a broad consensus that the neocortex is the final storage site for declarative long-term memories [7]. The representations are formed on the similarity principle. The use of overlapping representations allows the cortex to represent the shared structure of events, it therefore enables generalisation.

The hippocampus is associated with episodic memory encoding. In addition it directs the consolidation process by reactivation of the stored patterns in some cortical areas and allowing synapses in these areas to change. The functionality of the hippocampus by the memory formation and consolidation processes is mediated by the theta rhythm.

The theta rhythm is a major brain oscillation in the range of 4-10Hz. As a rhythmic activity it originates at the septum[18]. The theta bursts are sent from the medial septum and the diagonal band of the septum to all parts of the hippocampus [14]. The hippocampus then relays these rhythmic firing bursts to the cortex.

The studies on anterograde amnesia strongly suggest that theta activity of the hippocampus may be related to the encoding and/or retrieval of new information. Positive evidence came from studies which have documented that there is a preference for long-term potentiation (LTP) to occur in the hippocampal formation, and that theta activity induces or at least enhances LTP [4]. The fact that LTP is considered the most important

electrophysiological correlate for encoding new information, underlines the potential importance of hippocampal theta for memory processes.

In trying to explain the possible functional significance of the theta rhythm in humans, we assume that synchronized bursts of a small set of hippocampal pyramidal cells induce theta activity in selected but distributed cortical regions which are relevant for performing a particular task.

There is a general consensus that memory recall involves reactivation of cortical patterns that characterize the original episodes. This reactivation can be instantiated by partial or noisy patterns. Kali and Dayan [8] define the computational role of memory system as cortical pattern completion. This view is a basis for their model of memory consolidation, performed by a Hopfield - type associative memory network. Obviously, their model of the consolidation process is restricted to the first view for consolidation, as listed in the beginning of this section. In general consensus with [8] our understanding of memory consolidation extends it in the following way:

- The central aspect of consolidation is an achievement of consensus between the two memory subsystems with different characteristics - the fast, on-line memory encoding by the hippocampus and the slowly changing representations of concepts in the neocortex.
- Memory consolidation is a twofold process of integration of novel information into the existing representational structures and updating the previously formed representations.
- The novel information is encoded by the theta rhythm. The so encoded information reaches the associative neocortex areas, where either update an older concept/representation or can form a new one.
- Once reactivated, the episode initiates a sequence of related events.

3 Computational Model

As a physiological process, memory consolidation allows for memories to become permanently encoded in the brain. Computationally it has to resolve the conflict between the two types of representations - hippocampal and neocortical. The hippocampus is specialized in on-line memorizing of specific events, while the neocortex learns slowly the regularities and concepts. The hippocampus assigns distinct representations to stimuli, thereby allowing rapid learning without suffering catastrophic forgetting. In contrast, the neocortex assigns similar representations to similar stimuli and uses overlapping representations. Therefore it can represent the shared structure of events and generalizes to previously experienced stimuli.

These arguments clarify that computationally the consolidation model has to deal with the trade-off of generalization and novelty encoding.

A biologically plausible model for hippocampal novelty encoding in one-shot learning is developed in [22]. According to it, the novel experience is encoded on the phase-locking principle by the theta rhythm and stored in the connections of the Hebbian network.

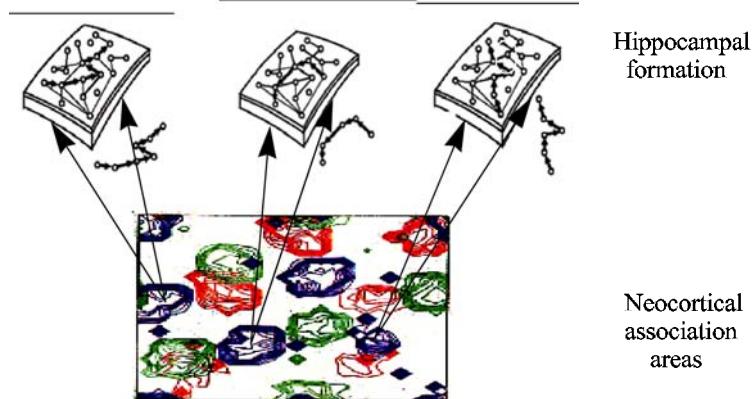


Figure 2 Schematic representation of computations in HF and NAA

The input information, characterizing a distinctive scenery from the environment is represented with a vector $I(t) = \{I_i(t)\}$. Its physical meaning in terms of the hippocampal model of Yamaguchi[22] is, that in an initially stable system of the hippocampal network is brought to oscillatory behaviour by introducing the positive valued input pattern $I(t)$. Since the entorhinal cortex (EC) provides the main cortical input to the CA1 and CA3 hippocampal areas, this layer is considered as an anatomical counterpart of the neural layer, where the inputs are initially supplied to. In EC layer, the neural oscillator that models the functionality of the $i-th$ EC unit is presented as:

$$\dot{\phi}_i(t) = \omega_i(t) + \{\beta_0 - I_i(t) - \cos\phi_0(t)\} \sin\phi_i \quad (1)$$

where $\omega_i(t)$ represents the native frequency and β_0 is a stabilization coefficient. Theta rhythm is modelled as an additional unit of the EC layer. Its oscillatory behaviour is described as:

$$\dot{\phi}_0(t) = \omega_0. \quad (2)$$

A mechanism, that alleviates spatial coding of the information is called phase precession [15] and constitutes in gradual change in the native frequency $\omega_i(t)$. Because the firing phase advances with the successive cycles, the spatial displacement of any two place fields or in general memory events is reflected in the firing order of the cells within the theta cycle. Thus, the temporal firing order is expressed in a compressed form during any given theta cycle. Functionally, this serve for encoding of event sequences by making the neural activity pattern change rapidly within the effective time window of LTP (about a theta cycle). Without compression, the patterns, that become associated over this short time of retrieval of sequences will suffer interference [17].

The phase-modulated input patterns are further fed into a categorization network. The categorization network is accomplished by a Growing neural gas algorithm[6]. It combines competitive Hebbian learning [10] with a growing cell structure. The purpose of the self-organizing algorithm is dimensionality reduction for the input vector. In computational terms, the low-dimensional representation is equivalent of the concept for-

mation. The competitive Hebbian learning adds to this the preservation of the topological structure of the input data [10]. In addition, the model is incremental. It starts with a small structure and grows until all the input space is represented, i.e. until all the concepts are represented. The method uses local computations therefore, it is prone to catastrophic forgetting. It adds new units any time a new concept is found.

Until now the two parts of the model, the one that approximates the hippocampal-like one-shot learning, and the one, that represents the concept formation as it is believed to happen in neocortex were described. The consolidation process, that features the interplay between the two structures takes place as schematically shown in Figure 2. The upper layer represents a number of stored sequences, found during different learning trials and encoded and stored by the hippocampal network. The lower level represents the concept formation. Some of the novel concepts, shown in red, are not distinctively formed yet. If a pattern is categorized as belonging to a certain concept, that is close to a wanted solution, the categorization cause recall of a stored Hebbian hippocampal network, and initiates a sequence of related events.

4 Experimental evidence

There is not a well established definition for an autobiographical robot, although there are few researchers that have been using this term. Intuitively, we define as an autobiographical robot to possess authobiographical memory. The definition will be of practical significance if it makes explicit the borders of functionality of an autobiographical robot. So we clarify, that autobiographical robot is able to:

- Recall old episodes.
- The recalled episodes provoke a sequence of related memories.
- The old experiences can be used in a creative way i.e. as an emergent novel solution.

The first step of creating an autobiographical robot is a gathering of relevant experiences trough a single trial learning. The useful (according to a certain criterion) sequences of experience acquired during the robot life span are stored as the fixed Hebbian structures, that can reproduce the experienced event sequence. This is done in an agreement with the new findings, that by the recall process both the hippocampal and neocortical representations are reactivated[17].

According to the proposed cortico-hippocampal model, the novel perception/event) is first categorized as belonging to a particular existing category, or a new category is created. If categorization to an existing cluster is made, this experience triggers a sequence of events, that have been following the same experience when it has happened in the past.

To be more specific, a new visual percept (the novel seen event or scenery) is categorized in the cortical structure. This novel event triggers a series of ideothetic information that has been experienced in the past, and represents an successful strategy for accomplishing a wished goal.

Figure 3 illustrates the concept formation maps in two different stages of life-long learning process. With the time of learning some close concepts has unified, and the others

have diverged and separated. This shows the recategorization feature of our model, an important element of a life-long learning system. For simplicity the illustration is made for ideothetic stream of information.

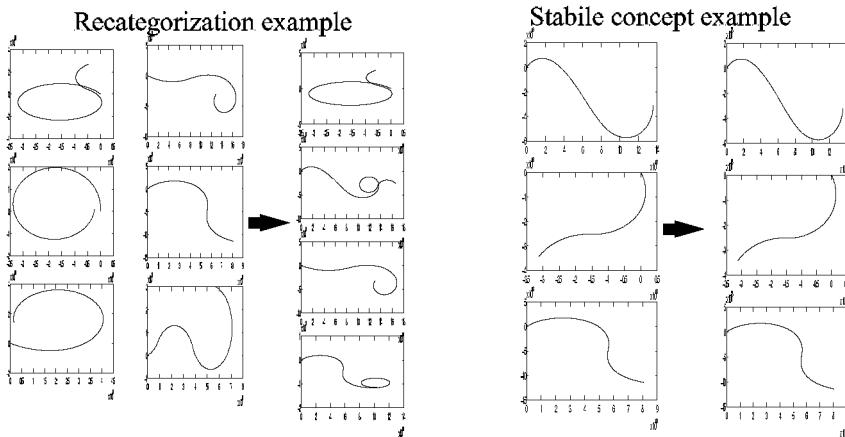


Figure 3 Two stages of concept formation by life-long learning

5 Discussion

Psychological studies[3][19] show that humans integrate and interpret new experiences on the basis of previous ones. Previous experiences are reconstructed with the actual body and context as the point of reference Conway[3]. This way the previously remembered event is recalled through the prism of the present-day knowledge, understanding and mood. With this respect memory is reconceptualization process as opposed to a storage. The developed model clearly shows this property, and therefore is a good basis for developing of a life-long learning system.

The division of the memory structures to fast learning for novelty encoding and slow changing for representing the knowledge on conceptual level is a powerful mechanism to cope with the trade-off between generalization and inference.

The advantages of categorization of theta encoded signals will be shown in follow-up paper.

The proposed model, goes to a different degree of detail in the functionality of the two brain regions, mostly associated with consolidation process. The near future developments include creating an on-line version of the hippocampal encoding network and dynamical environment framework.

6 References

- [1] Barakova E.I. and U. R. Zimmer, Dynamical Situation and Trajectory Discrimination of Raw Range Measurements, *AISTA*, Canberra, Feb. 2000.

- [2] Cohen NJ and Eichenbaum H, *Memory, amnesia and the hippocampal system*. Cambridge, MA: MIT Press, 1995.
- [3] Conway, M. A., & Beckerian, D. A. (1987). Organization in autobiographical memory. *Memory & Cognition*, **15**, 119~132.
- [4] Pavlides C, Greenstein YJ, Grudman M, Winson J., Long-term potentiation in the dentate gyrus is induced preferentially on the positive phase of theta-rhythm. *Brain Res* 1988, **439**:383-387.
- [5] Izquierdo I, Medina JH, Memory formation: the sequence of biochemical events in the hippocampus and its connection to activity in other brain structures. *Neurobiol. Learning and Memory* 1997, **68**:285-316.
- [6] Frizke, B. (1995). A growing neural gas network learns topologies. *Advances in Neural Information Processing Systems* 7. MIT Press, Cambridge MA.
- [7] Fuster, J.M. *Memory in the Cerebral Cortex*. MIT Press, 1994.
- [8] Kali S, Dayan P. 2000. Hippocampus-dependent consolidation in a hierarchical model of neocortex. *NIPS* 2000:24~30.
- [9] Lavenex P. and David G. Amaral, Hippocampal-Neocortical Interaction: A Hierarchy of Associativity, *Hippocampus* 10:42-430 (2000).
- [10] Martinetz, T., & Schulten, K. (1991). A neural-gas network learns topologies. In T. Kohonen et al. (Eds.), (pp. 397~402). Proc. of ICANN, Amsterdam, Netherlands.
- [11] McClelland JL, McNaughton, B.L., O'Reilly, R.C. Why there are complementary learning systems in the hippocampus and neocortex. *Psychol. Rev.* 1995, **102**, 419-457
- [12] Marr D: Simple memory: a theory of archicortex. *Philos Trans R Soc Lond B* 1971, **262**:23-81.
- [13] McClelland JL and Goddard, NH, 1997. Considerations arising from a complementary learning systems perspective on hippocampus and neocortex. *Hippocampus*, **6**:654-665.
- [14] Miller, R. Cortico-hippocampal interplay: Self-Organized, Phase-locked loops for Indexing Memory, *Psychobiology*, Vol. 17(2), 115-128, 1989.
- [15] O'Keefe, J. and Recce M., 1993, Phase relationship between hippocampal place units and the EEG theta rhythm. *Hippocampus*, **3**:317-330.
- [16] Squire, L. R. Memory and the hippocampus: A synthesis from findings with rats, monkeys, and humans. *Psychological Review*, 99:195-231, 1992.
- [17] Sutherland, G.R. and McNaughton, B. (2000) Memory trace reactivation in the hippocampal and neocortical neuronal ensembles. *Curr. Opin. Neurobiol.* 10, 180~186
- [18] Swanson, L.V., Kohler, C. and Bjorklund A. The limbic region: I: The septohippocampal system. In A. Bjorklund, T. Hokfeld, and L.W. Swanson, eds. *Handbook of chemical neuroanatomy*, Vol. 5, pages 125-277. Elsevier, 1987.
- [19] Rubin, D. C. (Ed.). (1996). Remembering our past: Studies in autobiographical memory. New York: Cambridge University Press.
- [20] Thrun, S.B., and T. M. Mitchell. Integrating inductive neural network learning and explanation-based learning. In *Proceedings of IJCAI-93*, Chamberry, France, 1993.
- [21] Wiener, S.I., V. Korshunov, R.Garcia, and A . Berthoz, 1995. Internal, substratal and landmark que control of hippocampal CA1 place cell activity. *European Journal of Neuroscience*, **7**:2206-2219, 1995
- [22] Yamaguchi, Y., 2003. A Theory of Hippocampal Memory Based on Theta Phase Precession *Biological Cybernetics*, In press.

Real-time Sound Source Localization and Separation based on Active Audio-Visual Integration

Hiroshi G. Okuno¹ and Kazuhiro Nakadai²

¹ Graduate School of Informatics, Kyoto University, Kyoto 606-8501 Japan
okuno@i.kyoto-u.ac.jp <http://winnie.kuis.kyoto-u.ac.jp/~okuno/>
² Kitano Symbiotic Systems Project, ERATO, Japan Science and Technology Corporation
nakadai@nakadai.com <http://www.symbio.jst.go.jp/~nakadai/>

Abstract. Robot audition in the real world should cope with environment noises and reverberation and motor noises caused by the robot's own movements. This paper presents the *active direction-pass filter* (ADPF) to separate sounds originating from the specified direction with a pair of microphones. The ADPF is implemented by hierarchical integration of visual and auditory processing with hypothetical reasoning on interaural phase difference (IPD) and interaural intensity difference (IID) for each subband. In creating hypotheses, the reference data of IPD and IID is calculated by the *auditory epipolar geometry* on demand. Since the performance of the ADPF depends on the direction, the ADPF controls the direction by motor movement. The human tracking and sound source separation based on the ADPF is implemented on an upper-torso humanoid and runs in real-time with 4 PCs connected over Gigabit ethernet. The signal-to-noise ratio (SNR) of each sound separated by the ADPF from a mixture of two speeches with the same loudness is improved to about 10 dB from 0 dB.

1 Introduction

Robot audition can be improved by active motion and multi-modal integration as well as *active vision* [1]. As a technique for robust robot audition, *active audition* is proposed to control microphone parameters to attain better auditory perception [9, 12]. The difficulty in active audition lies in sound source separation under real world environments. The technical issues include as follows:

1. cancellation of motor noise in motion,
2. auditory processing without using *Head Related Transfer Function (HRTF)* and measuring acoustic environments in advance, and
3. sound source separation techniques.

Robot's active motion makes inevitable motor noise, and it makes auditory processing more difficult. Therefore, robots with audition adopt the "stop-hear-act" principle, that is, a robot stops to hear. Otherwise it uses a microphone attached near the mouth of each speaker for automatic speech recognition [4, 7].

HRTFs are often used for sound source localization. Since, HRTFs are measured in an anechoic chamber, actual room characteristics should be measured in advance to use HRTFs. In addition HRTFs are available only at discrete points due to discrete



Fig. 1. *SIG*



Fig. 2. *SIG*'s ear

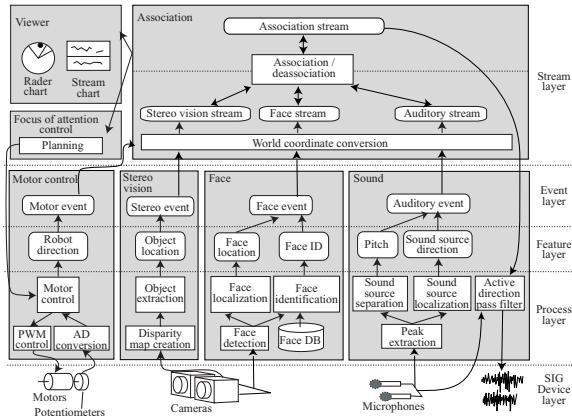


Fig. 3. Hierarchical Architecture of Real-Time Tracking System

measurement. Therefore, such sound source localization methods cannot be applied to a robot that changes its position by moving or rotation.

Common current sound source separation techniques, such as beamformers with a microphone array [2, 13], independent component analysis as blind source separation [3], and computational auditory scene analysis techniques based on human auditory system [8], have not met the above requirements for robot audition yet. A real-time real-time sound localization and separation system based on the minimum-variance beamformer with eight microphones placed on a circle is developed for robot audition [2], which needs the database of location vector for the near-field measured in advance.

Robots with microphones as ears for sound source localization or sound source separation have attained little in auditory tracking or sound source separation. *Kismet* has a pair of omni-directional microphones outside the simplified pinnae [4]. Since it is designed for one-to-one communication in social interaction based on visual attention, the auditory tracking has not been implemented so far. *Hadaly* uses a microphone array to perform sound source localization, but the microphone array is mounted in the body and its absolute position is fixed during head movements [7]. In the both cases, a microphone for speech recognition is attached to the speaker.

To cope with these technical issues, a new sound source separation method called the *active direction-pass filter (ADPF)* is proposed. The ADPF does sound source localization by *auditory epipolar geometry* without using HRTFs [9]. The ADPF is implemented as a part of the real-time multiple speaker tracking system installed on an upper-torso humanoid called *SIG* [9]. Motor noises and other noises caused by its movement are canceled by using cover acoustics. The tracking system attains accurate localization of multiple speakers by integrating face detection by stereo vision, multiple face recognition, and active motion control of *SIG* [9–11].

The rest of this paper is organized as follows: Section 2 describes our humanoid *SIG* and the real-time human tracking system. Section 3 explains sound source separation by the ADPF. Section 4 evaluates the performance of the ADPF. The last section provides discussion and conclusion.

2 The Real-Time Human Tracking System

We use the upper torso humanoid *SIG* shown in Fig. 1 as a platform for multi-modal integration. *SIG* has a cover made of FRP (fiber reinforced plastic). It is designed to separate the *SIG* inner world from the external world acoustically. A pair of CCD camera (Sony EVI-G20) is used for stereo vision. Two pairs of microphones are used for auditory processing. One pair is located in the left and right ear position for sound source localization (Fig. 2). The other is installed inside the cover mainly for canceling self-motor noise in motion. *SIG* has 4 DC motors (4 DOFs) with functions of position and velocity control by using potentiometers.

The architecture of the real-time human tracking system using *SIG* shown in Fig. 3 consists of seven modules, i.e., Sound, Face, Stereo Vision, Association, Focus-of-Attention, Motor Control and Viewer.

Sound, Face and Stereo Vision generate an *event* by feature extraction. Motor Control also generates an event of motion. Sound separates sound sources and localizes them. In addition, by feedback of streams in Association, it can extract a specific sound source by the ADPF. Face detects multiple faces by combining skin-color extraction, correlation based matching, and multiple scale image generation [5]. It identifies each face by Linear Discriminant Analysis (LDA), which creates an optimal subspace to distinguish classes and continuously update a subspace on demand with a small amount of computation [6]. In addition, the faces are localized in 3-D world coordinates by assuming average face size. Stereo Vision localizes lengthwise objects such as people precisely by using a disparity map. Association forms *streams* as temporal sequences of these events and associates these streams into a higher level representation, that is, an *association* stream according to the proximity in location. Focus-of-Attention plans *SIG*'s movement based on the status of streams. The precedence is An association stream, a visual stream, and an auditory stream in decreasing order. Motor Control is activated by the Focus-of-Attention module and generates PWM (Pulse Width Modulation) signals to DC motors. Viewer shows the status of auditory, visual and association streams in the radar and scrolling windows. The whole system works in real-time with a small latency of 500 ms by distributed processing with 4 PCs and combination of Gigabit and Fast Ethernet.

Strream Formation and Association: Association synchronizes the results (events) given by other modules. It forms an auditory, visual or associated stream by their proximity. Events are stored in the short-term memory only for 2 seconds. Synchronization process runs with the delay of 200 msec, which is the largest delay of vision module.

An auditory event is merged to the nearest auditory stream within $\pm 10^\circ$ and with similar harmonic pitch. A visual event is connected to the nearest visual stream within 40 cm and with common face ID. In either case, if there are plural candidates, the most reliable one is selected. If any appropriate stream is found, such an event becomes a new stream. In case that no event is connected to an existing stream, such a stream remains alive for up to 500 msec. After 500 msec of keep-alive state, the stream terminates.

An auditory and a visual streams are associated if their direction difference is within $\pm 10^\circ$ and this situation continues for more than 50% of the 1 sec period. If either auditory or visual event has not been found for more than 3 sec, such an associated stream

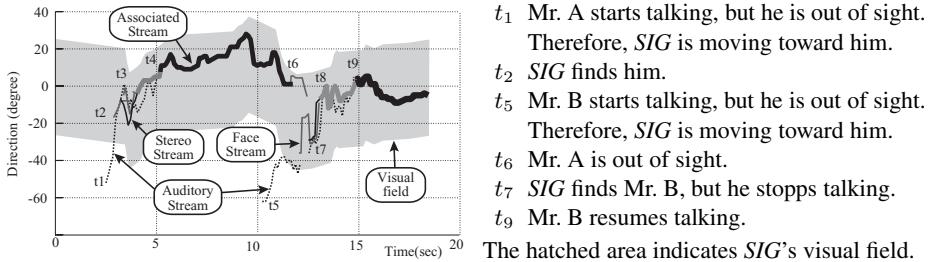


Fig. 4. Stream formation and association during tracking of two talkers

is deassociated and only existing auditory or visual stream remains. If the auditory and visual direction difference has been more than 30° for 3 sec, such an associated stream is disassociated to two separate streams.

A snapshot of stream formation and association is depicted in Fig. 4. The scenario of the experiment with two talkers is described in the same figure. Stream formation and association occur as follows:

- t_1 : An auditory stream of Mr. A is created, and *SIG* turns to its direction.
- t_2 : *SIG* finds Mr. A and a face and a stereo vision stream are created.
- t_3 : These three streams are associated.
- t_4 : *SIG* tracks Mr. A based on the associated stream.
- t_5 : Since Mr. B starts talking from out of *SIG*'s visual field, an auditory stream of Mr. B is created. *SIG* turns to its direction to obtain further information on it.
- t_6 : Since Mr. A becomes out of sight, its associated stream is disassociated and only its auditory stream remains.
- t_7 : Since Mr. B is in sight, its face and a stereo vision stream are created. Since he stopps talking, association does not occur.
- t_8 : Its face and a stereo vision stream are associated.
- t_9 : Since Mr.B resumes talking, all streams of Mr. B are associated.

3 Sound Source Localization

Sound source localization in the system uses auditory epipolar geometry without using HRTFs. In addition, several sound clues are integrated to make it more robust. The rest of this section describes the flow of the sound source localization in detail.

Peak Extraction and Sound Source Separation: First, a STFT (Short-Time Fourier Transform) is applied to the input sound. A peak on spectrum is extracted by a subband selection, which a subband with a frequency between 90 Hz and 3 KHz is selected if its power is a local maximum and more than the threshold. This threshold is automatically determined by stable auditory conditions of the room. Then, extracted peaks are clustered according to *harmonicity*. A frequency of F_n is grouped as an overtone (integer multiple) of F_0 if the relation $| \frac{F_n}{F_0} - \lfloor \frac{F_n}{F_0} \rfloor | \leq 0.06$ holds. The constant, 0.06, is determined by trial and error. By applying an Inverse FFT to a set of peaks in harmonicity, a harmonic sound is reconstructed, and thus separated from a mixture of sounds.

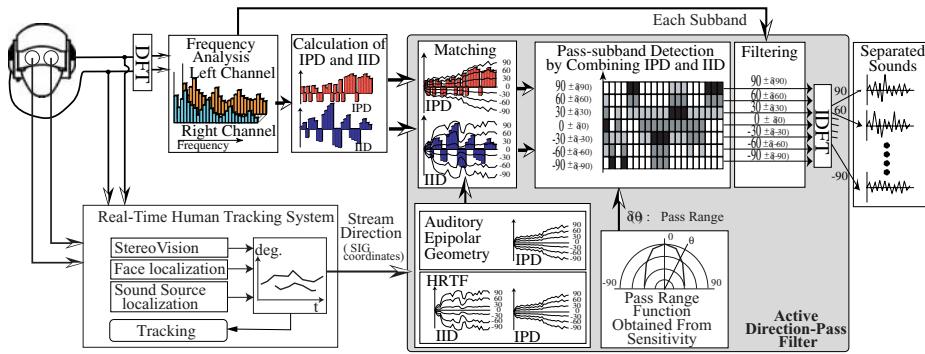


Fig. 5. Active Direction-Pass Filter

Sound Source Localization by Integration: Sound source localization in the real world consists of four stages of processing, i.e.,

1. localization by interaural phase difference (IPD) and auditory epipolar geometry,
2. localization by interaural intensity difference (IID),
3. integration of overtones, and
4. integration of 2. and 3. by Dempster-Shafer theory.

For localization by IPD, a hypothesis of the IPD for each 5° candidate is generated by auditory epipolar geometry. The distance between each hypothesis and the IPD of the input sound is calculated. IPDs of all overtones are summed up by using a weighted function. It is converted into belief factor B_P by using a probability density function.

For localization by IID, by calculating summation of IID of all overtones, belief factors supported by the left, front, and right direction are estimated.

Thus, sound directions are estimated by IPD and by IID with belief factors. Then, the belief factors of B_P and B_I are integrated into a new belief factor of B_{P+I} supported by both of them using Dempster-Shafer theory defined by

$$B_{P+I}(\theta) = B_P(\theta)B_I(\theta) + (1 - B_P(\theta))B_I(\theta) + B_P(\theta)(1 - B_I(\theta)). \quad (1)$$

Finally, an auditory event consisting of pitch (F_0) and a list of 20-best directions (θ) with reliability factors and observation times for each harmonics is generated.

4 Active Direction-Pass Filter

To improve auditory processing in the real world, the ADPF based on auditory epipolar geometry, which is shown in Fig. 5, uses techniques such as calculation of IPD and IID without using HRTFs and control of pass range taken the sensitivity of the direction-pass filter into account. The algorithm is described as follows:

1. Direction of a stream with current attention is obtained from Association.
2. Because the stream direction is obtained in world coordinates, it is converted into azimuth θ_s in the SIG coordinate system by considering latency of processing.

3. The pass range $\delta(\theta_s)$ of the ADPF is selected according to θ_s . The pass range function δ has a minimum value in the *SIG* front direction, because it has maximum sensitivity. δ has a larger value at the side directions because of lower sensitivity. Let us $\theta_l = \theta_s - \delta(\theta_s)$ and $\theta_h = \theta_s + \delta(\theta_s)$.
4. The IPD $\Delta\varphi_E(\theta)$ and IID $\Delta\rho_E(\theta)$ are calculated for each subband by auditory epipolar geometry. The IPD $\Delta\varphi_H(\theta)$ and IID $\Delta\rho_H(\theta)$ are obtained from HRTFs.
5. Peaks are extracted from the input, and IPD $\Delta\varphi'$ and IID $\Delta\rho'$ is calculated.
6. The sub-bands are collected if the IPD and IID satisfy the specified condition. Four kinds of conditions shown below are investigated:
 - A:** if $f < f_{th}$ then $\Delta\varphi_E(\theta_l) \leq \Delta\varphi' \leq \Delta\varphi_E(\theta_h)$,
 - B:** if $f \leq f_{th}$ then $\Delta\varphi_H(\theta_l) \leq \Delta\varphi' \leq \Delta\varphi_H(\theta_h)$ else $\Delta\rho_H(\theta_l) \leq \Delta\rho' \leq \Delta\rho_H(\theta_h)$
 - C:** $\Delta\varphi_E(\theta_l) \leq \Delta\varphi' \leq \Delta\varphi_E(\theta_h)$ for every frequency,
 - D:** if $f \leq f_{th}$ then $\Delta\varphi_E(\theta_l) \leq \Delta\varphi' \leq \Delta\varphi_E(\theta_h)$ else $\Delta\rho_H(\theta_l) \leq \Delta\rho' \leq \Delta\rho_H(\theta_h)$.
 The f_{th} is the upper boundary of frequency which is efficient for localization by IPD. It depends on the baseline of the ears. In *SIG*'s case, the f_{th} is 1500 Hz.
7. A wave consisting of collected sub-bands is constructed.

Note that the direction of an association stream is specified by visual information not by auditory one to obtain more accurate direction.

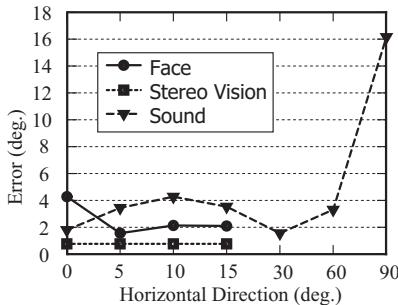


Fig. 6. Error of localization by Face, Stereo Vision and Sound

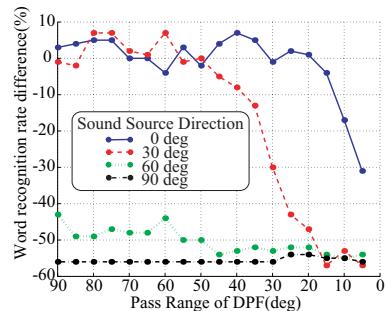


Fig. 7. Difference of recognition rate by sound source direction

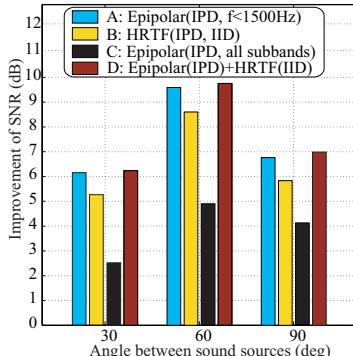


Fig. 8. Front speech extraction

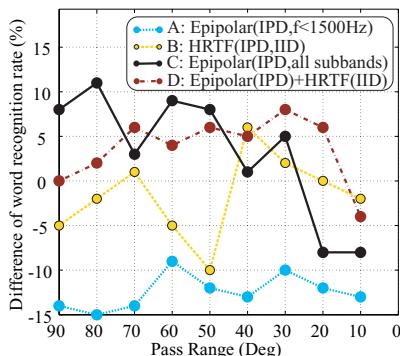


Fig. 9. recognition rates by filtering methods

5 Experiments

The performance of the ADPF is evaluated by two experiments. In these experiments, *SIG* and loud speakers are located in a room of 10 square meters. The distance and elevation between *SIG* and the speakers is about 80cm and -40° , respectively. The direction of a loud speaker is represented as 0° for *SIG* front direction.

A metric for evaluation is difference of SNR (signal-noise ratio) between input and separated speech defined by Eq. 2. For speech data, 20 sentences read by men and women from the Mainichi Newspapers are used.

$$SNR = 10 \log_{10} \frac{\sum_n (s(n) - \beta s_o(n))^2}{\sum_n (s(n) - \beta s_s(n))^2} \quad (2)$$

where, $s(n)$, $s_o(n)$, and $s_s(n)$ are the original signal, the signal observed by robot microphones and the signal separated by the ADPF, respectively. β is the attenuation ratio of amplitude between original and observed signals.

Experiment 1: The errors of sound source localization of Sound, Face and Stereo Vision are measured. The result is shown in Fig. 6 when sound source direction varies from 0° to 90° .

Experiment 2: The first loud speaker is fixed at 0° , the second one is located in 30° , 60° and 90° of *SIG*. Two loud speakers play different speeches with the same loudness simultaneously. Speech from the first loud speaker is extracted by the ADPF. The filter pass range function $\delta(\theta)$ is defined by localization errors obtained from *Experiment 1*. Fig. 8 shows the improvement of SNR by the ADPF by each filtering condition **A** to **D**.

Observation: Sound source localization by Stereo Vision is the most accurate shown in Fig. 6. The error is within 1° . Generally, localization by vision is more accurate than by audition. However, Sound has the advantage of an omni-directional sensor. That is, Sound can estimate the direction of sound from more than $\pm 15^\circ$ of azimuth. The sensitivity of localization by Sound depends on sound source direction. It is the best in the front direction. The error is within $\pm 5^\circ$ from 0° to 30° , and it is getting worse at more than 30° . This proves that active motion such as turning to face a sound source improves sound source localization.

The filtering condition **D** has the best improvement of SNR regardless of direction between two sound sources in Fig. 8. This shows that the efficiency of the ADPF is $6 - 10dB$ in case of two speakers. But separation of two speakers closer together than 30° would be more difficult. The improvement by the filtering condition **B** based on HRTFs is lower than by the filtering condition **A** and **D** based on auditory epipolar geometry. This proves the efficiency of auditory epipolar geometry in sound source separation under real environments. The difference of improvement between the filtering condition **A** and **D** is small, because subbands with frequencies of more than 1500Hz collected by IID have lower power. But, it is expected that the difference of recognition rate becomes bigger in case of automatic speech recognition (ASR), because ASR uses information from subbands with higher frequencies. In case of the filtering condition **C**, most subbands with more than 1500Hz are collected because of a limitation of the baseline between the *SIG*'s ears. Therefore, the improvement of SNR is not so big.

6 Conclusion

This paper reports sound source separation by an ADPF connected with a real-time multiple speaker tracking system. The ADPF with adaptive sensitivity control is shown to be effective in improving sound source separation. The sensitivity of the ADPF has not been reported so far in the literature and the idea of the ADPF resides in active motion to face a sound source to make the best use of the sensitivity. The combination of most up-to-date robust automatic speech recognition with the ADPF filter is one of exciting future work. The authors thank Dr. Hiroaki Kitano, the Project Director of Kitano Symbiotic Systems Project, JST. This research was partially supported by MEXT Grant-in-Aid for Informatics, No.14019051, and COE program of MEXT for Informatics Research Center at Kyoto University.

References

1. ALOIMONOS, Y., WEISS, I., AND BANDYOPADHYAY., A. Active vision. *International Journal of Computer Vision* 1, 4 (1987), 333–356.
2. ASANO, F., GOTO, M., ITOU, K., AND ASOH, H. Real-time sound source localization and separation system and its application to automatic speech recognition. *Proc. of International Conf. on Speech Processing (Eurospeech 2001)*, ESCA, 1013–1016.
3. BELL, A.J., AND SEJNOWSKI, T.J. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation* 7, 6 (Jun 1995), 1129–1159.
4. BREAZEAL, C., AND SCASSELLATI, B. A context-dependent attention system for a social robot. *Proc. of 16th Intern'l Joint Conf. on Artificial Intelligence (IJCAI-1999)*, 1146–1151.
5. HIDAI, K., MIZOGUCHI, H., HIRAKO, K., TANAKA, M., SHIGEHARA, T., AND MISHIMA, T. Robust face detection against brightness fluctuation and size variation. *Proc. of IEEE/RAS Intern'l Conf. on Intelligent Robots and Systems (IROS 2000)*, IEEE, 1397–1384.
6. HIRAKO, K., HAMAHIRA, M., HIDAI, K., MIZOGUCHI, H., YOSHIZAWA, S. Fast algorithm for online linear discriminant analysis. *Proc. of ITC-2000*, IEEE, 274–277.
7. MATSUSAKA, Y., TOJO, T., KUOTA, S., FURUKAWA, K., TAMIYA, D., HAYATA, K., NAKANO, Y., AND KOBAYASHI, T. Multi-person conversation via multi-modal interface — a robot who communicates with multi-user. *Proc. of 6th European Conf. on Speech Communication Technology (EUROSPEECH-1999)*, ESCA, 1723–1726.
8. MIZUMACHI, M., AND AKAGI, M. Noise reduction by paired-microphones using spectral subtraction. *Proc. of 1998 Intern'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP-98)* (1998), IEEE, 1113–1116.
9. NAKADAI, K., LOURENS, T., OKUNO, H.G., AND KITANO, H. Active audition for humanoid. *Proc. of 17th National Conf. on Artificial Intelligence (AAAI-2000)*, 832–839.
10. NAKADAI, K., HIDAI, K., MIZOGUCHI, H., OKUNO, H.G., AND KITANO, H. Real-time auditory and visual multiple-object tracking for robots. *Proc. of 17th Intern'l Joint Conf. on Artificial Intelligence (IJCAI-2001)*, IJCAI, 1425–1432.
11. NAKADAI, K., OKUNO, H.G., KITANO, H. Exploiting auditory fovea in humanoid-human interaction. *Proc. of 18th National Conf. on Artificial Intelligence (AAAI-2002)*, 431–438.
12. OKUNO, H., NAKADAI, K., HIDAI, K., MIZOGUCHI, H., AND KITANO, H. Human-robot non-verbal interaction empowered by real-time auditory and visual multiple-talker tracking. *Advanced Robotics* 17, 2 (2003), *in print*.
13. SARUWATARI, H., KAJITA, S., TAKEDA, K., AND ITAKURA, F. Speech enhancement using nonlinear microphone array based on complementary beamforming. *IEICE Trans. Fundamentals E82-E*, 8 (Aug. 1999), 1501–1510.

Hierarchical Neuro-Fuzzy Systems

M. Vellasco^{1,2}, M. Pacheco^{1,2}, K. Figueiredo¹

¹ Applied Computational Intelligence Lab

Department of Electrical Engineering – PUC-Rio

Rua Marquês de São Vicente, 225, Rio de Janeiro, 22453-900 RJ, Brazil

marley@ele.puc-rio.br

² Department of Computer and Systems Engineering - UERJ

Abstract. This work introduces a new class of neuro-fuzzy models called Hierarchical Neuro-Fuzzy BSP Systems (HNFB). These models employ the BSP partitioning (Binary Space Partitioning) of the input space and has been developed in order to bypass the traditional drawbacks of neuro-fuzzy systems: the reduced number of allowed inputs and the poor capacity to create their own structure. First the paper briefly introduces the HNFB model based on supervised learning algorithm. Then it details the RL_HNFB model, which is a hierarchical neuro-fuzzy system with reinforcement learning process. The RL_HNFB model was evaluated in a benchmark application – mountain car – yielding good performance when compared with different reinforcement learning models.

1 Introduction

Neuro-Fuzzy Systems (NFS) [1-6] are hybrid systems which match the learning capability of artificial neural networks (ANNs) [7] with the linguistic interpretation power of the Fuzzy Inference Systems (FISs) [8]. The basic idea of a Neuro-Fuzzy System is to implement a Fuzzy Inference System in a distributed parallel architecture in such a way that the learning paradigms used in the neural networks can be employed in this hybrid architecture.

This work proposes the use of recursive partitioning methods of the input/output space, resulting in a new class of neuro-fuzzy system, called Hierarchical Neuro-Fuzzy (HNF) [6] that overcomes two of the main weaknesses of current Neuro-Fuzzy Systems: their fixed structure and limited number of inputs. The hierarchical neuro-fuzzy models presented in this article use the BSP – Binary Space Partitioning.

This article is organized in 5 additional sections. Section 2 briefly describes the BSP partitioning scheme. Section 3 introduces the supervised HNFB model, describing its basic cell, its architecture and its learning method. Sections 4 details the reinforcement learning model (RL_HNFB) and section 5 presents the results obtained with the mountain car application. Finally, section 6 presents the conclusions of this work.

2 BSP partitioning

The BSP – Binary Space Partitioning, which is very common in image compression applications [9-10], divides the space successively, in a recursive way, in two regions. Figure 1 shows an example of a two dimensional input partitioned using the BSP method.

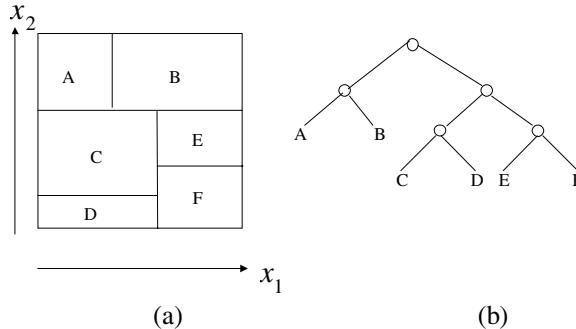


Fig. 1. (a) BSP Partitioning. (b) BSP Tree representing the BSP partitioning of Fig 1(a).

The BSP partitioning is flexible and minimizes the problem of the exponential growth of rules, since it only creates new rules locally, according to the training set. Its main advantage is to build automatically its own structure. This type of partitioning is considered recursive because it uses recursive processes in its generation, which results in models with hierarchy in their structure and, consequently, hierarchical rules.

3 Hierarchical Neuro-Fuzzy BSP (HNFB) Model

3.1 Basic Neuro-Fuzzy BSP (Binary Space Partitioning) Cell

An HNFB (Hierarchical Neuro-Fuzzy BSP) cell is a neuro-fuzzy mini-system that performs binary fuzzy partitioning of the input space. The HNFB cell generates a precise (crisp) output after a defuzzification process.

Figure 2(a) illustrates the cell's functionality. In this cell, 'x' represents the input variable, while $\rho(x)$ and $\mu(x)$ are the membership functions *low* and *high*, respectively, which generate the antecedents of the two fuzzy rules. Figure 2(b) illustrates the representation of this cell in a simplified manner and Figure 2(c) presents the main blocks of the neuro-fuzzy inference system: *fuzzification*, *rules/inference* and *defuzzification*.

The linguistic interpretation of the mapping implemented by the HNFB cell (see Figure 2(c)) is given by the following set of rules:

- Rule 1: If $x \in \rho$ then $y = d_1$.
- Rule 2: If $x \in \mu$ then $y = d_2$.

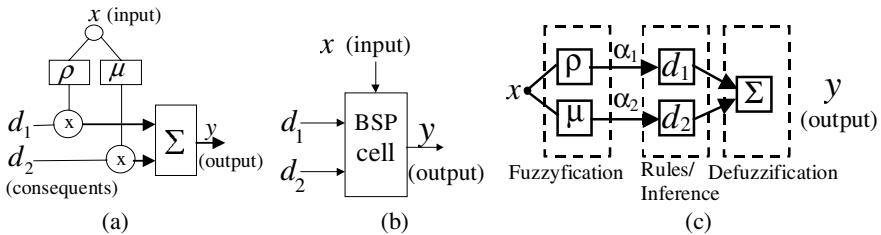


Fig. 2. (a) Interior of the Neuro-Fuzzy BSP cell. (b) Simplified Neuro-Fuzzy BSP Cell. (c) HNFB cell as a mini neuro-fuzzy inference system.

Each rule corresponds to one of the two partitions generated by the BSP partitioning. When the inputs belong to partition 1 (*low* part of the partition), it is rule 1 that has a higher firing level. When they are incident to partition 2 (*high* part of the partition), it is rule 2 that has a higher firing level. Each partition can in turn be subdivided into two parts by means of another HNFB cell.

The profiles of membership functions $\rho(x)$ and $\mu(x)$ are sigmodal with parameters a and b that determine, respectively, the fuzzy set inclination and its inflexion point.

The (crisp) output ‘ y ’ of an HNFB cell (defuzzification process) is given by the weighted average. Due to the fact that the membership function $\rho(x)$ is the complement to 1 of the membership function $\mu(x)$ in the HNFB cell the following equation applies:

$$y = \rho(x)*d_1 + \mu(x)*d_2 \text{ or } y = \sum_{i=1}^2 \alpha_i * d_i. \quad (1)$$

where the α_i ’s symbolize the firing level of the rules and are given by: $\alpha_1 = \rho(x)$; $\alpha_2 = \mu(x)$ (see Figure 2(c)).

Each d_i of equation (1) corresponds to one of the three possible consequents below:

- A **singleton**: The case where $d_i = \text{constant}$.
- A **linear combination** of the inputs: The case where $d_i = \sum_{k=1}^n w_k x_k + w_0$.
- The **output of a stage** of a previous level: The case where $d_i = y_j$, where y_j represents the output of a generic cell ‘ j ’, whose value is also calculated by eq. (1).

3.2 HNFB Architecture

An HNFB model may be described as a system that is made up of interconnections of HNFB cells. Figure 3 illustrates an HNFB system along with the respective partitioning of the input space. In this system, the initial partitions 1 and 2 (‘BSP 0’ cell) have been subdivided; hence, the consequents of its rules are the outputs of subsystem 1 and subsystem 2, respectively. In turn, these subsystems have, as

consequents, values d_{11} , y_{12} , d_{21} and d_{22} , respectively. Consequent y_{12} is the output of the ‘BSP 12’ cell. The output of the system in figure 3(a) is given by equation 2.

$$y = \alpha_1 \cdot (\alpha_{11} \cdot d_{11} + \alpha_{12} \cdot (\alpha_{121} \cdot d_{121} + \alpha_{122} \cdot d_{122})) + \alpha_2 \cdot (\alpha_{21} \cdot d_{21} + \alpha_{22} \cdot d_{22}) \quad (2)$$

It must be stressed that, although each BSP cell divides the input space only in two fuzzy set (*low* and *high*), the complete HNFB architecture divides the universe of discourse of each variable in as many partitions as necessary. The number of partitions is determined during the learning algorithm. In Figure 3(b), for instance, the upper left part of the input space (partition 12 in gray) has been further subdivided by the horizontal variable x_1 , resulting in three fuzzy sets for the complete universe of discourse of this specific variable.

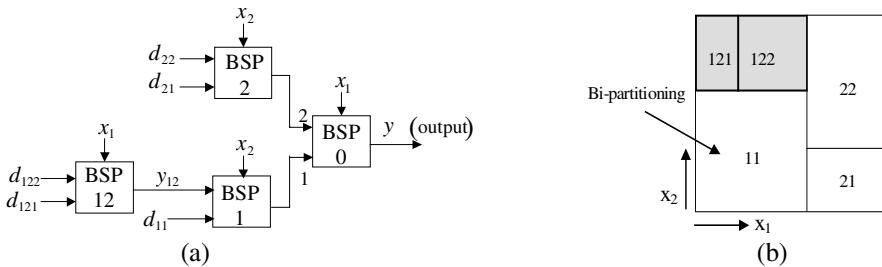


Fig. 3. (a) Example of an HNFB system. (b) Input space Partitioning of the HNFB system. (c) Final partition of the variable x_1 , where ρ_i and μ_i are related to membership functions of partition i .

3.3 Learning Algorithm

The HNFB system has a training algorithm based on the gradient descent method [6] for learning the structure of the model and consequently, the linguistic rules. The parameters that define the profiles of the membership functions of the antecedents and consequents are regarded as the fuzzy weights of the neuro-fuzzy system. Thus, the d_i 's and parameters ‘a’ and ‘b’ are the fuzzy weights of the model.

In order to prevent the system’s structure from growing indefinitely, a parameter, named decomposition rate (δ), was created. It is an adimensional parameter and acts as a limiting factor for the decomposition process. Information on how this algorithm functions may be found in greater detail in [4][12].

In order to demonstrate the potential of the HNFB model in problems with a larger number of inputs than what is normally found in neuro-fuzzy system applications it has been applied to classification and forecasting time series problems. The results obtained have been presented in [5] and [11-12].

4 Reinforcement Learning Neuro-Fuzzy Hierarchical BSP

This new model descends from the hierarchical neuro-fuzzy models NFHB [12], which use supervised learning. Using this partitioning method, together with the

Reinforcement Learning, a new class of Neuro-Fuzzy Systems (SNF) was obtained, which executes, other than the structure learning, the autonomous learning of action to be taken by an agent when it is on a given state. These characteristics represent an important differential when compared to the existing intelligent agents learning systems.

In this new hybrid model, the reinforcement learning process is based on the SARSA[13] method. The state identification process, which is not previously known, is accomplished by the learning phase.

The RL-NFHB model is composed of one or various standard cells called RL-neuro-fuzzy bsp (RL-NFB). These cells are laid out in a hierarchical structure with the shape of a binary tree. The following sections describe the basic cell, the hierarchical structure and the learning algorithm. Section 5 presents the mountain car case study whose results show the good applicability of this model in the control area.

4.1 RL Neuro-Fuzzy BSP Basic Cell

The RL_HNFB basic cell follows the same patterns defined by the supervised NFHB basic cell. However, though the two consequents of the cell are singleton or the output of a previous level stage, they are not known in advance, that is, there is not a target output for each input pattern. A set of actions (a_1, a_2, \dots, a_n) is related to each consequent, and each action is associated to a Q value function. By means of RL-based learning method, one action of each bi-partition (a_i and a_j) will be defined as the one representing the desired system behavior whenever the system is in a given state. Figure 4 shows an RL-NFB cell, where each bi-partition is related to a set of actions.

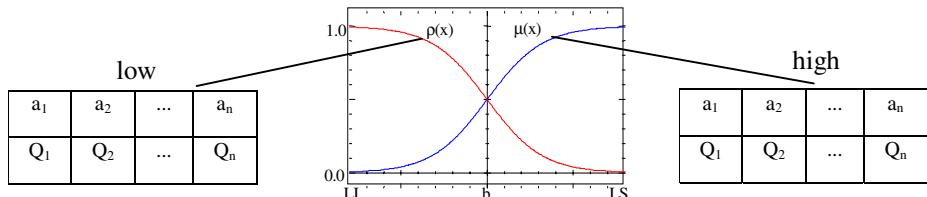


Fig. 4. Interior of the RL-NFB cell

4.2 RL-HNFB Architecture

The RL-HNFB architecture is made up of interconnected RL-HNFB cells, following the same pattern of the architecture of the model NFHB. For this model (RL-HNFB), the input variables are alternated at every hierarchical level of the structure.

4.3 Learning Algorithm of RL Hierarchical Neuro-Fuzzy BSP

The flowchart in figure 5 describes the RL-NFHB model's learning algorithm. The 6 steps highlighted in the flowchart are described below.

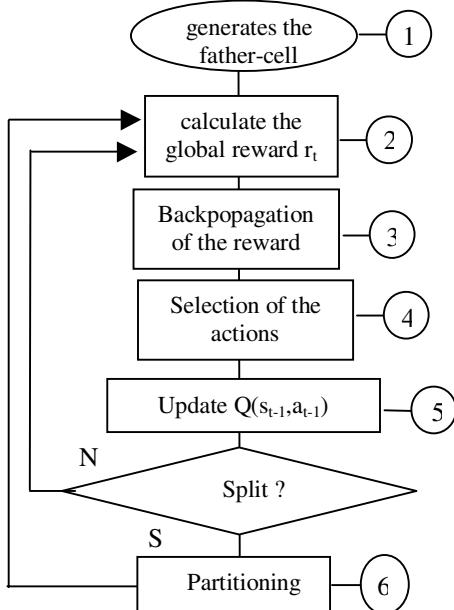


Fig. 5. Learning Algorithm of the RL-NFHB

The learning process starts with the definition of the relevant inputs for the system/environment where the agent is and the sets of actions it may use in order to achieve its objectives. Initial Q-values associated to the actions must also be defined in advance. Applications using SARSA or Q-Learning usually start their Q-values at zero.

The agent must run many cycles to ensure learning in the system/environment where the agent is. A cycle is defined by the number of steps the agent takes in the environment, which extends from the point he is initiated to the target point.

Learning Steps:

Step 1 - Generating a father-cell (or root-cell)

A root-cell is created with fuzzy sets whose domain is equal to the universe of discourse of the cell's input variable x . Then, the (normalized) x value is evaluated in the high and low fuzzy sets, resulting in two membership degrees, $\rho(x)$ e $\mu(x)$, respectively. Each bi-partition (high and low) chooses one of the actions (from its set of actions), based on the methods described in step 4 of this algorithm. The cell output is calculated by the defuzzification process given by eq. 1 and represents the action that will be executed by the agents' actuators.

Step 2 - Global Reward:

After the action is carried out, the environment is read once again. This reading enables calculation of the environment reward value that will be used to evaluate the action taken by the agent. This value must be calculated by means of an evaluation function defined according to the agent's objectives, making for a more efficient process of guiding this agent during the learning.

Step 3 - Rewards Backpropagation:

At each step, the reinforcement is calculated for each partition of all active cells, by means of its participation in the resulting action. Thus, the environment reinforcement calculated by the evaluation function is backpropagated from the root-cell to the leaf-cells, according to figure 6 and to the calculations shown in equations 3.

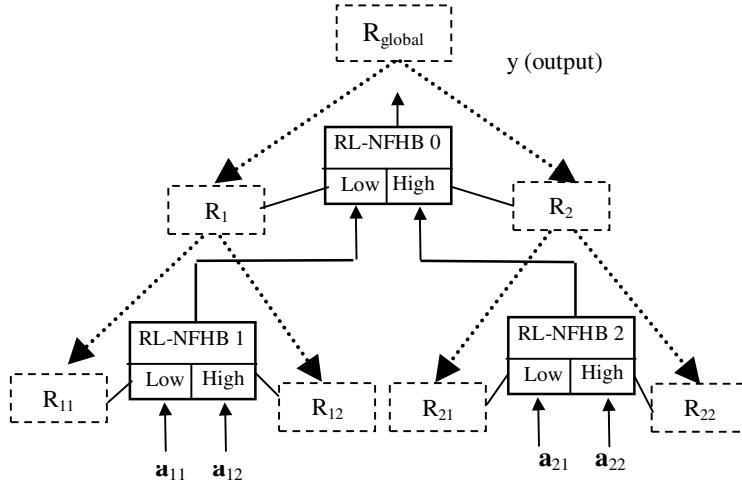


Fig. 6. Backpropagation of the global reinforcement

$$\begin{aligned}
 R_1 &= \rho(x_1) R_{\text{global}} \\
 R_2 &= \mu(x_1) R_{\text{global}} \\
 R_{11} &= \rho(x_2) R_1 \\
 R_{12} &= \mu(x_2) R_1 \\
 R_{21} &= \mu(x_2) R_2 \\
 R_{22} &= \rho(x_2) R_2
 \end{aligned} \tag{3}$$

where R_{global} is the global reward.

Step 4 - Selection of the actions:

The actions are associated to Q-value functions and compose a set of actions that are selected and experimented during the RL learning. Exploring the space of states is key to discover actions that correspond to the best agent response (looking at achieving an objective) when the agent is in a given state of the environment. So, the ϵ -greedy [13] method was used, which selects an action associated to the highest expected Q-value with $(1 - \epsilon)$ probability; and with ϵ probability it randomly selects any action. The maximum value of ϵ is 0.1 (10%) [13].

Step 5 - Update $Q(s_{t-1}, a_{t-1})$:

From the value of reinforcements calculated for each cell in the structure, the Q-values associated to actions that have contributed to the resulting action carried out by the agent must be updated.

This update is based on the evaluation between the current and previous global returns. The Q-values update takes place in two distinct forms: when the current

global return value is higher than the previous global return ($R_{t+1} > R_t$), the objective is to reward by increasing the Q's value; when the current global return is lower than or equal to the previous global return ($R_t \geq R_{t+1}$), the objective is to punish by reducing the Q's value.

Step 6 - Partitioning:

In order to implement the partitioning, the growth variable and the growth function were created to allow or prevent structure growth. As the cells are updated, the Q value variation percentage (ΔQ) of the actions associated to low and high bi-partition of the cell are observed. When the Q value variation associated to the updated action is greater than a percentage of the greatest variation ever for this cell partition, this bi-partition is considered to present growth potential, and the growth variable is increased; otherwise, it is reduced. In the first case, this variation may indicate that the actions that are being taken in this bi-partition are not adequate for the sub-domain relative to this bi-partition.

When a bi-partition has all the necessary requirements for partitioning, a daughter cell is created and connected to that bi-partition. Its domain will be the subdomain that corresponds to the low or high bi-partition of its closest ancestor. Daughter cells also inherit the set of actions with their respective Q values from this ancestor.

When a cell has all the descending cells, all of the ancestor cell's actions become the output actions of the daughter cells and, in this case, the ancestor cell's membership functions are voided; that is, the ancestor cell's membership degree ρ and μ will take value 1. The procedure to deactivate an ancestor cell that already has all of its descendants avoids the problem of having very low action values at the output of the structure (when compared to values of actions selected in the bi-partitions) due to the multiple layers of structure, and it does not jeopardise mapping (state-action) since the father cell domain is already represented in daughter cells with a higher degree of precision.

5 Case Studies

The mountain car [14] is a highly relevant benchmark, since it has been used by different researchers [14-18] with the purpose of testing their function approach methods.

Table 1 compares the results from the learning phases and from testing the RL-HNFB model [19] with different RL-based models described in [19].

Table 1. Comparing the RL-NFHB model with other models [18] for the test case mountain-car problem.

Model	No. of parameters	Learning Fase	Test Fase
Neural-Q-learning	4	1724	2189
CMAC Q-learning	343	262	85
FQL	25	112	61
RL-NFHB	0	131	71

The first column in table 3 indicates the model being evaluated, the second column indicates the number of parameters, that is, the number of neurons in the hidden layer for the Neural Q-Learning case, the number of grids for the CMAC case, and the number of rules for the FQL. The third and fourth columns indicate the performance obtained for each model in terms of number of steps.

Despite FQL's result being slightly better than the result obtained by the RL-NFHB model, just like the CMAC it starts from previous information (fuzzy rules and sets) relative to the learning process. Concerning the FQL model, though the RL-NFHB model uses a more elaborate return function, it concerns the objective in sight and not the learning.

6 Conclusions

The objective of this paper was to introduce a new class of neuro-fuzzy models which aims to improve the weak points of conventional neuro-fuzzy systems. The models HNFB and RL-NFHB belongs to this new class of neuro-fuzzy systems called Hierarchical Neuro-Fuzzy System.

The HNFB model was developed with the purpose of improving the weaknesses of traditional neuro-fuzzy systems. The results obtained by the HNFB model showed that the HNFB model yields a good performance as a classifier of database patterns.

This model is able to create its own structure and allows the extraction of knowledge in a fuzzy rule-base format. Another important characteristic of this class of neuro-fuzzy models is its performance in terms of processing time, which is far faster than neural network models for instance, allowing its usage in complex problems.

The RL-NFHB model was able to create and expand the structure of rules without any prior knowledge (fuzzy rules or sets); extract knowledge from the agent's direct interaction with large and/or continuous environments (through reinforcement learning), in order to learn which actions are to be carried out; and produce interpretable fuzzy rules, which compose the agent's intelligence to achieve his goal(s). The agent was able to generalize its actions, showing adequate behavior when the agent was in states whose actions had not been specifically learned. This capacity increases the agent's autonomy. Normalizing the inputs also enabled the model to adequately respond to changes in the limits of input variables. This characteristic was also very interesting, because it further increases the autonomy desired for this agent [19].

References

1. Halgamuge, S. K., Glesner,M.: Neural Networks in Designing Fuzzy Systems for Real World Applications. *Fuzzy Sets and Systems* No.65 (1994) 1-12
2. Jang, J.S.R., Sun, C.T., Mizutani, E.: *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice-Hall (1997)

3. Kruse, R., Nauck, D.: NEFFCLASS – A Neuro-Fuzzy Approach for the Classification of Data. Proc. Of the ACM Symposium on Applied Computing. Nashville (1995)
4. Souza, F.J.: Modelos Neuro-Fuzzy Hierárquicos. PhD Thesis. Department of Electrical Engineering, Pontifícia Universidade Católica do Rio de Janeiro (1999).
5. Souza, F.J., Vellasco, M.M.B.R., Pacheco, M.A.: Hierarchical Neuro-Fuzzy BSP Model – HNFB, Proceedings of the VIth Brazilian Symposium on Neural Networks – Volume 2 – (ISBN 0-7695-0856-1), IEEE Computer Society, Rio de Janeiro, RJ, 22-25 November (2000) 286
6. Souza, F.J., Vellasco, M.M.B.R., Pacheco, M.A.: Hierarchical Neuro-Fuzzy QuadTree Models, Fuzzy Sets & Systems, (ISSN 0165-0114). vol 130/2 August (2002) 189-205
7. Haykin, S.: Neural Networks – A Comprehensive Foundation, Macmillan College Publishing Company, Inc.(1999).
8. Mendel, J.: Fuzzy Logic Systems for Engineering: A Tutorial. Proceedings of the IEEE, Vol.83, n.3 (1995) 345-377.
9. Chin, M., Feiner, S.: Near Real-Time Shadow Generation Using BSP Trees. Computer Graphics, (SIGGRAPH '89 Proceedings), 23(3) July (1989) 99-106
10. Chrysanthou, Y., Slater, M.: Computing dynamic changes to BSP trees. EUROGRAPHICS 92 Proceedings, 11(3) September (1992) 321-332
11. Souza, F.J., Vellasco, M.M.B.R., Pacheco, M.A.C.: Load Forecasting with The Hierarchical Neuro-Fuzzy Binary Space Partitioning Model. International Journal of Computers Systems and Signals (ISSN 1608-5655) Inter. Assoc. for the Advancement of Methods for System Analysis and Design, South Africa (2003).
12. Vellasco, M.M.B.R., Pacheco, M.A.C., Ribeiro Neto, L.S., Souza, F.J.: Electric Load Forecasting: Evaluating the Novel Hierarchical Neuro-Fuzzy BSP Model, Inter. Journal of Electrical Power & Energy Systems, Elsevier Science Ltd (2003).
13. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. (1998).
14. Moore, A.W. Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces, Proc. 8th Int. Conf. Machine Learning, L.A., Birnbaum and G.C.Collins, eds., Morgan Kaufmann, (1991) 333-337.
15. Boyan, J.A.and Moore, A.W. Generalization in reinforcement learning: Safely approximating the value function, G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, Advances in Neural Information Processing Systems 7 (1995) Cambridge, MA, The MIT Press.
16. Gordon, G.J.: Stable function approximation in dynamic programming, In Armand Prieditis and Stuart Russell, editors, Proceedings of the Twelfth International Conference on Machine Learning, San Francisco, CA,.Morgan Kaufmann (1995).
17. Sutton, R.S.: Generalization in Reinforcement learning: Succesful examples using sparse coarse coding, In Touretzky, D.S., Mozer, M.C., and Hasselmo, M.E., editors, Advances in Neural Information Processing Systems 8 (1996) 1038-1044, MIT Press.
18. Jouffe, L.: Fuzzy Inference System Learning by Reinforcement Methods, IEEE Transactions on Systems, Man and Cybernetics. part c vol.28 n. 3 (1998).338-355.
19. Figueiredo, K.: Novos Modelos Neuro-Fuzzy Hierárquicos com Aprendizado por Reforço para Agentes Inteligentes. PhD Thesis. Department of Electrical Engineering, Pontifícia Universidade Católica do Rio de Janeiro (2003).

A functional spiking neuron hardware oriented model

Andres Upegui, Carlos Andrés Peña-Reyes, Eduardo Sanchez

Swiss Federal Institute of Technology at Lausanne, Logic Systems Laboratory,
1015 Lausanne, Switzerland
[andres.upegui, carlos.pena, eduardo.sanchez]@epfl.ch

Abstract. In this paper we present a functional model of spiking neuron intended for hardware implementation. The model allows the design of speed-and/or area-optimized architectures. Some features of biological spiking neurons are abstracted, while preserving the functionality of the network, in order to define an architecture easily implementable in hardware, mainly in field programmable gate arrays (FPGA). The model permits to optimize the architecture following area or speed criteria according to the application. In the same way, several parameters and features are optional, so as to allow more biologically plausible models by increasing the complexity and hardware requirements of the model. We present the results of three example applications performed to verify the computing capabilities of a simple instance of our model.

1 Introduction

1.1 Spiking neuron models

The human brain contains more than 10^{11} neurons connected among them in an intricate network. In every volume of cortex, thousands of spikes are emitted each millisecond. Issues like the information contained in these pulses, the code used to transmit information, or the decoding of signals by receptive neurons, have a fundamental importance in the problem of neuronal coding. They are, however, still not fully resolved.

"The neuronal signal consists of short voltage pulses called action potentials or spikes. These pulses travel along the axon and are distributed to several postsynaptic neurons where they evoke postsynaptic potentials. If a postsynaptic neuron receives several spikes from several presynaptic neurons within a short time window, its membrane potential may reach a critical value and an action potential is triggered. This action potential is the output signal which is, in turn, transmitted to other neurons." [1]

Biological neurons are extremely complex biophysical and biochemical entities. Before designing a model it is therefore necessary to develop an intuition for what is important and what can be safely neglected. The Hodgkin-Huxley model [3] describes the generation of action potentials in terms of differential equations resulting on input spike responses as shown in figure 1. In formal spiking neuron models, spikes are fully characterized by their firing time. Typical spiking neuron models as Integrate-

and-fire (I&F) and Spike Response Model (SRM) [1] have been conceived for software implementation, using kernels and numeric integration to represent the neuron response.

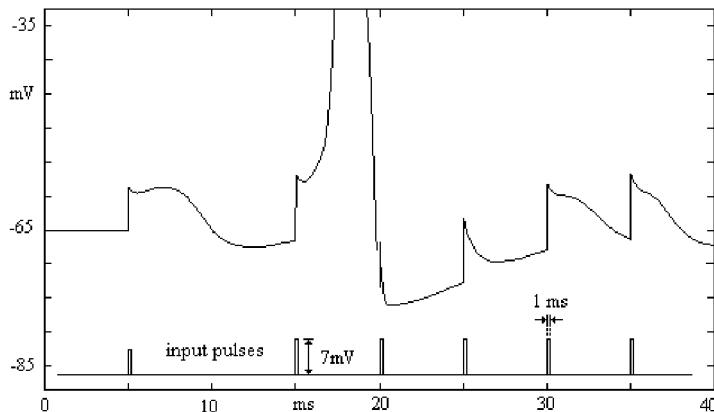


Fig. 1. Hodgkin-Huxley model response. The neuron receives input spikes at $t = 5, 15, 20 \dots 35$. The first spike is not strong enough to fire the neuron (i.e. to reach the threshold potential) and generates a postsynaptic response. The spike at time 15 is stronger, and produces a firing; the neuron enters in a refractory period, where the actions of the input spikes are negligible.

1.1 Neural Hardware

The inherent parallelism of neural network models has led to the design of a large variety of hardware implementations. Applications as image processing, speech synthesis and analysis, high energy physics, and so on, have found in neural hardware a field with promising solutions. When implemented on hardware, neural networks can take full advantage of their inherent parallelism and run orders of magnitude faster than software simulations becoming thus adequate for real-world applications [2, 4].

While hardware-neural solutions can provide real-time response and learning for networks with large numbers of neurons and synapses, even the fastest sequential processor cannot. Parallel processing with multiple simple processing elements such as SIMD multi-processors provides impressive speedups. Hardware solutions as neu-rochips (embedded neural units with relatively small complexity, but with high degree of parallelism) obtain largely higher speeds. As parallelism is expensive in terms of chip area, a compromise between speed and cost must be found.

Usually, neuron models process, as inputs and outputs, continuous values which are computed using logistic functions [4, 7, 8]. These characteristics lead to more complex models than those of spiking neurons. Spiking-neuron models process discrete values representing the presence or absence of spikes; this fact allows a simpler connectionism structure at the network level, and a striking simplicity at the neuron level as we show in the next section. Implementing models like SRM and I&F on hardware

is largely inefficient, wasting many hardware resources and adding larger delays to implement kernels and numeric integration. This is why a hardware-oriented model is necessary to achieve fast architectures at a reasonable chip area cost.

2 The proposed model

Spiking neuron models are characterized by their biological inspiration, relative simplicity, and higher capability of representation compared with traditional models of neurons [4, 7, 8]. They have an inherent capability to represent temporal dynamic patterns due to the information contained in the spike trains frequency and phase.

To exploit the coding capabilities of spiking neurons, our model includes, as other standard spiking models do, the following five elements: (1) membrane potential, (2) resting potential, (3) threshold potential, (4) postsynaptic response, and (5) after-spike response (see Figure 2). Each time a pulse, representing a spike, is received by an excitatory (or inhibitory) synapse, the counter that represents the membrane potential is increased (or decreased) a certain value to then decrease (or increase) with a constant slope until the arrival to the resting value. If a pulse arrives when a previous postsynaptic potential is still active, its action is added to the previous one. When the membrane reaches the threshold potential, the neuron fires generating an output spike while its membrane potential goes down to the after-spike potential. After firing, the neuron enters in a refractory period in which it recovers from the after-spike potential to the resting potential. Two kinds of refractoriness are allowed: absolute and partial. Under absolute refractoriness, input spikes are ignored. Under partial refractoriness, the effect of input spikes is attenuated by a constant factor.

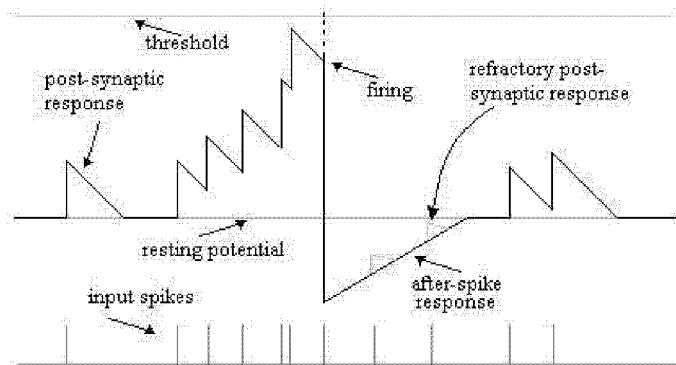


Fig. 2. Response of the model to a train of input spikes. The first input spike produces a post-synaptic response. With a train of spikes, the potential grows until it reaches the threshold potential, then the neuron fires and enters a refractory period.

The model is implemented as a Moore finite state machine, a basic hardware description technique to specify the behavior of sequential entities [6]. Two states, op-

erational and refractory, are allowed (see Figure 3). During the operational state, the neuron receives inputs from other neurons. With each input spike the potential increases (excitation) or decreases (inhibition). When the firing condition is fulfilled (i.e., potential \geq threshold) the neuron fires, the potential takes on the after-spike potential value, and the neuron passes then to the refractory state. The refractory state represents the period required by the neuron to recover from firing. When the membrane potential reaches the resting potential, the neuron passes to the operational state (see Figure 3). As mentioned before, refractoriness can be absolute or partial, depending on whether it takes or not into account new input spikes.

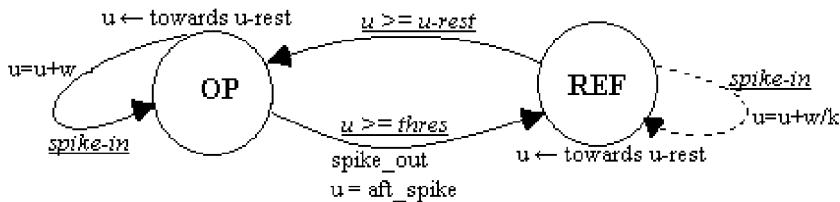


Fig. 3. Finite state machine describing the operation of the neuron. u is the membrane potential, w is the weight of the synapse receiving the spike, k is an attenuation factor, $u\text{-rest}$ is the resting potential, $thres$ is the threshold potential, aft_spike is the after-spike potential, and $spike_in$ and $spike_out$ represent respectively the post and pre synaptic events in the neuron. Underlined statements represent conditions, while not underlined represent actions.

For a hardware architecture conception, the neuron would be defined as an embedded processor where the control unit is driven by the finite state machine. The processing unit would be given by a counter with some special features, such as being incremental or decremental at different rates (post-synaptic responses, slopes for operational and refractory states) and it must allow to preset several values (after-spike potential, resting potential).

3 Experimental setup

Three experiments have been performed in order to measure the flexibility and the representation power of the model. Our goal was to test the functionality and capabilities of a network of neurons, initially with a static problem (i.e., a XOR gate), then with a simple dynamic problem (i.e., a temporal 3-pattern recognizer), and then with a more-complex dynamic problem (i.e., a temporal 10-pattern number recognizer). A generic network topology is proposed for all the problems (see figure 4). It consists on a network with 3 layers: input, hidden, and output with recurrent connections allowed only on the hidden layer. To provide the input to the network, the logical values '1' and '0' of the patterns are represented, respectively by a train of three spikes and by the absence of spikes. Due to the propagation of spikes throughout the network, the classification spikes arrive a certain time after the presentation of the full pattern.

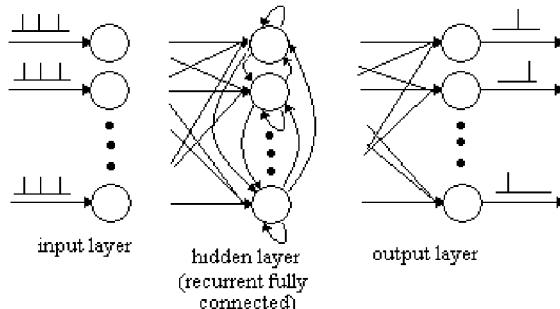


Fig. 4. Topology of the network for pattern recognition.

The patterns for the temporal pattern recognition problem are: +, \times and \diamond , drawn on a grid of 5×5 pixels (Figure 5). A pattern is presented to the network row after row. The topology of the network consists of an input layer of 5 neurons (one for each column), a recurrent hidden layer with 10 neurons, and an output layer with 3 neurons, one for each pattern (see Figure 4).

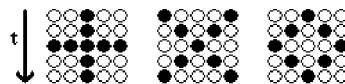


Fig. 5. Training patterns. Patterns are presented as the time flows. Each row is presented at a sample period of n iterations.

For the number-recognition problem, the ten numbers are represented with a grid of 4 columns and 5 rows (see Figure 6). The network has 8 input neurons, 30 hidden neurons, and 10 output neurons (we use the negation of the pattern as input, as it increases the amount of useful information). In order to reduce cases where several output neurons fire with the same input, we used a competitive strategy inhibiting reading outputs after a first output fires (i.e., the pattern is considered as classified).

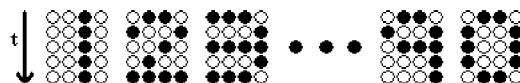


Fig. 6. Number patterns on a grid 4×5 .

3.1 Genetic weight determination

We used a simple genetic algorithm [5] to search for the connection weights. For each problem the basic genome encodes up to three groups of weights: input-to-hidden,

hidden-to-hidden and hidden-to-output (note that hidden-to-hidden weights exist only in recurrent networks). The number of neurons in the involved layers defines thus the number of weights.

For the pattern-recognition problem the genome encodes 180 synapse weights taking values from -255 to 256 using 7-bit resolution for a total genome length of 1260 bits. Crossover probability: 0.9, mutation probability: 0.001, 200 individuals and 500 generations. The fitness evaluates the classification error and penalizes high weights values (*fitness = 9 - classification error - average normalized weights*). The classification error is given by both undesired firing neurons and not-firing neurons expected to do it.

For the number-recognition problem the genome encodes, beside the weights, some neuron parameters: the resting potential, the threshold potential and the postsynaptic slope. The weights take on values from -127 to 128 with 6-bit resolution; the potential values go from 0 to 255 with 8-bit resolution, and the slope from 1 to 9 with 3-bit resolution. The total length of the genome is thus 8659 bits.

For this latter problem we used a two-stage incremental evolution to search for the solution. The first stage uses an ‘easy’ fitness criterion to perform coarse tuning; its goal is to find several individuals with acceptable performance. The second stage uses a ‘harder’ fitness criterion to finely tune the parameters of a set of the best individuals previously found which are used as initial population. The ‘easy’ fitness computes two scores: the number of patterns correctly classified without considering if the classification was ambiguous or not (see Table 1) and a second score obtained by comparing the non-fired outputs in both the desired and the obtained classification vectors. We refer to it as ‘easy’ because the fitness difference between good individuals is short. The fitness function is given by: *fitness = sum (correctly classified patterns) /10 + sum(correctly not classified patterns)/90*. The scale factors 1/10 and 1/90 normalize the expression, leading to a maximum fitness of 2.

Table 1. Criteria used to describe the quality of classifications.

Criterion	Description	Output example
Accurate classification	The desired output vector is obtained (i.e., the classification error is 0)	1 0 0 (desired) 1 0 0 (obtained)
Ambiguous classification	The desired output is activated, but another one fires too.	1 0 0 (desired) 1 0 1 (obtained)
Undetected pattern	There is no classification.	1 0 0 (desired) 0 0 0 (obtained)
Misclassification	The pattern is classified in at least one wrong class, but not in the desired one.	1 0 0 (desired) 0 1 0 (obtained)

The ‘harder’ fitness criterion has a more rugged landscape, due to a larger difference between good individuals; it penalizes harder the ambiguous classifications, while the previous strategy was very indulgent with them. The fitness evaluates each pattern and assigns a score according to the percentage of good classifications (1 point), ambiguous classifications (0.5 points), undetected classification (0.5 points), and misclassifications (0 points), and penalizes the number of output spikes generated,

in order to reduce the classification of the same pattern on several classes. The fitness is given by: (*accurate classifications* x 1) + (*ambiguous classifications* x 0.5) + (*undetected patterns* x 0.5) – (*total number of fired outputs* /50). The maximum fitness, that would be obtained with 10 accurate classifications and 10 fired outputs, values 9.8.

Each evolutionary stage was carried out using different parameters. For the first one we used a crossover probability of 0.9, a mutation probability of 0.0001, an elitism of 2, 200 individuals, and 500 generations. For the second one we used a crossover probability of 0.1, a mutation probability of 0.001, an elitism of 1, 10 individuals, and 500 generations. The second one uses atypical values for the probabilities of the genetic operators and for the size of the population since it is intended to perform a mutation-driven tuning for some good individuals.

4 Results

Several evolutionary runs were carried out for each genetic algorithm, for the pattern-recognition problem the genetic algorithm always finds a solution that classifies correctly the three patterns. The best individual has a fitness of 8.54, it means that the classification error is 0 and the mean of $\text{abs}(\text{weights})$ is 117, the minimum value is -255, and the maximum is 248, what lead us to suspect that the weight optimization is not performing very well, the evolution loses easily the best individual, because of that we introduce elitism in the number-recognition problem. For the sensitivity test we found, for 10 runs, an average accurate classification of 170.3 on 300 noisy patterns, 80.2 undetected patterns, 36.7 ambiguous classifications, and 12.8 misclassifications (It has to be considered that the sensitivity test is done on an already trained pattern without any generalization method, and the training does not takes into account any test or validation set).

For the number-recognition problem the first stage finds an individual that accurately classifies 3 patterns, the remaining 7 patterns are ambiguously classified. The achieved fitness is 1.844, what means that all patterns are correctly classified due to all expected neurons fired, but there were also 14 not expected output firings (over 100 possible firings).

At the second stage, the best individual found using the first criterion exhibits a fitness of 6.02 (3 *accurately classified* x 1 + 7 *ambiguous classification* x 0.5 – 24 *output spikes* / 50). After the fine tuning at the second stage we obtain accurate classification for 6 patterns (1, 3, 4, 6, 7, 0) and undetected pattern for the remaining 4, leading to a fitness of 7.88 (6 *accurately classified* x 1 + 4 *undetected pattern* x 0.5 – 6 *output spikes* / 50). This indicates that the network should not have the complexity required to solve the problem.

5 Conclusions and future work

We have presented a functional spiking neuron model suitable for hardware implementation, the proposed model neglects a lot of characteristics from biological and

software oriented models. Nevertheless it keeps its functionality and it is able to solve a relative complex task like temporal pattern recognition. Since the neuron model is highly simplified the lack of representation power of single neurons must be compensated by a higher number of neurons, which in terms of hardware resources could be a reasonable trade-off considering the architectural simplicity allowed by the model.

The next step is to design and test a hardware specification using a hardware description language such as VHDL, in order to compare the results with the simulated ones. The model should lead us to neuron architectures with low area requirements. An important issue to consider is the connectionism of the network since its complexity could increase dramatically as the application requires higher number of neurons.

Genetic algorithms training have demonstrated to be time consuming on simulations, in addition they require a considerable amount of memory, what makes them inappropriate for a hardware modeling (memory is extremely expensive in hardware applications), so we have to conceive a learning algorithm suitable for this model. The learning algorithm must be suitable for hardware implementation and must have the flexibility to be performed online and onchip, in order to implement it on a field programmable gate array (FPGA).

These encouraging results lead us to explore some variations on the model such as inclusion of partial refractoriness and inclusion of differential slopes in order to have a more biologic approach, evaluating the convenience of the trade-off complexity vs functionality. Variations in the search and optimization method should also be considered such as evolution of architectures, mixture of evolution and learning like weight learning with neuron parameters and network architecture evolution.

References

1. W., Kistler W. Spiking neuron models. Cambridge University Press. 2002.
2. Liao Y. Neural networks in hardware: A survey. <http://wwwcsif.cs.ucdavis.edu/~liaoy/research/NNhardware.pdf>
3. Hodgkin, A. L. and Huxley, A. F. (1952). A quantitative description of ion currents and its applications to conduction and excitation in nerve membranes. *J. Physiol. (Lond.)*, 117:500-544
4. Perez-Uribe A. Structure-adaptable digital neural networks. PhD thesis. 1999.EPFL. http://lslwww.epfl.ch/pages/publications/rcnt_theses/perez/PerezU_thesis.pdf
5. Vose M. The Simple Genetic Algorithm: Foundations and Theory
6. Mange D. and Tomassini M. (eds.) *Bio-Inspired Computing Machines*, Presses Polytechniques et Universitaires Romandes, Lausanne, Switzerland, 1998
7. Hikawa H. Frequency-based multilayer neural network with on-chip learning and enhanced neuron characteristics, *IEEE Trans. Neural Networks*, 10(3): 545-553, May 1999.
8. Maya S., Reynoso R., Torres C., Arias-Estrada M. Compact Spiking Neural Network Implementation in FPGA, *Field Programmable Logic Conference (FPL'2000)*. Austria. Pages 270-276, 2000.

SO(2)-Networks as Neural Oscillators

Frank Pasemann, Manfred Hild, Keyan Zahedi

Fraunhofer Institute for Autonomous Intelligent Systems (AiS)

Schloss Birlinghoven, D-53754 Sankt Augustin, Germany

frank.pasemann, manfred.hild, keyan.zahedi@ais.fraunhofer.de

Abstract. Using discrete-time dynamics of a two neuron network with recurrent connectivity it is shown that for specific parameter configurations the output signals of neurons can be of almost sinusoidal shape. These networks live near the Sacker-Neimark bifurcation set, and are termed SO(2)-networks, because their weight matrices correspond to rotations in the plane. The discretized sinus-shaped waveform is due to the existence of quasi-periodic attractors. It is shown that the frequency of the oscillators can be controlled by only one parameter. Signals from the neurons have a phase shift of $\pi/2$ and may be useful for various kinds of applications; for instance controlling the gait of legged robots.

1 Introduction

Aspects of discrete-time dynamics of two neuron networks with recurrent connectivity have been studied for a long time, e. g. [5], [8], [1], [2], [6]. This is because they are the simplest neural networks having non-trivial dynamical properties: for certain parameter domains one finds not only stationary attractors but oscillations of various periodicity, quasi-periodic and chaotic dynamics. There are various hysteresis phenomena observable, and different bifurcation scenarios involved. Most of these complex dynamical properties can be observed for recurrently coupled inhibitory and excitatory neurons with appropriate self-connections [6].

Biologically this setup was motivated, among others, by the Wilson-Cowan model of neural populations [9]. Therefore it is assumed that the analysis of dynamical properties of 2-neuron networks can help to understand and explain phenomena observed in artificial recurrent networks in general, and possibly also those observed in biological neural systems. Furthermore, if one is interested more in the qualitative dynamical aspects of recurrent neural networks, than it is appropriate to study discrete-time dynamics, because it reflects all of those properties, which can be observed also for some (in general higher-dimensional) continuous-time dynamical systems (probably with time-delays).

The parameterization of these systems (synaptic weights, "slow" external inputs) allows to study the appearance and destruction of attractors, the delicate balance of stability and instability, features which seem to be crucial for many of the adaptive and higher-information processing capabilities of biological

systems. A better understanding of these principles, eventually will allow alternative design methods, for instance robust neural controllers [3], in comparison to standard (and often not very efficient) learning algorithms.

Oscillatory dynamics in biological and artificial systems is of general interest, e.g. associated with various kinds of central pattern generators. But often smooth oscillations are desired, for instance when driving the legs of walking machines. For discrete-time dynamical systems, the sinusoidal shape of neural output signals is in general associated with appropriate quasi-periodic attractors. For 2-dimensional systems quasi-periodic orbits are dense on attractors which are compact 1-dimensional manifolds homeomorphic to a circle. They do appear after the system has passed a so-called Neimark-Sacker bifurcation [7]. For this reason, networks “living” near the Neimark-Sacker bifurcation set have been studied e.g. in [2], showing that the frequency of such systems can be controlled by external inputs. Here we choose a special type of 2-neuron network with this property, the weight matrix w of which corresponds to a rotation in the plane; i.e. the matrix w is an element in the *special orthogonal group* $\text{SO}(2)$.

If an attractor of a 2-dimensional system is a perfect circle (e.g. harmonic oscillator), the resulting motion is called *harmonic*; the neural output then will correspond to a perfect sine wave. To describe the deviation from this perfect waveform we introduce a special measure of harmonicity for $\text{SO}(2)$ -networks, and study frequency and harmonicity of the oscillations in dependence of the rotation angle and a scaling factor.

2 Two neuron networks

In the following the discrete-time dynamics of two neuron networks with standard additive neurons is discussed. In general it is given by a 6-parameter family of maps $f_\rho : \mathbf{R}^2 \rightarrow \mathbf{R}^2$, $\rho = (\theta_1, w_{12}, w_{11}, \theta_2, w_{21}, w_{22}) \in \mathbf{R}^6$, where θ_i denotes the bias term of neuron i , and w_{ij} the synaptic weight from neuron j to neuron i . The output of a neuron is in general given by a sigmoidal transfer function σ , which here is chosen to be the hyperbolic tangent $\sigma = \tanh$. The presented results will also hold for other types of sigmoids. In fact, choosing the standard sigmoid $\sigma(x) = (1 + e^{-x})^{-1}$ will give networks with topologically equivalent dynamics. This is due to the relation $\tanh(x) = 2 \cdot \sigma(2x) - 1$ and to a transformation of the corresponding parameters θ_i and w_{ij} , as was shown in [2], [6].

Furthermore, for convenience we will set $\theta_1 = \theta_2 = 0$ in the following. The resulting two neuron dynamics is then given by the equations

$$\begin{aligned} a_1(t+1) &:= w_{11} \sigma(a_1(t)) + w_{12} \sigma(a_2(t)), \\ a_2(t+1) &:= w_{21} \sigma(a_1(t)) + w_{22} \sigma(a_2(t)). \end{aligned} \quad (1)$$

where a_i denotes the activity of neuron i .

Following the standard procedure, first the stability properties of fixed points

$$a_i^* = \sum_{j=1}^2 w_{ij} \sigma(a_j^*) , \quad i = 1, 2 . \quad (2)$$

have to be analyzed. Recall that the origin $a^* = (0, 0)$ is always a fixed point of the dynamics (1). A fixed point a^* is *asymptotically stable* if the eigenvalues of the Jacobian $Df_\rho(a^*)$ of the dynamics f_ρ at a^* all have modulus less than one. The Jacobian $Df_\rho(a^*)$ of the dynamics (1) is given by

$$Df_\rho(a^*) = \begin{pmatrix} w_{11}\sigma'(a_1^*) & w_{12}\sigma'(a_2^*) \\ w_{21}\sigma'(a_1^*) & w_{22}\sigma'(a_2^*) \end{pmatrix}. \quad (3)$$

Stability criteria for stationary states a^* can be efficiently discussed in terms

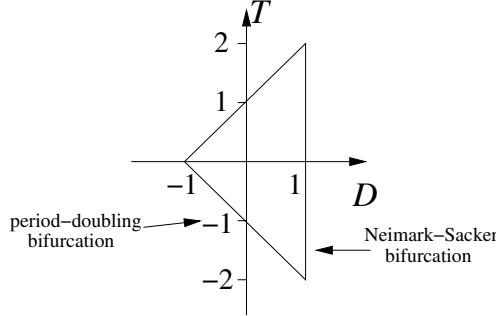


Fig. 1. The stability domain for a fixed point in terms of trace T and determinant \mathcal{D} of the Jacobian Df_ρ .

of the trace T and the determinant \mathcal{D} of $Df_\rho(a^*)$, which are here given by

$$\mathcal{T} = w_{11}\sigma'(a_1^*) + w_{22}\sigma'(a_2^*), \quad \mathcal{D} = (w_{11}w_{22} - w_{12}w_{21})\sigma'(a_1^*)\sigma'(a_2^*). \quad (4)$$

The eigenvalues of $Df_\rho(a^*)$ are then determined to be

$$\lambda_{1,2} = \frac{1}{2} (\mathcal{T} \pm \sqrt{\mathcal{T}^2 - 4\mathcal{D}}).$$

The domain of stability for a fixed point a^* in the (T, \mathcal{D}) -plane is given by a triangle bounded by the three straight lines $T - \mathcal{D} = 1$, $T + \mathcal{D} = -1$, and $\mathcal{D} = 1$ [7]. For instance, along the line $T + \mathcal{D} = -1$ there will be a *period-doubling bifurcation* from a fixed point attractor to a period-2 attractor; along the line $\mathcal{D} = 1$, $|\mathcal{T}| < 2$ there will be Neimark-Sacker bifurcations from a fixed point attractor to a highly periodic or quasi-periodic attractor [7]. Of course, this last type of bifurcation is interesting because of the different types of oscillations one has to expect for configurations where the determinant \mathcal{D} at a fixed point a^* is larger than one.

Choosing as transfer functions *tanh* and setting the bias terms $\theta_i = 0$, $i = 1, 2$, will result in the fact that the origin $a^* = (0, 0)$ is always a fixed point for the dynamics (1). Recall that the derivative of *tanh* satisfies $\tanh'(0) = 1$. Thus the Jacobian $Df_\rho(0)$ (3) is identical with the weight matrix $w = (w_{ij})$ of the network.

3 SO(2)-networks

Having identified the Jacobian $Df_\rho(0)$ at the origin with the weight matrix w of the network, it is now easy to construct networks which correspond to configurations guaranteeing that the origin as a fixed point attractor undergoes a Neimark-Sacker bifurcation. Such networks have a weight matrix w satisfying $\det w = 1$. A special type of matrices, satisfying this condition, are the elements of the *special orthogonal group* $\text{SO}(2)$. They are associated with rotations in the plane and a standard representation of these elements is given in terms of $\sin(\varphi)$ and $\cos(\varphi)$ of the rotation angle φ . Thus, convenient weight matrices are of the form

$$w = Df_\varphi(0) = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} = \begin{pmatrix} \cos(\varphi) & \sin(\varphi) \\ -\sin(\varphi) & \cos(\varphi) \end{pmatrix}, \quad (5)$$

and we now can consider the dynamics (1) as a one parameter family of maps with parameter $-\pi \leq \varphi \leq \pi$:

$$\begin{aligned} a_1(t+1) &:= \cos(\varphi) \tanh(a_1(t)) + \sin(\varphi) \tanh(a_2(t)), \\ a_2(t+1) &:= -\sin(\varphi) \tanh(a_1(t)) + \cos(\varphi) \tanh(a_2(t)). \end{aligned} \quad (6)$$

Networks with these weight matrices will always have the origin as a non-hyperbolic fixed point; i.e. the eigenvalues $\lambda_{1,2}$ of the Jacobian $Df_\varphi(0)$ satisfy $\|\lambda_{1,2}\| = 1$; they are complex numbers $\lambda_{1,2} = \cos(\varphi) \pm i \sin(\varphi)$. Varying φ from $-\pi$ to π now will keep the determinant $\mathcal{D} = 1$ and at the same time \mathcal{T} will vary between -2 and 2 , so that one moves along the line $\mathcal{D} = 1$ in $(\mathcal{T}, \mathcal{D})$ -space (figure 1).

Because here one wants to obtain almost sinusoidal output signals from the network, quasi-periodic attractors are preferred. Therefore one has to go slightly beyond the line $\mathcal{D} = 1$ crossing the Neimark-Sacker bifurcation set. To do this one may introduce a second parameter $\alpha > 1$ to obtain $\mathcal{D} > 1$ for the determinant of the Jacobian $Df_\rho(0)$, and the weight matrix of such a network is given by

$$w = Df_{(\alpha, \varphi)}(0) = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} = \alpha \cdot \begin{pmatrix} \cos(\varphi) & \sin(\varphi) \\ -\sin(\varphi) & \cos(\varphi) \end{pmatrix}, \quad (7)$$

and the eigenvalues $\lambda_{1,2}$ of the Jacobian satisfy $\lambda_{1,2} = \alpha \cdot e^{\pm i\varphi}$. For obvious reasons networks with a weight matrix of the type (7) will be called *SO(2)-networks*. The parameter α can also be understood as controlling the slope of the transfer function *tanh*; i.e. in equation 1 we may replace the hyperbolic tangent by the function $\tau_\alpha(x) = \tanh(\alpha \cdot x)$; i.e. $\tau'_\alpha(x) = \alpha \tanh'(\alpha x)$.

Staying in the parameter domains beyond the Neimark-Sacker bifurcation set, the frequency of the oscillations will change with varying α and φ . As was discussed in [2], the overall width of possible frequency range increases with α and depends crucially on φ . With $\alpha = 1.0 + \epsilon$, and $\epsilon \ll 1$ amplitudes of oscillations will be small and the waveform almost sine-shaped. With growing ϵ , i.e. growing amplitude the nonlinearities get more pronounced and waveforms get more and more distorted. This qualitative considerations are supported by the following numerical analysis.

4 Simulations

To get a first impression about the systems dynamics we depict in figure 2 the (φ, α) -parameter domains for higher periodic attractors, showing frequency locking domains around $\varphi = 0.5\pi, 1.0\pi$ corresponding to strong resonances (gray areas) of order 4 and 2, respectively. Higher order resonances, for instance 8 at $\varphi = 0.25\pi, 0.75\pi$, are observed for larger α -values. Black domains indicate quasi-periodic attractors. Setting $\alpha = 1.5$, this is for instance verified in figure 3, where the largest Liapunov exponent $L1$ is always zero outside the frequency locking domains. Also for $\alpha = 1.5$, the φ -dependence of the frequency ω of oscillations, calculated by the number of zero passages, is depicted in figure 4; it is almost linear, with frequency locking around 0.5π and π . Around $\varphi = 0$ we have only decoupled or weakly coupled neurons with super-critical self-connections resulting in four possible fixed point attractors.

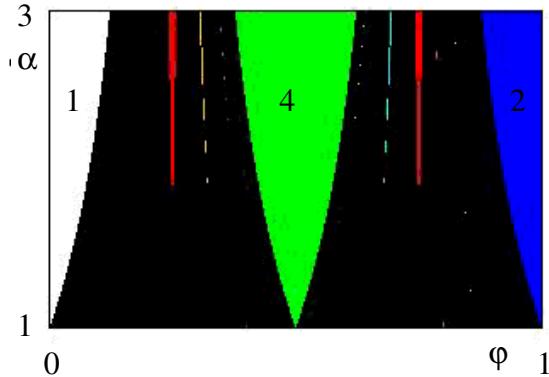


Fig. 2. (φ, α) -parameter domains for quasi-periodic attractors (black), showing frequency locking domains around $\varphi = 0.25\pi, 0.5\pi, 0.75\pi$ and π .

That for small α -values quasi-periodic attractors can be of almost circular shape is demonstrated in figure 5 for parameter values $\varphi = 0.1\pi$ and $\alpha = 1.05$. Thus the dynamics corresponds to an almost harmonic motion, and output signals of the network have almost sinusoidal shapes as can be read from figure 5b.

With increasing amplitude of the oscillations, i.e. increasing α , the wave shape will deviate from that of harmonic oscillation. To describe this the following measure of *harmonicity* h for an attractor of the $SO(2)$ -systems is introduced:

$$h := \frac{\min(A)}{\max(A)}, \quad A = \{ \|a(t)\| \mid t_c \leq t \leq t_c + N \}, \quad (8)$$

where t_c denotes the number of convergent iterations, after which the dynamics is assumed to be near the attractor, and N is the number of counting iterations,

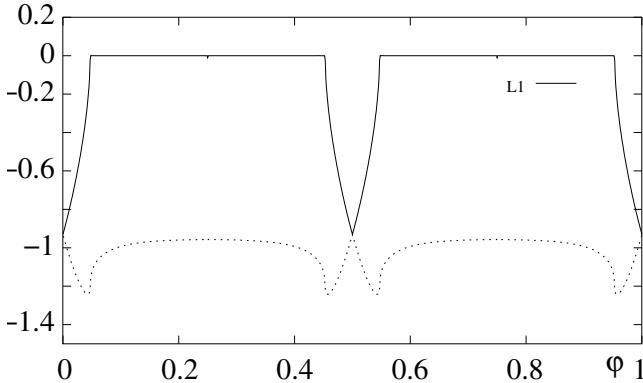


Fig. 3. Liapunov exponents for varying φ and $\alpha = 1.5$ fixed, showing that quasi-periodic attractors ($L_1 = 0$) exist everywhere outside the frequency locking domains.

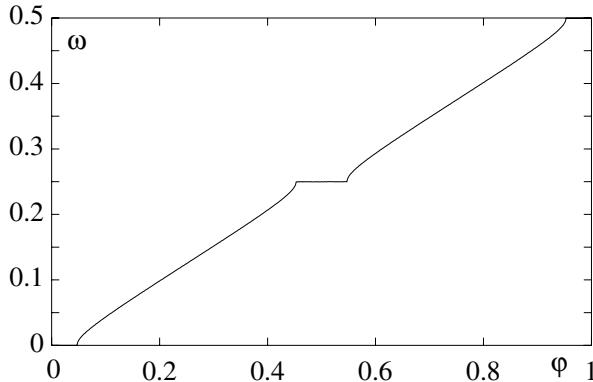


Fig. 4. Frequency ω depending on the rotation angle $0 < \varphi < \pi$ for $\alpha = 1.5$ fixed.

which should be large enough to represent a full period. Of course, $h = 1$ describes a circle, and $h = 1/\sqrt{2} = 0.707$ a perfect square. Fixing φ one observes a strong dependence of the harmonicity h of attractors on the parameter α , as in figures 6 and 7; convergent and counting iterations were set to $t_c = N = 5000$. For an angle $\varphi = 0.1\pi$ there is a smaller oscillatory α -regime - as can be seen from figure 2 - over which harmonicity h is decreasing with increasing α . After it almost reaches the $h = 0.707$, it suddenly jumps back to $h = 1$ when the stable fixed point appears. At the same time the frequency is decreasing over this α -interval (figure 6). For a larger angle $\varphi = 0.3\pi$ (figure 7) the oscillatory α -regime is much larger, and harmonicity h is decreasing in a highly nonlinear fashion with increasing α . The frequency of oscillations stays almost constant over this α -interval. Again, the decreasing harmonicity corresponds to attractors changing their shapes in output space from almost circular to almost quadratic.

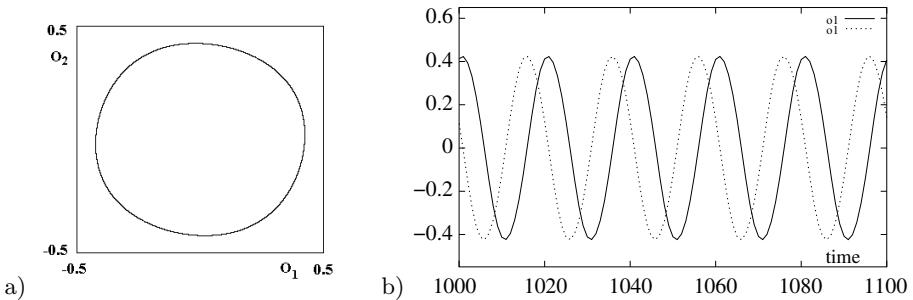


Fig. 5. a.) Attractor in (o_1, o_2) -space, and b) output signals of neurons 1 and 2 for $\alpha = 1.05$, $\varphi = 0.1 \pi$.

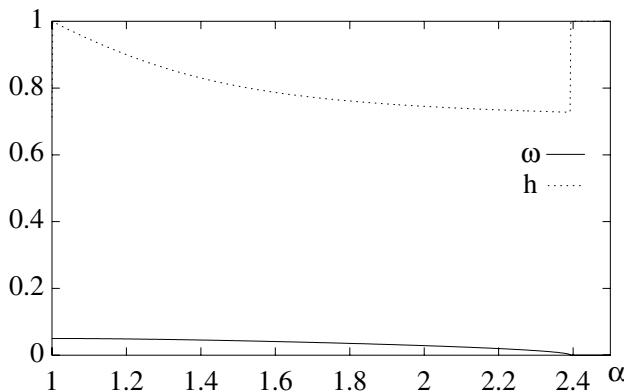


Fig. 6. Harmonicity h of attractors in dependence of α with $\varphi = 0.1 \pi$ fixed.

5 Conclusions

It was demonstrated that a specific class of 2-neuron networks, called SO(2)-networks, can be used as frequency variable oscillators producing almost sinusoidal waveforms. The frequency can be controlled over a large domain mainly by the rotation angle φ of the SO(2) weight matrix, and the waveform, characterized by the harmonicity h of the quasi-periodic attractor, can be adjusted mainly by the parameter α controlling the slope of the transfer function. Of course other variants of networks can be equally effective (see e.g. [2]), because parameters only have to guarantee that the determinant of the Jacobian is slightly larger than one. Therefore a convenient parameter may be also introduced as a modulation factor on one of the networks synaptic weights.

One possible application of these frequency variable oscillators is in the field of neural control for walking machines [4]. The described oscillators can control the type of walking and the walking speed of legged robots simply by using sensor inputs of the robot for weight or slope modulation. In addition, sensor inputs driving the neural system to and fro over the Neimark-Sacker bifurcation

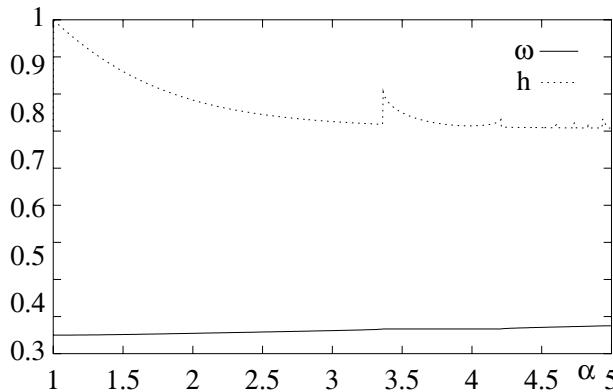


Fig. 7. Harmonicity h of attractors in dependence of α with $\varphi = 0.3 \pi$ fixed.

set will switch on and off the oscillations. Operating the neural oscillator near the bifurcation set, i.e. $\alpha = 1 + \epsilon$, $\epsilon \ll 1$, will cause long transients, and thus will result in a smooth type of switching. As an example, this will be demonstrated for a quadruped robot elsewhere.

References

- Chapeau-Blondeau, F., and Chauvet G. (1992), Stable, oscillatory, and chaotic regimes in the dynamics of small neural networks with delay, *Neural Networks*, **5**, 735–743.
- Haschke, R., Steil, J. J., and Ritter, H. (2001), Controlling oscillatory behaviour of a two neuron recurrent neural network using inputs, in: G. Dorffner, H. Bischof, K. Hornik (Eds.): *Artificial Neural Networks - ICANN 2001*, International Conference Vienna, Austria, August 21-25, 2001, Proceedings. LNCS 2130, Springer Verlag, Berlin, p. 1109 ff.
- Hülse, M., and Pasemann, F. (2002) Dynamical neural Schmitt trigger for robot control, in: J.R. Dorronsoro (Ed.): *Artificial Neural Networks - ICANN 2002*, International Conference, Madrid, Spain, August 28-30, 2002. Proceedings. LNCS 2415, Springer Verlag, Berlin, pp. 783–788.
- Kimura, H., Akiyama, S., and Sakurama, K. (1999), Realization of dynamic walking and running of the quadruped using neural oscillator, *Autonomous Robots*, **7**, pp. 247-258.
- Marcus, C. M., and Westervelt R. M. (1989), Dynamics of iterated-map neural networks, *Phys. Rev.*, **A 40**, 501-504.
- Pasemann, F. (2002) Complex dynamics and the structure of small neural networks, *Network: Computation in Neural Systems*, **13**, 195–216.
- Thompson, J. M. T., and Stewart, H. B. (1986), *Nonlinear Dynamics and Chaos*, Chichester: John Wiley.
- Wang, X. (1991) Period-doublings to chaos in a simple neural network: An analytic proof, *Complex Systems*, **5**, 425–441.
- Wilson, H. R., and Cowan, J. D. (1972), Excitatory and inhibitory interactions in localized populations of model neurons, *Biophys. J.*, **12**, 1–23.

A Rotated Kernel Probabilistic Neural Network (RKPNN) for Multi-class Classification

Ingo Galleske and Juan Castellanos

Departamento de Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid, Spain

Abstract. An improvement to the Probabilistic Neural Network (PNN) is presented that overcomes two weaknesses of the original model. In the new model, due to the fact that each neuron uses its own Gaussian kernel function, a better generalization ability is achieved by the means of stretching and rotation leading to the Rotated Kernel Probabilistic Neural Network (RKPNN). Furthermore, an algorithm is presented that calculates automatically the kernel parameters of each Gaussian function. The covariance matrices will be subdivided into two other matrices R and S that are calculated separately. This training is slower than that of the original PNN, but in its complexity comparable with other classification methods. A real-world example finally proves that the proposed model shows good generalization capacity with similar or even slightly better results than other approaches.

1 Introduction

The Probabilistic Neural Network (PNN) is a kind of neural networks that tries to resolve classification problems (i.e. the separation of the input space into different regions) that relies on the Bayes decision theory rule applied to a multi-dimensional input space. This architecture uses one unit for each training pattern and a one-pass learning process, that is independent from other patterns. As a consequence, it is very fast to train and easily extendable with new training patterns. Application can be found, for example, in the area of real-time classification [3].

Specht introduces in his original contributions [9], [10] "smoothing parameters" for the different kinds of kernel types applicable to this network architecture [7], that have a great influence on the generalization ability of the network [6], [5]. Instead of having only one "smoothing parameter" for the complete network, other researchers tried to estimate different kernel parameters for all training patterns (heteroscedastic probabilistic neural networks [11]). This paper follows this idea, proposing a constructive algorithm to calculate the kernel parameters. Nonetheless the computational complexity arises, it is comparable with other classic statistic or neural network classification methods.

The presented model allows the automatic calculation of the kernel parameters so that the network designer is freed from the task to estimate them. A

second, and also very important advantage is the enhancement of the generalization ability with similar or even better results than other methods.

This paper begins with an introduction into Probabilistic Neural Networks and its improvements that lead to the model proposed here. The recursive algorithm to calculate the kernel parameters is described in a further section. Different neural network classification model are compared on a real-world experiment, that shows the generalization ability advantage over other models.

2 Architecture of Probabilistic Neural Networks

The original Probabilistic Neural Network (PNN): Probabilistic Neural Networks are simple 3-layer networks. The input layer has n (dimensionality of input space) neurons, the pattern layer contains p (number of training patterns) units, and the class layer has one processor for each class of a given problem. The input layer is completely connected with the pattern layer, whereas the connections to the class layer exist only for those neurons, that are members of the corresponding class. The latter cannot be modified and are fixed to the reciprocal value of the amount of connections of that class. The former are learnt in a one-pass training setting each link to the value of the coordinate of the pattern in the n -dimensional input space of the classification problem.

The class layer uses the summation function as activation function, whereas the pattern layer has the kernel activation function. Considering that each neuron has as output its own activation and the Gaussian function is the kernel, then each class unit h calculates its activation for a test pattern X as follows:

$$a(h) = \sum_{i=k}^l \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \cdot \exp \left(-\frac{1}{2}(X - X_i)^T \Sigma^{-1} (X - X_i) \right) \quad (1)$$

where the units k to l participate in that specific class h , Σ is the covariance matrix of the Gaussian kernel function, X_i is the i -th training pattern. The class unit with the highest activation is defined as the winner of the classification problem, indicating the class X should be assigned to.

In this model, to set up a Probabilistic Neural Network, the network designer has to find a suitable kernel function and its associated kernel parameters for a given classification problem.

The Rotated Kernel Probabilistic Neural Network (RKPNN): The model proposed in this article uses the Gaussian kernel functions and different kernel parameters Σ_i for each pattern neuron i that are estimated automatically through the training process.

To obtain the kernel parameters for each training pattern i , the covariance matrix Σ_i is divided into two other matrices: R_i , a rotation matrix, and S_i , a diagonal matrix, in the following way:

$$\Sigma_i = R_i^T S_i S_i R_i. \quad (2)$$

The multidimensional Gaussian kernel functions have Constant Potential Surfaces (CPS) that are hyper-ellipsoids, whose principal axes are the elements of the diagonal matrix S_i and whose rotation can be represented by R_i . To achieve a good generalization capacity, the ellipsoid should be rotated (represented by the rotation matrix R) to obtain principal axes as big as possible. The following Fig. 1 demonstrates this in a three-dimensional example: The principal axes ($\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$) are rotated and stretched regarding the original coordinate system (x, y, z).

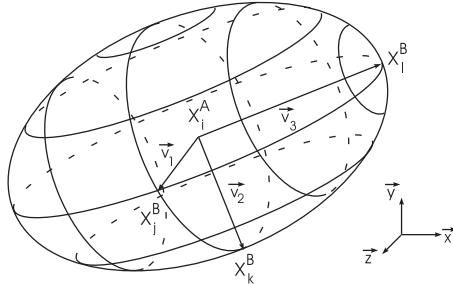


Fig. 1. The rotated ellipsoid for training pattern X_i^A .

Training of RKPNNs: The training, or the estimation of the Gaussian kernel parameters, consists in the calculation of the matrices R_i and S_i to obtain the matrix of covariances $\Sigma_i = R_i^T S_i S_i R_i$. The here proposed algorithm has to be executed for each training pattern of each class. In the case of a multi-class problem, the class that is trained will be compared against the joint set of all other classes.

The first step of the algorithm to train pattern X_i^A (pattern i belonging to class A) consists in finding the closest neuron of the joint class set B , denominated X_j^B , where the Euclidean metric is used (see Fig. 1). The connection vector \mathbf{v}_1 pointing from X_i^A to X_j^B produces the first components of the matrices R_i and S_i , and besides reveals important information for the search for the second closest training pattern X_k^B . Considering that the connection vector \mathbf{v}_1 represents the first principal axis of the hyper-ellipsoid, its length will be the last element of the matrix of variances S_i , and its direction, normalized to unit length, is the last row vector of the rotation matrix R_i . \mathbf{v}_2 is the connection vector from X_i^A to the second closest training pattern X_k^B and yields the second principal axis (see Fig. 1) and permits the calculation of the second last row vector of R_i , while its length is the second last component of S_i . With the information given by the first two principal axes, we can start with the search for the third closest training pattern X_l^B , and so on until both matrices R_i and S_i are completely defined.

As mentioned above, to find the closest training pattern, the Euclidean distance measure is used, but starting with the second closest training pattern, a

special "elliptical" distance measure is used. To calculate this distance measure, we start from the general formula for a hyper-ellipsoid in the n -dimensional space:

$$\frac{x_1^2}{a_1^2} + \frac{x_2^2}{a_2^2} + \cdots + \frac{x_i^2}{a_i^2} + \cdots + \frac{x_n^2}{a_n^2} = 1 \quad (3)$$

The general idea is to find a certain axis a_i with all other parameters x_1 to x_n , and a_1 to a_n given. In the following, we describe, how to obtain these parameters executing the algorithm above. a_1 is determined very easily by taking the length of the connection vector \mathbf{v}_1 to the closest training pattern using the Euclidean distance metric. With the knowledge of a_1 , the algorithm tries to find the next variable a_2 . To continue with the algorithm, we set $a_2 = \cdots = a_n = b$. This simplification is possible, because the remaining $(n - 1)$ principal axes of the hyper-ellipsoid aren't defined yet and can be rotated freely on the $(n - 1)$ -dimensional subspace that is perpendicular to \mathbf{v}_1 . The parameters x_1, \dots, x_n are the coordinates of the training pattern of class set B transformed into a coordinate system originated in X_i^A . The calculation of $b = a_2$ is possible, it being the length of the connection vector from X_i^A to X_k^B using the "elliptical" distance measure. Following the same procedure, once found the second closest training pattern, a_1 and a_2 are known, and we find the next closest training pattern setting $a_3 = \cdots = a_n = b$. The general idea of this recursive algorithm is: To calculate the i -th closest training pattern, a_1, \dots, a_{i-1} are known from anterior calculations, and $a_i = \cdots = a_n = b$ are set equal to find the i -th closest training pattern using the following "elliptical" distance measure:

$$b = \begin{cases} \left(\frac{\prod_{j=i+1}^n a_j^2 \sum_{j=1}^i x_j^2}{\prod_{j=i+1}^n a_j^2 - \left(\sum_{k=i+1}^n \left(\prod_{\substack{j=i+1 \\ j \neq k}}^n a_j^2 \right) x_k^2 \right)} \right)^{\frac{1}{2}}, & \text{if square root } \geq 0 \\ \infty, & \text{else} \end{cases} \quad (4)$$

To find the i -th closest training pattern, this procedure tries to construct a hyper-ellipsoid through the training pattern under observation and all $(i - 1)$ already known closest training patterns from class set B (see Fig. 1). The pattern of the class that yields the smallest hyper-ellipsoid is chosen, because it is at the same time the biggest possible one, that doesn't contain any other training pattern from the class set B . From this geometric point of view stems the term "elliptical" distance measure.

Through the application of this procedure, the calculation of the elements of the two matrices R_i and S_i are possible, and using the equation $\Sigma_i = R_i^T S_i S_i^T R_i$, all hyper-ellipsoids for all training patterns can be estimated. In the Gaussian kernel function (see Eq. 1) Σ_i is not needed, but its determinant $|\Sigma_i|$ and its inverse Σ_i^{-1} . Σ_i can be inverted starting from the equation $\Sigma_i^{-1} = (R_i^T S_i S_i^T R_i)^{-1} = R_i^{-1} S_i^{-1} S_i^{-1} R_i^{T-1}$. The inverse of the diagonal matrix S_i is easily calculated and for the rotation matrices, as they are used here, holds the

equation $R_i^{-1} = R_i^T$, that the calculation of Σ_i^{-1} reduces to $\Sigma_i^{-1} = R_i^T S_i^{-1} S_i^{-1} R_i$ and therefore isn't more complicated than the calculation of Σ_i .

3 Test on Real-world Data

A real-world example taken from the UCI machine learning repository [2] has been used to compare the generalization ability of different models: the Rotated Kernel Probabilistic Neural Network (RKPNN) as proposed in this article, the Support Vector Machine (SVM), a learning theory with similar complexity, and the original Probabilistic Neural Network (PNN) with its fast one-pass learning process.

The “Glass Identification Database” contains a set of 214 patterns divided into 6 classes with (70/17/76/13/9/29) members, that was reduced to 75% = 160 patterns to build the training set. This reduction was done without taking into account the classes: 54 patterns were deleted randomly, independent from its class memberships. The test set comprised the complete database, including the training set, yet exist learning algorithms that aren't able to classify correctly the set of training patterns that is required as well.

To not depend on the results of only one experiment, it was repeated ten times with randomly chosen training sets. Table 1 shows the results of these experiments, comparing the original PNN with the SVM (Simulator used: SVM Torch [4]) and the RKPNN, the last column shown the media of all 10 experiments.

Table 1. Number of misclassifications on ten experiments

Experiment	0	1	2	3	4	5	6	7	8	9	$\bar{\emptyset}$
original PNN	81	90	73	79	72	105	84	70	95	85	83.4
SVM	19	18	18	16	18	21	19	18	22	20	18.9
RKPNN	15	12	17	11	13	18	13	14	17	20	15.0

In all 10 experiments, the PNN delivered the most misclassifications, but it does have the advantage over the other models of the fast training algorithm and the extendability with new training patterns without complete retraining. The SVM and this model with slower training achieved better results, whereas the here proposed neural network (mean error: 15.0) has a small advantage over the SVM (mean error: 18.9) regarding its generalization capacity.

4 Conclusion

The here proposed model overcomes two weaknesses of the original Probabilistic Neural Network; first, it calculates automatically an “optimal” covariance matrix for each training pattern considering only its local environment and second, it enhances its generalization ability. The effectiveness of the new neural network

is proved in a real-world example for multi-class classification, where it showed a slightly smaller misclassification compared with the Support Vector Machine, a model of similar complexity.

Further work to improve the general weakness of probabilistic neural networks (one neuron on the pattern layer for each training pattern) is done using clustering techniques, that rely on the information obtained during the training process and the information provided by the Gaussian kernel parameters.

References

1. Berthold M.R., Diamond J. (1998), Constructive training of probabilistic neural networks. *Neurocomputing*, 19, 167-183
2. C.L. Blake and C.J. Merz (1998), UCI Repository of machine learning databases, <http://www.ics.uci.edu/~mlearn/MLRepository.html>, University of California, Irvine, Dept. of Information and Computer Sciences
3. Chen C.H., You G.H. (1992), ISBN Recognition Using a Modified Probabilistic Neural Network (PNN). *Proceedings 11th IAPR International Conference on Pattern Recognition*, Vol.II, 419-421
4. Collobert R., Bengio S. (2001), SVMTorch: Support Vector Machines for Large-Scale Regression Problems, *Journal of Machine Learning Research*, Vol 1, 143-160
5. Galleske I., Castellanos J. (1997), Probabilistic Neural Networks with Rotated Kernel Functions *Proceedings 7th International Conference on Artificial Neural Networks ICANN'97*, 379-384
6. Musavi M.T., Chan K.H., Hummels D.M., Kalantri K (1993), On the Generalization Ability of Neural Network Classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.16, No.6, 659-663
7. Parzen E. (1962), On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33, 1065-1076
8. Specht D.F. (1988), Probabilistic neural networks for classification mapping, or associative memory. *Proceeding, IEEE International Conference on Neural Networks*, 1, 525-532
9. Specht D.F. (1990), Probabilistic Neural Networks. *Neural Networks*, 3, 109-118
10. Specht D.F. (1991), Generalization accuracy of probabilistic neural networks compared with backpropagation networks. *IJCNN-91-Seattle: International Joint Conference on Neural Networks*, Vol.1, 887-892
11. Yang Z.R., Chen S. (1998), Robust maximum likelihood training of heteroscedastic probabilistic neural networks. *Neural Networks*, 11, 739-747

Independent Residual Analysis for Temporally Correlated Signals

L.-Q. Zhang^{†*} and A. Cichocki[‡]

[†] Department of Computer Sciences,
Shanghai Jiaotong University, Shanghai 200030, China
[‡]Brain-style Information Systems Research Group
RIKEN Brain Science Institute, Saitama 351-0198, Japan
Email: zhang-lq@cs.sjtu.edu.cn

Abstract. In this paper, we study the blind source separation problem of temporally correlated signals via exploring both the temporal structure and high-order statistics of source signals. First, we formulate the problem as independent residual analysis and present a simple cost function. Efficient learning algorithm is developed for the demixing matrix and the corresponding stability analysis is also provided. The formulation provides much more flexibility for us to identify learning algorithms with good learning performance and stability. Furthermore, the approach unifies the conventional high-order statistical method and the second-order statistical method. From stability analysis, we infer that if the temporal filters of sources are mutually different, the second order statistical algorithm will be sufficient to separate the sources from their linear mixtures.

1 Introduction

Blind source separation or independent component analysis (ICA) has been considered as a fundamental data analysis tool in the fields of signal processing and neural network. The theoretical framework has been established as independent component analysis. The blind source separation/deconvolution problem is to recover independent sources from sensor outputs without assuming any a priori knowledge of original signals besides some statistic features [2],[4],[7]. It is applicable to many fields, such as wireless telecommunication systems, sonar and radar systems, audio and acoustics, image enhancement and biomedical signal processing (EEG/MEG signals)[4, 6, 7, 9].

There have existed a number of learning algorithms for blind source separations[9, 7, 4, 3, 6, 8, 10]. Generally speaking, there exist two categories of methods for blind source separation. First category is to explore the temporal structures or linear predictability of source signals, such as SOBI [5]. The other is to explore the high order statistics of sources, such as the Infomax [4], the natural gradient algorithm [1, 3, 12] and FastICA [8].

In this paper, we formulate the blind source separation as independent residual analysis. The purpose is to make use of both the temporal correlations and

* Corresponding author

high order statistics positively for obtaining more efficient and reliable learning algorithms for dealing with practical problems. In this framework, we explore the mutual independency of the residual signals of the source signals. Theoretically, independence of the source signals implies independence of residual signals. However, in practice, when the data samples are not sufficient or the sensor signals are noisy, the correlations between different sources may not equal the zero. Thus, the separating solution obtained by using only the high order statistics may not capture the true one. In this paper, the independent residual analysis provides a new approach to study both the high order statistics and temporal structures of the source signals.

2 Problem Formulation

In this section, we formulate the blind separation of temporally correlated sources in the framework of independent residual analysis. The objective function is derived from the mutual information of the residual signals.

2.1 Temporally correlated sources

Assume that s_i , $i = 1, \dots, n$ are n mutually stochastically independent source signals, of which each temporally correlated. We assume that each signal has zero mean. Suppose that source $s_i(k)$ is modelled by a stationary AR model,

$$s_i(k) = \sum_{p=1}^L a_p^i s_i(k-p) + \varepsilon_i(k), \quad (1)$$

where L is the degree of the AR model and $\varepsilon_i(k)$ is a zero mean independent and identically distributed (that is, white) time series called the residuals. For the sake of simplicity, we use the notation $A_i(z) = 1 - \sum_{p=1}^L a_p^i z^{-p}$. where z is the z-transform variable.

If the source signals are known, the coefficients of the forward linear predictor $A_i(z)$ can be obtained by minimizing the prediction error in the least squares sense. The linear filter $A_i(z)$ is minimum phase, that is the zeros of $A_i(z)$ are located in the interior of the unit circle. Since in the blind separation setting, the source signals are unknown, we need to impose some constraints on the linear filters. We assume that the linear filters $A_i(z)$ are minimum phase throughout this paper. Denote

$$\mathbf{A}(z) = \text{diag}(A_1(z), \dots, A_n(z)) = \sum_{p=0}^L \mathbf{A}_p z^{-p}, \quad (2)$$

where \mathbf{A}_0 is the identity matrix and $\mathbf{A}_p = -\text{diag}(a_p^1, \dots, a_p^n)$, $p \geq 1$ is a diagonal matrix.

Now, we assume that the residual signals are spatially mutual independent and temporally i.i.d.. This property will be used to develop efficient algorithms for the demixing matrix and temporal structures.

2.2 Separation model

We consider the case when observed signals are instantaneous mixtures of the source signals. Let $\mathbf{x}(k) = [x_1(k), \dots, x_n(k)]^T$ be the set of linearly mixed signals,

$$\mathbf{x}(k) = \mathbf{H}\mathbf{s}(k). \quad (3)$$

Here, \mathbf{H} is an $n \times n$ unknown nonsingular mixing matrix. For the sake of simplicity, we assume that the number of observations is exactly the same as that of the source signals. Blind source separation is to find a linear transform which transforms the sensor signals into mutually independent components, which are used to recover the source signals. Let \mathbf{W} be an $n \times n$ nonsingular matrix which transforms observed $\mathbf{x}(k)$ to

$$\mathbf{y}(k) = \mathbf{W}\mathbf{x}(k). \quad (4)$$

This provides an estimate of the original source signals $\mathbf{s}(k)$.

2.3 Cost function

In this paper, we suggest to use the mutual information of residual signals as an objective function. Although from theoretical point of view, two formulations are equivalent, the residual independent analysis provides us a new way to explore both the temporal structures and high-order statistics of source signals simultaneously within one framework.

From the source model, we have

$$\mathbf{\varepsilon}(k) = \mathbf{A}(z)\mathbf{s}(k), \quad (5)$$

where $\mathbf{A}(z)$ can be estimated via the linear prediction method if the source signals $\mathbf{s}(k)$ are known. When the temporal structure $\mathbf{A}(z)$ and the demixing matrix \mathbf{W} is not well-estimated, the residual signals

$$\mathbf{r}(k) = \mathbf{A}(z)\mathbf{W}\mathbf{x}(k) \quad (6)$$

are not mutually independent. Denote $\mathbf{W}(z) = \mathbf{A}(z)\mathbf{W}$. We attempt to train the demixing model and temporal structures such that the are spatially mutually independent and temporally decorrelated. To this end, we employ the mutual information rate of random variables \mathbf{r} as a criterion.

Assume $q(\mathbf{r})$ is the probability density function of \mathbf{r} and $q_i(r_i)$ is the marginal probability density function of r_i , $i = 1, \dots, n$. Simplifying the mutual information of r_i , $i = 1, \dots, n$ leads to the following cost function [11]

$$L(\mathbf{y}, \mathbf{W}, \mathbf{A}(z)) = -\log(|\det(\mathbf{W})|) + \sum_{i=1}^n H(r_i), \quad (7)$$

where $H(r_i)$ is the entropy of the random variable r_i .

In the parameterized model, parameters are not equally important in finding the solution. The parameters, which directly influence the solution, are referred to as the parameters of interest. Otherwise, the parameters are referred to as the nuisance parameters. However, identifying the nuisance parameters will provide an approach way to efficiently estimate the parameters of interest. In the framework of independent residual analysis, there are two unknowns in the cost function: the probability density function q and the temporal filter $\mathbf{A}(z)$. These two unknowns are seen as the nuisance parameters in the semiparametric model. The demixing matrix \mathbf{W} is referred to as the parameters of interest. The semi-parametric approach [2] suggests to use an estimating function to estimate the parameter of interest, regardless of the nuisance parameters. In this paper, we will analyze how to identify the nuisance parameters to improve learning performance and stability.

3 Learning Algorithm

In this section, we derive a learning algorithm based on the gradient descent approach for the demixing matrix. The cost function for on-line statistical learning can be simplified as

$$l(\mathbf{y}, \mathbf{W}, \mathbf{A}(z)) = -\log(|\det(\mathbf{W})|) - \sum_{i=1}^n \log q_i(r_i) \quad (8)$$

where $q_i(r_i)$ is the probability density function of r_i , and $\mathbf{r} = (r_1, \dots, r_n)^T = \mathbf{A}(z)\mathbf{W}\mathbf{x}(k)$. To obtain the natural gradient of the cost function, we calculate the total differential dl with respect to a variation of $d\mathbf{W}$.

$$dl(\mathbf{y}, \mathbf{W}, \mathbf{A}(z)) = -d\log(|\det(\mathbf{W})|) - d \sum_{i=1}^n \log q_i(r_i) \quad (9)$$

It is not difficult to obtain the following relation [1]

$$d\log(|\det(\mathbf{W})|) = \text{tr}(d\mathbf{W}\mathbf{W}^{-1}) \quad (10)$$

The second term in (9) can be calculated by

$$-d \sum_{i=1}^n \log q_i(r_i) = \varphi(\mathbf{r})^T d\mathbf{r} = \varphi(\mathbf{r})^T \mathbf{A}(z) d\mathbf{W}\mathbf{W}^{-1} \mathbf{y} \quad (11)$$

where $\varphi_i(r_i) = -\frac{q'_i(r_i)}{q_i(r_i)}$. Introducing the non-holonomic transform $d\mathbf{X} = d\mathbf{W}\mathbf{W}^{-1}$, we have

$$\begin{aligned} dl(\mathbf{y}, \mathbf{W}, \mathbf{A}(z)) &= -\text{tr}(d\mathbf{W}\mathbf{W}^{-1}) + \varphi(\mathbf{r})^T \mathbf{A}(z) d\mathbf{W}\mathbf{W}^{-1} \mathbf{y} \\ &= -\text{tr}(d\mathbf{X}) + \varphi(\mathbf{r})^T \sum_{p=0}^L \mathbf{A}_p d\mathbf{X} \mathbf{y}(k-p) \end{aligned} \quad (12)$$

where φ is the vector whose components are $\varphi_i(r_i)$ and \mathbf{A}_p are diagonal matrices. Now, we obtain the derivative of $l(\mathbf{y}, \mathbf{W}, \mathbf{A}(z))$ with respect to \mathbf{X} as follows

$$\frac{dl(\mathbf{y}, \mathbf{W}, \mathbf{A}(z))}{d\mathbf{X}} = -\mathbf{I} + \left[\sum_{p=0}^L \mathbf{A}_p \varphi(\mathbf{r}) \mathbf{y}^T(k-p) \right], \quad (13)$$

The gradient descent algorithm with respect to $d\mathbf{X}$ is described as

$$\Delta \mathbf{X} = -\eta \frac{dl(\mathbf{y}, \mathbf{W}, \mathbf{A}(z))}{d\mathbf{X}}, \quad (14)$$

Using the non-holonomic transform, we obtain the gradient descent algorithm in terms of the original \mathbf{W} as

$$\Delta \mathbf{W} = \eta \left\{ \mathbf{I} - \sum_{p=0}^L [\mathbf{A}_p \varphi(\mathbf{r}) \mathbf{y}^T(k-p)] \right\} \mathbf{W}. \quad (15)$$

A number of factors in the algorithm need to be further elaborated. First, there are two unknowns in the algorithm, the activation function $\varphi(\mathbf{r})$ and the temporal filter $\mathbf{A}(z)$. If we formulate the independent residual analysis into the semi-parametric model, these two unknowns can be seen as the nuisance parameters. According to the semi-parametric theory, it is not necessary to estimate the nuisance parameters precisely. But appropriate nuisance parameters will help for the algorithm to have a good learning performance.

Second, when the temporal filters $\mathbf{A}(z)$ become the identity matrix, the learning algorithm is the same as the natural gradient for the instantaneous mixtures. This means that if \mathbf{W} is true solution, for any non-singular temporal filters $\mathbf{A}(z)$, \mathbf{W} satisfies the following equation

$$\mathbf{I} - E \left[\sum_{p=0}^L \mathbf{A}_p \varphi(\mathbf{r}) \mathbf{y}^T(k-p) \right] = \mathbf{0}. \quad (16)$$

4 Stability Analysis

In this section, we analyze the stability conditions for the natural gradient learning algorithm. Suppose that \mathbf{W} is the separating solution. Since the learning algorithm for updating \mathbf{W} is a linear combination of \mathbf{X} , the stability of learning algorithm for \mathbf{X} implies the stability of the learning algorithm (15). In order to analyze the stability of the learning algorithm, we suppose that the separating signals $\mathbf{r} = [r_1, \dots, r_n]^T$ are not only spatially mutually independent, but also temporally identically independently distributed for each component. To analyze the asymptotic properties of the learning algorithm, we take expectation for the learning algorithm for \mathbf{X}

$$\frac{d\mathbf{X}}{dt} = \eta \left(\mathbf{I} - E \left[\sum_{p=0}^L \mathbf{A}_p \varphi(\mathbf{r}) \mathbf{y}^T(k-p) \right] \right). \quad (17)$$

Taking a variation $\delta\mathbf{X}$ on \mathbf{X} , and using the mutual independence of signals \mathbf{y}_i and the normalized condition (15), we deduce

$$\frac{d\delta\mathbf{X}}{dt} = -\eta \left(\sum_{p=0}^L E [\mathbf{A}_p \varphi'(\mathbf{r}) \delta\mathbf{r}] \mathbf{y}^T(k-p) + \mathbf{A}_p \varphi(\mathbf{r}) \delta\mathbf{y}^T(k-p) \right) \quad (18)$$

We use the following notations

$$m_i = E[\varphi'(r_i)r_i^2], \kappa_i = E[\varphi'_i(r_i)], \quad (19)$$

$$\gamma_{ij}^2 = E[|\sum_{p=0}^L A_{pi}y_j(k-p)|^2], \quad i, j = 1, \dots, n. \quad (20)$$

Using mutual independence of r_i and their i.i.d. properties, we simplify the variational equations at the separating solution as

$$\frac{d\delta x_{ij}}{dt} = -\kappa_i \gamma_{ij} \delta x_{ij} - \delta x_{ji}, \quad (21)$$

$$\frac{d\delta x_{ji}}{dt} = -\kappa_j \gamma_{ji} \delta x_{ji} - \delta x_{ij}, \quad (22)$$

for $i \neq j$ and

$$\frac{d\delta x_{ii}}{dt} = -\eta(m_i + 1) \delta x_{ii}. \quad (23)$$

Thus, the stability conditions for (17) are given as follows

$$m_i + 1 > 0, \quad \kappa_i > 0, \quad \text{for } i = 1, \dots, n \quad (24)$$

$$\kappa_i \kappa_j \gamma_{ij}^2 \gamma_{ji}^2 > 1, \quad \text{for } i \neq j \quad (25)$$

The stability conditions depend on the temporal filters $\mathbf{A}(z)$ and the activation functions $\varphi(\mathbf{r})$. It should be noted here that the activation functions are functions of the residual variables, rather than the source variables. Therefore, when we estimate the probability density function, we should estimate the pdf of the residual variables.

There exists a number of ways to identify the learning algorithm which satisfies the above stability conditions. First way is to adapt the temporal structures, given the activation functions. If the activation functions are chosen as a linear function, the approach will become the second-order statistical method. Second way is to adapt the activation functions, given the temporal filters $\mathbf{A}(z)$. If the temporal filters $\mathbf{A}(z) = \mathbf{I}$, then the approach will reduce to the conventional high-order statistical method, such as the natural gradient algorithm. The third way is to adapt both the temporal filter and the activation functions.

4.1 Stability Analysis for Second-Order Methods

In this subsection, we further elaborate the stability conditions for second-order statistical method. If we do not consider the temporal structures of the source

signals, the second-order statistical method just provides the decorrelated solution, rather than the independent solutions. However, if we take the temporal structures into account in the learning algorithm, the second-order statistical method can also provide the independent solution.

Now we assume that the activation functions are linear, i.e. $\varphi_i(r_i) = r_i$, for $i = 1, \dots, n$. In this case, $\kappa_i = 1$, $m_i = 1$. Thus the stability conditions are simplified as

$$\gamma_{ij}^2 \gamma_{ji}^2 > 1, \text{ for } i \neq j \quad (26)$$

To further simplify the stability conditions (26), we consider the sufficient conditions. If the following conditions are satisfied,

$$\gamma_{ij}^2 > 1, \text{ for } i \neq j \quad (27)$$

then, the stability conditions (26) hold. Suppose that $A_i(z)$ is the forward linear filter of source $s_i(k)$ given by the linear prediction model, satisfying

$$r_i(k) = A_i(z)s_i(k), \text{ for } i = 1, \dots, n. \quad (28)$$

Assume that $A_i(z)$, $i = 1, \dots, n$ are mutually different

$$A_i(z) \neq A_j(z), \text{ for } i \neq j. \quad (29)$$

This means that different sources have different temporal structures. Now, we are going to prove that under the conditions (29), the stability conditions (27) are satisfied. To this end, we introduce the following lemma:

Lemma 1. *If $B(z) \neq A_i(z)$, then*

$$E \left[\left| \sum_{p=0}^L B_p s_i(k-p) \right|^2 \right] > E \left[\left| \sum_{p=0}^L A_{pi} s_i(k-p) \right|^2 \right] \quad (30)$$

Proof: The result follows from the i.i.d. assumption on the residual signals.

Theorem 1. *Assume that $A_i(z)$ are the forward linear filter of source s_i , for $i = 1, \dots, n$. If the forward linear filters differ mutually for different sources, then the stability conditions are satisfied.*

Proof. Since $A_i(z) \neq A_j(z)$, we have the following inequality

$$\begin{aligned} \gamma_{ij}^2 &= E \left[\left| \sum_{p=0}^L A_{pi} y_j(k-p) \right|^2 \right] \\ &> E \left[\left| \sum_{p=0}^L A_{pj} y_j(k-p) \right|^2 \right] = E[|r_j(k)|^2] = 1, \end{aligned} \quad (31)$$

for any $i \neq j$. This proves our result.

From the above analysis, we see that if the forward linear filters differ mutually for different sources, we can use the second order statistics to recover the source signals if their forward linear filters are known. This suggests that adaptation of the temporal filters of the output signals should make the learning algorithm for the demixing model more stable and reliable.

5 Conclusions

In this paper, we formulate the blind source separation as in the framework of independent residual analysis. Instead of analyzing the original source signals, we suggest to analyze the mutual independency of the residual signals. Learning algorithm for training the demixing model is derived by minimizing the mutual information of the residual signals. Stability analysis is also provided. We further elaborate the stability conditions for the second order statistic method, inferring that if the forward linear filters differ mutually for different sources, the second order statistics method can also recover the source signals if their forward linear filters are well estimated. Detailed analysis on the adaptation of the forward linear filters and computer simulations will provided in the full paper.

References

1. S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10:251–276, 1998.
2. S. Amari and J.-F. Cardoso. Blind source separation– semiparametric statistical approach. *IEEE Trans. Signal Processing*, 45:2692–2700, Nov. 1997.
3. S. Amari, A. Cichocki, and H.H. Yang. A new learning algorithm for blind signal separation. In G. Tesauro, D.S. Touretzky, and T.K. Leen, editors, *Advances in Neural Information Processing Systems 8 (NIPS*95)*, pages 757–763, 1996.
4. A.J. Bell and T.J. Sejnowski. An information maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7:1129–1159, 1995.
5. A. Belouchrani, K. Abed-Meraim, J.-F. Cardoso, and E. Moulines. A blind source separation technique using second order statistics. *IEEE Trans. Signal Processing*, 45:434–444, 1997.
6. J.-F. Cardoso and B. Laheld. Equivariant adaptive source separation. *IEEE Trans. Signal Processing*, SP-43:3017–3029, Dec 1996.
7. P. Comon. Independent component analysis: a new concept? *Signal Processing*, 36:287–314, 1994.
8. A. Hyvärinen and E. Oja. A fast fixed-point algorithm for independent component analysis. *Neural Computation*, 9(7):1483–1492, 1997.
9. C. Jutten and J. Herault. Blind separation of sources, Part I: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24:1–10, 1991.
10. T.W. Lee, M. Girolami, and T. Sejnowski. Independent component analysis using an extended infomax algorithm for mixed subgaussian and supergaussian sources. *Neural Computation*, 11:417–41, 1999.
11. L. Zhang, S. Amari, and A. Cichocki. Semiparametric model and superefficiency in blind deconvolution. *Signal Processing*, pages 2535–2553, 2001.
12. L. Zhang, A. Cichocki, and S. Amari. Natural gradient algorithm for blind separation of overdetermined mixture with additive noise. *IEEE Signal Processing Letters*, 6(11):293–295, 1999.

MLP+H: a Hybrid Neural Architecture Formed by the Interaction of Hopfield and Multi-Layer Perceptron Neural Networks

Clayton Silva Oliveira e Emílio Del Moral Hernandez¹

Polytechnic School of the University of São Paulo

Department of Electronic Systems Engineering

Cidade Universitária – Av. Prof. Luciano Gualberto

São Paulo, SP, CEP 05508-900 – Brazil

e-mails: edmoral@usp.br, clayton@uol.com.br

Abstract: This paper addresses the design and experimental characterization of a novel hybrid neural network, in which two distinct classical architectures interact: the Hopfield neural network and the Multi-Layer Perceptron. This hybrid neural system, named MLP+H (from MLP + Hopfield), presents a better performance than each one of the two classical architectures when considered in separate. In addition, it has the ability to deal with different classes of data than that normally allowed by the two conventional architectures. For example, while the Hopfield networks deal with binary patterns and the MLPs with information that always have some analog characteristics (due to the continuous nature of the MLP nodes), the MLP+H deals with analog inputs and purely digital outputs. Moreover, the MLP+H allows reduced training times when compared with the MLP architecture, also presenting compactness and flexibility in dealing with different applications, as implementation of anti-noise filters, an application currently under study.

1 Introduction

Multi-Layer Perceptrons and Hopfield neural networks have unique characteristics that allow the solution of specific classes of problems in the universe of artificial intelligence. Nevertheless, certain classes of problems present specific requirements that are not entirely or well addressed by either one of these architectures individually. One possible approach to cope with some of these classes of problems is the development of hybrid neural architectures. As it will be discussed later, the hybrid architecture here studied puts together some of the fundamental features of the two mentioned classical architectures, and it offers to the application designer other features that are new and unique, resulting in better performances and sometimes shorter computation and training times than the ones obtained with classical architectures taken in isolation.

2 Characteristics and Limitations of Hopfield and MLP Architectures

The Hopfield network can be defined as an *associative memory*, which is able to recover previously stored patterns, starting from prompting inputs that correspond to a

¹ Acknowledgments to the University of São Paulo (USP) and Foundation for the Support of Research in the State of São Paulo (FAPESP) for the financial support to this work.

partial or noisy version of one of such stored patterns [3]. We can list here some of the relevant characteristics and limitations of the Hopfield network, which will be of particular relevance for the discussions in the following sections:

- we have the storage of undesired patterns (spurious and mirror states, for example);
- being an *auto-associative memory*, it can only deal with inputs and outputs of same nature and dimensions (binary inputs and outputs, with the same number of bits);
- after the programming, the Hopfield network can only generate as output one of the previously stored patterns, i. e., given an input, this neural network will give as output the stored pattern that is more similar to that input pattern. Moreover, it can fix some input errors, as flipped bits.

The Multi-Layer Perceptron (MLP) has different functionalities and operation principles. Among the differences between the Hopfield network and the MLP network, one regards the analog nature of the output patterns in the MLP. While the Hopfield architecture deals with binary vectors, the MLP can deal with analog vectors [7]. Some aspects of the MLP important to this article can be listed:

- it can deal with training sets having different dimensions and natures with respect to inputs and outputs. We can have for example digital inputs with N_I dimensions, and analog outputs with N_O dimensions;
- after trained, when given an input pattern, the MLP will give as output a vector that is function of the input, using the learned mapping that relates the input and output vectors of the training set. Thus, the MLP architecture can be characterized as an *interpolator* and *extrapolator*, without *error correction*;
- because of its high flexibility, which allows the solution of complex problems such as classification of data beyond the linear separability (XOR problem is a classical example [7]), its training is slower when compared with the Hopfield network training time, what represents high computational processing times.

With the presentation of these key characteristics of the two neural networks that are the main components of the hybrid architecture discussed in this article, we can proceed now with its detailed description in the following section.

3 The Proposed Hybrid Neural Architecture – Functional and Structural Characteristics of the MLP+H Network

As it will be seen ahead, the developed hybrid architecture here presented can implement inputs-outputs relationships in which the inputs can be either analog or discrete, and the output is always discrete, no matter what the nature of the input is. Another of its characteristics is the possibility of having different dimensions for the inputs and outputs vectors. Thus, this hybrid neural network has a feature that both of the classic neural networks here discussed do not have individually: to relate *analog inputs* to *discrete outputs* with error correction. Notice that the Hopfield network does not deal with *analog* vectors, while the MLP does not present *error correction* [3, 4, 7]. Structurally, the hybrid architecture MLP+H is defined as presented in the Fig. 1.

As it can be seen from Fig. 1, the input dimension (N_I) can be different from the output dimension (N_O), what is not possible in the Hopfield network separately. In this way,

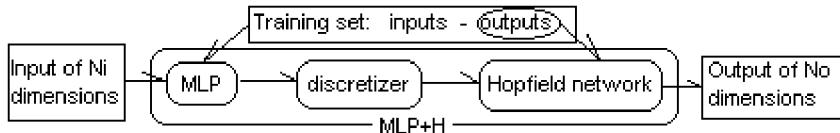


Fig. 1. The structure of the hybrid architecture MLP+H. The chain formed by the MLP and the Hopfield neural networks give to this hybrid architecture distinct characteristics when compared with its component neural networks in isolation. For example, here it is possible to have analog inputs with N_I dimensions and purely digital outputs with N_O dimensions

the hybrid network MLP+H is structurally composed by a chain with one MLP followed by one Hopfield network. During the conception of the hybrid architecture MLP+H, one of the options initially considered was the direct coupling between the MLP output and the Hopfield network. As it will be presented in the Section 5, the performance of this hybrid network without discretizer was worst than that achieved by an MLP alone with its output discretized. Another possibility later considered was the discretization (binarization to be more precise) of the MLP outputs, before using them as inputs to the Hopfield network. This option, which is the one represented in Fig. 1, resulted in significant performance improvement.

The reason why the insertion of one discretizer between the MLP and the Hopfield network improves the hybrid network performance, as it will be shown in the experimental section, is the following: the Hopfield network in the MLP+H is given binary inputs and not analog as in the original idea, what makes the Hopfield network deal with binary inputs, what is more similar to its binary nature.

The recovery of the stored patterns in an MLP+H architecture can be briefly described as follows:

- given one input to the MLP (noisy or not) with N_I dimensions, this first network produces an output with N_O dimensions according to the interconnections between its neurons defined in the MLP training phase (described ahead);
- this output of the MLP after discretized, will be the prompting pattern to the Hopfield network whose state will then evolve until one pattern previously stored is recovered as a stable state. This state will be the output of the hybrid network MLP+H, with N_O bits, and it can be one of the programmed patterns or not (it can be also a spurious state).

In its functionality, this architecture allows, for example, that the inputs and outputs be different in their nature, i. e., the inputs can be binary or analog, while the outputs are always binary (due to the Hopfield network in the final section of the MLP+H architecture). Notice that this is not possible for an MLP alone, since this type of neural network, strictly speaking, deals with input and output patterns that always have an analog nature.

Regarding the training algorithm, it is defined as the following:

- first of all, the output patterns (always binary) of the training set are stored in the Hopfield network that is part of the MLP+H architecture;
- then, the MLP of the hybrid architecture is trained with the input-output pairs of the training set, up to the point when the Hopfield network is capable of recognizing the outputs given by the MLP as one of the previously stored output patterns.

From the discussed above, we can expect that the MLP+H architecture exhibit a faster training phase than an MLP being trained alone, because the MLP in the hybrid architecture does not have to be trained until its RMS error is less than a required value. Instead, it just has to be trained until its outputs are recognized by the Hopfield network. As it will be shown in Section 5, this proposed training strategy in fact allows a significant reduction in the hybrid neural network learning time.

The hybrid network can also obtain better performances against spurious states when compared with the Hopfield network in separate, as its MLP (with the discretizer in its output) gives as input to the Hopfield network binary patterns that are closer to the previously programmed ones. This enhances the Hopfield ability to recognize these stored patterns, since one condition to be satisfied at end of the training phase is that the Hopfield network is capable of recognizing all vectors that the MLP is giving as inputs as one of its programmed patterns.

In Section 5 we will show several experimental results obtained with the MLP+H hybrid neural network, some of them illustrating many of the aspects here discussed. Before we discuss such experiments though, we will use the following section to present an important tool that allows better analyses of the obtained results.

4 The Confusion Matrix Concept to Measure the Performance of Neural Networks of Discrete Nature

The concept of confusion matrix has its origin in the field of telecommunications, allowing the quantification of inter symbol confusion in noisy channels. With the increasing complexity of the experiments designed for the MLP+H characterization, an adapted version of the concept of confusion matrix became useful in our studies.

In our context, it is needed to know how a neural network deals with a noisy version of a previously stored pattern P , when trying to recover it. More specifically, we are interested in knowing what are the probabilities that the neural network recognizes it as the pattern P (success), as another stored pattern, or even as a spurious state. For a better understanding of this concept, it is shown a confusion matrix in Table 1 that resulted from an experiment with a neural network with three stored patterns.

Table 1 – example of results from an experiment with a neural network with three stored patterns

		Outputs generated by the neural network		
		Pattern 1	Pattern 2	Pattern 3
<i>n</i> noisy inputs of the pattern <i>i</i>	Pattern 1	60%	20%	10%
	Pattern 2	10%	80%	10%
	Pattern 3	25%	30%	45%

In Table 1, the first row relates to n noisy input versions to pattern 1, and the first column in this row relates to the subset of them resulting in successful outputs (perfect recoveries of pattern 1); the second and third columns in the row present the number of mistaken recoveries of pattern 1, as patterns 2 and 3, respectively. In this example, we see from the first row that 60% of the n pattern 1 noisy inputs were correctly recognized as pattern 1, and 20% and 10% were wrongly recognized as patterns 2 and 3,

respectively. Since $60\% + 20\% + 10\% = 90\%$, 10% of the outputs corresponded to spurious states. Additionally, the average of success for the network operation with respect to all the three patterns can be given by the average of the principal diagonal in the confusion matrix: $(60\% + 80\% + 45\%) / 3 = 61,67\%$. Thus, through this single number, we have a practical and concise way to evaluate a neural architecture performance: the largest this average, the better the immunity to noisy conditions.

This concept was very important to characterize the proposed hybrid network, as it represents an easy and practical way to obtain the MLP+H performance. Moreover, to obtain more specific information about the neural network behavior, we can analyze the entries in confusion matrix individually. It will become clear in the next section how important this methodology was to the MLP+H studies.

5 Experiments with the Hybrid Neural Architecture MLP+H

This section has the objective of presenting experiments performed with the MLP+H, which were designed to concretely verify its operation, as well as to evaluate its performance under noisy input vectors. First, it was performed a series of experiments for architecture operation verification with the training set illustrated in the Fig. 2a.

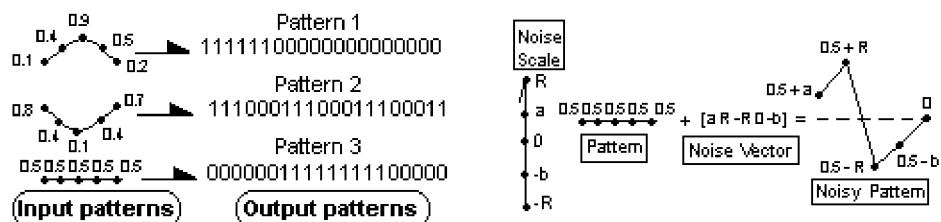


Fig. 2a. (left) Training set used to verify the operation of the hybrid network MLP+H. Notice the analog nature of the inputs versus the binary nature of the outputs. **Fig. 2b. (right)** Illustration of the noise definition used during experiments to measure the MLP+H performance. To the input test vector, we add a noise vector formed by components that always are in the range between $[-R, R]$, with uniform distribution

From Fig. 2a, we can see that the inputs of the hybrid architecture are analog, while the outputs are binary. The network parameters for the MLP were 3 layers, with **6** ($5 + 1$ bias)/**25** / **40** ($20 + 20$) nodes per layer; for the Hopfield network we used 40 nodes, as a Hopfield network with 20 nodes does not have the necessary storage capacity to store the three output patterns of the training set of the Fig. 2a [1, 5]. Taking this need in consideration, the 20 bits output patterns were replicated in length, so to be compatible with the network needed size.

It is important realizing that the output layer of the MLP must have the same number of nodes that the Hopfield network has, since the outputs of the MLP feed the inputs of the Hopfield network.

To characterize the MLP+H performance with respect to noisy input vectors, it was defined a way to measure and to add noise to the input patterns of the training set (Fig. 2a). This added noise was defined in a given range, i. e., we had the noise limited to a maximum value R and a minimum value $-R$. During our characterization tests, this

noisy component was added to the input patterns, as represented in Fig. 2b. With the training set and the architecture characteristics previously defined, the network was prompted with 900 test vectors (noisy versions of the input patterns); each 300 vectors of these were related to only one of the three patterns of the training set, distorted by noise as illustrated in Fig. 2b.

In Fig. 3, we present a graphic that shows the obtained performance (average value in the principal diagonal of the confusion matrix) *versus* the maximum noise module $|RI|$ for two different versions of the hybrid neural network - MLP+H with and without discretized coupling - and we contrast these performances with that obtained with a traditional MLP with discretized output, all of them used to solve the same problem.

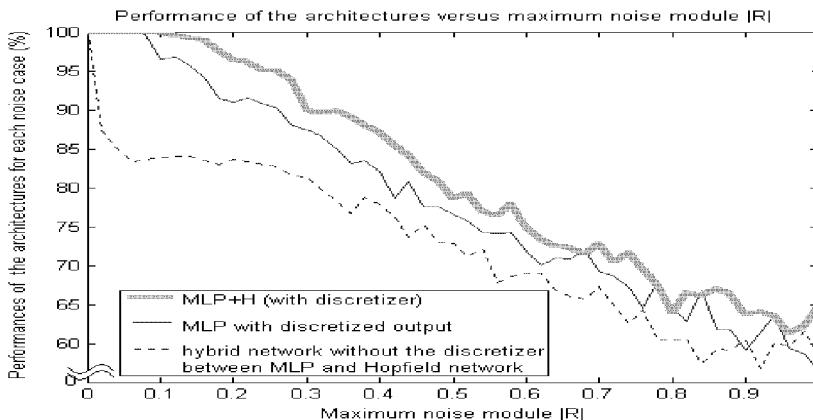


Fig. 3. Performance of the studied architectures. The horizontal axis represents the level of noise (the magnitude $|RI|$) added to the input patterns. The vertical axis shows the network performance (average value in the principal diagonal of the confusion matrix). We see clearly that the MLP+H performance (thick line) is always higher or equal than the discretized output MLP performance (narrow line). Its performance is also higher than the obtained with an MLP+H without the discretizer between its MLP and Hopfield stages (dashed line). When the maximum noise module is equal to 1 (100% of noise), the average performance of the MLP+H is about 65% of successes, while the MLP with its binarized output presents in average of 55% of successes. The MLP with its binarized output and the hybrid neural network without the discretizer were trained by the same strategy used to the MLP+H training. The MLP trained in separate was equal in size to that used in the both hybrid architectures here discussed

From the graphic in Fig. 3, we can evaluate the validity of this hybrid architecture and its good performance when submitted to noisy inputs; only when the noise level $|RI|$ goes beyond 0.5, the average of successes drops below the 80% of successes. It is important to notice that the input patterns have analog values between 0 and 1, and $|RI| = 0.5$ means 50% of noise in the minimum (when analog value is 1), what can be considered a high level of noise. Moreover, as already mentioned, the MLP+H performance was higher than that of the MLP with discretized output, considered here as the natural alternative to the hybrid architecture, in problems with analog inputs and digital outputs.

To verify the influence of the number of nodes of the Hopfield network on the hybrid architecture MLP+H, two specific series of tests were performed. In one of these, a similar graphic to that of Fig. 3 (performance *versus* level of noise) was produced,

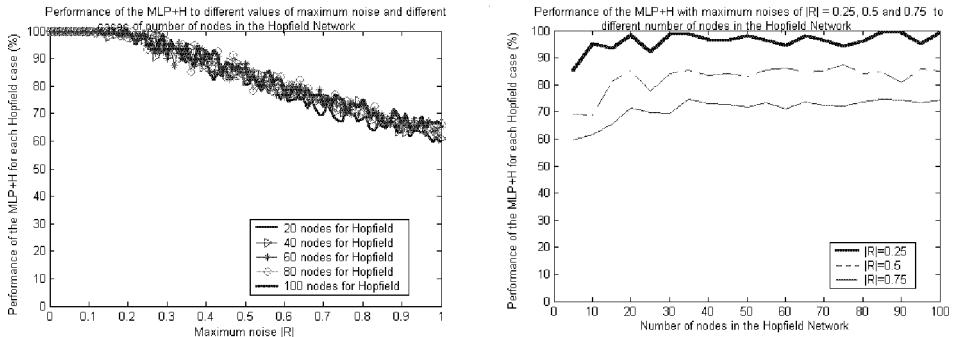


Fig. 4a. (left) Performances of the hybrid network MLP+H with different numbers of nodes in the Hopfield network that composes the MLP+H. Each plotted line is related to either 20, 40, 60, 80 or 100 nodes in the Hopfield network. By this graphic, we can verify that the performance of the hybrid network does not vary so much when the number of nodes in the Hopfield network is changed. This shows that, even if the MLP+H is implemented with a Hopfield network of few nodes, the MLP (the complementary component of the MLP+H), takes care of a primary treatment of the noisy input patterns and guarantees a good proximity with the prompting patterns programmed in the Hopfield network. Thus, the Hopfield network has to do a limited effort to recover the previously stored patterns. Specially notice the performances in the cases of 20 and 100 nodes for Hopfield. **Fig. 4b. (right)** Performances of the hybrid network for series of experiments where the maximum noise modules are $|R| = 0.25, 0.5$ and 0.75 , and the number of nodes in the Hopfield network is changed. The number of nodes starts with 5 and goes up to 100, with a 5 nodes step

for different sizes of the Hopfield network that composes the MLP+H. Fig. 4a shows the obtained results.

In order to enhance the analysis of the results obtained and represented in Fig. 4a, additional experiments were performed, with maximum noise modules of $|R| = 0.25$, $|R| = 0.5$ and $|R| = 0.75$, and variable number of nodes in the Hopfield network. The obtained results are represented in Fig. 4b. It can be seen that when the number of nodes in the Hopfield network is lower than 20, the average performance of the MLP+H is lower than the average performance observed for number of nodes 20 and higher. This occurs because, from the theoretical limits on storage capacity of the Hopfield network [2, 5], the minimum number of nodes to store 3 patterns (which is the number of patterns used in the experiments represented in Fig. 3 and the following ones) is a little higher than 20 nodes, so to guarantee that the Hopfield network presents a good performance. Nevertheless, even in this case, the performance of the hybrid architecture is about 85% when $|R| = 0.25$ and 5 nodes in the Hopfield network, what confirms its little dependency from the number of nodes in the Hopfield network. Beyond 20 nodes, the MLP+H performance does not vary so much as the number of nodes of the Hopfield network grows, similarly to what happens in the results presented in Fig. 4a.

Another test was performed aiming to contrast the performance of the MLP+H versus the isolated MLP in what regards the number of iterations required for the network learning. For this test, the isolated MLP had 3 layers (4/15/20 nodes), while the hybrid network MLP+H had a similar MLP, plus a Hopfield network with 20 nodes, being used the same training set for both architectures. The numbers of needed iterations to the training of each one of the networks were the following:

- the hybrid network MLP+H was trained with 50 iterations (null RMS error);
- the MLP, for a required RMS error of 0.07, was trained with 163 iterations;
- the MLP, for an RMS error of 0.05, 230 iterations;
- the MLP, for an RMS error of 0.05, 230 iterations;
- the MLP, for an RMS error of 0.03, 434 iterations;
- and at last, the MLP, for an RMS error of 0.01, 1805 iterations.

In this way, as previously expected, the hybrid network can learn faster a given training set than the MLP alone, confirming related arguments presented in Section 3. Thus, we see that the MLP+H network has unique features that can result in better performances to solve problems such as the ones here presented, what justifies its use as a good alternative to the classical neural networks here mentioned.

6 Conclusions and Future Works

As a result of the presented studies, the proposed hybrid neural network, the MLP+H, had its validity demonstrated, and many of the advantages initially foreseen during its conception were in fact verified during the experimental studies:

- better performance under situations of noisy inputs, as compared to the other studied architectures;
- smaller number of iterations in the training phase, as compared to the MLP;
- compactness, which allows that small architecture sizes (such as few nodes in the Hopfield network) still can solve problems with good performance and small computational times;
- unique features, which are different from those of the MLP and Hopfield networks, allowing thus the solution of problems which would not be addressed by either one of those two, or would be addressed with a low performance.

With these results, we could verify the validity of the MLP+H hybrid architecture. In respect to future practical applications involving the hybrid architecture MLP+H, we can mention that the use of the concepts and methods here presented and developed is being considered to the implementation of frequency filters with features markedly distinct and superior than the ones obtained with traditional analog and digital filtering techniques [6], such as pass bands with high selectivity.

References

1. Amit, D. J. *Modeling Brain Function: the World of Attractor Neural Networks*, New York: Cambridge University Press, 1989.
2. Haykin, Simon. *Neural Networks: a comprehensive foundation*. MacMillan Pub. Co., 1994.
3. Hopfield, J. J.. "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", in Proc. Nat. Acad. of Sciences, USA, Ap. 1982, V. 79, p.2554-58.
4. Kosko, B., "Bidirectional Associative Memories", IEEE Trans. on Systems, Man, and Cybernetics, Vol. 18, pp.49-60. Jan-Feb, 1988.
5. McEliece, R., E. Posner, E., and S. Venkatesh, "The Capacity of the Hopfield Associative Memory" Information Theory, IEEE Trans. on , Vol.: 33 Issue: 4 , Jul 1987 pp. 461 –482
6. Oppenheim, Alan V., and R. W. Schafer, *Discrete-Time Signal Processing*, Englewood Cliffs, NJ, Prentice Hall, 2000.
7. Rumelhart, D. E. and J. L. McClelland, eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, V1, Cambridge, MA: MIT Press, 1986.

Linear unit relevance in multiclass NLDA networks

José R. Dorronsoro, Ana González, Eduardo Serrano *

Depto. de Ingeniería Informática and Instituto de Ingeniería del Conocimiento
Universidad Autónoma de Madrid, 28049 Madrid, Spain

Abstract. In this work we shall discuss how to apply classical input relevance results for linear Fisher discriminants to measure the relevance of the linear last hidden layer of a Non Linear Discriminant Analysis (NLDA) network. We shall quickly review first possible ways to extend classical and non linear Fisher analysis to multiclass problems and introduce a criterion function very well suited computationally to NLDA networks. After defining a relevance statistic for linear NLDA units, we shall numerically illustrate the resulting procedures on a synthetic 3 class classification problem.

1 Introduction

One way of looking at Multilayer Perceptrons (MLPs) is to see them as performing a standard linear regression on the outputs of the last hidden layer that is preceded by a nonlinear transformation of the original MLP inputs. Non Linear Discriminant Analysis (NLDA) networks are a novel feature extraction procedure in which the same nonlinear MLP input transformation is followed by Fisher's well known Linear Discriminant Analysis. More precisely, in a NLDA network a standard MLP architecture is used, but the usual MLP square error criterion is replaced by one of those typically used in Fisher's analysis. The situation is particularly simple for two class problems. For them, a NLDA network has just a single output unit (in general it has $C - 1$ outputs in a C class problem) and a typical criterion one could use is the quotient $J(W) = s_W/s_B$, where s_B is the between class covariance and s_W is the within class covariance (see their definition below), that in this case turn out to be scalars. There are several ways to extend this 2 class setting to C class problems, for which s_W and s_B are now full matrices. Probably the better known is the determinant ratio $J(W) = |s_W|/|s_B|$; however, this can result in too costly NLDA gradient computations and in section II we shall review this and other choices and will settle for a simpler, efficient Fisher criterion choice.

In any case, and as with MLPs, optimal NLDA network architectures have to be somehow decided upon, and the purpose of this work is to analyze how this can be done for the linear units of the last hidden layer of an NLDA network. As it is customary in Fisher's analysis, the number of output units is fixed at $C - 1$ for a

* With partial support of Spain's CICYT, TIC 01-572 and CAM 02-18

C class problem. Moreover, input pattern dimension corresponds to the number of input units, although if the relevance of the original features is discussed, this number cannot be taken to be fixed in advance, and may have to be lowered. In any case, the number of hidden layers and of units in each layer have still to be decided. This is a long studied question for MLPs, and many approaches, such as bootstrapping, pruning, regularization or stopped training, have been given (see [8], section 5.6). For NLDA networks the situation is very different and in fact one has to treat separately the last, linear hidden layer and all the other nonlinear layers. For the latter the approach in [2] should give these weights' asymptotic distribution. In any case, this approach, to be studied elsewhere, requires an isolated minimum structure for the optimal weights, something that is not true for the last linear weights. They are in fact obtained by Fisher's eigen analysis are unique only up to a scalar factor. They do not thus correspond to isolated minima and we shall present here a different approach for their study, based on classical results for the relevance of input features in Fisher's analysis. These results shall be given in section III, while in section IV we shall present a numerical illustration of the resulting methods. The paper will close with a short conclusions section.

2 Multiclass Fisher Discriminant Analysis

The general objective of Fisher's Discriminant Analysis is to linearly transform a original feature vector X in a new vector $Y = W^t X$ so that the new features concentrate each class c , $c = 1, \dots, C$, around its mean \bar{Y}^c while keeping these means apart (in what follows A^t will denote the transpose of a vector or matrix A). The dimension of the new features is $C - 1$. The whithin class s_W and between class s_B covariance matrices can be used to capture this objective. They are defined as $s_W = \sum_{c=1}^C \pi_c E_c[(Y - \bar{Y}^c)(Y - \bar{Y}^c)^t]$ for the whithin class covariance and as $s_B = \sum_{c=1}^C \pi_c (\bar{Y}^c - \bar{Y})(\bar{Y}^c - \bar{Y})^t$ for the between class covariance. Here $\bar{Y} = E[Y]$ denotes the overall output mean, $\bar{Y}^c = E_c[Y] = E[Y|c]$ denotes the class conditional output means and π_c are the class prior probabilities. We shall use the total output covariance matrix $s_T = E[(Y - \bar{Y})(Y - \bar{Y})^t]$ instead of s_W ; in any case, these three matrices are related as $s_T = s_W + s_B$.

When $C = 2$, the new features are one dimensional and the above matrices become thus scalars. We can use as projecting weights W those that minimize either the quotient $J'(W) = s_W/s_B$ or equivalently, $J(W) = s_T/s_B = 1 + J'(W)$. For more than 2 classes a suitable scalar criterion has to be defined from the matrices s_T and s_B . The most widely used is $J_1(W) = |s_T|/|s_B|$, as the determinants $|\cdot|$ somewhat measure the scatter of the points of each class. An alternative could be to define the criterion $J'(W) = \text{tr}(s_T)/\text{tr}(s_B)$, with tr denoting the trace operation, but this ignores [1] the effect on class separability of correlations in feature components. To overcome this, s_T can be restricted to be the $C - 1$ identity matrix I_{C-1} , something that can be achieved by the transformation $u^t s_T u$ with $u = (s_T)^{-1/2}$. Applying it to both terms in J' , we arrive at a new criterion $J_2(W) = 1/\text{tr}((s_T)^{-1}s_B)$.

Although seemingly distinct, both J_1 and J_2 have a common solution. In fact, since $s_T = W^t S_T W$ and $s_B = W^t S_B W$, it easily follows [1, 4] that

$$\nabla J_1(W) = 2J_1 (S_T W(s_T)^{-1} - S_B W(s_B)^{-1}), \quad (1)$$

$$\nabla J_2(W) = 2J_2^2 (S_T W(s_T)^{-1} s_B(s_T)^{-1} - S_B W(s_T)^{-1}). \quad (2)$$

Now, if S_T is definite positive, there is an invertible matrix $U = (U_1, \dots, U_D)$ with vector columns U_i that simultaneously diagonalizes S_T to the identity matrix I_D and S_B to a diagonal Λ' , i.e.,

$$U^t S_T U = I_D, \quad U^t S_B U = \Lambda'.$$

Moreover, it is easily seen that $(S_T)^{-1} S_B U = U \Lambda$, i.e., U is made up of the eigenvectors of $(S_T)^{-1} S_B$. If we now take $W_0 = (U_1, \dots, U_{C-1})$, it follows that $s_T = W_0^t S_T W_0 = I_{C-1}$ and $s_B = W_0^t S_B W_0 = \Lambda$, with Λ the $C-1 \times C-1$ upper left submatrix of Λ' . It is now easily seen from this that for W_0 , we have $\nabla J_1(W_0) = \nabla J_2(W_0) = 0$. Moreover,

$$J_1(W_0) = \frac{1}{\prod_1^{C-1} \lambda_i}, \quad J_2(W_0) = \frac{1}{\sum_1^{C-1} \lambda_i},$$

and to minimize them we simply have to take in W_0 the eigenvectors associated to the $C-1$ largest eigenvalues of $(S_T)^{-1} S_B$. We will also assume that they are arranged in descending eigenvalue order, i.e., $\lambda_1 > \dots > \lambda_{C-1}$.

We could use either J_1 or J_2 as criterion functions of NLDA networks. However, their gradients with respect to hidden weights would then be quite costly to compute. To simplify this, we will consider [1] a third criterion, $J_3(W) = \text{tr}(\tilde{\Lambda} s_T)/\text{tr}(s_B)$, where $\tilde{\Lambda} = \frac{1}{\lambda_1} \Lambda$, i.e., $\tilde{\lambda}_1 = 1$. We have

$$\nabla J_3(W) = \frac{2}{\text{tr}(s_B)} (S_T W \tilde{\Lambda} - S_B W J_3),$$

from which it is easily seen that we have again $\nabla J_3(W_0) = 0$. Moreover, since for W_0 we also have $s_T = I_{C-1}$ and $s_B = \Lambda$, it follows that

$$J_3(W_0) = \frac{\text{tr}(\tilde{\Lambda})}{\text{tr}(\Lambda)} = \frac{1}{\lambda_1}.$$

As a consequence, $J_3(W) \geq J_3(W_0) \geq 1$. In fact, since $S_T = S_B + S_W$, $U^t S_W U = I - \Lambda'$. Moreover, S_W is also definite positive and its eigenvalues are thus ≥ 0 ; in particular, its first eigenvalue μ_1 verifies $\mu_1 = 1 - \lambda_1 \geq 0$. Therefore $\lambda_1 \leq 1$ and $J_3 \geq 1$.

The extension of these facts to NLDA networks is now straightforward. They use a standard Multilayer Perceptron (MLP) architecture, i.e., they will have a number N_H of hidden layers, each of them with n_h units, $h = 1, \dots, N_H$. A $n_{h-1} \times n_h$ matrix W^h connects layer $h-1$ to layer h , and the h -th layer

activations $A^h = (a_1^h, \dots, a_{n_h}^h)^t$ are obtained from the $h - 1$ -th layer outputs $O^{h-1} = (o_1^h, \dots, o_{n_{h-1}}^h)^t$ as $A^h = (W^h)^t O^{h-1}$. In turn, the h -th layer outputs are obtained as $o_l^h = \sigma(a_l^h)$, with σ usually the standard sigmoidal or the hyperbolic tangent. In this notation, the input layer is taken as the 0-th hidden layer and the preceding formulae are applied to the h units with $h = 1, \dots, n_h - 1$; we'll assume for bias effects that the n_h output has a constant value of 1. However, no bias is assumed for the last hidden layer and the $C - 1$ dimensional (as customary in Fisher Analysis) network outputs Y are simply obtained by a linear transform $Y = (W^O)^t O^{N_H}$, with W^O now a $n_{N_H} \times (C - 1)$ matrix. The network transfer function $Y = F(X, W) = F(X, W_1, \dots, W_{N_h}, W^O)$ is thus just that of a standard MLP. The difference with standard MLPs lies in the choice of the optimal weights. NLDA optimal weights will be those that minimize the just defined Fisher criterion J_3 . In what follows we shall drop the subindex.

3 Linear hidden unit relevance for NLDA networks

Optimal NLDA network architectures, just as it happens with MLPs, have to be somehow decided upon. A possible way to deal with all layers except the last one (including network inputs) could be to find the asymptotic distributions of the weights near an optimum weight set W^* . In the 2 class case, the general procedure for this can be found in [2]. The starting point is the following general result [5].

Theorem 1. *Let $\Psi(X, W)$ be a vector valued function depending on a random variable X and a parameter vector W such that $\nabla_W \Psi(X, W)$ and $\nabla_W^2 \Psi(X, W)$ exist for all X and are bounded by integrable functions. If $E[\Psi(X, W_*)] = 0$ at an isolated point W_* and the matrices*

$$\mathcal{H}^* = \mathcal{H}(W_*) = E[\nabla_W \Psi(X, W_*)], \quad \mathcal{I}^* = \mathcal{I}(W_*) = E[\Psi(X, W_*) \Psi(X, W_*)^t]$$

are respectively definite positive and non singular. Then, if X_n is a sequence of i.i.d. random vectors distributed as X , the equation $\sum_1^N \Psi(X_n, W) = 0$ has a solution sequence \widetilde{W}_n converging in probability to W_ and such that $\sqrt{N}(\widetilde{W}_n - W_*)$ converges in distribution to a multivariate normal with 0 mean and variance $\mathcal{C}^* = \mathcal{C}(W^*) = (\mathcal{H}^*)^{-1} \mathcal{I}^* (\mathcal{H}^*)^{-1}$.*

To apply this for multiclass NLDA networks we just have to express the gradient ∇J as the expectation of a certain random vector. This has been done in [3], where in the case for instance of an NLDA network with just one hidden layer it is shown that ∇J can be written as

$$\frac{\partial J}{\partial w_{kl}}(W) = E[\Psi_{kl}(X, W)] = E[z_{kl}] - \lambda_{kl},$$

where we have

$$z_{kl}(X, W) = \frac{2J}{\text{tr}(s_B)} \left(\sum_{i=1}^{C-1} \lambda_i w_{li} y_i \right) f'(a_l) o_k$$

and the non random term $\lambda_{kl}(W)$ is given by

$$\lambda_{kl}(W) = \frac{2J}{\text{tr}(s_B)} \left(\sum_1^C \pi_c \sum_1^{C-1} w_{li} E_c[y_i] E_c[f'(a_l)] - \sum_i (\lambda_i - 1) w_{li} E[y_i] E[f'(a_l) o_k] \right).$$

The application of the resulting criteria to input and nonlinear hidden units will be studied elsewhere. In any case, this approach cannot be applied to the last hidden layer and as an alternative we shall consider input relevance tests used in classical Fisher analysis. Remember that in our context, the values of the last hidden outputs would be taken as the inputs of a classical Fisher transformation. Among the several options available, we shall follow the approach essentially proposed by Rao [7] (see also [6]). We begin with the particularly simple situation of discrimination into two classes. Assume we have an input vector $Z = (z_1, \dots, z_D)^t$ for classical linear discriminant for which we want to test the discrimination relevance of the last input component z_D . The linear Fisher weights are then given by $W = S_T^{-1}(\mu_1 - \mu_2)$, where μ_i denotes the mean of the class $i = 1, 2$, and we can assess the relevance of z_D testing whether $w_D = 0$. Using the notations $\mu_1 - \mu_2 = \delta = (\delta', \delta_D)^t$ with δ_D the D -th component of δ and δ' its complementary vector, and writing $W = (W', w_D)^t$ and

$$S_T = \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix}$$

with $S_{11} = \text{cov}(z_1, \dots, z_{D-1})^t$ and $S_{22} = \text{cov}(z_D)$, it follows that

$$\begin{pmatrix} W' \\ w_D \end{pmatrix} = \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix}^{-1} \begin{pmatrix} \delta' \\ \delta_D \end{pmatrix} = \begin{pmatrix} S^{11} & S^{12} \\ S^{21} & S^{22} \end{pmatrix} \begin{pmatrix} \delta' \\ \delta_D \end{pmatrix}$$

Therefore, $w_D = 0$ if and only if $S^{21}\delta' + S^{22}\delta_D = S^{22}((S^{22})^{-1}S^{21}\delta' + \delta_D) = 0$. Since it follows from the definition of S^{22} and S^{21} we have $(S^{22})^{-1}S^{21} = -S_{21}S_{11}^{-1}$, we can test the relevance of z_D for 2 classes in terms of the null hypothesis H_0

$$\delta_2 - S_{21}S_{11}^{-1}\delta_1 = \mu_D^1 - \mu_D^2 + S_{21}S_{11}^{-1}((\mu^1)' - (\mu^2)') = 0.$$

For the general C class case, we can apply the previous reasoning for all pairs of classes i and j with $i \neq j$, which leads to the general null hypothesis H_0 in this situation

$$\mu_D^i - \mu_D^j + S_{21}S_{11}^{-1}((\mu^i)' - (\mu^j)') = 0, \quad i \neq j = 1, \dots, C.$$

# hid. units	2	3	4	5	6	7	8	9	10	11	12	13	14
# conv. runs	19	8	9	14	15	15	16	16	18	17	16	16	19

Table 1. Converging runs for each number of hidden units. A run is considered to converge for 2 and 3 unit networks if its training error is below 40% and 20% respectively, while it must be below 1 % for 4 and more unit networks.

If we assume that the random vector Z follows a homoscedastic normal model, that is, that the conditional distributions $Z|i$ are multivariate normals with a common covariance Σ , and we define

$$t = \frac{|S_W|}{|S_B + S_W|}, \quad t' = \frac{|S_W^{11}|}{|S_B^{11} + S_W^{11}|},$$

with S_W^{11} and S_B^{11} denoting the whithin and between class covariance matrices of the random vector $Z^1 = (z_1, \dots, z_{D-1})^t$ of the first $D - 1$ components of Z , and set $\tilde{t} = t - t'/1 + t'$, Rao [7] has shown that for a sample of size N and under the null hypothesis H_0 , the statistic

$$F = \frac{N - C - D + 1}{C - 1} \left(\frac{1 - \tilde{t}}{\tilde{t}} \right)$$

follows an F distribution with $C - 1$ and $N - C - D + 1$ degrees of freedom. Although the homoscedastic normal hypothesis is not likely to be true for many problems, the statistic F can still be used to assess the relevance of a given unit in the last hidden layer. We shall see how this can be done in the next section.

4 Numerical experiments

For NLDA linear units, and as just mentioned, the statistic \tilde{t} will follow an F distribution if the output vector O has a homoscedastic normal distribution, something not likely to be true in practice. Because of this we shall use as an alternative a direct comparison of \tilde{t}_h values for the different hidden units. The simplest way to proceed is to just compare for different architectures some ratio of the statistic F values for different units, such as, for instance, the ratio R between the maximum of F to its minimum value that we shall use here. When all hidden units are relevant, we should expect $R \simeq 1$, while R should be $>> 1$ when redundant hidden units are introduced.

We shall illustrate how to use the above procedures to arrive at an optimal number of hidden units. We shall work with a 3-class synthetic problem with two dimensional patterns. Gaussian distributions centered at the opposite corners of the bidimensional unit cube define two of these classes, while the third one is given by a single Gaussian centered at $(0.5, 0.5)$. These gaussians are isotropic with a common standard deviation of 0.1, resulting in a mean error probability (MEP) for the optimal Bayes classifier of about 0.53 %. We have trained

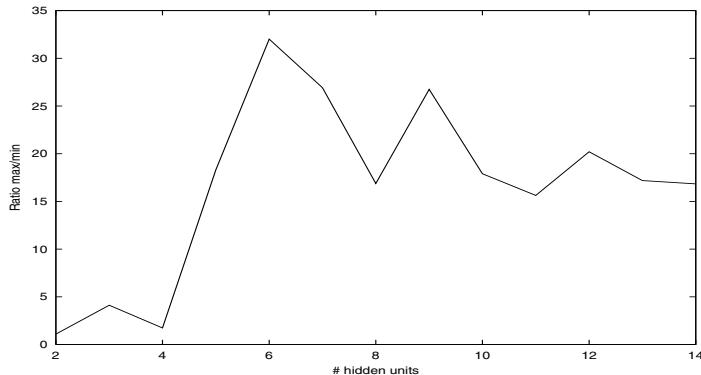


Fig. 1. Ratio of maximum to minimum F statistic values.

NLDA networks with a 3×165 point sample, whose error probability is about 0.2 %, that is, smaller than the optimal MEP. On the other hand, the test set has 3×500 points, and its error probability is also about 0.5 %. The optimal architecture is not clear a priori. Notice that although the classes are well separated, the problem is somewhat difficult, as these classes are geometrically quite intertwined. Furthermore, since the number of classes is 3, the smallest NLDA architecture must have at least 2 hidden units. We have started our runs at this number of hidden units going all the way to 14. Each run had 20 trials starting at random initial weights and NLDA training was done using natural gradient descent [3]. Notice that, independently of the statistic's behavior, an acceptable architecture must give a test classification error not too far away from the 0.53 % test set value. This is only the case for 4 or more units, and while for 2 and 3 units we have kept runs with a training error below 40 % and 20% respectively, for 4 and more hidden units, we have only kept those runs with an error below 1 %. Table 1 gives for the different hidden units the number of converging runs with the above error restrictions, while figure 1 shows for each number of hidden units the ratios between the maximum and minimum statistic values. There is clearly a jump at 5 units while for 2, 3 and 4 units the ratio is close to one. In other words, the F statistic has similar values for all the hidden units of the 2, 3 and 4 hidden unit networks, while for networks with 5 and more hidden units, the smallest statistic value starts dropping well below the maximum value. This figure suggests that 4 is an adequate number of hidden units. Figure 2 gives additional weight to this assertion. It gives for each number of hidden units the corresponding average error values measured upon the test set. Observe that the average test error oscillate between 0.6 and 0.7 %, that is, quite close to the optimal MEP. In other words, an a posteriori check on test set error values gives also 4 as an optimal number of hidden units.

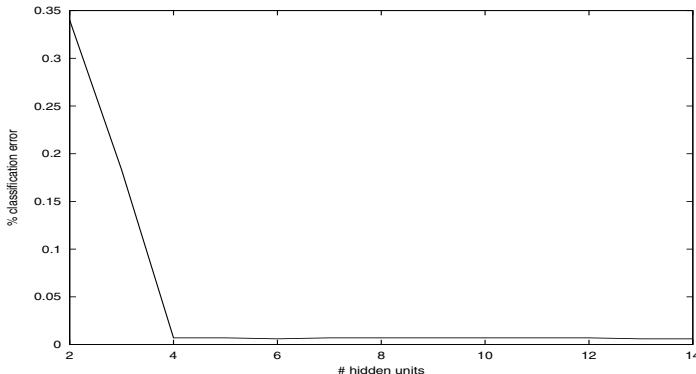


Fig. 2. Average test set classification errors.

5 Conclusions and further work

After reviewing possible multiclass extensions of Non Linear Discriminant Analysis networks, in this work we have proposed a statistic to analyze the optimal number of linear units for such a network. Although based on classical relevance results for Fisher's linear discriminants, the usual homoscedastic assumptions from which the statistic distribution are derived in the classical setting do not hold here. We have used instead the ratio of the maximum of the statistic to its minimum to assess the optimal number of hidden units and have shown on a synthetic example that this gives good results. Further work shall delve more deeply on this statistic while also considering how to define relevance criteria for the nonlinear units of an NLDA network.

References

1. P.A. Devijver, J. Kittler, **Pattern Recognition: A Statistical Approach**, Prentice Hall, 1982.
2. J.R. Dorronsoro, A. González, C. Santa Cruz, "Arquitecture selection in NLDA networks", Proceedings of ICANN 2001, LNCS 2130, Springer Verlag 2001, 27–32.
3. J.R. Dorronsoro, A. González, "Natural gradient learning and Multiclass NLDA networks", Proceedings of ICANN 2002, LNCS 2415, Springer Verlag 2002, 673–678.
4. K. Fukunaga, **Introduction to Statistical Pattern Recognition**, Academic Press, 1972.
5. E.B. Manoukian, **Modern Concepts and Theorems of Mathematical Statistics**, Springer, 1986.
6. G. McLachlan, **Discriminant analysis and statistical pattern recognition**, John Wiley, 1992.
7. C.R. Rao, **Linear Statistical Inference and its Applications**, Wiley, 1973.
8. B.D. Ripley, **Pattern Recognition and Neural Networks**, Cambridge U. Press, 1996.

Bootstrap for model selection: linear approximation of the optimism

G. Simon¹, A. Lendasse², M. Verleysen^{1,‡}

Université catholique de Louvain

¹ DICE - Place du Levant 3, B-1348 Louvain-la-Neuve, Belgium,
Phone : +32-10-47-25-40, Fax : +32-10-47-21-80
{gsimon, verleysen}@dice.ucl.ac.be

² CESAME - Avenue G. Lemaître 4, B-1348 Louvain-la-Neuve, Belgium,
lendasse@auto.ucl.ac.be

Abstract. The bootstrap resampling method may be efficiently used to estimate the generalization error of nonlinear regression models, as artificial neural networks. Nevertheless, the use of the bootstrap implies a high computational load. In this paper we present a simple procedure to obtain a fast approximation of this generalization error with a reduced computation time. This proposal is based on empirical evidence and included in a suggested simulation procedure.

1 Introduction

A large variety of models may be used to describe processes: linear ones, nonlinear, artificial neural networks, and many others. It is thus necessary to compare the various models (for example with regards to their performances and complexity) and choose the best one. The ranking of the models is made according to some criterion like the generalization error, usually defined as the average error that a model would make on an infinite-size and unknown test set independent from the learning one.

In practice the generalization error can only be estimated, but there exists some methods to provide such an estimation: the AIC or BIC criteria and the like [1], [2], [3] as well as other well-known statistical techniques: the cross-validation and k-fold [3, 6], the leave-one-out [3, 6], the bootstrap [4, 6] and its unbiased extension the .632 bootstrap [4, 6]. The ideas presented in this paper can be applied both to the bootstrap and the .632 bootstrap.

Although these methods are roughly asymptotically equivalent (see for example [5] and [6]), and despite the fact that the use of the bootstrap is not an irrefutable question, it seems that using the bootstrap can be advantageous in many “real world”

[‡] G. Simon is funded by the Belgian F.R.I.A. M. Verleysen is Senior Research Associate of the Belgian F.N.R.S. The work of A. Lendasse is supported by the Interuniversity Attraction Poles (IAP), initiated by the Belgian Federal State, Ministry of Sciences, Technologies and Culture. The scientific responsibility rests with the authors.

modeling cases (i.e. when the number of samples is limited, the dimension of the space is high, etc.) [6].

But the bootstrap main limitation in practice is the computation time required for assessing an approximation of sufficient reliability (or *accuracy*). A second limitation, in our context of model selection, is the fact that the selected best model is picked up from a set of a priori chosen models, leading to a restricted choice.

In a previous work [7], we have proposed a fast approximation of the generalization error using the bootstrap, based on linear and exponential approximations of the optimism and apparent error (as defined by Efron [4]) respectively. In this paper, we prove experimentally the validity of the linear approximation of the optimism, and show how to use this approximation to perform efficient bootstrap simulations with reasonable computational complexity.

2 Model Selection Using Bootstrap Technique

The fundament of the bootstrap is the plug-in principle [4]. This general principle allows to obtain an estimator of a statistic according to an empirical distribution. In our context of model selection, our statistic of interest is the generalization error. We thus use the bootstrap to estimate the generalization error (or the prediction error in Efron's vocabulary) in order to rank the models and choose the best one.

The bootstrap estimator of the generalization error is computed according to the bootstrap resampling approach. Given an original sample (or data set) x , we generate B new samples, denoted x^b , $1 \leq b \leq B$. The new samples x^b are obtained from the original sample x by drawing with replacement. For each bootstrap sample x^b , we compute a bootstrap estimator of our statistic of interest. The final value is obtained by taking the mean of the estimators over the B bootstrap replications. In the following, we will use the notation $e_{A,B}$ for the error of a model built (learned) on a sample A and tested on a sample B .

In model selection context, Efron defines in [4] the bootstrap estimator of the generalization (prediction) error:

$$\hat{e}_{gen} = e_{app} + optimism, \quad (1)$$

where \hat{e}_{gen} is the estimate of the generalization error e_{gen} given by bootstrap, e_{app} is the apparent error (computed on the learning set A), and *optimism* is an estimator of the correction term for the difference between a learning and a generalization error, which in fact aims to approximate the difference of errors obtained on the finite sample x and an (infinite) unknown ideal sample. The optimism is computed according to:

$$optimism = E_B [optimism^b], \quad (2)$$

where $E_B[\cdot]$ is the statistical expectation computed over the B bootstrap replications and:

$$\text{optimism}^b = e_{x^b, x} - e_{x^b, x^b}. \quad (3)$$

With our notation, (1) becomes:

$$\hat{e}_{gen} = e_{x, x} + E_B [e_{x^b, x} - e_{x^b, x^b}] \quad (4)$$

In order to approach the theoretical value of the final bootstrap estimation of the generalization error, we can increase the number B of bootstrap replications, but this increases considerably the computation time. We have proposed in [7] a way to reduce this computation load with a limited loss of accuracy.

Note that the .632 bootstrap [4] aims to reduce the slight bias introduced by the *optimism* correction. This bias is due to e_{x^b, x^b} where the error is computed on the same set than the one used for the learning stage. The linear approximation of the *optimism* term, presented in the following, is applicable to the bootstrap and the .632 bootstrap.

3 Framework

Assuming a linear relation between the optimism and the number p of parameters in the model is probably an unexpected hypothesis. Nevertheless, this hypothesis is strengthened by the fact that the general formulation of a structure selection criterion can also be written as

$$\hat{e}_{prediction} = e_{app} + \text{correction} \quad (5)$$

where the *correction* term is $2p\sigma/n$ for AIC and $\ln(n)p\sigma/n$ for BIC, with σ the estimated quadratic error on the learning set containing n elements. In AIC, BIC criteria and the like, we can see that the *correction* term is directly proportional to the number of parameter p . Though the apparent error e_{app} is also a function of p , we will focus here on the second term, the *correction*.

Although the *correction* term is computed by bootstrap, and therefore called the *optimism*, its value depends, as the apparent error, on the initialization conditions of the learning process. In practice, a "good" local minimum of a learning error (either on x or on x^b) is obtained by repeating the learning with different initial conditions. Nevertheless, when including this in a bootstrap procedure, the number of learnings is again multiplied by B , resulting in an excessive computation time.

In comparison with the AIC and BIC criteria, we assume that the *correction* term is linearly increasing. Our first goal is then to show experimentally that the *optimism* term is a linear function of p , like a_1p+a_2 .

Under this hypothesis, if we compute the value of the *optimism* term for a limited number of models, we can determine (in mean square sense) constants a_1 and a_2 . Our second goal is thus, under the linearity hypothesis, to propose a method to reduce the number of tested models and the number of bootstrap replications.

Since the values of a_1 and a_2 result from an experimental procedure, an obvious advantages of our proposal is these parameters are set specifically for each application, avoiding the use of asymptotic results.

4 Methodology

In the experimental results shown below, we used Radial Basis Function Networks (RBFNs) as approximation models. We would like to emphasize on the fact that this choice is made a priori and that the goal is not to compare the results with those that could be obtained with other approximators. The learning procedure to fit the parameters of the model is described in [8], [9].

For the RBFNs models, we consider p in expression a_1p+a_2 as the number of Gaussian units or Gaussian kernels (the total number of parameter in RBFNs is in fact proportional to the number of Gaussian kernels). To observe the linearity, we use the R^2 statistics, also called the square correlation coefficient. The R^2 statistics is here computed between the *optimism* estimated for each model (different values of p) and the linear approximation (a_1p+a_2) of these values. The more this R^2 is close to 1, the most our linear approximation is valid.

Remember that each *optimism*^b, in the context of nonlinear models, is usually the result of several learnings (Q learnings) with different initial conditions. To estimate one value of the *optimism* (i.e. the *optimism* for a specific model complexity p), we should therefore learn QB models, what could be excessive in our context. Now notice that in practice we are not interested in a specific value of the *optimism* but only in the linear approximation a_1p+a_2 . A lower accuracy on each value of the *optimism* can thus be balanced by the number of different complexities p , i.e. the number of points (larger than 2) used for the linear approximation.

5 Experimental Results

5.1 Artificial Example

We first illustrate the validity of the linear approximation of the *optimism* described in the previous sections on a toy example. We generate a set of 1000 datas (x, y) , with x randomly drawn in $[0, 1]$ and y defined by:

$$y = \sin(5x) + \sin(15x) + \sin(25x) + noise \quad (6)$$

where *noise* is a uniform random variable in $[-0.5, 0.5]$.

We then use the bootstrap resampling method in a model selection procedure, observing the generalization error corresponding to a specific model characterized by its number p of Gaussian kernels. Figure 1 presents the evolution of our R^2 criterion versus the number B of bootstrap replications. We clearly see that R^2 is getting closer and closer to one while B increases. Fishers' test (with a p-value of $2.2584 \cdot 10^{-11}$) leads to accept the linear hypothesis from $B = 15$.

Since we admit the linear approximation hypothesis, we can go one step further and address the reduction of computation time. We then look to the evolution of a_1 and a_2 in function of B respectively in Figures 2.1 and 2.2. Here again, when B is greater or equal to 15, we have a roughly constant value. Figure 3 shows the graph of the optimism according to the number p of Gaussian kernels in the model. Figure 4 is the graph of the generalization error versus the number of bootstrap replications, where we can see that the “best” model for our toy example has 20 Gaussian kernels. Finally, Figure 5 shows the 1000 learning data and the predictions we got with the selected model.

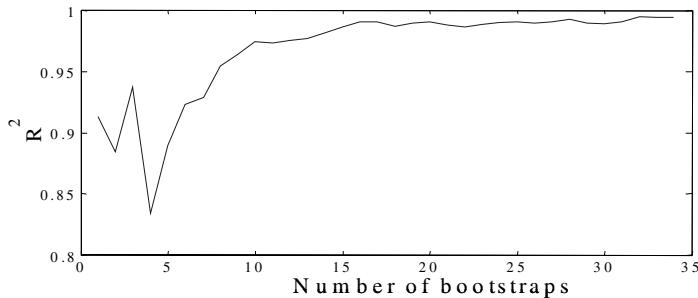


Fig. 1. Toy example: evolution of R^2 versus B

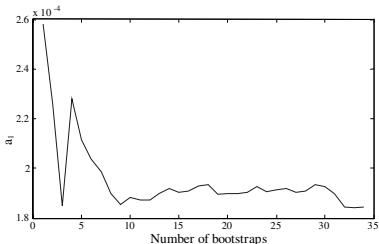


Fig. 2.1 Toy example: evolution of coefficient a_1 versus B

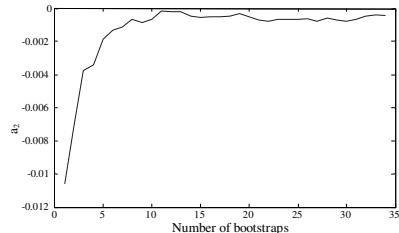


Fig. 2.2. Toy example: evolution of coefficient a_2 versus B

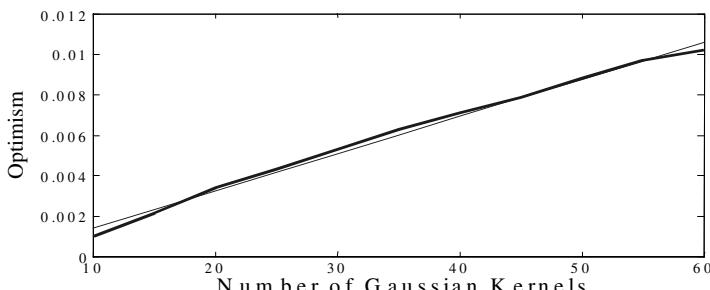


Fig. 3. Toy example: approximation of the *optimism* term (thin line) versus the number of Gaussian kernels with $B = 20$ (thick line : values obtained for the tested models)

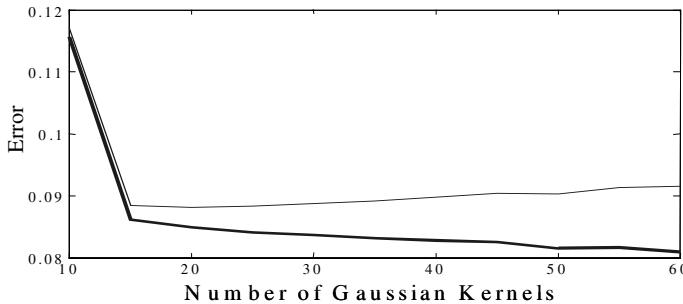


Fig. 4. Toy example: Learning (thick) and generalization (thin) errors versus p

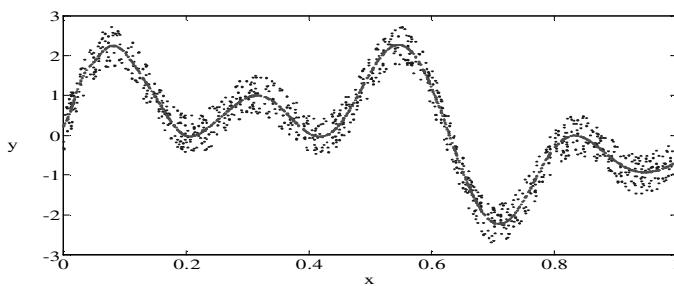


Fig. 5. Toy example: learning data (dots) and predictions (solid line) with the selected model

5.2 Real Data Set (Abalone)

We use the abalone dataset [10] as a second example to validate the linearity hypothesis, with a more realistic (and difficult) approximation problem. Here again, we use 1000 data for the learning. Figure 6 is the evolution of R^2 with respect to B . In this case, Fisher's test p-value is rounded by the computer to 0. Figure 7.1 is the graph of a_1 and figure 7.2 is the evolution of a_2 in function of B . According to these graphs, we suggest to use $B = 40$. Figure 8 shows the reported optimism with respect to p . Figure 9 shows the learning and generalization errors versus B . The minimum corresponding to the “best” model for the Abalone data set has 62 Gaussian kernels.

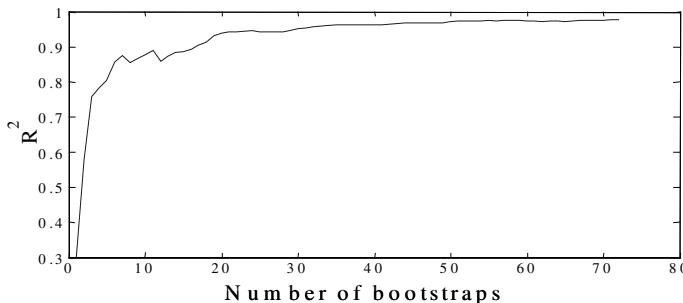


Fig. 6. Abalone: evolution of R^2 versus B

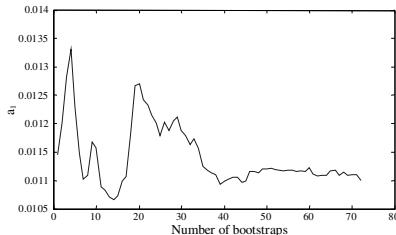


Fig. 7.1 Abalone: evolution of coefficient a_1 versus B

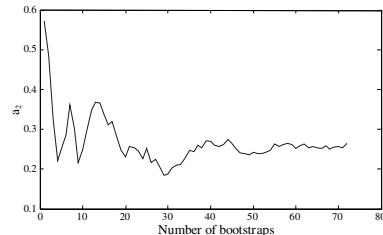


Fig. 7.2 Abalone: evolution of coefficient a_2 versus B

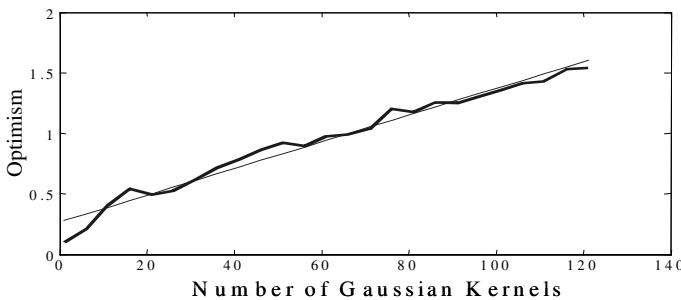


Fig. 8. Abalone: approximation of the *optimism* term (thin) versus the number of Gaussian kernels with $B = 40$ (thick: values obtained for the tested models)

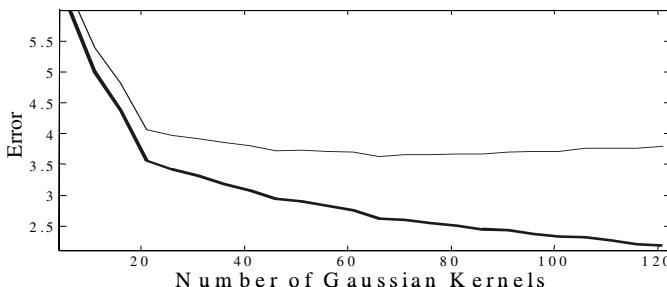


Fig. 9. Abalone: Learning (thick) and generalization (thin) errors versus the number p of Gaussian kernels

6 Conclusion

In this paper we have shown that the *optimism* term of the bootstrap estimator of the prediction error is a linear expression of the number of parameters p .

Furthermore, we illustrate the time saving procedure proposed in [7], enhanced here with the early stop criterion based on the R^2 of the linear approximation. According to the two results shown here and to other ones not illustrated in this paper,

we recommend a conservative value of 50 for the number B of bootstrap replications before stopping the approximation computation.

We would like to emphasize on the fact that the limited loss of accuracy is balanced by a considerable saving in computation load, this last fact being the main disadvantage of the bootstrap resampling procedure in practical situations. This saving is due to the reduced number of tested models and to the limited number of bootstrap replications.

Although this procedure has only been tested in a neural network model selection context, this simple and time saving method could easily be extended to other contexts of nonlinear regression, classification, etc., where computation time and complexity play a role. It can also be applied to other resampling procedures, as the .632 bootstrap.

References

- [1] H. Akaike, “*Information theory and an extension of the maximum likelihood principle*”, 2nd Int. Symp. on information Theory, 267-81, Budapest, 1973
- [2] G. Schwarz, “*Estimating the dimension of a model*”, Ann. Stat. 6, 461-464, 1978.
- [3] L. Ljung, “*System Identification - Theory for the user*”, 2nd ed, Prentice Hall, 1999.
- [4] B. Efron, R. J. Tibshirani, “*An introduction to the bootstrap*”, Chapman & Hall, 1993.
- [5] M. Stone, “*An asymptotic equivalence of choice of model by cross-validation and Akaike's criterion*”, J. Royal. Statist. Soc., B39, 44-7, 1977.
- [6] R. Kohavi, “*A study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection*”, Proc. of the 14th Int. Joint Conf. on A.I., Vol. 2, Canada, 1995.
- [7] G. Simon, A. Lendasse, V. Wertz, M. Verleysen, “*Fast approximation of the bootstrap for model selection*”, accepted for publication in Proc. of ESANN'2003, d-side, Brussels, 2003.
- [8] N. Benoudjit, C. Archambeau, A. Lendasse, J. Lee, M. Verleysen, “*Width optimization of the Gaussian kernels in Radial Basis Function Networks*”, Proc. of ESANN'2002, d-side, Brussels, 2002.
- [9] M. J. Orr, “*Optimising the Widths of Radial Basis Functions*”, in Proc. of Vth Brazilian Symposium on Neural Networks, Belo Horizonte, Brazil, december 1998
- [10] W.J. Nash, T.L. Sellers, S.R. Talbot, A.J. Cawthron and W.B. Ford, “*The Population Biology of Abalone (*Haliotis* species) in Tasmania. I. Blacklip Abalone (*H. rubra*) from the North Coast and Islands of Bass Strait*”, Sea Fisheries Division, Technical Report No. 48, 1994.

A learning rule to model the development of orientation selectivity in visual cortex

Jose M. Jerez¹, Miguel Atencia², Francisco J. Vico¹, and Enrique Dominguez¹

¹ Escuela Técnica Superior de Ingeniería en Informática
Departamento de Lenguajes y Ciencias de la Computación
Universidad de Málaga

jja@lcc.uma.es

² Escuela Técnica Superior de Ingeniería en Informática
Departamento de Matemática Aplicada
Universidad de Málaga

Abstract. This paper presents a learning rule, CBA, to develop oriented receptive fields similar to those founded in cat striate cortex. The inherent complexity of the development of selectivity in visual cortex has led most authors to test their models by using a restricted input environment. Only recently, some learning rules (PCA and BCM rules) have been studied in a realistic visual environment. The CBA rule proposed in this work is tested in different input visual environments and the results are compared to those achieved by the BCM and PCA rules. The final results show that the CBA rule is appropriate for studying the biologically process of receptive field formation in visual cortex.

1 Introduction

Among the different approaches to imitate the perceptual capabilities of biological systems, neural-based models have been proposed in the last decades [1–5], and some have been tested in natural scenarios [6, 7]. Stimulating a single neuron model with natural images, PCA [8] and BCM [3] learning rules were shown to develop receptive fields (RFs) similar to those found in visual cortex in the early experiments of Hubel and Wiesel [9, 10]. Exposing the neuron to some stimulation trials transformed a random receptive field in one selective to orientation. Each trial included the presentation of a patch of size 13×13 pixels, obtained from a set of 24 grey-scale 256×256 pixels images, that has been processed with a DOG filter. Preferred orientations of the resulting RFs spread out widely, and concentrated slightly in the range from 80 to 120 since the images contained vegetal forms, that aligned more in vertical orientation.

The resulting RFs contained excitatory and inhibitory regions arranged in a preferred orientation. The emergence of these regions have to do with the potentiating (LTP) and depressing (LTD) character of the learning rule. According to the Hebbian postulate, both rules include LTP terms, but differ in the way they implement LTD. While PCA incorporates heterosynaptic competition, BCM produces a similar effect through homosynaptic competition. These

two forms of LTD reinforce inhibition by means of spatial competition among the afferents of a neuron in the case of PCA, or temporal competition in the case of BCM. The fact that each of these learning rules rely on a single mechanism for LTD strongly influences the final shape of the RFs, and, consequently, the type of processing performed by the neuron. The RFs resulting of a PCA training are sensitive to low spatial frequencies (only two regions are differentiated), while those obtained with CBA show selectivity to high frequencies (three or more bands). Both, homosynaptic and heterosynaptic competition have been described in the nervous system [11–13], and its combined effect might yield the wide range of spatial frequencies that are captured by the RFs of the striate cortex cells [14]. Although, in principle, the BCM learning rule seems to be more suitable to achieve sensitivity to both low and high frequencies with a proper parameter set, the temporal competition that implements its LTD mechanism makes hard the fitting process. This problem arises when the BCM theory is tested using the images from a camera mounted on a freely moving robot [15].

Taking into account these functional limitations and biological constraints, we propose here a new learning rule that incorporates homosynaptic and heterosynaptic competition. This rule is derived from the one proposed in [16] for neural assemblies formation, with the only difference that incorporates a decay term.

The rest of this paper is organized as follows. In Section 2 the model is presented. In Section 3, first, the rule is simulated within a restricted visual environment and then, realistic images are presented to the model. Both experiments suggest that receptive fields are formed, which mix properties of the BCM and the PCA rules. Finally, Section 4 summarizes the main conclusions, and some lines for future research are provided.

2 The model

The neuron single model consists of a vector \mathbf{x} of inputs, representing an averaged presynaptic activity originated from another cell, a vector \mathbf{w} of synaptic weights, and a scalar output y , given by $y = \mathbf{w} \cdot \mathbf{x}$, that represents an averaged postsynaptic activity. The weight vector \mathbf{w} can take negative values, since they can be considered as effective synapses, made up of multiple excitatory and inhibitory connections. Once the activation equation is defined, we face the problem of modelling the weight modification process that represents learning. In a previous work [16] we proposed a new correlational learning rule (BA, for bounded activity) that formed stable neural attractors in a recurrent network. The CBA learning rule is essentially a modification of the BA rule in which an extra term to implement the heterosynaptic LTD has been incorporated. Thus, the resulting synaptic modification equation for the CBA rule is a Hebbian-type learning rule with an specific form of stabilization, defined as

$$\frac{d w_i}{dt} = \alpha x_i y (y - \tau)(\lambda - y) - \beta y w_i = f(w) \quad (1)$$

where α is the learning rate. The rationale behind the introduction of the remaining parameters is now explained. The term λ avoids the unbounded weight growing in a plausible way, and can be interpreted as a neuronal parameter representing the maximum level of activity at which the neuron might work. The CBA modification equation also defines a threshold τ that determines whether depression or potentiation occurs when both the pre- and postsynaptic neurons fire. Finally, the parameter β controls the heterosynaptic competition effect, such that the strength of synapses can change even in the absence of presynaptic activity to those synapses. The β value should be lower than α to preserve the dominant character of LTP and homosynaptic LTD over heterosynaptic LTD. All these parameters adopt positive values. The effect of this adaptation mechanism in the receptive fields formation process is that the graded response elicited after stimulus presentation leads the neural activity either to high or resting levels.

3 Simulation results

Before doing simulations in a realistic visual environment, the simulations performed on a simpler input environment (one and two dimensions) will provide a qualitative insight on the system. In this context, dimension means number of afferent synaptic connections.

3.1 Low dimensional environment

In the one-dimensional case we have only one differential equation, where both the input x and the weight y are scalars. The fixed points would be the weight values w that satisfy the condition $f(w) = 0$:

$$w_0 = 0, \quad w_1 = \frac{\gamma + R}{2x^3\alpha}, \quad w_2 = \frac{\gamma - R}{2x^3\alpha} \quad (2)$$

where the parameters γ , ρ and R have been defined as

$$\gamma = -\beta + x^2\alpha(\lambda + \tau), \quad \rho = 2x^2\alpha\sqrt{\lambda\tau}, \quad R = \sqrt{\gamma^2 - \rho^2} \quad (3)$$

It is instructive to observe the function $f(w)$ in equation (1), which has been drawn in Figure 1. The geometrical intuition suggests that if $f(a) > 0$ (e.g. if a is largely negative) and the initial state of the system is $w = a$, w will increase. On the other hand, starting from $w = a$, w will decrease if $f(a) < 0$, e.g. if a is largely positive. The increasing or decreasing evolution will continue until a fixed point is reached, but the system "corrects" itself so that its state does not blow up to $\pm\infty$.

Although the one dimensional model gives us an idea about the system dynamics, one cannot obtain selectivity with this restricted environment. In this sense, we define a two-dimensional environment composed by two input patterns, x_1 and x_2 , presented to the neuron with equal probabilities. Figure 2 illustrates

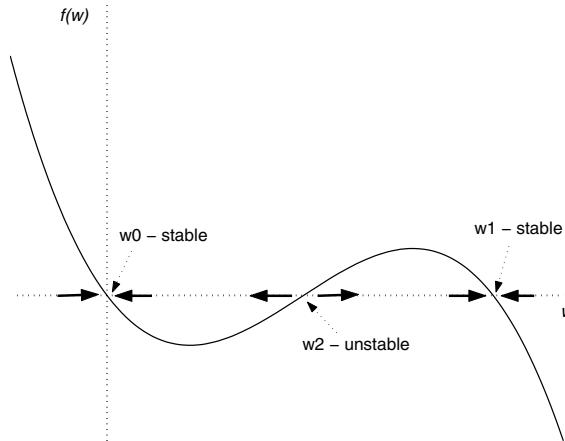


Fig. 1. Nonlinear differential equation modelling the behaviour of the system.

the trajectories followed by different weight initializations (drawn as circles) in a states space, what provides an description about the weights dynamics in this restricted environment.

In this figure we can observe one attractor fixed point, w_1 , one saddle point, labelled as w_2 , and a set of initial stabled points, w_0 , located in a perpendicular plane to the attractor point. The figure shows an attractor basin towards a line of points crossing by zero, such a way that every weight initialization inside the region located between this line and the parallel one crossing by the unstable point, will yield the system not to develop selectivity.

Blais [17] proposed the analysis of the output distribution of the neuron at the fixed points as a useful tool to compare the behavior of different learning rules. In Figure 3 the output distributions for PCA, BCM and CBA learning rules are compared for the two-input environment. The results show that the PCA rule is trying to have most of its responses strong, BCM rule tries to have a small subset of its responses strong and the others weak, and CBA gives the maximum response strong to an input pattern.

These results might help us to predict the structures of the receptive fields achieved by these three learning rules trained in a more realistic visual environment.

3.2 Realistic visual environment

The visual environment used in this section is similar to that described in Law et al. [6], and it is composed by 24 natural images scanned into 256x256 pixel images, where man-made objects have been avoided, since they would make easier to achieve receptive fields, given their sharp edges and straight lines characteristics. The retina model is composed of square arrays of receptors which have antagonistic center-surround receptive fields that approximate a difference of

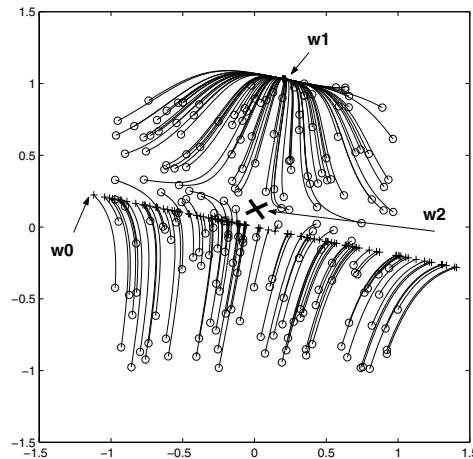


Fig. 2. States-space indicating the weights dynamics from different initial values (represented as circles). These results were achieved setting the learning constant, α , to the value 0.05, the maximum activity level, λ , was set to the value 1.0, the threshold, τ , was 0.25, and the heterosynaptic competition term, β , was set to 0.0025.

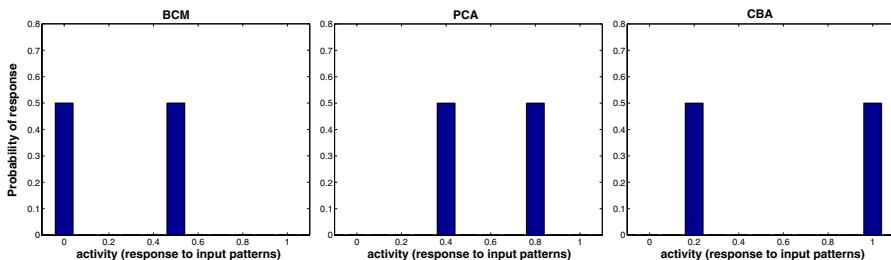


Fig. 3. Output distributions for BCM, PCA and CBA learning rules. BCM seeks orthogonality to one of the input vectors, PCA tries to maximize responses to the set of input vectors, while CBA maximizes the projection to one input vector (giving the maximum response strong).

Gaussian (DOG) filter. The ratio of the surround to the center of the Gaussian distribution is approximately 3:1, which has been biologically observed in [18].

The model neuron was trained with 13x13 pixels patches randomly taken from the images. For every simulation step, the activity of the input cells in the retina is determined by randomly picking one of the 24 images and randomly shifting the receptive field mask. The activity of each input in the model is determined by the intensity of a pixel in the image. The exact time course of these simulations depend on the parameter chosen, so we have examined these over a large range. Table 1 shows the range of parameters used to obtain the results presented at Figure 4.

Table 1. Setting of learning rule parameters for simulations in a realistic visual environment.

Learning constant, α	0.005
Maximum level of activity, λ	1.0
Threshold level, τ	0.15
Heterosynaptic competition term, β	0.00025
Input values range, x	[−0.10, 0.10]
Weights initialization range, w_0	[−0.15, 0.15]
Number of iterations	250000

Figure 4 shows the weights resulting from these simulations starting from different initial conditions. With this realistic input environment, the CBA neuron develops receptive fields with distinct excitatory and inhibitory regions. Notice, also, that the variety of oriented receptive fields structures obtained is significant enough.

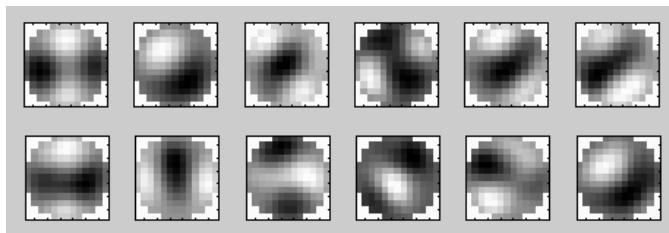


Fig. 4. Different types of cortical receptive fields arising from the CBA learning rule. The individual plots show the weights vector for two-dimensional receptive field with white denoting positive values and black negative values (synaptic efficacies).

Figure 5 shows examples receptive for BCM and PCA rules trained in the same visual environment as CBA rule. These oriented receptive fields are similar to those experimentally observed by Hubel and Wiesel [9, 10]. However, BCM receptive fields are clearly selective to bars of lights at different orientations, whereas PCA develops receptive fields always divided into two antagonist regions, one of them with synaptic potentiation and the other one with synaptic depression. At this point, establishing a comparison to the receptive fields structures in Figure 4, we can assess that the CBA learning rule can develop receptive fields with properties similar to those achieved by both PCA and BCM rules. Effectively, Figure 4 shows examples receptive fields with the same structure as PCA receptive fields, and others becoming selective to bars of lights at different positions, but with an spatial frequency less than the receptive fields achieved by the BCM rule.

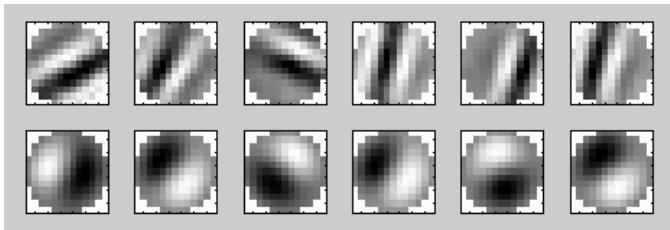


Fig. 5. Examples receptive fields achieved by BCM (top) and PCA (bottom) trained in a realistic visual environment composed by natural images.

4 Conclusions and future work

This paper has shown that the CBA learning rule is appropriate to develop cells with oriented receptive fields in visual cortex.

This learning rule contributes with a term for controlling the synaptic growing, such that any additional weight saturation and normalization constraint is avoided. Besides, the CBA rule integrates both heterosynaptic and homosynaptic methods through different parameters in the synaptic modification equation. The results have shown that, in a realistic visual environment, the CBA rule develops oriented receptive fields similar to those achieved by both BCM and PCA learning rules. In addition, the simulation results presented robustness and a high level of stability on a wide range of parameter values.

Two immediate steps arise from this research as future works. On the one hand, it is preceptive to study the properties of CBA modification dynamics and the influence of the learning rule parameters in normal and deprived environments through experiments of visual deprivation. Also, the process of direction selective receptive fields formation in visual complex cells can be studied in terms of the CBA rule. On the other hand, the receptive fields achieved by this learning rule might be considered as filters susceptible of being applied as the first stage in the features extraction process carried out in image processing and artificial vision tasks. Finally, an exhaustive mathematical analysis of the CBA rule must be done in both one- and n-dimensional environment, determining the stability conditions as well as the basins of attraction for the system fixed points. This analysis will provide a better understanding of the CBA fundamental properties, and a mathematical relation among the parameters of the learning rule identifying a region where the system works properly.

References

1. Sejnowsky, T.: Storing covariance with nonlinearly interacting neurons. *Journal of Math. Biology* **4** (1977) 303–321
2. Von der Malsburg, C.: Self-organization of orientation sensitivity cells in striate cortex. *Kybernetik* **14** (1973) 85–100

3. Bienenstock, E., Cooper, L., Munro, P.: Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. *Journal of Neuroscience* **2** (1982) 32–48
4. Linsker, R.: From basic network principles to neural architecture: Emergence of orientation-selective cells. *Proceedings of the National Academy of Science* **83** (1986) 8390–8394
5. Miller, K.: A model for the development of simple cell receptive fields and the ordered arrangement of orientation columns through activity-dependent competition between on-and off-center inputs. *Journal of Neuroscience* **14** (1994) 409–441
6. Law, C., Cooper, L.: Formation of receptive fields according to the bcm theory of synaptic plasticity in realistic visual environment. *Proceedings of the National Academy of Science* **91** (1994) 7797–7801
7. Shouval, H., Liu, Y.: Principal component neurons in a realistic visual environment. *Network* **7** (1996) 501–515
8. Oja, E.: A simplified neuron model as a principal component analyzer. *Mathematical Biology* **15** (1982) 267–273
9. Hubbel, D., Wiesel, T.: Receptive fields, binocular interaction and functional architecture in the cats visual cortex. *Journal of Physiology* **160** (1962) 106–154
10. Hubbel, D., Wiesel, T.: Receptive fields and functional architecture of monkey striate cortex. *Journal of Physiology* **195** (1968) 215–243
11. Artola, A., Brcher, S., Singer, W.: Different voltage-dependent threshold for inducing long-term depression and long-term potentiation in slices of rat visual cortex. *Nature* **347** (1990) 69–72
12. Abraham, W., Goddard, G.: Asymmetric relationships between homosynaptic long-term potentiation and heterosynaptic long-term depression. *Nature* **305** (1983) 717–719
13. Lynch, G., Dunwiddie, T., Gribkoff, V.: Heterosynaptic depression: A postsynaptic correlate of long-term potentiation. *Nature* **266** (1977) 737–739
14. Miller, K.: Synaptic economics: Competition and cooperation in correlation-based synaptic plasticity. *Neuron* **17** (1996) 371–374
15. Neskovic, P., Verschure, P.: Evaluation of a biologically realistic learning rule, bcm, in an active vision system. (1998)
16. Vico, F., Jerez, J.: Stable neural attractors formation: Learning rules and network dynamics. *Neural Processing Letters* **to appear** (2003)
17. Blais, B.: The role of the environment in synaptic plasticity: Towards an understanding of learning and memory. PhD thesis, Brown University (1998)
18. Linsenmeier, R., Frishman, L., Jakielka, H., Enroth-Cugell, C.: Receptive field properties of x and y cells in the cat retina derived from contrast sensitivity measurements. *Vision Research* **22** (1982) 11731183

Sequence learning using the neural coding

Sorin Moga¹, Philippe Gaussier², and Jean-Paul Banquet³

¹ Département IASC / ENST Bretagne
BP 832, Brest, 29285, France

sorin.moga@enst-bretagne.fr

² ETIS / University of Cergy
6, av. Ponceau, Cergy, 95014, France
gaussier@ensea.fr

³ IFR Neurosciences / INSERM U483
47, bv. de l'Hopital, 75651 Paris, France
banquet@ccr.jussieu.fr

Abstract. This article introduces a neural network capable of learning a temporal sequence. Directly inspired from a hippocampus model [2], this architecture allows an autonomous robot to learn how to imitate a sequence of movements with the correct timing. The results show that the network model is fast, accurate and robust.

1 Introduction

This article considers the problem of learning to predict events, i.e. to forecast the future behavior of a system using past experience. This problem has often been viewed and formalized in neural network theory as so-called temporal sequence learning. Studying such sequences is a topic of research in several domains such as robotic trajectory planning, speech or vision processing. For most of these, neural networks provide two distinct mechanisms: one for spatial information and the other for temporal information. The main mechanism stores the sequence events regardless of the temporal dependences between them. In parallel, or later, the second mechanism, the so-called short time memory (STM), extracts and learns the temporal relationships between the events.

Moreover, we have shown in previous works [4, 1] that the capability of learning a temporal sequence is one of most important features of a learning by imitation system. As a capability of learning by observation, imitation is a strong learning paradigm for autonomous systems. Imitation can improve and accelerate the learning of sensory/motor associations. In our work, oriented to the design of a neural network architecture allowing learning by imitation, we are involved in the first level of imitation [8]. This “proto-imitation” level plays a key role in understanding the principles of the perception/action mechanisms necessary to perform higher order behaviors and it is likely that the protoimitation is triggered by a perception ambiguity. In our approach, the starting point for an “imitating behavior” implementation is the capability of learning temporal sequences of movements.

2 Temporal sequence learning model

Almost all neural models of sequence learning use a discrete temporal dimension by sampling the continuous time at regular intervals. In these models, time proceeds by intervals of Δt , and the interval between 2 items of a sequence is considered as a few units of Δt (usually less than 10). In this section, we introduce the model⁴ for timing sequence learning with a variable and a long range time interval and we evaluate it.

2.1 Timing learning model

Our model (Fig. 1) is based on the idea that a prediction (P-type) neuron learns the timing between 2 items, or, to be more precise, learns to predict the end of this time interval. This time interval starts with the firing of a derivation (D-type) neuron and ends with the firing of an input (E-type) neuron. The P-type neuron learns this interval using the activity of the granular (G-type) group of neurons.

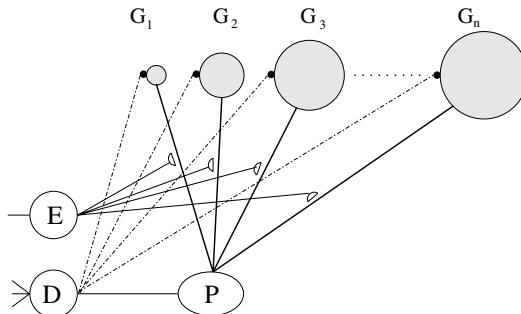


Fig. 1. Model allowing time prediction. The G_i are time base neurons, P is the prediction neuron, D and E are formal neurons.

The D-type neuron acts as a novelty detector. Its output becomes high (i.e. it fires) if the derivative of its inputs is positive:

$$D(t) = \begin{cases} 1, & \text{if } \frac{\Delta D(t)}{\Delta t} > 0 \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

⁴ The model is inspired by the functions of two brain structures involved in memory and time learning: the cerebellum and the hippocampus (see [2] for further neurobiological references)

The firing of the D neuron resets the activity of all G neurons by means of direct links (Fig. 1) between them. The E neuron act as the D neuron does. It fires at the end of the interval to be predicted. When the E neuron fires, the links between the P neuron and the G neurons are updated using Eq. (4). The G neurons are aggregated in batteries of cells associated with the recognition of a given stimulus. We suppose the reaction time of a given cell depends for instance on its size (like the granular cells in the cerebellum [5]). Hence the batteries of G cells are useful for a spectral timing decomposition of the input signal. The activity of a battery of G neurons is reset by the activation of the next input signal (first reset of the different batteries, next activation of the selected battery). For sake of simplicity, a group of G neurons is arranged in a line. After the reset, by the way of the D neuron, the activity of a G neuron is computed as follows:

$$G_i(t) = \frac{m_0}{m_i} \exp - \frac{(t - \tau_i) - m_i)^2}{2 \cdot \sigma_i^2} \quad (2)$$

i position of the neuron in the group (the i th cell of the battery);
 m_i and σ_i time constant and the standard deviation associated with the
 i th neuron;
 τ_i instant of the last reset of the battery.

The temporal activity of six neurons of one battery is shown in Fig. 2-left.

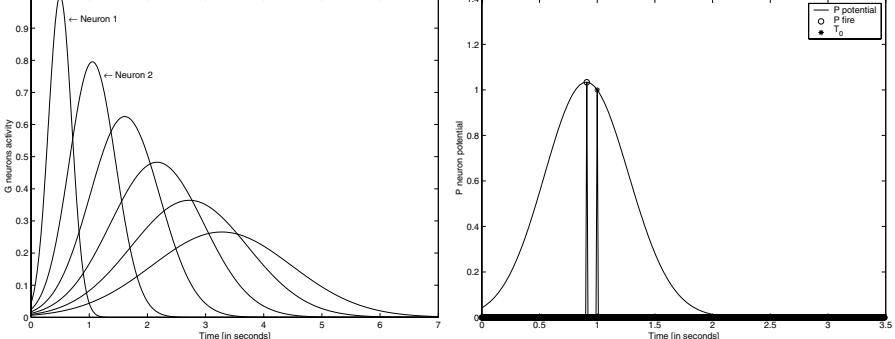


Fig. 2. Left: time activity of a group of 6 G neurons (cells) that allows us to measure time with $\tau_i = 0$. Right: detailed activity of the P-type neuron.

A P neuron receives inputs from the D neuron (the start of the interval) and from G neurons (a temporal trace). The potential of the P neuron is the weighted sum of the inputs from the D neuron and the delayed activity in G neurons. It is computed as

$$P^{pot}(t) = \sum_{l=1}^N W_P^{G_l} \cdot G_l(t) + W_P^D \cdot D(t) \quad (3)$$

with $W_P^{G_i}$ the weight of the link from the i^{th} G neuron and the P neuron and W_D^P the weight of the link from D to P.

The time between the D neuron activation and the E neuron activation is learned in the G-P connection weights. The weights of links between the P neuron and the G neurons are updated according to

$$W_P^{G_i} = \begin{cases} \sum_j \frac{G_i}{G_j^2}, & \text{if } E(t) > 0 \text{ (i.e. E fires)} \\ \text{unmodified, otherwise.} \end{cases} \quad (4)$$

The P neuron fires only when its potential reaches its maximum value, which corresponds to the prediction of the end of the time interval.

$$P(t) = P^{\text{act}}(t) = f_P(P^{\text{pot}}(t)) \quad (5)$$

$$f_P(x(t)) = \begin{cases} 1, \text{ if } \frac{dx(t)}{dt} < 0 \text{ and } \frac{dx(t-\tau)}{dt} > 0 \\ 0, \text{ otherwise.} \end{cases} \quad (6)$$

Let us consider a simple example: we present the first item and, one second later, we present the second item. The length of the interval to be learned is $T_0 = 1\text{s}$. The first item forces the D neuron to fire. The firing of the D neuron resets all the activity of the G neurons. Starting at this instant, the G neuron's activity is expressed by Eq. 2 (Fig. 2-left). One second later, the second item forces the E neuron to fire. The firing of the E neuron enables the update of the weight between the P and the G neurons (see Eq. 4), i.e. enables the learning of the T_0 interval. Finally, when the first item is presented again the D neuron fires again. 920 milliseconds later, i.e. 80 milliseconds before t_0 , the P neuron fires (Fig. 2-right) and predicts an imminent firing of the E neuron.

The firing of the P neuron before the firing of the E neuron is the main characteristic of this model. Let $T_{ISI} = [0, t_{ISI}]$ be a time interval with $t_{ISI} \in]0, \infty[$ the length of the learned time interval. If the P neuron fires at $t_p > t_{ISI}$ then it is impossible to decide if P fires due to the D neuron firing or due to the past learning (firing forced by G neurons). Therefore, we consider that there is a prediction if and only if $t_p < t_{ISI}$.

We introduce $Pr = t_{ISI} - t_p$ the difference between the learned time interval and the time of the firing of the P neuron. If we have $Pr > 0$ then we have a “prediction”. A prediction Pr_1 is better than Pr_2 if and only if $Pr_1, Pr_2 > 0$ and $Pr_1 < Pr_2$, i.e. t_{p_1} is closer to t_{ISI} than t_{p_2} . We also introduce the “negative prediction” to represent the $Pr < 0$ case. On the other hand, if we intend to use our model as a novelty detector and we have a *poor* prediction (the P fires a “long” time before t_{ISI}) then we have a good *anticipation*. The results shown below are interpreted from the prediction point of view. In order to visualize the results, we plot Pr as a t_{ISI} function (Fig. 3). The graph is divided into 2 intervals with different characteristics. For $t_{ISI} \in [0, 3]$, the prediction is *unstable* i.e. Pr is negative or positive. If $t_{ISI} \in [3, 7]$ then the prediction is always positive and linear. Prediction precision decreases with the t_{ISI} increase: shorter t_{ISI}

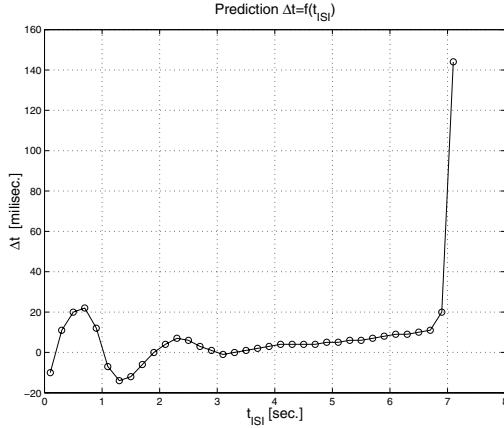


Fig. 3. The prediction as an ISI function ($N = 40$, Gauss, $m_0 = m_1 = 1$, $m_N = 10$, lin, $\sigma_1 = 0.1$, $\sigma_N = 0.2$, lin). The circles are the computed values; the line is the interpolation result.

determines a better precision and we find the same results as for a system which verifies the psycho-physics of Weber law⁵. These results are correlated with the results of well-known timing learning models [5, 7].

3 Sequence learning architecture

Timing learning models are currently used by neurobiologist modelers for conditioning simulation [3]. Alternatively, the proposed model permits the temporal sequence of events to be learned and predicted. In this section, we introduce a neural architecture destined to learn simple or cyclic sequences. In our context, a simple sequence is defined as an enumeration of events with the associated timing (eg. “A,B,C”) and a cyclic sequence as periodic simple sequence (eg. “A,B,C,A,B,C,A,...”) with the associated timing.

3.1 Learning simple sequences

The main idea is to use several batteries of G neurons for learning the timing between two consecutive events and a group of P type neurons for learning the event sequencing. The global architecture is shown in Fig. 4. The input group (CC) can be viewed as the input interface. Any neuron of this group represents a sequence event and it is ON while the corresponding input event is present; otherwise it is OFF. Each CC neuron is one-to-one linked with EC group of

⁵ The concept that a just-noticeable difference in a stimulus is proportional to the magnitude of the original stimulus. Weber’s Law states that the ratio of the increment threshold to the background intensity is a constant.

neurons. The EC group is made up of D-type neurons (Eq. 1). Each EC neuron is linked with unconditional links to all neurons of a battery in the DG group. In the same way, an EC neuron is connected with unconditional links to all neurons of the corresponding column of the CA3 group. The DG group integrates

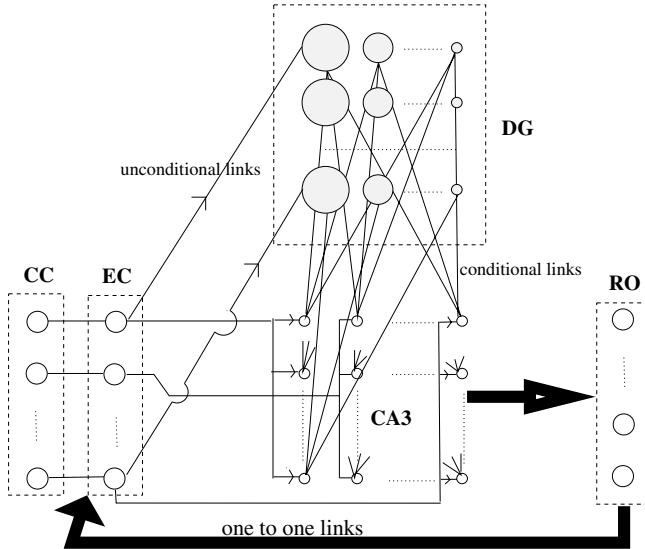


Fig. 4. Detailed connectivity of the event prediction network. The circle size in TB is associated with the time constants (m_j) of the neurons.

several batteries of G-type neurons. A DG battery is equivalent to a G_i group of neurons shown in section 2.1. Each DG neuron is connected via conditional links to all neurons of the corresponding row in the CA3 group. The size of the CC group (respectively EC) is constrained by the maximum length of sequences. Alternatively, the size of a DG battery is a function of the prediction precision of the time interval between two events of the sequence. This architecture can learn all event combinations, in other words, all possible sequences. Consequently, the size of the CA3 group is the square of the input group (CC). The output group (RO) is a Winner Take All neurons group. A neuron of the RO group has the same signification as a CC neuron. The RO outputs are connected to the EC inputs via one-to-one secondary unconditional links.

Now we will describe the system behavior. For the sake of simplicity, let us consider a three-event sequence $((A; t_A), (B; t_B), (C; t_C))$. At the start, all conditional link weights are reset and the unconditional links weights are set to 1. First of all, event A is activated. The activation of the first input neuron (CC_1) determines the corresponding neuron (EC_1) to fire (Fig. 5-left-a). The activation of the EC neuron resets the activity of the first DG battery and the

τ_1 parameter in Eq. (2) (the G_i^1 neuron activation) is set to the precise value of the computer clock at this instant (to be sure the computation will not depend on the computer speed nor depend on any task switching problem at the operating system level).

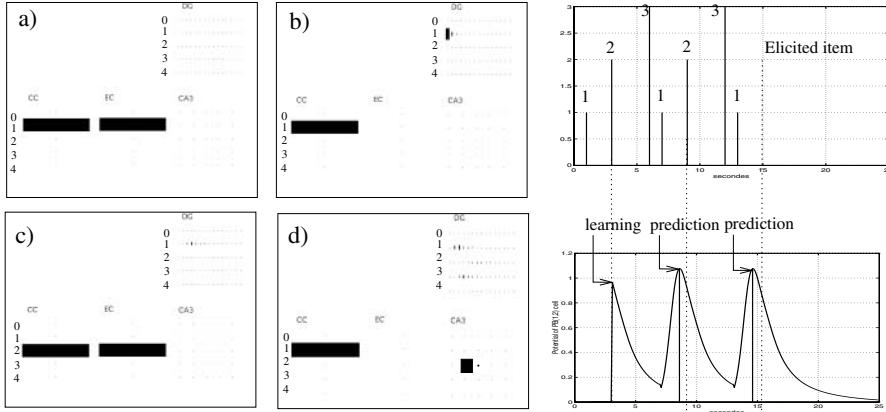


Fig. 5. Left: The detailed activation for the A-B interval learning. Right: top, a three-event sequence $((A; t_A), (B; t_B), (C; t_C))$ (the heights of events are different only for visual comprehension purposes); bottom, the potential of the $CA3_1^1$ neuron, which learns and predicts the transition from A to B.

While the CC_1 neuron remains activated, the EC_1 neuron is silent (there is no variation on its input) and the first DG battery fires (Fig. 5-left-b). When the B event occurs, the CC_2 neuron fires and allows the EC_2 to fire (Fig. 5-left-c). Due to EC_2 firing, two actions are enabled: first, the update of the weights from all the DG neurons to the second column of the CA3 group, in other words, the timing between event A and B ($[t_a, t_b]$) is learned; second, the activity of the second battery of the DG group is reset. For the third event we have the same behavior as for the second. At this time, the sequence “A,B,C,” is learned and may be used. A new “A” event allows the system to predict the “A to B” transition (Fig 5-left-d) and triggers the firing of the RO_2 . Due to the architecture (Fig. 4), the RO_2 firing is equivalent to the B input (the EC_2 firing) and allows the system to predict the “B to C” transition, and so on.

In general, any sequence will be learned in the same manner. The size of learned sequences is limited only by the system memory capacity.

3.2 Cyclic sequence learning

Supposing that we try to learn a cyclic sequence “A,B,C,A,B,...” shown in Fig. 5-right top. Regarding the network architecture and simple sequence learning, this

is quite simple. The behavior of the system is identical till the end of the “simple” sequence. After that, when the A event occurs again, the timing between the last (C) and the first (A) event of the sequence is learned and allows the learning of cyclic sequences. The network keeps its predictive behavior as shown in Fig. 5.

4 Conclusion

The timing leaning model and the associated neural networks were successfully utilized in [4, 6] to teach an autonomous robot different “dances”. The proposed model allows the correct timing to be predicted and it concords with Weber’s law. Even if Weber’s law concordance is not a prerequisite, it corresponds to a strong constraint of neurobiological inspired models of perception and learning: if the prediction precision is constant then it is not possible to have reinforcement due to repetitive experiments. In addition, this model was successfully employed [1] to build a learning model based on the prediction of rhythms as a reward signal⁶ and for spatio-temporal transition learning in autonomous robot navigation. These results prove that the proposed neural network architecture can serve both as a starting point for understanding imitation mechanisms, and as an effective learning algorithm for autonomous robotics.

References

1. P. Andry, P. Gaussier, S. Moga, and J. Nadel. Learning and communication via imitation: an autonomous robot perspective. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 31(5):431–442, 2001.
2. J.P. Banquet, P. Gaussier, J.L. Contreras-Vidal, and Y. Burnod. The cortical-hippocampal system as a multirange temporal processor: A neural model. In R. Park and D. Levin, editors, *Fundamentals of neural network modeling for neuropsychologists*, Boston, 1998. MIT Press.
3. D. Bullock, J.C. Fiala, and S. Grossberg. A neural model of timed response learning in the cerebellum. *Neural Networks*, 7(6/7):1101–1114, 1994.
4. P. Gaussier and S. Moga. From perception-action loops to imitation processes: A bottom-up approach of learning by imitation. *Applied Artificial Intelligence: An international Journal*, 12(7–8):701–727, 1998.
5. S. Grossberg and N. Schmajuk. Neural dynamics of adaptive timing and temporal discrimination during associative learning. *Neural Networks*, 2:79–102, 1989.
6. S. Moga. *Apprendre par imitation : une nouvelle voie d’apprentissage pour les robots autonomes*. PhD thesis, Université de Cergy-Pontoise, 2000.
7. M Reiss and J G Taylor. Storing temporal sequences. *Neural networks*, 4:773–787, 1991.
8. A. Whiten and R. Ham. On the nature and evolution of imitation in the animal kingdom: Reappraisal of a century of research. In P.J.B. Slater, J.S. Rosenblatt, C. Beer, and M. Milinski, editors, *Advances in the study of behavior*, pages 239–283, San Diego, CA, 1992. Academic Press.

⁶ see http://www-etis.ensea.fr/~neurocyber/G_neurocyber.html for videos.

Improved Kernel Learning Using Smoothing Parameter Based Linear Kernel

A B M Shawkat Ali¹ and Ajith Abraham²

¹School of Computing and Information Technology, Monash University, Victoria 3842, Australia. Shawkat.Ali@infotech.monash.edu.au

²Department of Computer Science, Oklahoma State University, Tulsa, OK 74106, USA. ajith.abraham@ieee.org

Abstract. Kernel based learning has found wide applications in several data mining problems. In this paper, we propose a modified classical linear kernel using an automatic smoothing parameter (Sp) selection compared with the existing approach. We designed the Sp values using the Eigen values computed from the dataset. Experiment results using some classification related benchmark datasets reveal that the improved linear kernel method performed better than some of the existing kernel techniques.

1. Introduction

The classification problem has been introduced since Fischer's theory of linear discrimination in the mid 30's of the last century. Until the backpropagation algorithm [1] found success in several application domains, the perceptron method could not get enough attention mainly because of some theoretic limitations. Since the mid 80's neural networks have become very popular in the field of pattern recognition and machine learning. In the early nineties of the last century emerged a new methodology of learning from data called Support Vector Machines (SVM) [2-4]. In the beginning, it was proposed for binary classification only, but later on, it has been explored for solving regression problems [5] densities estimation [6] and so on. Previous studies [7-12] show that there exists no special kernel, which has the best generalization performance for all problem domains. Our paper is organised as follows. In Section 2, we present some theoretical frameworks related to SVM followed by kernel space in Section 3. Experimentation setup and results are provided in Section 4 and some conclusions are provided towards the end.

2. Support Vector Machine

SVMs always follow to structural risk minimisation principle, closely related to regularisation theory. This principle incorporates capacity control to prevent overfitting and thus is a partial solution to the bias-variance trade-off dilemma [7]. SVM has two key techniques, one is the mathematical programming and the other one is kernel functions. The parameters are found by solving a quadratic programming problem with linear equality and inequality constraints; rather than by solving a non-

convex, unconstrained optimisation problem. The flexibility of kernel functions allows the SVM to search a wide variety of hypothesis spaces. Now we need to focus on SVMs for two-class classification, the classes being T, F for $\{y_i = +1, -1\}$ respectively. This can easily be extended to k -class classification by constructing k two-class classifiers [13]. The geometrical interpretation of support vector classification (SVC) is that the algorithm searches for the optimal separating hyperplane that is, in a sense, equidistant from the two classes [14]. This optimal separating hyperplane has many nice statistical properties [13]. SVC is outlined first for the linearly separable case. Kernel functions are then introduced in order to construct non-linear decision surfaces. Finally, for noisy data, when complete separation of the two classes may not be desirable, slack variables are introduced to allow for training errors. When the training data (x 's) are linearly separable then we get a pair (ω, ω_0) such that

$$\begin{aligned} \omega^T \mathbf{x}_i + \omega_0 &\geq +1, & \text{for all } \mathbf{x}_i \in T \\ \omega^T \mathbf{x}_i + \omega_0 &\leq -1, & \text{for all } \mathbf{x}_i \in F \end{aligned} \quad (1)$$

with the activate function given by

$$f_{\omega, \omega_0}(\mathbf{x}) = \text{sign}(\omega^T \mathbf{x}_i + \omega_0) \quad (2)$$

ω and ω_0 are termed as the weight vector and bias respectively. The inequality constraints (1) can be combined to give

$$y_i(\omega^T \mathbf{x}_i + \omega_0) \geq 1, \quad \text{for all } \mathbf{x}_i \in T \cup F \quad (3)$$

Without loss of generality the pair (ω, ω_0) can be rescaled such that

$$\min_{i=1, \dots, l} |\omega^T \mathbf{x}_i + \omega_0| = 1$$

this constraint defines the set of canonical hyperplanes [13] on \Re^N . In order to restrict the expressiveness of the hypothesis space, the SVM searches for the simplest solution that classifies the data correctly. The learning problem is hence reformulated as: minimize $\|\omega\|^2 = \omega^T \omega$, subject to the constraints of linear separability (3). This is equivalent to maximising the distance, normal to the hyperplane, between the convex hulls of the two classes (margin). The optimisation is now a convex quadratic programming (QP) problem

$$\begin{aligned} \text{Minimize}_{\omega, \omega_0} \Phi(\omega) &= \frac{1}{2} \|\omega\|^2 \\ \text{subject to } y_i(\omega^T \mathbf{x}_i + \omega_0) &\geq 1, \quad i = 1, \dots, l. \end{aligned} \quad (4)$$

3. Kernel Space

SVM can be used to learn non-linear decision functions by first mapping the data to some higher dimensional *feature space* and constructing a separating hyper-plane in this space. Denoting the mapping to feature space by

$$\begin{aligned} X &\rightarrow H \\ \mathbf{x} &\mapsto \Phi(\mathbf{x}) \end{aligned}$$

the decision functions (4) become

$$\begin{aligned} f(\mathbf{x}) &= \text{sign}(\Phi(\mathbf{x})^T \omega^* + \omega_0^*) \\ &= \text{sign}\left(\sum_{i=1}^l y_i \lambda_i^* \Phi(\mathbf{x})^T \Phi(\mathbf{x}_i) + \omega_0^*\right) \end{aligned} \quad (5)$$

Note that the input data appear in the training (4) and activation functions (5) only in the form of inner products $\mathbf{x}^T \mathbf{z}$, and in the decision function (2) only in the form of inner products $\Phi(\mathbf{x})^T \Phi(\mathbf{z})$. Mapping the data to H is time consuming and storing it may be impossible, e.g. if H is infinite dimensional. Since the data only appear in inner products we require a computable function that gives the value of the inner product in H without explicitly performing the mapping. The $*$ indicating the optimal values. Hence, introduce a *kernel function*,

$$K(\mathbf{x}, \mathbf{z}) \equiv \Phi(\mathbf{x})^T \Phi(\mathbf{z}). \quad (6)$$

The kernel function allows us to construct an optimal separating hyperplane in the space H without explicitly performing calculations in this space. Training is the same as (4) with the matrix D having entries $D_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$, i.e. instead of calculating inner products we compute the value of K . This requires that K be an easily computable function. For instance the polynomial kernel $K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z} + 1)^d$, which corresponds to a map Φ into the space spanned by products of up to d dimensions of \Re^N . The decision function (4) becomes

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^l y_i \lambda_i^* K(\mathbf{x}, \mathbf{x}_i) + \omega_0^*\right) \quad (7)$$

where the bias is given by

$$\omega_0^* = y_i - \omega^* T \Phi(\mathbf{x}_i) = y_i - \sum_{j=1}^l y_j \lambda_j^* K(\mathbf{x}_j, \mathbf{x}_i) \quad (8)$$

for any support vector \mathbf{x}_i . Among all the known kernel functions, polynomial, rbf, and sigmoid are the most widely used:

- *Polynomial kernel (poly_c):* $K(\mathbf{x}, \mathbf{x}_i) = (\mathbf{x} \cdot \mathbf{x}_i + 1)^p$.
The result of an SVM with polynomial kernel is a polynomial of degree p .

- *Classical Radial Basis Function (RBF) kernel (rbf_c):* $K(\mathbf{x}, \mathbf{x}_i) = \exp(-\|\mathbf{x} - \mathbf{x}_i\|^2)$
- *Modified Radial Basis Function (RBF) kernel (rbf_m):* $K(\mathbf{x}, \mathbf{x}_i) = \exp(-\|\mathbf{x} - \mathbf{x}_i\|^2/2\sigma^2)$
The result of an SVM with RBF kernel is an RBF network with σ^2 as the variance of the RBF function (bandwidth).
- *Sigmoid kernel:* $K(\mathbf{x}, \mathbf{x}_i) = \tanh(\kappa \mathbf{x} \cdot \mathbf{x}_i - \delta)$.
We consider the classical linear kernel and the improved version as follows:
- *Classical Linear kernel (linear_c):* $K(\mathbf{x}, \mathbf{x}_i) = (\mathbf{x} \cdot \mathbf{x}_i)$.
- *Improved Linear kernel (linear_m):* $K(\mathbf{x}, \mathbf{x}_i) = (\mathbf{x} \cdot \mathbf{x}_i)/Sp$.

We observed that the variance of the data set is inversely proportional to the smoothing parameter. Due to this, we calculated the variance from the Eigen values, which can give the maximum variance of a dataset.

Algorithm for improved linear kernel with SVM

Step 1. Splitting the dataset

Randomly split the data set into V (say 10) disjoint subsets

- each set size is roughly N/V
- N is the total number of examples

Step 2. Smoothing parameter (Sp) estimation

Construct the covariance matrix among the data points $\mathfrak{R} = \frac{1}{p} \sum_{p_i=1}^p x(p_i)'x(p_i)$

To get the maximum Eigen value E_{i_max}

$$\text{Calculate } Sp = \frac{1}{E_{i_max}}$$

Step 3

For $i = 1:10$,

construct the model using 90% of the examples

$$f(\mathbf{x}) = \text{sgn} \left(\sum_{i=1}^l y_i \lambda_i^* \mathbf{x}^T \mathbf{x}_i + \omega_0^* \right)$$

Calculate accuracy $[ACC(i)]$ for the rest of 10% examples

end

Step 4

$$\text{Cross Validation Accuracy} = \frac{1}{V} \sum_{i=10}^V ACC(i).$$

4. Experimental Setup and Results

We compared the performance of the improved linear kernel with different kernel functions. We considered the binary class as well as multi-class data from two different sources [15,16]. While *bos*, *iris* and *wine* datasets belong to multi-class the

remaining six belong to binary class. To adopt the split-sample strategy we selected 10 fold cross validation. We followed the method suggested in [17] to implement the codes using Matlab. The codes were implemented using a Pentium III, 933 MHz with 256 MB RAM. The models were optimized from a set of different predictive variables. Figure 1 (a) explains the normal distribution of a binary class dataset. The ‘+’ and ‘x’ signs indicates the classes. Figure 1 (b) represents the linear kernel performance with decision boundary. Figure 2 (a) explains a natural scenario of an artificial dataset. Figure 2 (b) illustrates the same data points after smoothing with Sp . The results of the classification accuracy with dataset names and number of instances is presented in Table 1. Table 2 illustrates the execution time for different kernels. The bold face indicates the highest accuracy. The classification rate of *pima*, *promoter* and *wine* is higher using the improved linear kernel approach. The modified version of RBF also performed well for 3 datasets. As evident from Table 2, for some datasets improved linear kernels took more computational time than classical linear kernels. But for some datasets the computational time using modified and classical linear kernels were identical.

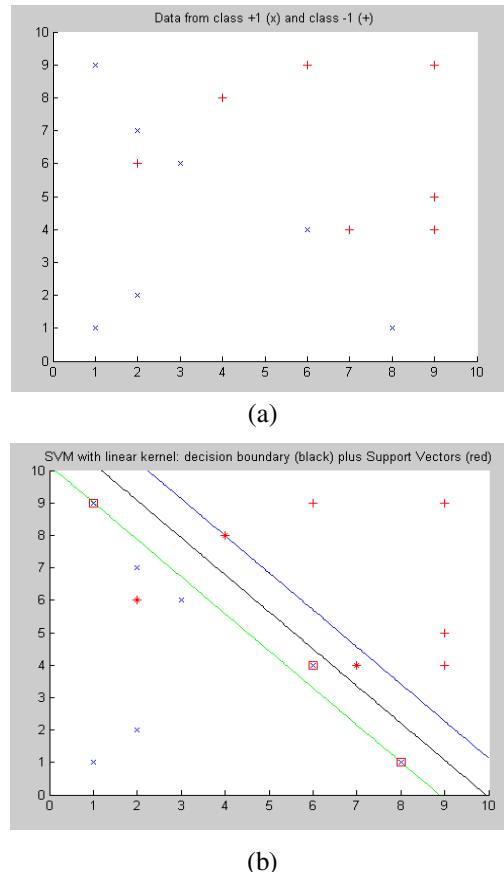


Fig. 1. (a) Dataset with normal distribution (b) decision boundary formed by SVM using linear kernel

Table 1. The percentage of accuracy for different kernels

Dataset (# of instances)	linear_c	linear_m	poly_c				rbf_c	rbf_m
			2	3	4	5		
bcw (699)	84.06	79.42	78.55	87.39	89.13	88.12	90.73	92.75
bos (455)	59.33	63.56	42.44	62.22	60.00	62.67	76.00	64.22
bupa (345)	58.24	61.47	60.88	62.06	54.41	52.94	65.29	59.12
German(1000)	67.98	67.68	50.51	50.51	50.51	50.51	68.59	69.60
h-d (303)	76.90	79.66	73.45	81.38	80.00	76.55	80.35	82.41
iris (150)	66.43	66.43	15.00	10.71	49.29	22.14	92.86	90.71
pima (768)	72.37	74.87	65.66	60.13	58.68	64.87	71.18	69.47
promoter (106)	68.00	77.00	75.00	68.00	74.00	76.00	74.00	74.00
sonar (208)	65.00	69.00	77.50	87.00	82.50	88.50	83.00	77.00
wine (178)	87.06	98.24	90.59	92.35	92.35	92.94	93.53	92.94

Table 2. Execution time for classification with different kernels

Dataset	linear_c	linear_m	poly_c				rbf_c	rbf_m
			2	3	4	5		
bcw	325.72	324.62	645.38	508.99	523.50	692.34	247.78	193.26
bos	626.07	1330.71	704.85	1344.71	1344.65	1342.94	364.92	260.40
bupa	51.74	94.93	53.72	60.96	66.49	63.23	36.75	29.41
german	833.41	813.34	4682.65	4940.42	5039.26	5142.54	657.31	525.96
h-d	39.29	83.76	43.07	68.22	53.36	35.84	27.45	20.37
iris	43.48	41.71	193.39	195.59	186.67	200.38	26.9	19.17
pima	378.161	672.43	402.056	449.846	409.166	562.419	302.911	235.493
promoter	4.246	4.617	4.076	4.036	4.046	4.306	3.946	3.886
sonar	14.061	18.577	12.729	12.548	13.35	18.256	11.557	10.285
wine	36.052	127.837	40.208	90.653	118.172	122.91	35.071	24.245

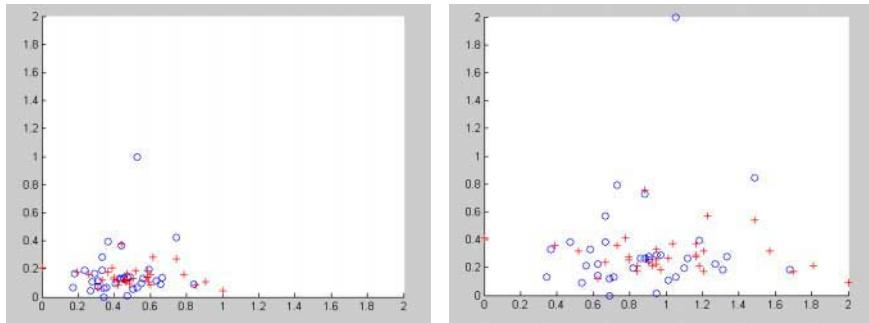


Fig. 2. (a) Normal target position of a dataset (b) target position of the dataset after smoothing

5. Conclusions

The goal of the present research was to verify the modification of the classical linear kernel approach and study its performance using some benchmark classification problems. It is interesting to note that the performance of the direct classical linear kernel approach could not deliver optimal performance for any of the datasets. Our preliminary studies and the empirical results showed that improved linear kernel could deliver better performance at the expense of some additional computational cost in a few cases. Computation of the smoothing parameter is an easy task but however we need to consider some heuristics to estimate optimal values.

References

1. Rumelhart, D.E., Hinton, G.E., Williams, R. J.: Learning internal representations by error propagation. In *parallel distributed processing: Explorations in the macrostructures of cognition*, volume 1, pages 318–362, Cambridge, MA, 1986.
2. Boser, B.E., Guyon, I. M., Vapnik, V.: A training algorithm for optimal margin classifiers, In *Fifth Annual Workshop on Computational Learning Theory*, Pittsburgh, 1992. ACM.
3. Cortes, C. and Vapnik, V.: Support vector networks. *Machine Learning*, 20: 273–297, 1995.
4. Vapnik, V.: *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.
5. Smola, A.J. and Schölkopf, B.: A tutorial on Support Vector Regression. *NeuroCOLT2 Technical Report Series, NC2-TR-1998-030*. October, 1998.
6. Weston, J. A., Gammerman, M., Stitson, Vapnik, V., Vovk, V., Watkins, C.: Density Estimation using Support Vector Machines. *Technical Report, Csd-TR-97-23*. February 1998.

7. Zien, A., Rätsch, G., Mika, S., Schölkopf, B., Lemmen, C., Smola, A., Lengauer, T., Müller, K.R.: Engineering support vector machine kernels that recognize translation initiation sites. *German Conference on Bioinformatics*, 1999.
8. Schölkopf, B., Smola, A., Learning with Kernels : Support Vector Machines, Regularization, Optimisation, and Beyond. *The MIT Press*, England, 2002.
9. Ali, S., Abraham, A.: An Empirical Comparison of Kernel Selection for Support Vector Machines, *2nd International Conference on Hybrid Intelligent Systems, Soft Computing Systems: Design, Management and Applications*, IOS Press, The Netherlands, pp. 321-330, 2002.
10. Cristianini, N., and Shawe-Taylor, J.: An Introduction to Support Vector Machines. *Cambridge University Press*, 2000.
11. Burges, C., Crisp, D.: Uniqueness of the SVM solution. In *Proceedings of the Twelfth Conference on Neural Information Processing Systems*. S. A. Solla, T. K. Leen, and K.-R. Müller (Eds.), MIT Press, 1999.
12. Chapelle, O., and Vapnik, V.: Model selection for support vector machines. In *Proceedings of the Twelfth Conference on Neural Information Processing Systems*. S.A. Solla, T. K. Leen, and K.-R. Müller (Eds.), MIT Press, 1999.
13. Vapnik, V.: *Statistical Learning Theory*. John Wiley & Sons, 1998.
14. Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):1-47, 1998.
15. Blake, C., Merz, C. J.: *UCI Repository of Machine Learning Databases*. Irvine, CA: University of California, 1998.
16. Loh. W.: Knowledge Discovery Central, Data Sets, <<http://www.KDCentral.com/>>, 2002.
17. Weston, J.W.: Multi-class support vector machines, presented at the Proc. ESANN99, M. Verleysen, Ed., Brussels, Belgium, 1999.

Automatic Car Parking: A Reinforcement Learning Approach

Darío Maravall¹, Miguel Ángel Patricio², and Javier de Lope¹

¹ Department of Artificial Intelligence

Faculty of Computer Science

Universidad Politécnica de Madrid

Campus de Montegancedo, 28660 Madrid, Spain

{dmaravall, jdlope}@dia.fi.upm.es

² Departamento de Informática

Universidad Carlos III de Madrid

mpatrici@inf.uc3m.es

Abstract. The automatic parking of a car-like robot is the problem considered in this paper to evaluate the role played by formal representations and models in neural-based controllers. First, a model-free control scheme is introduced. The respective control actions are sensory-based and consist of a dynamic, neural-based process in which the neurocontroller optimizes *ad hoc* performance functions. Afterwards, a model-based neurocontroller that builds without supervised a formal representation of its interaction with the environment is proposed. The resulting model is eventually utilized to generate the control actions. Simulated experimentation has shown that there is an improvement in robot behavior when a model is used, at the cost of higher complexity and computational load.

1 Introduction

In this paper we investigate the role played by representations and models of the environment in artificial neurocontrollers. We have chosen the automatic parking of car-like robots to illustrate this discussion. Previous work on this subject has been based either on hard computing techniques [1]-[3] –i.e., analytical methods from applied mathematics and control theory- or soft computing techniques, like fuzzy logic and artificial neural networks (ANN), [4]-[6]. A common feature of both hard computing and soft computing methods for automatic parking maneuvers is that they are based on *a priori* known models of the environment, as well as known car kinematics and dynamics. Sometimes, particularly for ANN-based methods, the aprioristic models of the environment are replaced by supervised trajectories that allow the vehicle controller to learn the proper maneuvers.

We depart from the usual model-based and supervised approaches to propose a model-free and unsupervised technique to solve the automatic parking problem. The paper is organized as follows. First, we succinctly present the problem at hand and introduce the general description of our method, which is based

on sensory-driven reinforcement control. Afterwards, we introduce the ANN implementation of the proposed method and present experimental results. Finally, we discuss in detail the role played by *a posteriori* formal representations or models of the interaction between the vehicle and the environment, emphasizing the differences between such learned models and the usual *a priori* models or equivalent supervised techniques –i.e., induced trajectories–.

2 Automatic Parking with Reinforcement Control

Commercial cars can be modelled as dynamic systems with, basically, two control inputs: speed, v , and steering angle, ϕ . By admitting the usual hypothesis of the car being on a plane, the state variables are (x, y, θ) , as illustrated in Figure 1, in which we have considered the two typical kinematics of commercial cars; i.e., rear traction and front traction. The respective dynamical equations are

$$(a) \dot{x} = |v| \cos \theta; \quad \dot{y} = |v| \sin \theta; \quad \dot{\theta} = \frac{|v|}{L} \tan \phi \quad (1)$$

$$(b) \dot{x} = |v| \cos(\theta + \phi); \quad \dot{y} = |v| \sin(\theta + \phi); \quad \dot{\theta} = \frac{|v|}{L} \sin \phi$$

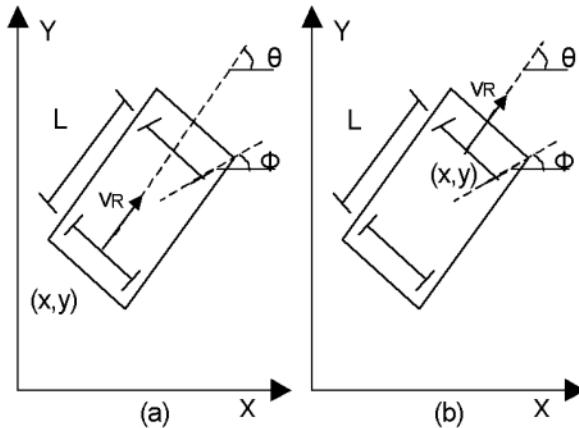


Fig. 1. Two typical kinematics of commercial cars: (a) rear traction and front steering and (b) front traction and front steering.

Let us suppose that our car-like robot is controlled only by the steering angle –i.e., it moves at a constant speed-. The automatic car parking task using conventional control theory [7] is based on the typical feedback loop. In this case, $\theta_d(t)$ is the desired orientation of the car, which is compared with its current

direction and, as a result of the error, a control law generates the steering angle that eventually drives $\theta(t)$ towards $\theta_d(t)$. Another two similar control loops for the remaining state variables, x and y , are needed for complete lateral control of the vehicle. As mentioned before, the main difficulty in applying control theory to automatic car parking is the computation of the desired control trajectories $\theta_d(t)$, $x_d(t)$ and $y_d(t)$, which are always pre-computed and injected into the car-like robot in real-time. When using soft computing techniques, either fuzzy logic or ANN, the rigid control loops take another shape.

In fuzzy logic-based automatic parking of a car, [4], the control laws are converted into linguistic rules and the environment's responses are also converted into linguistic variables. As is well known, the design process and, particularly, the tuning of membership functions is very cumbersome and tricky. Another problem is the possible combinatorial explosion of the fuzzy control rules, [6].

When applying ANN techniques, [4] and [5], the control laws of the car-like robot are learnt by sensory-based experience and, particularly, by following a teacher that provides the correct actions for each specific situation. Although the ANN approach simplifies the design process substantially, the fact is that it calls for an exhaustive battery of training examples and, as is very well known, the system can get into serious trouble in the operating stage, if it comes up against situations for which it has not been trained.

We have introduced elsewhere, [8] and [9], a method based on the perception-planning-action cycle that is able to solve the navigation of robotic mechanisms without explicit knowledge of their kinematics and without using formal representations or models of their interaction with unknown environments. This method is bio-inspired at the functional level and its particularization to the automatic parking problem is shown in Figure 2.

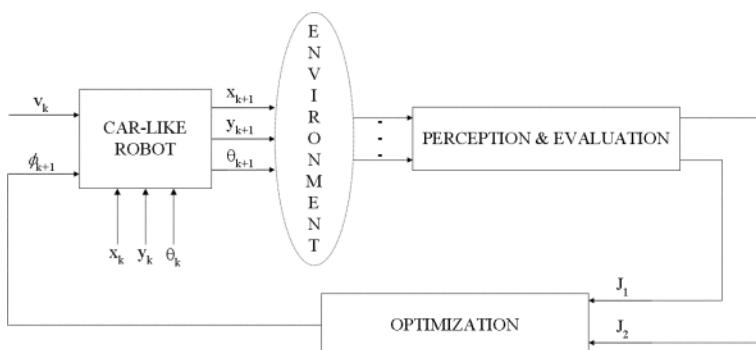


Fig. 2. Block-diagram of the proposed system for automatic parking maneuvers.

In this particular case, there are two performance functions J_1 and J_2 , which represent (1) the objective of the car-like robot being parked in the final position (x_d, y_d) and (2) pointing in a desired orientation $\theta_d(t)$. Therefore,

$$\begin{aligned} J_1 &= \frac{1}{2} [(x - x_d)^2 + (y - y_d)^2] \\ J_2 &= \frac{1}{2} (\theta - \theta_d)^2 \end{aligned} \quad (2)$$

As mentioned above, let us suppose that the robot is driven by a negative constant speed -ie., it is maneuvering to reverse- so that there is a single control variable ϕ . Obviously, the control law is aimed at minimizing both performance indices:

$$\dot{\phi}(t) = -\mu_1 \frac{\partial J_1}{\partial \phi} - \mu_2 \frac{\partial J_2}{\partial \phi} \quad (3)$$

where μ_1 and μ_2 weight the relative importance of each objective. The discrete-time version of this control law is

$$\phi_{k+1} = \phi_k - \mu_1 \left. \frac{\partial J_1}{\partial \phi} \right|_{\phi(k)} - \mu_2 \left. \frac{\partial J_2}{\partial \phi} \right|_{\phi(k)} \quad (4)$$

In other words, the car steering angle is computed as an increment/decrement step of the current steering angle, whose sign and module depend on the gradients of both performance functions.

3 Model-Free Behavior

By computing the empirically estimated gradient of the sensory-based performance indices $J_1(\phi)$ and $J_2(\phi)$, our robot is acting as a model-free agent, immerse in a dynamic environment. The agent decisions -i.e., its control actions- exclusively rely on the direct and instantaneous information gathered from the environment by its sensors :

$$\left. \frac{\partial \hat{J}_1}{\partial \phi} \right|_{\phi(k)} = \frac{J_1(k) - J_1(k-1)}{\phi(k) - \phi(k-1)}, \quad \left. \frac{\partial \hat{J}_2}{\partial \phi} \right|_{\phi(k)} = \frac{J_2(k) - J_2(k-1)}{\phi(k) - \phi(k-1)} \quad (5)$$

This sensory-based behavior can be interpreted as purely reactive, in the sense that perception and action are directly coupled, without any type of intermediate processing. However, this simple and model-free behavior is extremely powerful, as we find by looking at the experimental results shown in Figure 3, in which the car-like robot has successfully performed several parking maneuvers.

The secret of the excellent performance achieved by the robot lies in two elements: (1) the objective functions evaluating its performance, which are in

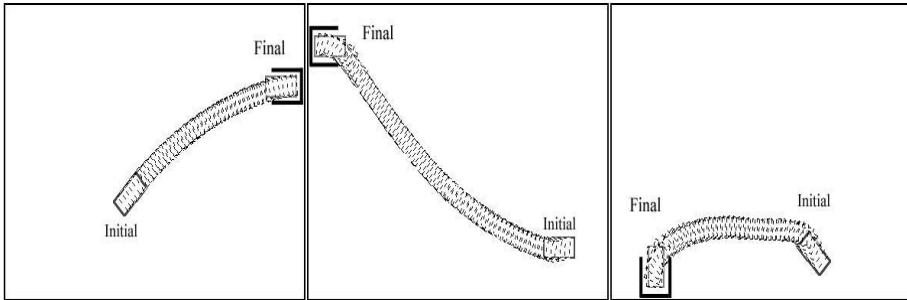


Fig. 3. Several parking maneuvers executed by the robot using a simple, model-free behavior.

turn based on sensory information, and (2) the action strategy based on the optimization of the performance functions. Furnished with these two elements, the car-like robot is able to interact with its environment without using a formal representation or model of its interaction with the environment, and without making explicit use of its own kinematics –i.e., the transformation between its internal coordinates and the environment coordinates–.

4 Model-Based Behavior

Let us now investigate the use of formal representations of the vehicle-environment ensemble. We emphasize that we are talking about modelling the joint dynamics of the agent and its particular environment, rather than modelling each part individually. Then, we are going to evaluate robot performance in parking when its behavior is based on a formal representation or model of its interaction with the environment. Figure 4 represents the modelling process undertaken by the car-like robot by means of an ANN. Note that the model built by the ANN represents the relationship between the vehicle actions –i.e., the steering angles– and the responses of the environment, as quantified by performance indices. This ANN has been implemented via a multilayer perceptron with backpropagation of the model error.

It should be noted that we have explicitly introduced a second ANN to produce the control actions. This ANN, as shown in Figure 4, is based on the same idea of utilizing the empirically estimated gradients of the indices and it has been implemented by means of a simple perceptron. Then, we can conclude that the agent's formal representation of its interaction with the environment is not used for control purposes, as the steering angle is exclusively based on the gradient of the performance functions obtained from the sensor readings. However, once an accurate model of the vehicle-environment ensemble has been built by the respective ANN, then the agent actions can also be generated by the model. This kind of model-based control is known as indirect control, as

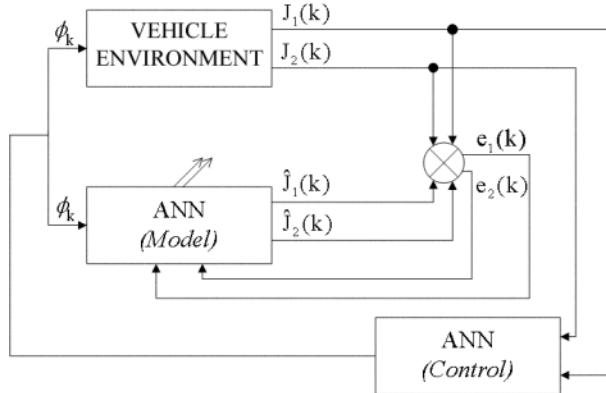


Fig. 4. Conceptual block-diagram of the modelling process using an ANN. Note that the control actions are generated by another ANN minimizing the performance functions.

it is obtained from a model of the plant under control [10]. Figure 5 shows the block-diagram of an hybrid control architecture, in which we have simultaneously considered both type of control actions: (1) model-free or direct control and (2) model-based or indirect control. Note also the ANN in charge of the tricky task of properly combining both control actions.

In both direct and indirect control, the respective control actions ϕ_k^e and ϕ_k^m , respectively, depend on the sensory-based gradients, either as measured by the sensors or as estimated by an ANN. Due to this sensory dependency, the modules of the respective control outputs tend to be disproportionated to the internal dynamics of the vehicle: sometimes the sensory-based gradients are too strong or, inversely, sometimes they are too small, as far as the vehicle's natural dynamics is concerned. To solve this problem of scale, we have devised a look-up table mapping the module of the sensory-based gradients to the natural dynamics of the car-like robot, for both the model-free and the model-based behaviors.

Figure 6 presents the results obtained with the model-based or indirect control actions, for the same situations considered for the direct actions illustrated in Figure 3. We find that the use of a formal representation of the robot-environment couple has appreciably improved the smoothness of the vehicle trajectories. As a conclusion, at least for automatic parking maneuvers, the use of models improves the robot performance. We must add, however, that this improvement is accomplished at the cost of a more complex design process. Tuning the ANN in charge of the modelling process and tuning the ANN combining the direct and the indirect control actions is a tricky and cumbersome process. As a general conclusion, the use of formal representations involves, as in other fields, a subtle trade-off between efficiency and complexity.

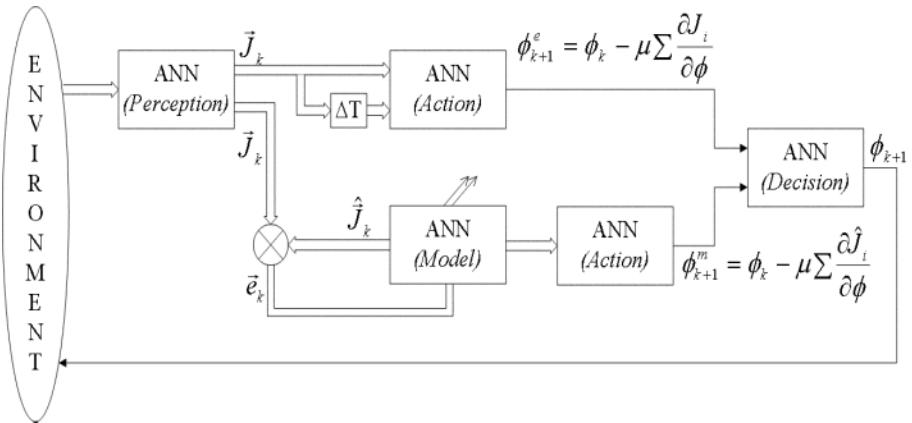


Fig. 5. Block-diagram of the combined direct and indirect control actions. Note the vectors representing more than just one signal or function.

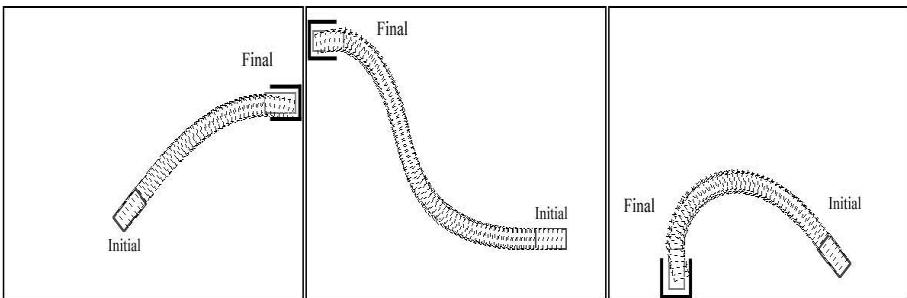


Fig. 6. Experimental results obtained with the model-based or indirect control.

5 Conclusions

Automatic car parking maneuvers is the problem considered in this paper to evaluate the role played by and the pros and cons of formal representations of the interaction of a physical agent with its environment. First, we discussed our previous work on reinforcement control without the use of models of the environment and, then, we introduced a model-free and ANN-based control scheme for performing parking maneuvers. Afterwards, we introduced a model-based or indirect control scheme, also implemented using ANN, for the same task. Simulated experimentation of both direct and indirect reinforcement control has shown that model-based control actions are more efficient –in particular, they perform the parking maneuvers smoother- than model-free or direct actions, at the cost, however, of increased design complexity and computational burden.

Acknowledgements

This work has been partially supported by the Spanish Ministry of Science and Technology, project DPI2002-04064-CO5-05.

References

1. Laumond, J.-P., Jacobs, P.E., Taix, M., Murray, R.M. (1996) A motion planner for non-holonomic mobile robots. *IEEE Trans. On Robotics and Automation*, 10(5), 577–593.
2. Paromtchik, I.E., Laugier, C. (1996) Motion generation and control for parking an autonomous vehicle. *Proc. IEEE Int. Conference on Robotics and Automation*, Minneapolis, MN, 3117-3122.
3. Laugier, C., Fraichard, Th., Garnier, Ph., Paromtchik, I.E., Scheuer, A. (1999) Sensor-based control architecture for a car-like vehicle. *Autonomous Robots* 6, 165-185.
4. Kong, S.G., Kosko, B. (1992) "Comparison of fuzzy and neural track backer-upper control systems". In B. Kosko (ed). *Neural Networks and Fuzzy Systems*. Prentice-Hall. Englewood Cliffs, NJ, 339-361.
5. Gu, D., Hu, H. (2002) Neural predictive control for a car-like mobile robot. *Robotics and Autonomous Systems* 39, 73-86.
6. Hitchings, M., Vlacic, L., Kecman, V. (2001) "Fuzzy control". In L. Vlacic, M. Parent, F. Harashima (eds). *Intelligent Vehicle Technologies*. Butterworth&Heinemann, Oxford, 289-331.
7. Canudas de Wit, C. (1998) "Trends in mobile robots and vehicle control". In B. Siciliano, K.P. Valavanis (eds). *Control Problems in Robotics and Automation. LNCIS 230*, Springer, Berlin, 151-176.
8. Maravall, D., de Lope, J. (2002) "A reinforcement learning method for dynamic obstacle avoidance in robotic mechanisms". In D. Ruan, P. D'hondt, E.E. Kerre (eds). *Computational Intelligent Systems*. World Scientific, Singapore, 485-494.
9. Maravall, D., de Lope, J. (2003) "A bio-inspired robotic mechanism for autonomous locomotion in unconventional environments". In C. Zhou, D. Maravall, D. Ruan (eds). *Autonomous Robotic Systems: Soft Computing and Hard Computing Methodologies and Applications*. Physica-Verlag, Springer, Heidelberg, 263-292.
10. Zhou, C., Meng, Q. (2003) Dynamic balance of a biped robot using fuzzy reinforcement learning agents. *Fuzzy Sets and Systems*, 134(1), 169-187.

Discriminative Training of the Scanning N-Tuple Classifier

Simon M. Lucas

Dept. of Computer Science
University of Essex, Colchester CO4 3SQ, UK
sml@essex.ac.uk

Abstract. The Scanning N-Tuple classifier (SNT) was introduced by Lucas and Amiri [1, 2] as an efficient and accurate classifier for chain-coded hand-written digits. The SNT operates as speeds of tens of thousands of sequences per second, during both the training and the recognition phases.

The main contribution of this paper is to present a new discriminative training rule for the SNT. Two versions of the rule are provided, based on minimizing the mean-squared error and the cross-entropy, respectively. The discriminative training rule offers improved accuracy at the cost of slower training time, since the training is now iterative instead of single pass. The cross-entropy trained SNT offers the best results, with an error rate of 2.5% on sequences derived from the MNIST test set.

1 Introduction

The Scanning N-Tuple classifier (SNT) was introduced by Lucas and Amiri [1, 2] as an efficient and accurate classifier for chain-coded hand-written digits. The SNT operates as speeds of tens of thousands of sequences per second, both during the training and the recognition phases. The SNT was developed as a way of applying n-tuple neural networks to sequences.

The main contribution of this paper is to present a new discriminative training rule for the SNT. We give two versions of this rule, one for minimizing the Cross Entropy error, and one for minimizing the Mean Square Error. In addition to this, a conditional probability version of the SNT is given, as an alternative to the usual joint-probability interpretation. Results for each of these SNTs are given for chain-coded sequences derived from the MNIST dataset of hand-written digit images.

The discriminative training rules offer improved accuracy at the cost of slower training time. The training is now iterative instead of single pass. The cross-entropy trained SNT offers the best results, with an error rate of 2.5% on the MNIST test set.

Recent approaches to optimizing the SNT have included unsupervised clustering of the classes [3], and also using bit-plane decomposition SNTs in combination with the standard ones [4]. It has also been successfully applied to on-line (dynamic) hand-writing recognition [5].

The rest of this paper is structured as follows. Section 2 reviews the basic structure of the SNT, and offers a new simple interpretation based on sequence re-ordering. Section 3 explains the new discriminative training rules. Section 4 describes some useful implementation techniques. Section 5 presents results on the MNIST dataset, and Section 6 concludes.

2 The Scanning N-Tuple Classifier (SNT)

The SNT is based on an n-gram model of a sequence, but with two differences from the standard n-gram model. The first is that it introduces gaps into the elements of the sequence that are modelled. The second is that by using a number of SNTs (each with a different gap), we can build an ensemble classifier, which usually improves performance compared to using just one SNT.

A particular SNT is hence defined by two main parameters, the number of sample points n and the gap between each sample point g . We denote this as SNT(n,g). For example, a conventional bigram model is similar to an SNT(2,1). However, in the original treatment the SNT estimates the probability of a sequence as the product of the joint probabilities of its constituent subsequences. The value of σ is largely dictated by the application, but applying appropriate transforms to the raw input sequences can improve the performance, and may also alter the value of σ . For example, the raw 4-direction chain-codes derived from the MNIST images are mapped into 8-direction chain codes, hence changing the σ from 4 to 8. We then append start and end symbols to the sequences, which further increases σ to 10. These boundary symbols add some extra context information and lead to an improvement in performance on some datasets.

Equations 1 – 3 illustrate the SNT address generation pipeline. This is the serial depiction for software implementation, which can also be done in a highly parallel manner. Incoming sequences are padded with start and end markers using the transform \rightarrow^p . The result is then re-ordered according to the gap parameter g using transform \rightarrow^g . The sequence is then broken up into a bag of integers using transform \rightarrow^f . This is done by interpreting each subsequence a radix σ integer composed of n digits, as shown in equation 14.

$$01234567 \rightarrow^p 880123456799 \quad (1)$$

$$\rightarrow^g 802469813579 \quad (2)$$

$$\rightarrow^f \{80, 2, 24, 46, 69, 98, 81, 13, 35, 57, 79\} \quad (3)$$

A conventional bigram model estimates the probability of a sequence s of length n :

$$P(s) = P(s_1) \prod_{i=1}^{n-1} P(s_{i+1}|s_i) \quad (4)$$

That is, the probability of the first symbol multiplied by the conditional probabilities of each subsequent symbol given the previous one.

On the other hand, prior treatments of the SNT model the estimate $P(s)$ thus:

$$P(s) = \prod_{i=1}^{n-1} P(s_{i+1}, s_i) \quad (5)$$

Let the value stored in the i th address of the k th class model look-up table (LUT) be m_{ki} . In the maximum likelihood estimate this is a count of the number of times address i occurred during training on patterns from class k . The standard SNT computes the likelihood for the sequence as a log-likelihood. The likelihood of the i th address in the LUT for class k is estimated as the number of times this address occurred, divided by the total number of address occurrences, and is denoted l_i . $N = \sigma^n$ is the size of the LUT for each class.

$$l_i = \log \frac{m_i}{\sum_{j=1}^N m_j} \quad (6)$$

Then, suppose we have a function $f(s)$ that generates all the n-tuple addresses for sequence s . Then the log-likelihood of s is given by a_k as follows:

$$a_k = \sum_{i \in f(s)} l_i \quad (7)$$

The a_k term is log-likelihood estimate of sequence s given the training data observations. It is usual to add a term ϵ to this to the m_i terms to prevent addresses that happened to be unseen in the training data having zero probability.

To interpret these outputs as posterior probabilities, we put the individual a_k through a *softmax* function [6, 7]:

$$y_k = \frac{\exp(a_k)}{\sum_{k'=1}^C \exp(a'_{k'})} \quad (8)$$

3 Discriminative Training

Given the interpretation of the output as y_k as $P(c_k|s)$ (probability of class k given input sequence s), we can apply two standard error measures to be minimized [7], the mean square error (MSE) and the cross entropy (CE). Both of these are zero when the outputs agree exactly with the targets. The MSE for a the k th output of the i th pattern is $e = (t_{ik} - o_{ik})^2$ hence over all training patterns:

$$E = \sum_{i,k} (t_{ik} - o_{ik})^2 \quad (9)$$

The cross entropy view on the other hand is to maximize the probability of correctly classifying the training set — thereby minimizing the cross-entropy:

$$P(y, t) = \prod_{i,k} y_{ik}^{t_{ik}} (1 - y_{ik})^{(1-t_{ik})} \quad (10)$$

We require the output the error terms to back-propagate to the LUT contents. This is straightforward. The Cross Entropy error term is:

$$\frac{\partial e}{\partial a_k} = y_k - t_k \quad (11)$$

While the MSE version is:

$$\frac{\partial e}{\partial a_k} = (y_k - t_k)y_k(1 - y_k) \quad (12)$$

See Bishop [7] pages 233 – 240 for the derivations. For training we use a stochastic update rule - each iteration we iterate over the entire training set in random order, choosing a different random order each time.

To implement the back-propagation for the SNT we simply loop over all the symbols in each sequence. The error vector specifies the adjustment to be made to the table entries l_i for each class.

Denote the error signal for a particular model and a particular pattern as δ . This adjustment δ is the same for all LUT entries for each access. Note that for the CE rule, the error signals have a very simple equation. If the pattern class equals the model class, then $\delta = 1 - y$, otherwise $\delta = y$.

To update the SNT LUT, we simply scan over the addresses j generated for that pattern, applying the following update rule:

$$l_j = l_j + \delta \quad (\forall j \in f(s)) \quad (13)$$

This is one of the simplest learning rules possible. Note that the underlying network architecture could also be viewed as a higher order neural network [8]. If an address j occurs n times then the value stored at since each term in the "bag of integers" mapping of the input sequence can occur any number of times, raising the value of that entry

Interestingly, although the l_i are interpreted as log-likelihoods, we never actually need to calculate the logs during training, nor do we need any prior estimates. All the entries in each table start at zero, and are just updated in accordance with equation 13.

4 Implementation Techniques

In this section we provide some tips for implementing efficient versions of the SNT. Where timings are given, these are based on a Java (1.4.1) implementation running on a 1ghz AMD Athlon laptop.

4.1 Loop Order

In the original Lucas and Amiri [1] paper the outer loop was specified over the set of all classes. Actually, this is inefficient, since it involves repeating the (identical) address calculations for each class. With the original training algorithm which

only updated the class model for the particular class of each training pattern, this did not affect the training speed, but it does affect the recognition speed. With discriminative training, both the training and recognition speed can be increased by moving the class-loop to the inner-most loop.

With this improvement we cut the recognition time for the entire MNist training plus test sets of 70,000 patterns ($60,000 + 10,000$) from 7 seconds to 3.3 seconds - i.e. a recognition rate of about 30,000 sequences per second.

4.2 Re-Ordering the Sequence

As explained above, the effect of sampling the sequence with gaps between each sample point is equivalent to sampling a appropriately permuted sequence with a standard n-gram window.

We can speed up iterative training algorithms by re-ordering all the sequences according to the value of g in this way prior to training. Then, the actual training reduces to estimating a standard n-gram model. This reduces the number of nested loops for each sequence from three to two, although the number of LUT address calculations remains the same. Note that the idea of transforming sequences (strings) to improve the performance of a simple language model has been explored in some depth by Garcia *et al* [9] and Vidal and Llorens [10], with the concept the *Morphic Generator Grammatical Inference*.

Indeed, the generation of addresses may also be viewed a transform of one input sequence into another input sequence. For the n gram address we compute the index j at position o in the sequence:

4.3 Address Calculation

Earlier we stated that each sub-sequence of re-ordered original could be viewed as an integer of radix σ .

$$j_o = \sum_{i=0}^n \sigma^i s_{i+o} \quad (14)$$

Of course, it is not necessary to calculate this value from scratch for each position in the sequence. Instead, each new address is calculated as a shift, mask and add operation. Where σ is a power of two, this can be implemented using bit-wise operators in high-level languages such as C and Java. When σ is not a power of two, then each new address is calculated by multiplication, addition and an integer division.

$$j_{o+1} = (j_o * \sigma + s_{i+o}) / \sigma \quad (15)$$

5 Results

We tested the system on a set of chain-code sequences¹ derived from the MNIST dataset [11] of hand-written digits. In the MNIST dataset each image is centred

¹ The dataset can be downloaded from <http://algoval.essex.ac.uk/tc5/datasets>

(according to the centre of mass of the pixel intensities) on a 28x28 grid. Although the original NIST images were binary, the MNIST versions are smoothed with a gaussian filter to give grey-level images with pixel intensities in the range 0...255. Our chain-code procedure only works with binary images, so we binarized the MNist images using a threshold of 128. It may be possible to improve the accuracy of the SNT by generating chain-codes from multiple different thresholds, thus introducing some systematic variation to help the SNT generalize, but we've not yet tried this. We used the procedure described in [2] to generate the chain codes. When mapped to eight direction codes, the resulting sequences had an average length of 59.3 symbols.

We investigated the use of various SNT versions with and training rules. The joint and conditional SNTs were training using Maximum Likelihood (ML) method, with ϵ set to 0.01; Note that the conventional SNT is J-SNT. While this performs very poorly as a bigram estimator J-SNT(2,1), it is more competitive as J-SNT(4,4). The results are shown in Table 1, with standard deviations in parentheses where applicable (based on 10 experiments). The ML estimates are exactly the same each time, but the discriminative models are estimated with a stochastic update rule which can give different results for each run. Note that with the bigram model the incorrect assumption involved in the product joint-probabilities calculation has a detrimental effect on J-SNT, but these becomes un-important for better performing system using higher values of n and g . In all cases the discriminative models (CE and MSE) outperform the ML estimated models (J and C).

While the test set accuracies for CE and MSE are very similar for the accurate models (e.g. SNT(4,4)), these have very different behaviors on the training set, with MSE only achieving about 98.5% training set accuracy versus 99.7% for CE. Hence, the CE rule offers a more effective optimization procedure for learning this data, but due to over-fitting this does not lead to a correspondingly large increase in test-set accuracy.

Estimation Method	$n=2;g=1$	$n=4;g=4$
J-SNT	67.4	96.1
C-SNT	70.2	96.0
CE-SNT	72.4 (3.4)	97.5 (0.03)
MSE-SNT	78.4 (0.4)	97.2 (0.02)

Table 1. MNIST test set accuracy for various SNT parameter estimation rules.

Table 2 shows how the performance varies for different values of n and g . Some of the poorer accuracies in this table have a high variance, due to the failure of the stochastic gradient descent procedure to converge, but the better accuracies have a very low variance. The best error rate shown here is 2.5%

n	g=1	g=2	g=3	g=4	g=5	g=6
2	74.1	84.5	90.7	92.2	91.1	90.7
3	92.1	95.6	96.5	96.7	96.4	96.1
4	94.7	96.6	97.4	97.5	97.0	96.6
5	95.3	96.6	97.1	97.1	96.8	96.2

Table 2. MNIST test set accuracy for CE-SNT given various values of n and g .

for CE-SNT(4,4). Table 3 shows this compared to a selection of other results reported on the MNIST web page².

Method	error rate
linear classifier (1-layer NN)	12.0
2-layer NN, 1000 hidden units	4.5
CE-SNT(4,4)	2.5
SVM deg 4 polynomial	1.1
LeNet 5 , distortions (conv. NN)	0.8
K-NN, shape context matching	0.67

Table 3. MNIST test set accuracy for various methods (only CE-SNT(4,4) results from this paper).

While the SNT error rate is considerably worse than the best quoted results on MNist, it should be noted that there is room for improvement. Some of the best performing methods on this dataset take a simple technique and introduce systematic variations in the input image. We can do this with the SNT also during either the training or the recognition stages. The fact that the discriminatively trained SNT(4,4) gets nearly 100% accuracy on the training set demonstrates that it has capacity for learning far more intra-class variations than were presented to it during training.

6 Discussion and Conclusions

The SNT has proven itself as an efficient sequence classifier which offers high accuracy on certain problems such as the recognition of chain-coded hand-written digits (both on and off-line). Previous treatments of the SNT, however, relied on a maximum likelihood classification rule. This paper presented two discriminative training rules based on minimizing the mean square error and the cross entropy respectively.

When using a bigram model the MSE rule significantly outperforms the other methods. For higher-order models that offer much better accuracy, the

² See <http://yann.lecun.com/exdb/mnist/>

cross-entropy rule performs best. Both discriminative methods significantly outperform the maximum likelihood (ML) estimated models.

Finally, it is worth emphasizing the speed of the method: Training the system for ten iterations over the 60,000 sequences derived from the MNist training set takes just 20 seconds when done using a discriminative rule. Training using a maximum likelihood rule involves just one pass taking less than 0.5 seconds. Classifying the 10,000 sequences derived from the MNist test set takes only 360ms seconds i.e. a rate of over 28,000 sequence classifications per second.

Acknowledgements

I thank the members of the Natural and Evolutionary Computation group at the University of Essex, UK for many stimulating discussions related to this work.

References

1. S.M. Lucas and A. Amiri, “Recognition of chain-coded handwritten characters with the scanning n-tuple method”, *Electronics Letters*, vol. 31, no. 24, pp. 2088 – 2089, 1995.
2. S.M. Lucas and A. Amiri, “Statistical syntactic methods for high performance ocr”, *IEE Proceedings on Vision, Image and Signal Processing*, vol. 143, pp. 23 – 30, (1996).
3. G. Tambouratzis, “Improving the clustering performance of the scanning n-tuple method by using self-supervised algorithms to introduce subclasses”, *IEEE Transactions on Pattern analysis and Machine Intelligence*, vol. 24, pp. 722 – 733, (2002).
4. S. Hoque, K. Sirlantzis, and M. C. Fairhurst, “Bit plane decomposition and the scanning n-tuple classifier”, *Proceedings of International Workshop on Frontiers in Handwriting Recognition (IWFHR-8)*, pp. 207 – 212, 2002.
5. E. H. Ratzlaff, “A scanning n-tuple classifier for online recognition of handwritten digits”, in *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, pp. 18 – 22. IEEE, Seattle, WA, (2001).
6. J.S. Bridle, “Alpha-nets: a recurrent “neural” network architecture with a hidden markov model interpretation”, *Speech Communication*, vol. 9, pp. 83 – 92, (1990).
7. C.M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, (1995).
8. C. L. Giles and T. Maxwell, “Learning, invariance and generalisation in high-order neural networks”, *Applied Optics*, vol. 26, pp. 4972 – 4978, (1987).
9. P. Garcia, E. Vidal, and F. Casacuberta, “Local languages, the successor method, and a step towards a general methodology for the inference of regular grammars”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, pp. 841 – 845, (1987).
10. E. Vidal and D. Llorens, “Using knowledge to improve N-gram language modelling through the MGCI methodology.”, in *Proceedings of the Third International Colloquium on Grammatical Inference (ICGI-96): Learning Syntax from Sentences*, Laurent Miclet and Colin de la Higuera, Eds., Berlin, 25–27 1996, vol. 1147, pp. 179–190, Springer.
11. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition”, *Proceedings of the IEEE*, vol. 86, 1998.

A Wrapper Approach with Support Vector Machines for Text Categorization *

Montañés E., Quevedo J.R., Díaz I.

Artificial Intelligence Center, University of Oviedo. Campus de Viesques, Gijón
(Asturias) Spain

Abstract. Text Categorization (TC)-the assignment of predefined categories to documents of a corpus-plays an important role in a wide variety of information organization and management tasks of Information Retrieval (IR). It involves the management of a lot of information, but some of them could be noisy or irrelevant and hence, a previous feature reduction could improve the performance of the classification. In this paper we proposed a wrapper approach. This kind of approach is time-consuming and sometimes could be infeasible. But our wrapper explores a reduced number of feature subsets and also it uses Support Vector Machines (SVM) as the evaluation system; and this two properties make the wrapper fast enough to deal with large number of features present in text domains. Taking the Reuters-21578 corpus, we also compare this wrapper with the common approach for feature reduction widely applied in TC, which consists of filtering according to scoring measures.

1 Introduction

Text Categorization (TC), which consists of assigning the documents of a corpus into a set of prefixed categories, plays an important role in a wide variety of information organization and management tasks. Since the information available is continuously growing, a lot of research is going on with the goal of automating this time-consuming task.

It is well known in TC environment [14] that, among the great amount of features present in text domains, most of them could be noisy or irrelevant and hence a previous feature reduction could allow the classifiers to perform better.

According to John [8], there are two main approaches to feature reduction: filtering and wrapping. In the former, a feature subset is selected independently of the learning method that will use the selected features. However, in the latter, a feature subset is selected using the evaluation function based on the same learning algorithm that will take the selected features. The widely adopted approach in TC is the filtering approach based on scoring the features, sorting them according to this score and selecting a predefined number of the best ones. The reason of choosing the filtering approach for TC is that the wrapper approach

* The research reported in this paper has been supported in part under MCyT and Feder grant TIC2001-3579 and FICYT grant BP01-114.

(although it has been shown to performs better [8]) can result in a rather time consuming process or sometimes it can be infeasible, since it must deal with a large number of features present in text domains.

Since Support Vector Machines (SVM) have been shown to perform well and fast in TC and it is a good handler of many features [7], we propose a new wrapper approach to perform feature reduction based on this classifier and especially designed for dealing with large set of features. We also compare them with the typical feature reduction approach adopted in TC.

The rest of the paper is organized as follows. In Section 2 it is discussed some previous work. In Section 3 the process of TC is described including the description of the proposed wrapper. The experiments are exposed in Section 4. Finally, in Section 5 some conclusions and ideas for further research are presented.

2 Previous Work

Feature selection is one of the filtering approach for feature reduction widely adopted in TC. It is performed by keeping the words with highest score according to a predetermined measure of the importance of the word for TC. Its main disadvantage is that it is necessary to predefined a threshold to indicate how many features are removed or selected. The tf measure is one of the most common measures used in Information Retrieval (IR) [11]. It simply counts the number of occurrences of each word. Another measure of this kind is $tf \times idf$ [14], which considers information about the dispersion of the word over the documents. Other measures are taken from the Information Theory (IT) like *information gain*, *mutual information* or $S - \chi^2$, being the first one that which has been shown to perform best [17]. In this paper we choose tf , $tf \times idf$ and *information gain* to compare them with our wrapper.

Although wrapper approaches have been shown to perform better [8], they are time-consuming and hence, the quality of the solution is traded for the time needed to find a solution, justified by the large number of features usually present in text domains. This is the reason why most of the research about wrapper approaches is not designed for situations with large number of features. Aha and Bankert [1] design a wrapper approach for feature reduction in instance based learning; Caruana and Freitag [3] and Cherkauer and Shavlik [4] also propose a wrapper approach but in decision tree induction. In this paper, taking advantage of the fast learning of SVM, we propose a new wrapper approach based on SVM for feature selection able to deal with large number of features. Another advantage of the wrapper approach is that the number of features selected is automatically obtained.

Independently of either filtering or wrapping, other previous feature reduction [14] is typically carried out consisting of eliminating unmeaningfull words like articles, conjunctions which are called *stop words*. Aditionally, the reduction of words to their common roots or stems (*stemming*) is usually performed. In this paper local reduction (consisting of taking only words occurring in docu-

ments of each category rather than in all categories) is performed since it has been proved that it performs better [11].

3 Process of Text Categorization

In this section it is described how we perform the process of TC.

3.1 Document Representation

The most widely used document representation is *vector of words* or *bag of words* (see [14]), which consists of identifying each document with a numerical vector whose components evaluate the importance of the different words in the document. We adopt the widely used (see [7] and [17]) *tf* (the absolute frequency of the word in the document) as the components of the vector.

3.2 Typical Filtering Feature Reduction

In this section the measures of filtering approach for feature reduction followed in this paper are described.

First, following Porter [12] *stop words* and *stemming* is performed and following [11] local reduction is considered before filtering with the measures *tf*, $tf \times idf$ and *IG*.

The words were scored according with *tf*, which simply counts the number of occurrences (*tf*). Considering information about the dispersion of the word over the documents, the words could be evaluated by $tf \times idf$ defined as

$$tf \times idf = tf \log\left(\frac{N}{df}\right)$$

where *tf* is the number of occurrences of the word, *N* is the total number of documents and *df* is the number of documents that contains the word. Recently, measures of the IT have been widely used since it consider the distribution of a word over the categories. Among these measures, there is *information gain* (*IG*), which has been showed to perform best [17]. It can be defined as (see, for instance [17])

$$IG(w, c) = P(w)P(c/w) \log\left(\frac{P(c/w)}{P(c)}\right) + P(\bar{w})P(c/\bar{w}) \log\left(\frac{P(c/\bar{w})}{P(c)}\right)$$

where $P(w)$ is the probability that the word w appears in a document, $P(c/w)$ is the probability that a document belongs to the category c knowing that the word w appears in it, $P(\bar{w})$ is the probability that the word w does not appear in a document and $P(c/\bar{w})$ is the probability that a document belongs to the category c if we know that the word w does not occur in it. Usually, these probabilities are estimated by means of the corresponding relative frequencies.

3.3 Our Wrapping Feature Reduction

In this section it is described the wrapper we propose for feature reduction.

First, as for typical filtering feature reduction, following Porter [12] *stop words* and *stemming* is performed and following [11] local reduction is considered before applying the wrapper.

In designing a wrapper for feature subset selection it is necessary to determine two issues: the searching method over the feature subset space and the evaluation function of a feature subset. There are several methods for searching in the feature subset space, for instance, Kohavi proposes in [9] a method based on the First Search Algorithm, but it is not adequate for the many features present in text domains because it explores too many subsets. In [13] it is proposed an aggressive algorithm for feature subset selection, that although it explores less subsets (so many as the number of features), it is not enough aggressive for text domains.

Since text domains have a lot of features and most of them are non relevant, the proposed wrapper searches over the subsets space using random sampling [10]. This wrapper initially takes the set of all features as the *global best subset*. Then, it generates a predefined number of times ($NRep$) random subsets of the *global best subset* (called *local subsets*) and the best of the local subsets is taken as the *global best subset*. If none of the *local subsets* is better than the current *global best subset* then the algorithm ends, otherwise the same process is repeated. This method is based on the idea previously applied for continuous categories in [13]. It consists of *if a feature subset of a set S is better than S, then the removed features are supposed to be not necessary or irrelevant*.

Once the searching algorithm is described, it is necessary to define the function that evaluates the subsets. The function consists of a combination (this combination is described later) of the results of a *Cross Validation* (CV) with 2 folds using SVM over the features of the subset. The use of SVM as the evaluation function is motivated by the fact that this classifier besides it has been shown to perform well in TC [7], it performs fast, which makes computationally possible the wrapper. The use of CV allows to avoid overfitting. More in detail, it is first computed the average of the True Positives (TP), False Positives (FP) and False Negatives (FN) of both CV executions of SVM and then the final evaluation of a subset is based on the widely adopted performance measures in TC [15], which is F_1 . So, we define the evaluation of a subset (V_{Subset}) as

$$V_{Subset}(TP, FP, FN) = \begin{cases} -FP - FN & \text{if } FP > TP \text{ or } FN > TP \\ 0 & \text{if } TP = 0 \text{ but } \sim (FP > TP \text{ or } FN > TP) \\ F_1(TP, FP, FN) = \frac{TP}{TP + \frac{FN + FP}{2}} & \text{otherwise} \end{cases}$$

This expression uses F_1 to evaluate a subset, but there are two exceptions: if TP is zero then F_1 can not be calculated, so in this case this function evaluates the subset as zero, which is lower than any possible value of F_1 . If $FP > TP$ or $FN > TP$ it means an awful learning, so the function evaluates the subset as a negative value, as negative as the sum of the two kinds of failures (FP and FN). The algorithm is detailed in figure 1.

```

model GFSSWsvm(DataSet Train,int NRep){
    Attribute Set GlobalBest=Attributes(Train);
    ValueGlobalBest=ValueOfSubSet(GlobalBest);
    Bool Mejora;
    do {
        Attribute Set LocalBest,Local;
        ValueLocalBest=-∞
        for(1 to NRep){ // Search for a local best
            Local=RandomSubSet(GlobalBest);
            if(ValueOfSubSet(Local)>ValueLocalBest)
                then {
                    LocalBest=Local;
                    ValueLocalBest=ValueOfSubSet(Local));
                }
        }
        if(ValueLocalBest>ValueGlobalBest)
            then { // There is one better than the best
                GlobalBest=LocalBest;
                ValueGlobalBest=ValueLocalBest;
                Mejora=true;
            }
        else Mejora=false; // GFSSWsvm ends
    }while(Mejora);
    return svm_learn(GlobalBest);
}
number ValueOfSubset(DataSet Data){
    {TP,FP,FN,TN}=CV2Folders(svm,Data);
    return VSubset(TP,FP,FN);
}

```

Fig. 1. Algorithm of the Greedy Feature Subset Selection Wrapper based on SVM. It generates NRep feature subsets of the current *global best subset*. If there is a feature subset S better than the current *global best subset* then S becomes the current *global best subset*

3.4 Classification

The aim of TC is to find out if a document is relevant to a category or not. Typically, the task of assigning a category to a document from a finite set of m categories is commonly converted into m binary problems each one consisting of determining whether a document belongs to a fixed category or not. This transformation makes possible the use of a wide number of binary classifiers to solve the multiclassification [2], [7] (*one-against-the-rest*).

One of the latest approaches showing the best results so far, is the use of SVM, since it is a good handler of many features and it deals so good with sparsity examples. These are universal binary classifiers able to find out linear or non-linear threshold functions to separate the examples of a certain category from the rest and they are based on the *Structural Minimization Risk* principle from computational learning theory [16]. We adopt this approach with a linear threshold function since the most of TC problems are linearly separable [7].

3.5 Evaluation of the Performance

Two measures of effectiveness commonly used in IR have been widely adopted [15]: *precision* and *recall*. Given the trade off existing between them [15], it is adequate to combine them and the most adequate choice is F_1 which gives the same relevance to both and it is defined as $F_1 = \frac{1}{0.5\frac{1}{Pr} + 0.5\frac{1}{Re}}$.

To compute the global performance over all the categories there exist two different approaches: *macroaveraging* (evaluating F_1 for each category and then averaging all of them) or *microaveraging* (summing TP , FP and FN for all the categories and then evaluating F_1).

4 Experiments

This section describes the corpus and shows the experiments carried out.

4.1 The Corpus

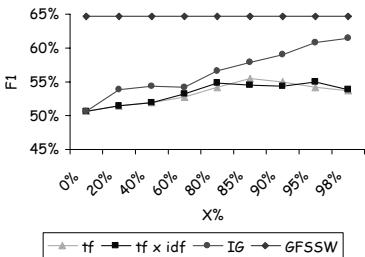
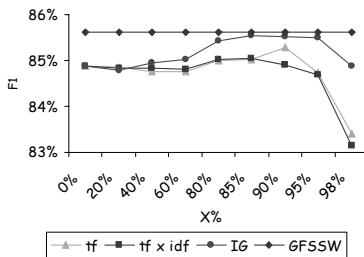
As in [6] and [11] we take the Reuters-21578 corpus. It contains short news related to economy published by Reuters during the year 1987 (it is publicly available at [5]). These stories are unbalanced distributed on 135 different prefixed categories. Each document is allowed to belong to one or more of these categories. We have chosen the split into train and test documents which was made by Apté (cf. [2]). After eliminating some documents which contain no body or no topics we finally have obtained 7063 train and 2742 test documents assigned to 90 different categories.

4.2 The Results

In the experiments with the filtering measures, a sweeping of filtering levels was made ranging from 20% to 98% (this level indicate the percentage of words of the vocabulary that are removed from the representation of the documents).

In the experiments with GFSSW, the algorithm must be executed several times, since it is a non deterministic system because it uses random sampling. A sample of 30 executions is enough to estimate the population mean by means of the sample mean. In this way, we estimate the *macroaverage* computing the average of the *macroaverage* of all executions and we estimate the *microaverage* computing first the average of TP , FP and FN of all executions. Also, the behaviour of GFSSW is shown to be steady since the mean, the minimum, the maximum and the Standard Deviation (SD) of the 30 executions are respectively: $M = 64.65\%$, $Min = 60.92\%$, $Max = 68.78\%$ and $SD = 2.05\%$ for the *macroaverage* and $M = 85.62\%$, $Min = 85.12\%$, $Max = 86.20\%$ and $SD = 0.30\%$ for the *microaverage*. The number of subsets generated in each iteration was taken as $NRep = 25$. The automatic filtering level obtained by GFSSW was 54.38%.

Figures 2 and 3 show the *macroaverage* and the *microaverage* of F_1 for tf , $tf \times idf$ and IG and for GFSSW.

**Fig. 2.** Macroaverage of F_1 **Fig. 3.** Microaverage of F_1

Looking at them, it is observed a different behaviour between the two different evaluation measures *macroaverage* and *microaverage*. This is because we are dealing with a collection that presents an unbalanced distribution of documents over the categories and meanwhile *macroaverage* gives equal importance to all categories, the *microaverage* assigns to each category a different weight, which is proportional to the number of documents.

In fact, for the *macroaverage*, GFSSW performs significantly better than the traditional filtering IR measures tf and $tf \times idf$ and the IT measure IG , which has been shown to be one of the best filtering measures.

For the *microaverage*, GFSSW also performs significantly better than tf and $tf \times idf$. However, GFSSW only is significantly better than IG when this measure filters at 20%, 40%, 60% and 98%. For the rest of filtering levels the performance of GFSSW is similar to IG .

5 Conclusions and Future Work

This paper proposes a wrapper approach based on SVM for feature reduction able to deal with large number of features present in text domains. The performance of this wrapper (called GFSSW) is better than the typical filtering measures widely used in TC, both for *macroaverage* and *microaverage*, but being similar for big categories (see *microaverage*) with respect to IG at high filtering levels. This different behavior arises when the collection has an unbalanced distribution of documents over the categories (like Reuters). The reason is because *macroaverage* gives equal importance to all the categories meanwhile *microaverage* assigns to each category an importance proportional to the number of documents of this category. Another advantage of the use of a wrapper instead of a filtering scoring measure is that it automatically selects the features and it does not need to define a predefined threshold to indicate how many features will be selected.

As future work, we are interesting in carrying out the same experiments over other text collections to check if the behaviour of the performance of GFSSW remains. We also plan to improve the method for searching in the feature space using a guided searching instead of random sampling.

References

1. D.W. Aha and R.L. Bankert. Feature selection for case-based classification of cloud types: An empirical comparison. In *Proceedings of the AAAI94 Workshop on Case-Based Reasoning*, 1994.
2. C. Apte, F. Damerau, and S. Weiss. Automated learning of decision rules for text categorization. *Information Systems*, 12(3):233–251, 1994.
3. R. Caruana and D. Freitag. Greedy attribute selection. In *Proceedings of the 11th International Conference on Machine Learning ICML94*, 1994.
4. K. J. Cherkauer and J. W. Shavlik. Growing simpler decision trees to facilitate knowledge discovery. In *Proceedings of the 2th International Conference on Knowledge Discovery and Data Mining KDD96*, 1996.
5. Reuters Collection. <http://www.research.att.com/lewis/reuters21578.html>.
6. E. F-Combarro, I. Díaz, E. Monta nés, A. M. Pea, and J. Ranilla. Aplicación de distintos métodos de aprendizaje automático a la clasificación documental. In *Conferencia Iberoamericana en Sistemas, Ciberntica e Informática CISCI 2002*, 2002.
7. T. Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveiro, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.
8. G.H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the 11th International Conference on Machine Learning ICML94*, 1994.
9. R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(12):273–324, 1997.
10. H. Liu and R. Setiono. A probabilistic approach to feature selection - a filter solution. In *Proceedings of the 13th International Conference on Machine Learning ICML96*, 1996.
11. E. Monta nés, J. Fernández, I. Díaz, E. F. Combarro, and J. Ranilla. Text categorisation with support vector machines and feature reduction. In *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation CIMCA2003*, 2003.
12. M. F. Porter. An algorithm for suffix stripping. *Program (Automated Library and Information Systems)*, 14(3):130–137, 1980.
13. J.R. Quevedo, E. Monta nés, and M.A. Alonso. Feature selection on modelling continuous systems by examples. In *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation CIMCA2003*, 2003.
14. G. Salton and M. J. McGill. *An introduction to modern information retrieval*. McGraw-Hill, 1983.
15. F. Sebastiani. Machine learning in automated text categorisation. *ACM Computing Survey*, 34(1), 2002.
16. V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.
17. T. Yang and J. P. Pedersen. A comparative study on feature selection in text categorisation. In *Proceedings of ICML'97, 14th International Conference on Machine Learning*, pages 412–420, 1997.

Robust Expectation Maximization Learning Algorithm for Mixture of Experts *

Romina Torres¹, Rodrigo Salas¹, Héctor Allende¹, and Claudio Moraga²

¹ Universidad Técnica Federico Santa María; Dept. de Informática;
Casilla 110-V; Valparaíso-Chile;
`{romina, rsalas, hallende}@inf.utfsm.cl`

² University of Dortmund; Department of Computer Science;
D-44221 Dortmund; Germany;
`moraga@cs.uni-dortmund.de`

Abstract. The Mixture of Experts (ME) model is a type of modular artificial neural network (MANN) specially suitable when the search space is stratified and whose architecture is composed by different kinds of networks which compete to learn several aspects of a complex problem. Training a ME architecture can be treated as a maximum likelihood estimation problem, where the Expectation Maximization (EM) algorithm decouples the estimation process in a manner that fits well with the modular structure of the ME architecture. However, the learning process relies on the data and so is the performance. When the data is exposed to outliers, the model is affected by being sensible to these deviations obtaining a poor performance as it is shown in this work. This paper proposes a Robust Expectation Maximization algorithm for learning a ME model (REM-ME) based on M-estimators. We show empirically that the REM-ME for these architectures prevents performance deterioration due to outliers and yields significantly faster convergence than other approaches.

Keywords: Modular Neural Networks, Mixtures of Experts, Expectation Maximization, Robust Learning Algorithm.

1 Introduction

Based on the idea that the brain is formed by a collection of modules each one specialized to an specific function, a novel modular connectionist architecture known as Mixture of Experts (ME) was developed by Jacobset al. [5]. This architecture performs task decomposition in the sense that it learns to partition a complex problem in two or more functionally independent subproblems. ME consists of two types of neural networks: the experts networks that compete to learn the training patterns and the gating network that allocates to each subproblem

* This work was supported in part by Research Grant Fondecyt 1010101 and 7010101, in part by Research Grant CHL-99/023 from the German Ministry of Education and Research (BMBF) and in part by Research Grant DGIP-UTFSM and in part by the Internship grant CONICYT-INRIA

the expert network whose topology is most appropriate to that subproblem. These networks are trained simultaneously so as to split the input space into regions where particular experts can specialize. The learning process of the ME model combines associative and competitive aspects of the learning. The competition enables that different networks learn different training patterns and, thus, learn to compute different functions.

The problem of training an ME architecture can be treated as a maximum likelihood estimation (MLE) problem. Although the gradient approach succeeded in finding reasonable parameter values, it does not appear to take advantage of the modularity of the ME architecture. An alternative approach was proposed by Jordan et al. [7] who introduced an Expectation Maximization (EM) algorithm for ME architectures, first presented by Dempster et. al. [2]. The EM algorithm decouples the estimation process in a manner that fits well with the modular structure of the architecture [6].

When the data is exposed to outliers, the model is affected by being sensible to these deviations obtaining a poor performance. The ME architecture is sensitive to the presence of outliers as shown in [10], motivating the research of robustness for these models. In this paper we propose to robustify the Expectation Maximization learning algorithm using M-estimators for the parameter estimation process to improve the performance of the Mixture of Experts model.

2 Mixture of Experts Models Architecture

As shown in Figure 1, the ME architecture, specified in [5], consists in K modules called *experts* and a *gating* network. The experts solve a function approximation problem over local regions of the input space, so the architecture needs a mechanism to identify for each input \underline{x} which experts are more adequate to model the desire output, work that is accomplished by the gating network.

To each expert there is an associated probabilistic model as follows $P(\underline{y}|\underline{x}, \underline{\theta}_j)$, $j = 1, \dots, K$, that relates input vectors $\underline{x} \in \mathbb{R}^n$ to output vectors $\underline{y} \in \mathbb{R}^m$, where the $\underline{\theta}_j$ are parameter vectors. Furthermore we assume that the probability densities $P(\underline{z}|\underline{\mu}, \Sigma)$ for $\underline{z} \in \mathbb{R}^p$ is a member of the family of p dimensional elliptically symmetric densities with mean vector $\underline{\mu}$ and covariance matrix Σ [8] of the following form:

$$|\Sigma|^{-1/2} \varphi\{(\underline{z} - \underline{\mu})^T \Sigma^{-1} (\underline{z} - \underline{\mu})\} \quad (1)$$

The j^{th} expert network produces as output a parameter vector $\underline{\mu}_j = f_j(\underline{x}, \underline{\theta}_j)$, $j = 1, \dots, K$, which is the location parameter for the j^{th} probability density. The gating network partitions the input space into regions corresponding to the various expert networks by assigning a probability vector $[g_1, g_2, \dots, g_K]^T$ to each point in the input space. g_j (output of the gating network) depends on the input \underline{x} by implementing a parameterized function $s(\underline{x}, \underline{\theta}_0)$ and a normalized function $g(s)$ as follows

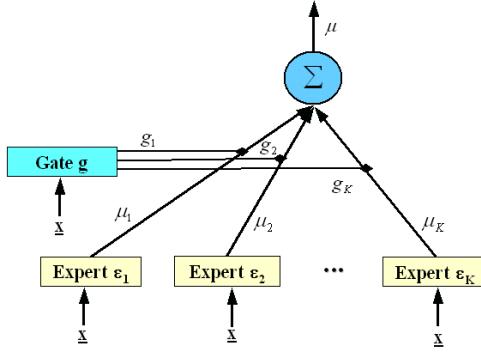


Fig. 1. Mixtures of Experts Architecture (ME): This architecture consists in a set of experts networks and a gating network. The experts compete for the learning of the problem and the gating mediates the competence.

$$g_j = g_j(\underline{x}, \underline{\theta}_0) = \frac{e^{s_j(\underline{x}, \underline{\theta}_0)}}{\sum_{i=1}^K e^{s_i(\underline{x}, \underline{\theta}_0)}}, \quad , j = 1, \dots, K, \quad (2)$$

The objective is to force each component of the output g_j to be non-negative and their sum be one. For each given \underline{x} , an expert j is selected with probability $P(j|\underline{x}, \underline{\theta}_0) = g_j(\underline{x}, \underline{\theta}_0)$. An output \underline{y} is then chosen with probability $P(\underline{y}|\underline{x}, \underline{\theta}_j)$. Thus the total probability of observing \underline{y} from \underline{x} is given by the following finite mixture of density

$$P(\underline{y}|\underline{x}) = \sum_{j=1}^K P(j|\underline{x}, \underline{\theta}_0) P(\underline{y}|\underline{x}, \underline{\theta}_j) = \sum_{j=1}^K g_j(\underline{x}, \underline{\theta}_0) P(\underline{y}|\underline{x}, \underline{\theta}_j) \quad (3)$$

So the expected value of the output will be the ME model output and is a weighted sum of the experts outputs: $\mu \equiv \mathbb{E}[\underline{y}|\underline{x}] = \sum_{j=1}^K g_j(\underline{x}, \underline{\theta}_0) \mu_j$

We assume that the training set $\Upsilon = \{(\underline{x}^{(t)}, \underline{y}^{(t)})\}_{t=1}^N$ is generated as an independent set of draws from this mixture density. Thus the total probability of the training set, for a specified set of input vectors $\{\underline{x}^{(t)}\}_{t=1}^N$, is given by the following likelihood function

$$L(\underline{\theta}, \Upsilon) = P(\{\underline{y}^{(t)}\}_{t=1}^N | \{\underline{x}^{(t)}\}_{t=1}^N) = \prod_{t=1}^N P(\underline{y}^{(t)} | \underline{x}^{(t)}) \quad (4)$$

where $\underline{\theta} = [\underline{\theta}_0^T, \underline{\theta}_1^T, \dots, \underline{\theta}_K^T, \text{vec}(\Sigma_1)^T, \dots, \text{vec}(\Sigma_K)^T]^T$ are the parameters of the model. The learning problem is treated as parameter estimation process of the ME architecture. The parameters $\underline{\theta}$ are chosen in the way that they maximize the likelihood function $L(\underline{\theta}, \Upsilon)$ given by (4), or, more conveniently, to maximize the log likelihood $l(\underline{\theta}, \Upsilon) = \ln(L(\underline{\theta}, \Upsilon))$. This process is known as maximum likelihood estimation (MLE).

3 Expectation Maximization Algorithm

Jacobs et al. introduce the EM algorithm [7]. EM is based on the idea of solving a succession of simplified problems that are obtained by augmenting the original observed variables with a set of additional "hidden" variables.

Given an observed data set \mathcal{Y} , this is augment with a set of additional variables \mathcal{Y}_{mis} , called "missing" or variaables, and consider a maximum likelihood problem for a "complete-data" set $Z = \{\mathcal{Y}, \mathcal{Y}_{mis}\}$. By choosing the missing variables in such a way that the resulting "complete-data log likelihood", given by $l_c(\underline{\Theta}, Z) = \ln P(\mathcal{Y}, \mathcal{Y}_{mis} | \underline{\Theta})$, is easy to maximize with respect to $\underline{\Theta}$. The probability model $P(\mathcal{Y}, \mathcal{Y}_{mis} | \underline{\Theta})$ must be choosen so that its marginal distributions across \mathcal{Y} , referred as the "incomplete-data" likelihood, is the original likelihood, i.e. $P(\mathcal{Y} | \underline{\Theta}) = \int P(\mathcal{Y}, \mathcal{Y}_{mis} | \underline{\Theta}) d\mathcal{Y}_{mis}$. Due to the fact that the likelihood is a random function of the "missing" random variables, we cannot work directly with the "complete-data" log likelihood. Is necessary to average out \mathcal{Y}_{mis} , that means, to maximize the *expected* complete-data log likelihood $\mathbb{E}_{\mathcal{Y}_{mis}} [\ln P(\mathcal{Y}, \mathcal{Y}_{mis} | \underline{\Theta})]$.

The EM algorithm for the Mixture of Experts models was derivated in [6] and is formulated as follows:

1. **The Expectation (E) step**, computes the folowing conditional expectation of the log likelihood

$$\begin{aligned} Q(\underline{\Theta} | \underline{\Theta}^{(k)}) &= \mathbb{E}_{\mathcal{Y}_{mis}} \left[\ln \left(P(Z | \underline{\Theta}, \mathcal{Y}, \underline{\Theta}^{(k)}) \right) \right] \\ &= \sum_{t=1}^N \sum_{j=1}^K h_j^{(k)}(t) \ln(g_j(\underline{x}^{(t)}, \underline{\theta}_0)) \\ &\quad + \sum_{j=1}^K \left[\sum_{t=1}^N h_j^{(k)}(t) \ln(P(\underline{y}^{(t)} | \underline{x}^{(t)}, \underline{\theta}_j)) \right] \end{aligned} \quad (5)$$

where $\underline{\Theta}^{(k)}$ is the value of the parameter vector at iteration k and

$$\begin{aligned} h_j^{(k)}(t) &= \mathbb{E}[I_j^{(t)} | \mathcal{Y}, \underline{\Theta}^{(k)}] = P(j | \underline{x}^{(t)}, \underline{y}^{(t)}) \\ &= \frac{g_j(\underline{x}^{(t)}, \underline{\theta}_0^{(k)}) P(\underline{y}^{(t)} | \underline{x}^{(t)}, \underline{\theta}_j^{(k)})}{\sum_{i=1}^K g_i(\underline{x}^{(t)}, \underline{\theta}_0^{(k)}) P(\underline{y}^{(t)} | \underline{x}^{(t)}, \underline{\theta}_i^{(k)})}, \end{aligned} \quad (6)$$

where $P(j | \underline{x}^{(t)}, \underline{y}^{(t)})$ denotes the probability that the pair $(\underline{x}^{(t)}, \underline{y}^{(t)})$ comes from the j^{th} probability model. Note that we always have $h_j^{(k)}(t) > 0$.

2. **The Maximization (M) step:** chooses a parameter value that increases the Q function, i.e., $\underline{\Theta}^{(k+1)} = \arg \max_{\underline{\Theta}} Q(\underline{\Theta} | \underline{\Theta}^{(k)})$

4 Robust Expectation Maximization Learning Algorithm

In this section we introduce a robust expectation maximization algorithm for Mixture of Experts (REM-ME). We will use the M-Estimators introduced by Huber [4] in the EM algorithm by modifying the espcetion E step of the algorithm in such a form that the influence introduced by an outlying observation will be bounded.

When the data is contaminated and the number of experts is low in order to not overfit the data, the performance of this algorithm is considerably reduced. The poor performance of the model is due to the fact that usually we assume, the data can be modelled by a mixture of gaussian, but when there is a presence of outliers this assumption no longer holds. Unfortunately real data are not free of outlying observations. In order to improve the performance more experts are introduced in the model but overfitting on the bulk of data occurs.

When the outlying observation is presented to the model, the magnitude of the likelihood that each expert and the gating network evaluate are very low (or very high absolute value), so during the learning process the step that each expert and the gating network must take towards the observation is of big magnitude, so the model gets away from the bulk of data. To avoid this problem we must downweight the influence of samples with low likelihood that are likely to be regarded as outliers. This task will be accomplished by introducing a $\rho(\cdot)$ function to each expert and the gating network after applying the natural logarithm $\ln(\cdot)$ as follows:

$$RQ(\underline{\Theta}|\underline{\Theta}^{(k)}) = \sum_{t=1}^N \sum_{j=1}^K h_j^{(k)}(t) \rho(\ln(g_j(\underline{x}^{(t)}, \underline{\theta}_0))) + \sum_{j=1}^K \left[\sum_{t=1}^N h_j^{(k)}(t) \rho(\ln(P(\underline{y}^{(t)}|\underline{x}^{(t)}, \underline{\theta}_j))) \right] \quad (7)$$

where $h_j^{(k)}(t)$ is given by the equation (6). We will denote RQ for $RQ(\underline{\Theta}|\underline{\Theta}^{(k)})$. The model will be robust by been insensitive to outlying observations due to the robust contribution of each expert.

The problem now arises in how to chose the right $\rho(\cdot)$ function to accomplish the robustification task. In [3] some special functions for M-estimation are discussed. The goal is to weight each observation according to the magnitude of likelihood evaluated at the observation. Samples with low likelihood are likely to be regarded as outliers and are downweighted. In particular, for the location problem, data that are far away must have a bounded impact in the estimation algorithm, we will use the Huber function given by

$$\rho_H(z) = \begin{cases} z + \frac{1}{2} \log(2\pi) & \text{if } z \geq \frac{1}{2}(-k^2 - \log(2\pi)) \\ -k\{-2z + \log(2\pi)\}^{\frac{1}{2}} - \frac{1}{2}k^2 & \text{otherwise} \end{cases} \quad (8)$$

$$\psi_H(z) = \begin{cases} 1 & \text{if } z \geq \frac{1}{2}(-k^2 - \log(2\pi)) \\ k\{-2z - \log(2\pi)\}^{-\frac{1}{2}} & \text{otherwise} \end{cases} \quad (9)$$

The function $\psi(\cdot)$ will be a type of weight function that will limit the influence of large atypical samples. In the E step, lets compute the following quantities:

$$\psi_{0,j}^{(k)}(t) = \psi(\ln(g_j(\underline{x}^{(t)}, \underline{\theta}_0))) \quad (10)$$

$$\psi_j^{(k)}(t) = \psi(\ln(P(\underline{y}^{(t)}|\underline{x}^{(t)}, \underline{\theta}_j))) \quad (11)$$

Now the Maximization (M) step calculates $\underline{\Theta}^{(k+1)} = \arg \max_{\underline{\Theta}} RQ(\underline{\Theta}|\underline{\Theta}^{(k)})$, or equivalently solves the estimation equations:

$$\frac{\partial RQ}{\partial \underline{\theta}_0} \Big|_{\underline{\theta}_0 = \underline{\theta}_0^{(k+1)}} = 0 \quad \frac{\partial RQ}{\partial \underline{\theta}_j} \Big|_{\underline{\theta}_j = \underline{\theta}_j^{(k+1)}} = 0 \quad \frac{\partial RQ}{\partial \Sigma_j} \Big|_{\Sigma_j = \Sigma_j^{(k+1)}} = 0 \quad (12)$$

Based on equations (2), (3), (7) and the distribution model assumed in (1), follows by taking the partial derivatives of RQ with respect to the parameters \underline{t}_{c_0} , \underline{t}_{c_j} and $\underline{\Sigma}$, where $\underline{\Sigma}$ is the covariance matrix obtained from the elliptical distribution assumed is (1):

$$\frac{\partial RQ}{\partial \underline{\theta}_0} = \sum_{t=1}^N \sum_{j=1}^K h_j^{(k)}(t) \psi_{0,j}^{(k)}(t) \left(g_j(\underline{x}^{(t)}, \underline{\theta}_0) \right)^{-1} \frac{\partial g_j(\underline{x}^{(t)}, \underline{\theta}_0)}{\partial \underline{\theta}_0} \quad (13)$$

$$\frac{\partial RQ}{\partial \underline{\theta}_j} = \sum_{t=1}^N h_j^{(k)}(t) \psi_j^{(k)}(t) \left(P(\underline{y}^{(t)} | \underline{x}^{(t)}, \underline{\theta}_j) \right)^{-1} \frac{\partial P(\underline{y}^{(t)} | \underline{x}^{(t)}, \underline{\theta}_j)}{\partial \underline{\theta}_j} \quad (14)$$

$$\frac{\partial RQ}{\partial \Sigma_j} = \sum_{t=1}^N h_j^{(k)}(t) \psi_j^{(k)}(t) \left(P(\underline{y}^{(t)} | \underline{x}^{(t)}, \underline{\theta}_j) \right)^{-1} \frac{\partial P(\underline{y}^{(t)} | \underline{x}^{(t)}, \underline{\theta}_j)}{\partial \Sigma_j} \quad (15)$$

where

$$\frac{\partial g_j(\underline{x}^{(t)}, \underline{\theta}_0)}{\partial \underline{\theta}_0} = g_j(\underline{x}^{(t)}, \underline{\theta}_0) \left[\frac{\partial s_j(\underline{x}^{(t)}, \underline{\theta}_0)}{\partial \underline{\theta}_0} - \sum_{i=1}^K \frac{\partial s_i(\underline{x}^{(t)}, \underline{\theta}_0)}{\partial \underline{\theta}_0} \right] \quad (16)$$

$$\frac{\partial P(\underline{y}^{(t)} | \underline{x}^{(t)}, \underline{\theta}_j)}{\partial \underline{\theta}_j} = -2|\Sigma|^{-1/2} \varphi'(\cdot) \frac{\partial f_j(\cdot)}{\partial \underline{\theta}_j} \times \Sigma^{-1} (\underline{y}^{(t)} - f_j(\cdot)) \quad (17)$$

$$\frac{\partial P(\underline{y}^{(t)} | \underline{x}^{(t)}, \underline{\theta}_j)}{\partial \Sigma_j} = -|\Sigma|^{-1/2} \Sigma^{-T} \left[\frac{1}{2} \varphi(\cdot) - \varphi'(\cdot) (\underline{y}^{(t)} - f_j(\cdot)) (\underline{y}^{(t)} - f_j(\cdot))^T \Sigma^{-T} \right] \quad (18)$$

$$\varphi(\cdot) = \varphi\{(\underline{y}^{(t)} - f_j(\cdot))^T \Sigma_j^{-1} (\underline{y}^{(t)} - f_j(\cdot))\}, \varphi'(z) = \frac{\partial \varphi(z)}{\partial z} \text{ and } f_j(\cdot) = f_j(\underline{x}^{(t)}, \underline{\theta}_j).$$

To obtain the update of the parameters of the gating network, experts networks and covariance matrix respectively, the Newton method can be used. Further simplifications can be made, for example, we can consider that the density $\varphi(z) = (2\pi)^{-1/2} \exp\{-z/2\}$ is assumed to be of the exponential type, for example Gaussian and experts are of the linear type. With this consideration, actualization of the covariance matrix is given by:

$$\Sigma_j^{(k+1)} = \frac{1}{\sum_{t=1}^N h_j^{(k)}(t) \psi_j^{(k)}(t)} \sum_{t=1}^N h_j^{(k)}(t) \psi_j^{(k)}(t) [\underline{y}^{(t)} - f_j(\cdot)] [\underline{y}^{(t)} - f_j(\cdot)]^T \quad (19)$$

The update for the parameters of the experts networks is:

$$\underline{\theta}_j^{(k+1)} = (\underline{R}_j^{(k)})^{-1} \underline{\mathcal{L}}_j^{(k)} \quad (20)$$

where $(X_t)_{i,\cdot} = \{0 \dots \underline{x}^{(t)} \dots 0 | 0 \dots 1 \dots 0\}$, $i = 1..K$

$$\begin{aligned}\underline{\mathcal{L}}_j^{(k)} &= \sum_{t=1}^N h_j^{(k)}(t) \psi_j^{(k)}(t) X_t (\Sigma_j^{(k)})^{-1} \underline{y}^{(t)} \\ \underline{R}_j^{(k)} &= \sum_{t=1}^N h_j^{(k)}(t) \psi_j^{(k)}(t) X_t (\Sigma_j^{(k)})^{-1} X_t^T\end{aligned}$$

Jordan and Jacobs realize that the gating network is a *multinomial logit* model, that can be fit efficiently with a variant of Newton's method known as *iteratively reweighted least squares* (IRLS). The update for the gating network parameters is obtained as follows:

$$\underline{\theta}_j^{(k+1)} = \gamma_g (\underline{R}_j^{(k)})^{-1} \underline{\mathcal{L}}_j^{(k)} \quad (21)$$

where γ_g is a learning rate, moreover the gradient vector and the Hessian matrix at iteration k respectively are

$$\begin{aligned}\underline{\mathcal{E}}_g^{(k)} &= \sum_{t=1}^N \sum_{j=1}^K \psi_j^{(k)}(t) [h_j^{(k)}(t) - g_j(\underline{x}^{(t)}, \underline{\theta}_0^{(k)})] \frac{\partial s_j}{\partial \underline{\theta}_0^{(k)}} \\ \underline{R}_g^{(k)} &= \sum_{t=1}^N \sum_{j=1}^K \psi_j^{(k)}(t) g_j(\underline{x}^{(t)}, \underline{\theta}_0^{(k)}) [1 - g_j(\underline{x}^{(t)}, \underline{\theta}_0^{(k)})] \frac{\partial s_j}{\partial \underline{\theta}_0} \frac{\partial s_j}{\partial \underline{\theta}_0^{(k)}}\end{aligned}$$

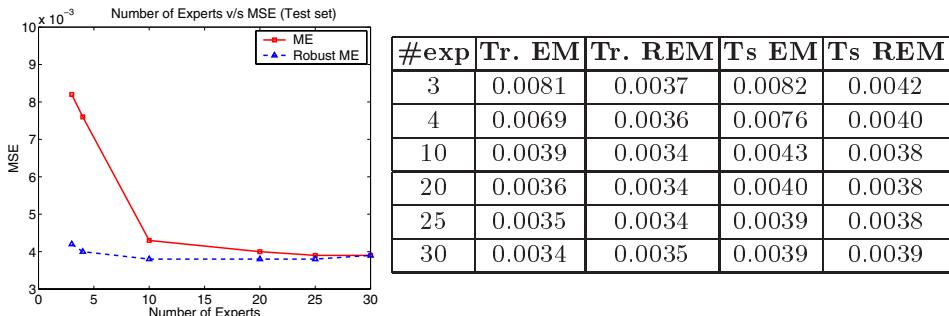
In summary, the parameter update by the REM-ME algorithm is given as follows:

1. **The E step:** Compute the $h_j^{(k)}(t)$, $\psi_{0,j}^{(k)}(t)$ and $\psi_j^{(k)}(t)$, $j = 1..K$, by equations (6), (10) and (11) respectively.
2. **The M step:** Estimate $\Sigma_j^{(k+1)}$, $\underline{\theta}_j^{(k+1)}$, $j = 1, \dots, K$ and $\underline{\theta}_0^{(k+1)}$ by equations (19), (20) and (21) respectively.

5 Simulations results applied to the Building Data

In this section, we report the results of our experiments comparing the REM-ME algorithm with the classical EM. The robust learning of the ME models were evaluated on a real life data consisting on a prediction problem. The *Building2* data set was obtained from the *PROBEN1* benchmark set [9]. The problem is to predict the hourly consumption of electrical energy, hot water, and cold water, based on the date, day of the week, time of day, outside temperature, outside air humidity, solar radiation, and wind speed. The data set is spread over six month from September to February. So, the input vector is dimension 14 and the output vector is 3. The data consists in 4208 samples.

The results obtained are shown in the figure 2, where in the left side of the figure shows that the REM-ME algorithm outperforms the classical EM algorithm when the number of experts is low, but asymptotically they show similar results. In the right side of the figure 2, the performance evaluation based in the mean square error for the training set (Tr.) and the test set (Ts.) for the EM y REM-ME algorithm, with different numbers of experts, is shown.

Fig. 2. Number of Experts vs. MSE

6 Concluding Remarks

In this work it is shown that the ME models with robust learning outperforms the results presented in similar works where ME models based on the classical EM algorithm. For the experiments studied in this work, it is not necessary to introduce more complexity by using localized model with robust learning, because less numbers of experts are required to obtain a desired performance, and the convergence of the learning algorithm is faster.

Further robust analysis should be made in this type of networks with possible extensions to Hierarchical ME. Robustness studies have been made for other neural networks [1].

References

1. H. Allende, C. Moraga, and R. Salas, *Robust estimator for the learning process in neural networks applied in time series*, LNCS **2415** (2002), 1080–1086.
2. A. P. Dempster, N. M. Laird, and D. B. Rubin, *Maximum likelihood from incomplete data via the EM algorithm*, **39** (1977), 1–38.
3. F.R. Hampel, E.M. Ronchetti, P.J. Rousseeuw, and W.A. Stahel, *Robust statistics*, Wiley Series in Probability and Mathematical Statistics, 1986.
4. Peter J. Huber, *Robust statistics*, Wiley Series in probability and mathematical statistics, 1981.
5. R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, *Adaptive mixtures of local experts*, Neural Computation **3** (1991), no. 1, 79–87.
6. M. I. Jordan and L. Xu, *Convergence properties of the EM approach to learning in mixture-of-experts architectures*, Neural Networks **8** (1995), 1409–1431.
7. M.I. Jordan and R.A. Jacobs, *Hierarchical mixtures of experts and the EM algorithm*, Neural Computation **6** (1994), no. 2, 181–214.
8. N.Cambell, *Mixture models and atypical values*, Math. Geol. **16** (1984), 465–477.
9. L. Prechelt, *Proben1 – a set of benchmarks and benchmarking rules for neural training algorithms.*, Technical Report 21/94, Fakultaet fur Informatik, Universitaet Karlsruhe, D-76128 Karlsruhe, Germany (1994).
10. R. Torres, R. Salas, H. Allende, and C. Moraga, *Estimador robusto en modelos de mezcla de expertos locales*, CLATSE V (2002).

Choosing among algorithms to improve accuracy*

José Ramón Quevedo, Elías F. Combarro, Antonio Bahamonde

Centro de Inteligencia Artificial. Universidad de Oviedo at Gijón.

Campus de Viesques. E-33271 Gijón, Spain

{quevedo, elias, antonio}@aic.uniovi.es

<http://www.aic.uniovi.es>

Abstract. It is a widely accepted fact that no single Machine Learning System (MLS) gets the smaller classification error on all data sets. Different algorithms fit better to certain problems and it is interesting to combine them in some way to improve the overall accuracy. In this paper, we propose a method to construct a new MLS from given ones. It is based on the selection of the system that will perform better on a particular data set. We study several ways of selecting the systems and carry out experiments with well-known MLS on the Holte data set.

1 Introduction

The existence of a wide variety of Machine Learning Systems based on different paradigms and showing different performances on different kinds of problems raises a question: How can we decide which one should be applied to a concrete problem?

Usually, we will have at our disposal a data set of train examples and several algorithms that we can use to induce classifiers by training on the data. From the observed accuracy of these classifiers, we want to be able to decide which algorithm will have the lowest classification error when presented with new examples.

In the literature, this question has been acknowledged specially for the particular case of the comparison of just two algorithms. The traditional approach implies the use of a statistical test that decides whether the performances of the algorithms are significantly different or not (at some level of confidence).

Among the most used tests we find the *paired t-test* [1]. This is usually applied in combination with *cross-validation* (*cv*), pairing the errors of classification of the algorithms on the same folds. The main drawback of this method is that it relies on the assumption of the independence of the estimated errors, which is not true in general. The same happens when procedures other than *cv* (for instance, resampling) are used to select the examples with which the algorithms are trained.

Another popular test is McNemar's one [2], which while requiring only one application of the inducers, presents the disadvantage of not measuring the variability inherent to the choice of the set of train examples.

* The research reported in this paper has been supported in part under MCyT and Feder grant TIC2001-3579

The difference of proportions [3] of the error rates of the algorithms is another test; however, it shares the drawbacks of McNemar's test and relies also on independence hypothesis which do not always hold.

In [4] Dietterich proposed a new test that he called the *5x2cv paired t test*. To apply this test, a *cv* of 2 folds is repeated 5 times with the same folds for the two systems. With the differences of the errors of the two algorithms, a test statistic that follows a *t* distribution with 5 degrees of freedom (under the null hypothesis) is constructed. From the way it is computed no independence hypothesis are violated in this case.

Among all these tests, this last ones show the lowest type I error and its use is suggested [4] when the cost of the execution of the *cv* is not critical.

In this paper, we explore the possibility of using this and other tests to construct, from different MLS, a new one with better accuracy. In an ideal situation, we could select the most appropriated system in every case and, then, the overall performance would be better than that of the individual constituents.

Several different methods for the combination of MLS or their predictions have been proposed and compared, like bagging, boosting, stacking and cascading in [5], combining Instance-based and Model-based Learning as Quinlan in [6] or using a MLS for symbolic labeled examples to manage symbolic attributes when learning continuous labeled data sets [7]. However, the construction we propose here is not easily compared to those other ones, since they work at the example level, deciding how to combine the predictions of the different systems for each individual case. Instead, our method selects the system to be applied to a whole problem, and it is complementary to those other approaches, rather than alternative.

2 The LearnWithChosen Algorithm

When accuracy of several Machine Learning Systems (MLS) over the Data Sets in a repository is compared, generally there is no one with the best accuracy for all the Data Sets. The objective of this section is to present a new MLS that tries to be as good as the best in each Data Set.

To develop this MLS we propose an algorithm that needs a decision function that, given a Data Set, chooses the *best* between the MLS candidates. The figure 1 shows this algorithm, that it is called *LearnWithChosen*.

```
Error LearnWithChosen(Function ChooseMLS,Vector MLS
VMLS, DataSet Train, DataSet Test)
{
    MLS Chosen=ChooseMLS(VMLS,Train);
    return Apply(Learn(Chosen,Train),Test);
}
```

Fig. 1. The *LearnWithChosen* algorithm chooses a MLS and learns with it. The function *ChooseMLS* is a parameter of this algorithm.

Even in the case that *LearnWithChosen* (LWC) uses the perfect *ChooseMLS* function, it is very important the election of the MLS candidates. If one MLS candidate is always better than the rest, the LWC will be as good as the first one, so

there is no improvement in the accuracy. If the MLS candidates have similar accuracy for all the Data Sets, there can not be significant improvement in the accuracy, because any of the MLS chosen would have similar accuracy.

LWC would get significant improvement in the accuracy only if the MLS candidates have different accuracy over the same Data Sets, and there is no one that is always better than the others in all Data Sets of the repository.

2.1 Using 5x2cv paired t test to choose

The Dietterich's *5x2cv paired t test* can be applied to choose between two MLS. This test has low type I error [4], so it is adequate to make high assurance decisions. However, this is not enough to develop a *ChooseMLS* function because it is necessary to choose among several (possible more than two) MLS. This problem can be solved using a Round Robin strategy. Fürnkranz shows in [8] that these kinds of strategies are fine for deciding among several classes using only binary tests. However a Round Robin algorithm can no decide (the voting could end in a tie), in this case, the MLS with lower average error in the *5x2cv* is chosen. Figure 2 details this function.

2.2 Using the average of the error to choose

The average of the errors in a cross validation can be used to choose among several MLS. However there is not assurance of low type I error, like in the *5x2cv paired t test*. This measure can decide among several MLS, not only between two, so it is not necessary a Round Robin competition.

One problem is to decide the number of folders and the minimum number of repetitions. In [4], Dietterich argues that with a five times two folders cross validation it is possible to make a fine test, so this number of folders and repetitions could be a good combination. To increase the precision of the test it could be good to increase the number of repetitions. We propose 30 repetitions to obtain a very precise test, but this number of repetitions could make unfeasible the use of the test for large data sets or slow MLS.

Both methods of choosing are no perfect: the wrong MLS can be chosen. An extra information that can help to this method is to make a previous select of MLS. Is convenient to select as less MLS as possible and select MLS that have different errors for the same Data Set.

We propose a heuristic to select N MLS from M available: choose the N MLS that minimizes the average of errors of the hypothetical MLS that has an accuracy equal to the minimum of the errors of each N MLS over a Data Set.

```

MLS 5x2cvPairedTTestRR(Vector MLS MLSV,DataSet Train)
{
  //The 10 Errors of a 5x2cv are calculated for each MLS
  for(i=0 to MLS.Dimension)
    Err[i]=5x2cv(MLS[i],Train);

  // Each MLS compete against the rest in Round Robin
  for(i=0;i<Number of MLS;i++)
    for(j=i+1;j<Number of MLS;j++) {
      Decision Dec=5x2cvPairedTTest(Err[i],Err[j]);
      if(Dec==1) {Score[i]++; Score[j]--;} // MLS i wins
      if(Dec==2) {Score[i]--; Score[j]++;} // MLS j wins
    }
  // The one with more 'Score' results is chosen
  Vector MLS best=GetTheMLSWithMoreScore(MLS,Score);
  if(best.Dimension==1) return best[1];

  //There are more than one MLS with the maximum Score
  //the one of these with lower error is chosen
  best=MinimumAverage(MLS,best,Err);
  return best;
}

```

Fig. 2. Chooses among the MLS for the given train Data Set applying the *5x2cv paired t test* in a Round Robin competition. If there is a tie in the voting, the one with less error in the *5x2cv* is chosen.

3 Experimentation

The experiments have been done in a MLC++ [9] environment using 2032 (laboratory telephone number) as seed for repeatability. An experiment consists of a 10 folders cross validation repeated 5 times (10x5cv).

The experiments are executed over the well-known set of 16 problems used by Holte in [10]. The *LearnWithChosen* system will be compared using this set of problems with the state-of-the-art and well known systems: c4.5[11] release 8, rise[12], ripper[13], oc1[14], naive-bayes and two versions of the *nearest neighbor*[15] family systems: 1NN and ANN-Man-D². The 1NN is a *nearest neighbor* system with k=1 and the HEOM[16] metric to determine distance between examples; the ANN-Man-D² is a *nearest neighbor* system with k=A, where A is the number of attributes, it uses a metric like HEOM but with Manhattan[16] distance instead of then Euclidian one and it weights the vote by the inverse of the square of the distance. This version is used because experimentally shows an important grown of the accuracy respect of the 1NN system.

The average errors of a 10x5cv experiment using these systems for each Holte Data Set are shown in table 1.

Table 1. Average error of a 10x5cv experiment for each System and Data Set.

	c4.5	rise	ripper	oc1	naive-bayes	1NN	ANN-Man-D ²
BC	25.58%	26.70%	28.54%	34.47%	27.68%	30.62%	24.81%
CH	1.09%	1.95%	1.36%	2.27%	13.24%	10.21%	3.87%
G2	22.85%	20.66%	20.06%	24.51%	38.07%	22.10%	18.70%
GL	30.56%	27.28%	34.11%	32.62%	52.70%	31.21%	27.29%
HD	24.45%	18.92%	20.36%	21.26%	16.81%	24.11%	17.25%
HE	19.89%	21.00%	19.78%	20.82%	16.13%	19.52%	16.92%
HO	15.00%	14.94%	15.11%	21.80%	20.87%	21.15%	17.66%
HY	0.75%	1.84%	0.81%	1.66%	2.12%	2.69%	3.12%
IR	5.20%	4.41%	5.20%	5.33%	4.80%	4.40%	4.67%
LA	19.07%	9.60%	14.87%	20.60%	9.47%	17.07%	7.33%
LY	23.14%	18.23%	23.07%	20.91%	20.74%	18.62%	15.53%
MU	0.00%	0.00%	0.00%	0.00%	0.31%	0.00%	0.00%
SE	2.04%	3.58%	2.33%	3.59%	15.88%	6.71%	6.65%
SO	2.90%	0.00%	2.10%	1.70%	9.00%	0.00%	0.00%
V0	3.49%	4.79%	4.27%	5.88%	9.83%	8.10%	7.25%
V1	10.15%	11.22%	10.89%	10.94%	12.72%	13.14%	10.75%
Av.	12.89%	11.57%	12.68%	14.27%	16.90%	14.35%	11.36%

3.1 Choosing between candidates

To test the algorithm, we have performed experiments choosing between 2, 3 and 4 MLS with either the 5x2cv test and the average error of 5x2cv and 30x2cv experiments. As MLS we have selected those which *a priori* are expected to combine better: those which present the smaller average error on all the data sets if the minimum error of all them is considered in every data set. For two candidates the best are c4.5 and ANN-Man-D², for three candidates c4.5, ANN-Man-D² and naive-bayes, and for four rise, c4.5, ANN-Man-D² and naive-bayes. Instead of rise in this last group we have selected ripper, because the average error obtained is very close but ripper is a MLS following a paradigm different to the other ones and it could improve the performance of the combination.

With this systems he have performed a stratified 10x5cv exactly as in the case of the constituents systems. We use this experiment because Kohavi [17] shows that stratified 10-fold cross-validation produces fairly good estimates of the error. The average errors on the Holte Data Sets are presented in table 2.

Table 2. Average errors of a 10x5cv experiment for the *LearnWithChosen* systems. Av5x2 means that average error of a 5x2cv is used to decide between the systems, RR that the 5x2cv paired *t test* together with Round Robin is used, and Av30x2 that the system with smaller error on a 30x2cv is selected.

	C4.5 and ANN-Man-D ²			C4.5, ANN-Man-D ² and naive-bayes			Ripper, C4.5, ANN-Man-D ² and naive-bayes		
	Av5x2	RR	Av30x2	Av5x2	RR	Av30x2	Av5x2	RR	Av30x2
BC	24.94%	25.01%	24.81%	24.94%	25.37%	24.81%	25.70%	26.34%	25.08%
CH	1.09%	1.09%	1.09%	1.09%	1.09%	1.09%	1.09%	1.09%	1.09%
G2	18.95%	18.95%	18.70%	18.95%	18.71%	18.70%	19.29%	19.06%	19.17%
GL	27.29%	27.29%	27.29%	27.29%	27.29%	27.29%	27.29%	27.29%	27.29%
HD	17.25%	17.25%	17.25%	18.25%	18.44%	18.71%	18.25%	18.57%	18.71%
HE	17.29%	17.29%	16.92%	16.13%	16.13%	16.13%	16.13%	16.13%	16.13%
HO	14.99%	15.04%	14.99%	14.99%	15.53%	14.99%	14.99%	15.37%	14.99%
HY	0.75%	0.75%	0.75%	0.75%	0.75%	0.75%	0.74%	0.73%	0.74%
IR	4.67%	4.67%	4.67%	5.47%	5.47%	5.47%	5.47%	5.60%	5.47%
LA	7.33%	7.33%	7.33%	7.33%	7.67%	7.33%	8.00%	9.67%	7.33%
LY	15.53%	15.53%	15.53%	15.53%	16.32%	15.53%	15.53%	16.32%	15.53%
MU	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
SE	2.04%	2.04%	2.04%	2.04%	2.04%	2.04%	2.04%	2.04%	2.04%
SO	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
V0	3.49%	3.49%	3.49%	3.49%	3.49%	3.49%	3.49%	4.18%	4.18%
V1	10.76%	10.76%	10.89%	10.76%	10.71%	10.89%	10.71%	10.80%	10.89%
Av.	10.40%	10.41%	10.36%	10.44%	10.56%	10.45%	10.59%	10.82%	10.56%

We have conducted test hypothesis to study whether these systems perform better than the original ones in these data sets. With the Wilcoxon [3] signed-rank test for paired data (we pair the errors of the different systems on the same data sets) we obtained that at a significance level of 95% all the *LearnWithChosen* systems have smaller classification error than rise, c4.5, naïve-bayes, oc1, 1NN and ripper. Additionally, the combination of c4.5 and ANN-Man-D² and the combination of naïve-bayes, c4.5 and ANN-Man-D² have smaller error than ANN-Man-D² when the selection of the system is made according to the average error on 30x2cv. Among the combinations of the same methods there are no significative differences, with just two exceptions: using Round Robin and the 5x2cv test leads to bigger errors than using the average error of the 5x2cv when more than two systems are involved.

To test the performance of the systems on the different data sets we have conducted paired t-test with the values of the 5 repetitions of 10 folds. As we have already mentioned the validity of these tests could be arguable, but it is a usual practice in the study of the performance of different MLS. For informative purposes, we present the

results of these tests in table 3. We present only the extreme situations: the *LearnWithChosen* systems with smaller and bigger average error.

Table 3. Results of the paired t-test between the basic systems and two *LearnWithChosen* systems. For each basic system the first column shows the result of the test agains the combination of c4.5 and ANN-Man-D² using Av30x2; the second shows the result of the test agains the combination of 4 systems with RR. A + indicates that the *LearnWithChosen* system performs better; a - that the basic system does.

	c4.5	rise	ripper	oc1	naive-bayes	1NN	ANN-Man-D ²
BC		+	+	+	+	+	-
CH		+	+	+	+	+	+
G2	+	+		+	+	+	+
GL	+	+		+	+	+	+
HD	+	+	+	+	+	+	-
HE	+	+	+	+	+	+	-
HO	+			+	+	+	+
HY		+	+	+	+	+	+
IR						-	-
LA	+	+		+	+	+	-
LY	+	+	+	+	+	+	+
MU					+	+	
SE		+	+	+	+	+	+
SO	+	+		+	+	+	+
V0	-	+	+	+	+	+	+
V1				+	+	+	+

4 Conclusions

We have presented a general algorithm for the combination of different MLS which depends on a choice function. We have also proposed three alternatives for this function. With them, we have conducted several experiments with well-known MLS on the Holte data set. The results of these systems seem to be better than those of the original systems, specially when we combine systems which perform very differently (for instance c4.5 and ANN-Man-D² and also naïve-bayes) and consider their errors on a 30x2-cv on the train set.

It would be interesting to study the influence of different choice functions, that may be based on other statistical tests (for instance McNemar's one) or not. It would be also interesting to study further the influence of the number of MLS in the performance of the method. Finally, we would like to extend these ideas to problems in which a real-valued function (rather than a classifier) has to be learned from sample data.

References

1. Cohen, P.R.: Empirical Methods for Artificial Intelligence. MIT Press. (1995)
2. Everitt, B.S.: The analysis of contingency tables. Chapman and Hall, London. (1977)
3. Snedecor, G.W., Cochran, W.G.: Statistical Methods. Iowa State University Press, Ames, IA, 8th edition. (1989)
4. Dietterich, T.G.: Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*. (1998) 10(7):1895-1923
5. Pavel, B., Brazdil, Soares, C.: A comparison of ranking methods for classification algorithm selection. In Proceedings of the 11th European Conference on Machine Learning (ECML-2000) Barcelona, Spain. Springer-Verlag. (2000) 63-74
6. Quinlan, J.R.: Combining instance-based and model-based learning. In Machine Learning: Proceedings of the Tenth International Conference, Amherst, Massachusetts. Morgan Kaufmann. (1993) 236-243
7. Quevedo, J.R., Bahamonde, A.: Aprendizaje de funciones usando inducción sobre clasificaciones discretas. Proceedings of CAEPIA'99-TTIA'99-VIII Conferencia de la Asociación Española para la Inteligencia Artificial – III Jornadas de Transferencia Tecnológica de Inteligencia Artificial, Murcia, Spain. (1999) 64-71
8. Fürnkranz, J.: Round Robin Classification. *Journal of Machine Learning Research*. (2002) 2:721-747
9. Kohavi, R., John, G., Long, R., Manley, D., Pfleger, K.: MLC++: A machine learning library in C++. IEEE Computer Society Press. In Proc. of the 6th International Conference on Tools with Artificial Intelligence. (1994) 740-743
10. Holte, R.C.: Very simple classification rules perform well on most commonly used datasets. *Machine Learning*. (1993) 11:63-91
11. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann. (1993)
12. Domingos, P.: Unifying Instance-Based and Rule-Based Induction. *Machine Learning*. (1996) 24:141-168
13. Cohen, W.W.: Fast Effective Rule Induction. Proceedings of the 12th International Conference on Machine Learning (ML95), Morgan Kaufmann, San Francisco. (1995) 115-123
14. Murthy, S.K., S. Kasif, S. Salzberg.: A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*. (1994) 2:1-33
15. Aha, D.W., Kibler, D., Albert, M.K.: Instance based learning algorithms. *Machine Learning*, Vol. 6. (1991) 37-66
16. Wilson, D., Martinez, T.: Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*. (1997) 6:1-34
17. Kohavi, R.: Wrappers for performance enhancement and oblivious decision graphs. Ph.D. thesis, Stanford University. (1995)

Evolutionary Approach to Overcome Initialization Parameters in Classification Problems

P. Isasi,F. Fernandez

Computer Science department
Universidad Carlos III de Madrid

Abstract The design of nearest neighbour classifiers is very dependent from some crucial parameters involved in learning, like the number of prototypes to use, the initial localization of these prototypes, and a smoothing parameter. These parameters have to be found by a trial and error process or by some automatic methods. In this work, an evolutionary approach based on Nearest Neighbour Classifier (ENNC), is described. Main property of this algorithm is that it does not require any of the above mentioned parameters. The algorithm is based on the evolution of a set of prototypes that can execute several operators in order to increase their quality in a local sense, and emerging a high classification accuracy for the whole classifier.

1 INTRODUCTION

Nearest Neighbour Classifiers are defined as the sort of classifiers that assign to each new unlabelled example, $v_r \in V = \{v_1, \dots, v_M\}$, the label of the nearest prototype, r_i , from a set, $C = \{r_1, \dots, r_N\}$, of N prototypes previously classified [DH73]. The set of different labels or classes is $S = \{s_1, \dots, s_L\}$.

The design of these classifiers is difficult, and rely in the way of defining the number of prototypes needed to achieve a good accuracy, as well as the initial set of prototypes used and, usually, a smoothing parameter. Many discussions about what is the right technique to use can be found in the literature. Some approaches based on clustering techniques are divided in two main steps. The first one is to cluster a set of unlabelled input data to obtain a reduced set of prototypes, for instance, with the LBG algorithm. The second step is to classify these prototypes basing on previous labelled examples and the nearest neighbour rule. Although this approach produces good results, it is obvious that to introduce information about the classification performance in the location of the prototypes seems to be needed to achieve a higher performance.

Neural networks approaches are also very common in the literature, like the LVQ algorithm [Koh84] and the works with radial basis functions. Some techniques try to introduce or to eliminate prototypes (or neurons) while designing the classifier following different heuristics, as the average quantization distortion or the accuracy in the classification. Other approaches try to define first the optimal size of the classifier and then to learn it using the previous value.

In this work, an evolutionary approach called Evolutionary Nearest Neighbour Classifier (ENNC), is introduced to dynamically define the number of prototypes of the classifier as well as the location of these prototypes.

2 THE ENNC ALGORITHM

2.1 Architecture

The system can be represented by a bidimensional matrix, where each row is associated to a region $r_i \in C$, and each column is associated to a class $s_j \in S$. Each position (i, j) of the matrix is a structure that contains features about the set of training examples that belongs to the region r_i and to the class s_j . These sets are called Region-Class Sets, and notated by V_{ij} . Furthermore, information about the elements that belong to a defined class (set VS_j), and that belong to a defined region (set VR_i) are stored. Last, some information can be obtained from the set V . Note that the structure does not store the whole region, class, and region-class sets, but only some features about them, as their size, or their centroid, that will be defined below. For all the sets, its membership function is given too.

Figure 1 shows the architecture used to keep all this information. Each of those nodes stores the features about the sets that represent. This structure is dynamic, in the sense that if any set is empty, the node is not created, solving some memory requirements in complex domains.

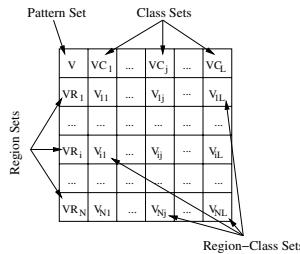


Figure 1. ENNC Architecture

From this architecture some important features are computed:

- $regions_{s_j}$ is the number of regions which prototype class is s_j .
- $expectation_{s_j}$ is the number of patterns that any prototype r_i of the class s_j is expected to correctly classify. This number is computed in a relative way, by the relationship among the number of patterns and the number of regions of each class s_j :

$$expectation_{s_j} = \frac{\|VS_j\|}{regions_{s_j}} \quad (1)$$

- $location_{r_i}$ is the location of the prototype r_i .
- $class_{r_i}$ is the class of the prototype r_i .
- $accuracy_{r_i}$ is the classification accuracy of the prototype. It is calculated as shown in equation 2, being $s_i = class_{r_i}$.

$$accuracy_{r_i} = \frac{\|V_{is_i}\|}{\|VR_i\|} \quad (2)$$

- $apportion_{r_i}$ is a relationship among the number of patterns correctly classified by this prototype and the number of patterns that it is suppose to classify:

$$apportion_{r_i} = \frac{\frac{\|V_{is_i}\|}{expectations_{s_i}}}{2} \quad (3)$$

- $quality_{r_i}$ is the quality of the prototype, calculated from equation 4. This value is a relationship among the accuracy of the prototypes, and its contribution to the classification of the whole input pattern set.

$$quality_{r_i} = \max(1, accuracy_{r_i} * apportion_{r_i}) \quad (4)$$

- $neighbours(r_i)$ is the set of regions that has a common border with the region r_i .

2.2 The Algorithm

The learning phase is an iterative process where the prototypes can execute several operators, once the features defined in section 2.1 have been computed. These operators are heuristics that allow the prototypes to change their location, to introduce new prototypes, etc. The algorithm begins with an initialization where the classifier is composed by only one prototype. After, the classifier evolves by executing, in a loop, all the designed operators, described below. These operators take advantage of some information gathered at the beginning of the loop.

Initialization One relevant feature of our method is the absolute elimination of initial conditions. These initials conditions are usually summarized in three: the number of prototypes, the initial set of prototypes and a smoothing parameter. The ENNC algorithm allows to learn without those parameters, given that the initial number of prototypes is always one, the initial location of that prototype is not relevant, and there are no learning parameters.

Mutation Operator The goal of this operator is to label each prototype with the most populate class in each region. Once the features are obtained, each prototype knows the number of patterns of each class located in its region, so the prototype changes, if needed, and becomes to the same class of the most abundant class of patterns in its region.

This way to get the main class is typically used when unsupervised learning is applied to supervised classification. In these works, the algorithms typically generate a set of clusters that takes into account only the distribution of the data. In a second phase, the clusters are labelled to the most populate class in the cluster. However, in this work, the supervision is included in each iteration of the algorithm, and not only in “a posteriori” phase. The formulation of this operator is defined in equation 5.

$$\forall r_i \in C, class_{r_i} = \arg_j \max \|V_{ij}\| \quad (5)$$

where $\|V_{ij}\|$ is the size of the set V_{ij} .

Reproduction Operator The goal of this operator is to introduce new prototypes in the classifier. The insertion of new prototypes is a decision that is taken by each prototype, in such a way that all of them contains in its region as much patterns as possible of the same class. The regions with patterns belonging to different classes, can create new regions containing the patterns of a different class to the class of the prototype.

Each prototype, r_i , of the class s_j , executes a roulette proportional to the number of elements of the set $V_{ij'}$ to which it represents. The possible results of the roulette are two. On one hand, if the resulting set $V_{ij'}$ is the set V_{ij} , i.e. $j = j'$, no reproduction is executed. On the other hand, if the resulting set $V_{ij'}$ is not the set V_{ij} , i.e. $j \neq j'$, the reproduction is executed, and a new region $r_{i'}$ is created to contain the patterns in $V_{ij'}$, that is renamed to be $V_{i'j'}$.

Fight Operator This operator provides the prototype the capability of getting patterns from other regions. Formally, this operator allows a prototype, r_i , to modify its sets V_{ij} from the sets $V_{i'j}$ of another prototype $r_{i'}$, for $i \neq i'$. This operator takes place in several steps:

1. Choose the prototype $r_{i'}$ against to fight proportionally to the difference between its quality and the quality of a neighbor r_i .
2. Decide whether to fight or not proportionally to the distance of their qualities:

$$P_{fight}(r_i, r_{i'}) = |quality_{r_i} - quality_{r_{i'}}| \quad (6)$$

3. If prototype r_i decides to fight with prototype $r_{i'}$, there are two possibilities. Given $class_{r_i} = s_i$, and $class_{r_{i'}} = s_{i'}$:

- If $s_i \neq s_{i'}$ (cooperation). Both prototypes belongs to different classes. In this case, the prototype $r_{i'}$ will give to the prototype r_i the patterns of the class s_i .
- If $s_i = s_{i'}$ (competition). In this case, patterns can be transferred from set $V_{i's_i}$ to the set $V_{i's_i}$, or vice versa, depending on who wins the fight. The amount of patterns that are transferred depends on a probability proportional to the qualities of both prototypes.

Move Operator The move operation implies to relocate each prototype in the best expected place. So each prototype, r_i , decides to move to the centroid of the set V_{ij} , being $class_{r_i} = j$, i.e. it takes as centroid the centroid of the set V_{ij} of its class, as shown in equation 7.

$$location_{r_i} = centroid_{V_{is_i}} \quad (7)$$

where $s_i = class_{r_i}$.

This operation based in the second step of Lloyd iteration allows to make a local optimization of the classifier, increasing its performance.

Die Operator The probability for a prototype of being eliminated is 1 minus the double of the quality, as defined in equation 8. In that case, successful prototypes will survive with probability of 1, while useless prototypes with quality less than 0.5 might die. A

wide range of heuristics about how to reduce the number of prototypes in a classifier can be found in the bibliography.

$$P_{die}(r_i) = \begin{cases} 0, & \text{when } quality_{r_i} > 0.5 \\ 1 - 2 * quality_{r_i}, & \text{when } quality_{r_i} \leq 0.5 \end{cases} \quad (8)$$

3 Experiments

This section shows the experiments performed over two different domains, where the ENNC algorithm is compared with other classification algorithms, as C4.5 [Qui93], decision rules [FW98], Naive Bayes [DH73], and IBK [AK91], for values of $k = 1$ and $k = 3$, executed in WEKA [WF00].

3.1 Spiral Data Set

Spiral data set is typically used in the bibliography as a challenge data set, where two interlaced spirals must be correctly classified, as it is shown in figure 2. Examples in the same spiral belong to the same class, and there are 500 examples in each spiral. From the whole set, 900 examples was used for learning, and 100 for test.

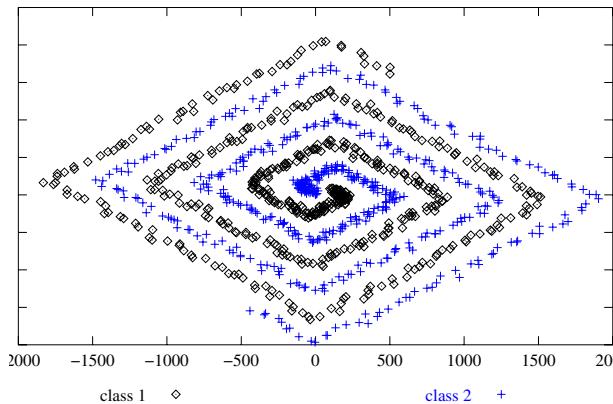


Figure2. Spiral Data Set

Given that the ENNC algorithm has an important stochastic component, the experiment has been executed 20 times, each of them of 300 iterations. The results are summarized in Table 1 where, for each execution, the iteration when the winner classifier was evolved is shown, as well as the number of prototypes of the classifier and the classification error for training and test sets. The table shows that the 99% of success is typically achieved on training, while success on test set ranges from 94% to 98%, what means a maximum difference of 4 errors over the test set. The number of prototypes of the classifiers achieved in each iteration is close to 80.

Iteration	Prototypes	Training (%)	Test (%)	Iteration	Prototypes	Training (%)	Test (%)
289	82	99,33	97,0	281	73	98,9	96,0
158	75	99,0	96,0	300	84	99,44	97,0
174	81	99,11	95,0	123	84	99,56	96,0
267	81	99,33	96,0	81	79	99,33	94,0
282	81	99,44	97,0	260	82	99,44	96,0
260	78	99,11	97,0	131	77	98,67	95,0
170	79	99,56	96,0	81	81	99,33	98,0
182	82	99,33	97,0	97	79	99,33	96,0
120	79	98,67	95,0	299	78	99,56	96,0
291	84	99,78	97,0	179	75	99,0	97,0

Table1. Results of different executions of the ENNC algorithm over spiral data set.

Table 2 shows comparative results of different classifiers over this domain, showing the best and the lowest results of the ENNC algorithm over the 20 executions. The same training and test sets was used with all the classifier systems. Results for C4.5, decision rules (PART) and Naive Bayes are poor, and they are far from the solutions obtained by ENNC. However, IBK achieves the best results of a 100% of success as is expected in a domain with the examples of different classes very separated, as shown in Figure 2.

C4.5	PART	Naive Bayes	IBK ($k = 1$)	IBK ($k = 3$)	ENNC (best)	ENNC (worst)
62	56	50	100	100	98	94

Table2. Comparative results over the spiral data set.

Figure 3 shows the evolution of one execution of the ENNC algorithm, with the number of prototypes and success obtained in each iteration over the training and the test sets. Figure shows that in only 25 iterations, the number of prototypes grows up to 64, obtaining a success of the 91,34% over the training set, and of the 85% over the test set. However, the number of prototypes still grows up to the range 70-80, when the search of better solutions go on, obtaining the best solution in iteration 176, with a success of 94% over the test set.

3.2 Uniformly Distributed Data

In this experiment, the ENNC algorithm has been executed over the data set shown in the Figure 4, as defined in [Bur91]. As in that work, data set contains 2000 instances of each class, using 500 for training and 1500 for test.

As in previous case, the ENNC algorithm is executed 20 times, with a maximum length of 300 iterations. The algorithm achieves a solution of only 2 prototypes in 16 of the 20 executions, achieving a 98.6% over the training and test sets. Other executions achieve better results for classifiers with more prototypes, arriving up to the 98.73% with 12 prototypes. Table 3 shows some comparative results with previous approaches, as LVQ [Koh84] and LVQ-PNN [Bur91] of different sizes, and a different version of

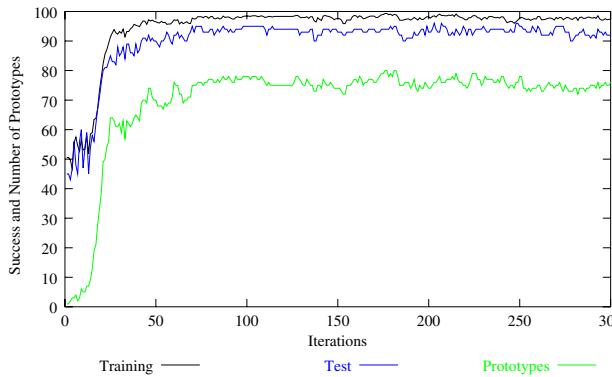


Figure3. Evolution of one execution of the ENNC algorithm over spiral data set.

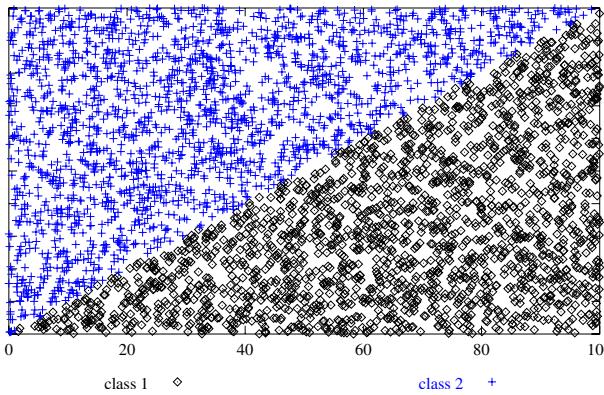


Figure4. Uniformly Distributed Data.

PNN [MTS00] that automatically defines the number of neurons. Furthermore, C4.5, PART, Naive Bayes and IBK (experimented with the default values defined in WEKA) are compared too.

The table shows that the worst result obtained by ENNC for two prototypes (98.6%) is very similar to the best solutions obtained with other different approaches, while the best solution obtained, with a 98.93% of success, is better than the rest of solutions reported.

4 CONCLUSIONS

The algorithm presented in this work is an evolutionary approach to solve the problem of finding a set of prototypes that are able to correctly classify the examples of a domain, following a 1-nearest neighbour approach. The main advantage of this method are, on

Algorithm	Data	Success
LVQ	10 prototypes	96.13
LVQ	100 prototypes	97.77
LVQ-PNN	10 Neurons	96.99
LVQ-PNN	100 Neurons	98.17
PNN (Mao)	8 Neurons	98.85
C4.5	-	96.97
PART	-	96.6
Naive Bayes	-	96.77
IBK	$k = 1$	98.7
IBK	$k = 3$	98.93
ENNC (best)	2 Prototypes	98.93
ENNC (worst)	2 Prototypes	98.6

Table3. Comparative results over Uniformly Distributed Data.

one hand, that it is able to achieve a high accuracy in most of the domains where it has been tested, even compared with other techniques from the literature. On the other hand, the achievement of these good results is done without the user defines the initial conditions for learning.

References

- [AK91] D. Aha and K. Kibler. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.
- [Bur91] Pietro Burrascano. Learning vector quantization for the probabilistic neural network. *IEEE Transactions on Neural Networks*, 2(4):458–461, July 1991.
- [DH73] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. John Wiley And Sons, 1973.
- [FW98] Eibe Frank and Ian H. Witten. Generating accurate rule sets without global optimization. In *Proceedings of the Fifteenth International Conference on Machine Learning*, 1998.
- [Koh84] Teuvo Kohonen. *Self-Organization and Associative Memory*. Springer, Berlin, Heidelberg, 1984. 3rd ed. 1989.
- [MTS00] K. Z. Mao, K.-C. Tan, and W. Ser. Probabilistic neural-network structure determination for pattern classification. *IEEE Transactions on Neural Networks*, 11(4):1009–1016, July 2000.
- [Qui93] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
- [WF00] I. H. Witten and E. Frank. *Data Mining. Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2000.

Text Categorisation Using a Partial-Matching Strategy *

J. Ranilla¹, I. Díaz¹, J. Fernández¹

Artificial Intelligence Center.

Campus de Viesques, E-33271 Gijón, Spain.

ir@aic.uniovi.es

Abstract. In this paper a family of rule learners whose application is carried out according to a partial-matching criterion based on different purity measures is presented. The behavior of these rule learners is tested by solving a Text Categorisation problem. To illustrate the advantages of each learner, the MDL-based method of C4-5 is replaced by a pruning process whose performance relies on an estimation of the quality of the rules. Empirical results show that, in general, inducing partial-matching rules yields more compact rule sets without degrading performance measured in terms of microaveraged F_1 which is one of the most common performance measure in Information Retrieval tasks. The experiments show that there are some purity measures which produces a number of rules significantly lesser than C4-5 meanwhile the performance measured with F_1 is not degraded.

1 Introduction

The wide availability of text documents in electronic form has carried out to develop methods automatically classify large collections of text files. This process is known in *Information Retrieval*(IR) as *Text Categorisation*(TC).

Formally, TC consists of determining whether a document d_i (from a set of documents D) belongs or not to a category c_j (from a set of categories C), consistently with the knowledge of the correct categories for a set of train documents [16]. That task is usually induced by an automatic classifier from a set of correctly labeled examples using statistical [21] or machine learning (ML) algorithms [16]. Then, this classifier is used to assign each new document to one or more of the categories.

In this paper we explore the use of a partial matching strategy when using a set of classification rules in a ML classifier applied to TC. In fact, given a new case (a new document), to return a class label we can use the conclusion of the rule whose conditions are completely fulfilled by case values, or instead we can follow the rule whose conditions are the nearest to case values. Usually we refer to these procedures as *full* and *partial matching* respectively. The main

* The research reported in this paper has been supported in part under MCyT and Feder grant TIC2001-3579

advantage of the partial matching strategy presented here is the possibility of adapting the decision areas of rules to regions with wavy frontiers.

The quantification of the performance of different systems in TC is usually measured in terms of two measures of effectiveness called *precision* and *recall* (see [16] for details) but given the trade off existing between both measures [16], it makes no sense to consider these parameters on their own. Several ways to combine their values have been proposed [19], being the function F_1 among the most popular because it gives the same relevance to both. Hence, we use this performance measure to check the goodness of the systems.

This work is divided in the following sections. In Section 2 it is presented a brief description about partial-matching rules. The alternative measures for test selection used in this paper as well as detail the algorithm template based on these measures are shown in Section 3. In Section 4 some considerations about the obtaining of the training set are presented. To close the paper, we report the results found when comparing the partial-matching learners amongst their full matcher counterpart, C4.5-rules.

2 Partial-matching rules

In partial-matching environments, a case is classified following a nearest-neighbor principle, so it is needed a function to compute distances between rules and cases. For this purpose it is used a HEOM-like [20] metric defined as:

$$\text{dist}(R, c) := \sqrt{\sum_{a=1}^m \text{diff}_a^2(R_a, c_a)} \quad (1)$$

where m is the number of attributes describing the examples, R_a is the condition on attribute a in rule R and c_a is the actual value of attribute a in case c . diff is defined as

$$\text{diff}_a(R_a, c_a) = \begin{cases} \text{overlap}(R_a, c_a), & \text{if } a \text{ is symbolic} \\ \text{eucl}(R_a, c_a), & \text{if } a \text{ is numeric} \end{cases} \quad (2)$$

The *overlap* metric [20] yields a difference of 1 when the symbolic value of the attribute is different than the value mentioned in the condition of the rule, and 0 otherwise. For numerical attributes we use

$$\text{eucl}(R_a, c_a) = \begin{cases} 0, & \text{if } c_a \text{ fulfills } R_a \\ \frac{|c_a - \text{value}_a|}{4\sigma_a}, & \text{otherwise} \end{cases} \quad (3)$$

where R_a may be of the form “ $a \leq \text{value}_a$ ”, or “ $a > \text{value}_a$ ”, or “ $a \in I$ ”, and value_a is the nearest border of interval I to c_a .

3 Purity measures and the Learner

On the basis of the framework of Quinlan C4.5-rules [11], our work can be placed into two categories: algorithms that *modify the test search* by means of

new selection measures such as the Laplace correction, Gini index [3] or Impurity Level [12, 13] instead of using Shannon's information-based measures. The other category is composed of algorithms using *alternative data structures*, specifically rules, obtained by new pruning methods [9] applied to induced trees.

The core for building a partial-matching rules learner is a *purity measure* capable of testing whether a selected group of training examples is coherent enough to somehow become a classification rule. To define these measures formally, let us consider a subset of training examples E (the whole set is TS) and a class C .

For ease of reference, the number of examples in E of class C is called e^+ , (positive examples), e^- will be the number of examples in E of a class different than C (negative examples), p the success probability, i.e. $p = \frac{e^+}{e^+ + e^-}$, $n = e^+ + e^-$ and $\#c$ the number of classes in training set TS . In this work, six different purity measures are studied. Next table shows those purity measures defined in terms of p , e^+ and e^- .

$$\begin{array}{ll} \text{Laplace} & \text{LAP} = \frac{e^+ + 1}{e^+ + e^- + \#c}; \\ \text{Furnkranz} & F(p) = e^+ - e^-; \end{array} \quad \begin{array}{ll} \text{Dietter G} & G(p) = 2\sqrt{p(1-p)}; \\ \text{Gini index} & \text{Gini} = 1 - (p^2 + (1-p)^2); \end{array}$$

All these measures were previously used in ML environment. In fact, Laplace measure was used in CN2 [4] and RISE [6], the G index was proposed by Dietrich et al. in [5] as well as Fürnkranz and Widmer proposed their measure in [7]. The Gini index is the heuristic of CART [3].

An heuristic that achieves good results is the *Impurity Level (IL)* [13]. It explicitly takes into account not only the success probability p , but also the difficulty of attaining that amount of examples of class C . The definition of *IL* is based on the criterion used by Aha in [1] for selecting a set of representative instances from a set of training examples: If a rule is applied n times and m successes, the confident interval of its success percentage is given by the formula of Spiegel [17] which takes into account the percentage of success (m/n). The distribution of the examples over the categories is included by the *canonical rule*, which is the rule that do not have any antecedent and has the category as consequent. Then *IL* is defined as the overlapping degree between the confident interval of p when predicting class C in subset TS (noted $[CI_l(TS, C), CI_h(TS, C)]$) and the confident interval of its correspondent canonical rule ($[CI_l(E, C), CI_h(E, C)]$) which is

$$IL(R) = \frac{CI_h(TS, C) - CI_l(E, C)}{CI_h(E, C) - CI_l(E, C)} \times 100. \quad (4)$$

The idea of estimating a rule's accuracy using a relative measure with respect to the accuracy of a default rule is also presented in [18] where it is used the *Weighted Relative Accuracy* (WRAcc) as a quality estimator of a rule; for a given rule this measure is defined as

$$\frac{n_R}{|E|}(p(R) - p(C)), \quad (5)$$

where n_R is the proportion of training examples covered by the rule R and C the canonical rule of class R .

The template used in the experiments is now described. This is a sketch of C4.5 in which the information gain ratio is substituted by one of the measures listed above to build the tree.

Our purity-rules differs in the purity measure used in building the trees but the main difference from the master full-matching learner is the rule generation process which is based on the method described in [12]. First, purity-rules tries to clean the antecedents or conditions list from each rule (*qualification*). Then, the algorithm *selects* the most promising subset of classification rules.

The qualification tries to drop the redundant or unnecessary conditions. In order to reduce the number of possibilities to consider, the conditions are first ordered according to the purity heuristic.

Once the conditions have been ordered, the process starts with an empty set of descriptions and progressively adds partial descriptions of the original rule being qualified.

Only those descriptions with a success probability higher than an *acceptance threshold* are saved. Once a set of partial descriptions has been obtained, the best of them is selected, namely R_{best} ; more partial descriptions are added by sequentially deleting each antecedent, from the penultimate to the first one. Only descriptions with higher success probability than R_{best} are added.

Rule qualification is followed by a selection process aimed at reducing the total number of rules induced so far, since compact rule sets are more comprehensible and they usually provide more accurate generalizations. The selection starts detecting and deleting those rules classifying too specific peculiarities of data caused by noisy examples. This procedure deletes rules whose success probability is lower than a noise threshold which is determined according to the one that yields the better subset of rules, in terms of accuracy.

At this stage rules compete to classify examples, being applied on the basis of a minimum distance criterion. The resulting subset is revised to eliminate useless rules still undeleted in prior steps. Each rule is considered useless if there is no accuracy loss when eliminated.

4 Feature Reduction Task

All the algorithms applied in TC need the documents to be represented in a way suitable for the inducing of the classifier. The most widely used representation consists of identifying each document with a numerical vector whose components are measures of the importance in the document of the different words occurring in the corpus. This representation is known as *vector of words* or *bag of words* (see [15]). Each component of that vector can be determined by different ways. We will adopt the absolute frequency of the word in the document in our experiments.

Usually, if all the words appearing in the corpus are considered for the representation, the resulting vectors have high dimensionality (of the order of tens of thousands of components) and this makes impossible the application of most of the usual decision algorithms. On the other hand, it is widely accepted among

the IR researchers [15] that only a small percentage of the words are really useful for the classification task. For this purpose, the importance of each word in the whole collection is evaluated and the most relevant words are selected. Several measures can be used to determine the importance of a word. In this work two of the most common measures are used to reduce the dimensionality. The first one is the number of occurrences of a word *tf* in the corpus. The second one, *information gain (ig)*, takes into account the distribution of a word on the different categories in which the corpus is initially divided (see, for instance [21]).

Finally we order the words with respect to such measures from the highest to the lowest and select the first 10% which is chosen according to [10].

5 Experimental results

In this section we present the scores reached by learners of the type purity-rules, which apply rules following a partial-matching strategy, in comparison with those obtained by *C4.5-rules* (release 8), which uses a full-matching strategy.

To carry out the experiments we have chosen the Reuters-21578 corpus [14]. This set of documents is very well-known to the IR community and contains short news related to economy published by Reuters during the year 1987. The distribution of Reuters collection into categories is quite unbalanced so that we only work with the 10 huger categories. We have used the split of the corpus into a set of train documents and a set of test documents which was originally made by Lewis and then slightly modified by Apté (cf. [2, 8]).

First of all we have proved that the behavior of each learner does not depend on the feature reduction measure (namely, *tf* and *ig*). To check that fact and taking into account that normality can not be supposed for these data, the non-parametric paired Wilcoxon signed rank test is used to check if two populations have the same distribution at a confidence level of 95%. The test shows non-significant differences between the number of rules and the F_1 obtained when the feature reduction is performed with either *tf* or *ig*.

Table 1 shows the number of rules for every learner and data set. The scores obtained with respect to the size of the induced rule set exhibit quite different behavior among learners. In fact, the behavior of each learner depends on the data set but it is more or less independent of the feature reduction measure. To check if these hypothesis can be generalized, an statistical analysis is performed. Hence, 6 different tests are performed in order to examine if some of the partial-matching learners uses significantly less rules than *C4.5-rules*. The results of these tests show that the number of rules for *F*, *Lap*, *IL* and *WRAcc* is significantly lesser than that of *C4.5-rules*, the number of rules used by *G* is significantly higher than that of *C4.5-rules* while there are not significant differences in the case of *Gini* learner. These results are valid when feature reduction is performed with both *tf* and *ig* measures.

However, reducing the number of rules of a certain learner does not have to cause a descent in the performance of that learner, that is, it is not acceptable

Table 1. Number of rules for every learner, data set and feature reduction measure

Set	<i>C4.5-rules</i>		<i>Lap</i>		<i>F</i>		<i>Gini</i>		<i>G</i>		<i>WRAcc</i>		<i>IL</i>	
	tf	ig	tf	ig	tf	ig	tf	ig	tf	ig	tf	ig	tf	ig
acq	36	44	33	28	2	2	38	43	3	4	43	50	22	34
corn	8	7	5	6	2	2	9	12	2	2	16	16	2	2
crude	18	25	14	14	2	2	18	27	3	3	25	28	8	11
earn	43	32	19	22	2	2	30	31	2	2	35	34	16	17
grain	14	14	2	11	2	2	19	15	3	3	18	26	3	8
interest	28	27	15	17	16	19	28	23	4	5	18	20	21	8
money-fx	25	25	17	16	2	9	26	22	6	6	22	24	10	14
ship	19	23	9	11	2	2	18	20	2	8	21	20	14	9
trade	27	29	18	18	3	3	24	28	3	3	25	25	3	3
wheat	7	7	5	5	2	2	12	8	2	3	8	12	5	4
average	23	23	14	15	4	5	22	23	3	4	23	26	10	11

that the cost of reducing the number of rules is a significant reduction in the F_1 of the method.

Hence, Wilcoxon test is again used to check whether there are significant differences between the F_1 of *C4.5-rules* and that of the partial-matching learners with a size of rule set significantly lesser. The results of these tests are successful for partial-matching learners in the sense that there are not significant differences between the F_1 of these learners and *C4.5-rules*. Therefore, it is possible to conclude that there are four partial-matching learners, namely, *F*, *Lap*, *IL* and *WRAcc* which use significantly lesser rules than *C4.5-rules* keeping their performance. Tables 2 and 3 shows the F_1 obtained by the different learners when feature reduction is performed with *tf* and *ig* measures respectively.

Table 2. F_1 value for the different partial-matching strategies when feature reduction is performed with *tf*

Set	<i>C4.5-rules</i>	<i>Lap</i>	<i>F</i>	<i>Gini</i>	<i>G</i>	<i>WRAcc</i>	<i>IL</i>
acq	91.6%	89.2%	83.8%	87.7%	89.7%	77.1%	87.9%
corn	88.0%	82.6%	89.3%	83.7%	81.2%	89.7%	91.1%
crude	78.0%	75.3%	74.2%	79.4%	82.2%	72.7%	79.3%
earn	96.6%	95.8%	94.6%	96.4%	95.7%	92.4%	97.1%
grain	91.0%	90.2%	93.2%	88.3%	84.8%	84.2%	90.3%
interest	54.4%	58.0%	58.9%	61.3%	63.6%	51.2%	67.0%
money-fx	64.3%	64.5%	64.3%	62.5%	60.9%	62.6%	68.9%
ship	78.7%	71.5%	72.8%	80.5%	71.3%	50.4%	80.7%
trade	56.1%	68.5%	57.0%	69.3%	65.6%	62.0%	67.2%
wheat	89.6%	90.3%	80.6%	86.1%	89.2%	89.6%	86.8%
average	78.8%	78.6%	76.9%	79.5%	78.4%	73.2%	81.6%

Table 3. F_1 value for the different partial-matching strategies when feature reduction is performed with *ig*

Set	<i>C4.5-rules</i>	<i>Lap</i>	<i>F</i>	<i>Gini</i>	<i>G</i>	<i>WRAcc</i>	<i>IL</i>
acq	91.2%	86.9%	83.8%	88.0%	89.2%	74.6%	90.8%
corn	87.6%	89.5%	89.7%	89.8%	86.0%	89.7%	91.1%
crude	78.0%	74.0%	74.2%	77.3%	76.7%	72.2%	81.5%
earn	96.8%	96.6%	94.5%	96.1%	96.1%	92.4%	97.0%
grain	91.9%	87.9%	93.2%	85.9%	85.9%	85.3%	86.7%
interest	60.2%	60.2%	59.0%	68.1%	51.4%	54.4%	62.3%
money-fx	66.2%	64.3%	45.7%	71.7%	64.2%	65.3%	69.1%
ship	70.3%	71.7%	74.2%	75.5%	72.0%	71.7%	82.8%
trade	64.3%	70.0%	57.0%	67.6%	66.7%	62.0%	67.2%
wheat	90.1%	85.1%	80.6%	87.0%	84.2%	89.0%	84.3%
average	79.7%	78.6%	75.2%	80.7%	77.2%	75.7%	81.3%

With regard to the percentage of cases classified by rules at distance greater than zero (uncovered cases), it depends on the problem itself, so we have noticed large differences among problems. For example, there is an average percentage of 52.26% of uncovered cases for the *ship* problem, *WRAcc* and *tf* feature reduction measure, while it is 0.00% for the *corn* problem, *Gini* measure and *ig* feature reduction measure.

6 Concluding remarks

We have presented a family of rule learners whose application is carried out according to a partial-matching mechanism based on minimal distance from rule conditions and case values. All the learners have a common template based on Quinlan's algorithm *C4.5-rules*; instead of using the information gain ratio to construct decision trees, our learners use different purity measures. The same purity measures are actively used to obtain a rule set using a pruning process according to a partial-matching criterion instead of the MDL-based method.

The purity measures used can be described as heuristics capable of quantifying the classification quality of a rule. Some of them are complex heuristics, such as the Laplace correction, the Gini index or the *IL*. However, all our purity measures can be computed with simple arithmetic expressions.

A total of seven learners thus built were compared, together with *C4.5-rules*, with regard to their F_1 and the size of their induced rule sets. The scores show that the partial-matching learner built with the assistance of the *WRAcc* gives the best results in rule-set size measures. However, *IL* produces the best results of the performance measured in terms of F_1 , keeping good results (but not the best) with regard to the rule-size.

References

- [1] D. W. Aha. *A Study of Instance-based Algorithms for Supervised Learning Tasks: Mathematical, Empirical, and Psychological Evaluations*. PhD thesis, University of California at Irvine, 1990.
- [2] C. Apte, F. Damerau, and S. Weiss. Automated learning of decision rules for text categorization. *Information Systems*, 12(3):233–251, 1994.
- [3] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International Group, Belmont, CA, 1984.
- [4] P. Clark and T. Niblett. The cn2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.
- [5] T. Dietterich, M. Kearns, and Y. Mansour. Applying the weak learning framework to understand and improve c4.5. In *Proc. 13th International Conference on Machine Learning*, pages 96–104. Morgan Kaufmann, 1996.
- [6] P. Domingos. Unifying instance-based and rule-based induction. *Machine Learning*, 24:141–168, 1996.
- [7] J. Fürnkranz and G. Widmer. Incremental reduced error pruning. In *International Conference on Machine Learning*, pages 70–77, 1994.
- [8] D. D. Lewis and M. Ringuelette. A comparison of two learning algorithms for text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93, Las Vegas, US, 1994.
- [9] O. Luaces, J. Alonso, E. de la Cal, J. Ranilla, and A. Bahamonde. Machine learning usefulness relies on accuracy and self-maintenance. In Springer-Verlag, editor, *Lecture Notes in Artificial Intelligence. Proc. of the 11th IEA & AIE*, volume 1416, pages 448–457, 1998.
- [10] E. Montañés, J. Fernández, I. Díaz, E. F. Combarro, and J. Ranilla. Text categorisation with support vector machines and feature reduction. In *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation CIMCA2003*, 2003.
- [11] J. R. Quinlan. Constructing decision tree in c4.5. In *Programs of Machine Learning*, pages 17–26. Morgan Kaufman, 1993.
- [12] J. Ranilla and A. Bahamonde. Fan: Finding accurate inductions. *International Journal of Human Computer Studies*, 56(4):445–474, 2002.
- [13] J. Ranilla, O. Luaces, and A. Bahamonde. A heuristic for learning decision trees and pruning them into classification rules. *AICOM (Artificial Intelligence Communication)*, 16(2):in press, 2003.
- [14] Reuters. Reuters collection. <http://www.research.att.com/lewis/reuters21578.html>.
- [15] G. Salton and M. J. McGill. *An introduction to modern information retrieval*. McGraw-Hill, 1983.
- [16] F. Sebastiani. Machine learning in automated text categorisation. *ACM Computing Survey*, 34(1), 2002.
- [17] M. R. Spiegel. *Estadística*. McGraw-Hill, 1970.
- [18] L. Todorovski, P. Flach, and N. Lavrač. Predictive performance of weighted relative accuracy. In *4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD2000)*, pages 255–264. Springer-Verlag, 2000.
- [19] C. J. Van-Rijsbergen. *Information retrieval*. Butterworths, 2 edition, 1979.
- [20] D. R. Wilson and T. R. Martínez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6(1):1–34, 1997.
- [21] T. Yang and J. P. Pedersen. Feature selection in statistical learning of text categorization. In *Proceedings of the 14th Int. Conf. on Machine Learning*, pages 412–420, 1997.

A new learning method for single layer neural networks based on a regularized cost function

Juan A. Suárez-Romero, Oscar Fontenla-Romero,
Bertha Guijarro-Berdiñas, and Amparo Alonso-Betanzos

Department of Computer Science, University of A Coruña,
Campus de Elviña s/n, 15071 A Coruña, Spain,
`{ciamparo, oscarfon, cibertha, ja}@udc.es`,
WWW home page: <http://www.dc.fi.udc.es/lidia>

Abstract. In this work, a new supervised learning method for single layer neural networks based on a regularized cost function is presented. This method obtains the optimal weights and biases by solving a system of linear equations and therefore it is always guaranteed the global optimum solution. In order to verify the soundness of the proposed learning algorithm and to analyze the effect of the regularization term, two simulations, one for a classification problem and another for a regression problem, were performed. The obtained results demonstrated the validity of the method.

1 Introduction

The objective of neural network learning is to build a statistical model of the process which generates a finite number of observable data. The network will try to map a, in general, nonlinear function between the inputs and the outputs of the system. However, there are usually two main situations that negatively affect the generalization capability of a neural network: 1) only a small number of samples are available, and 2) the data are affected by noise. In the first case, the neural network will tend to memorize the training data due to its higher nonlinear capability (overfitting problem). In the second case, the network could precisely fit the data, including noise, instead of the intrinsic model that generated the noiseless original data.

In order to mitigate these problems, regularization theory has been widely used [1, 2]. This technique allows to establish a commitment solution between the complexity of the mapping and a good fitting to the training data. Among the proposed regularization methods, weight decay [3] is one of the most employed for its simplicity and power. This method is used in this paper to propose a new regularized cost function for single layer neural networks. As it will be shown, in the next section, the presented cost function will allow to train the neural network using an analytical method that it is much simpler than the current iterative algorithms.

2 Algorithm for regularized learning

The architecture used in this work is presented in figure 1. The network is composed of a set of weights w_{ji} , $j = 1, \dots, J$, $i = 1, \dots, I$ and biases b_j , where I and J are the number of inputs and outputs of the network. The input of the network is formed by the variables x_{1s}, \dots, x_{Is} , $s = 1, \dots, S$, where S is the number of training samples. The j -th output of the network for a sample s , y_{js} , is computed as $y_{js} = f_j(z_{js})$, $j = 1, \dots, J$, where $z_{js} = \sum_{i=1}^I w_{ji}x_{is} + b_j$ and f_j is a nonlinear activation function. The learning of a single layer neural network

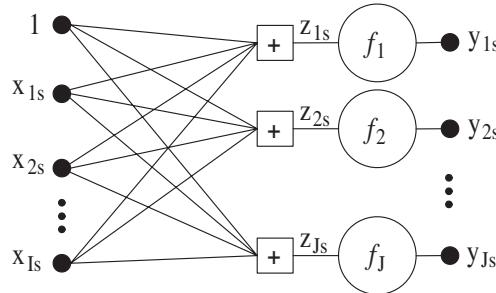


Fig. 1. Single layer neural networks

with J outputs can be decomposed in J different minimization problems, one for each output, as the weights of an output j are not related with the weights of the other outputs. Therefore, the results presented in this section deal with only one of these problems (for a fixed j).

The standard methods used for the supervised learning, with regularization, of feedforward neural networks employ, usually, the cost function in equation (1).

$$C_j^{(1)} = L_j + \alpha R_j \quad (1)$$

where α is a positive number. This cost function is composed of a) a loss function defined, commonly, as the quadratic error, i.e.,

$$L_j = \sum_{s=1}^S \varepsilon_{js}^2 = \sum_{s=1}^S (d_{js} - y_{js})^2, \quad (2)$$

where d_{js} is the desired output for the y_{js} output; and b) a regularization term $R_j = \sum_{i=1}^I w_{ji}^2 + b_j^2$ called *weight decay*. The first term (L_j) enforces closeness to the data and the second term (R_j) is introduced to control the smoothness properties of the mapping function. The regularization parameter (α) controls the tradeoff between both terms. Alternatively, in this work the following cost function is proposed:

$$C_i^{(2)} = \bar{L}_j + \alpha R_j, \quad (3)$$

where the regularization term (R_j) is the same as in equation (1) but the following alternative loss function is used:

$$\bar{L}_j = \sum_{s=1}^S (f'_j(\bar{d}_{js})\bar{\varepsilon}_{js})^2 = \sum_{s=1}^S (f'_j(\bar{d}_{js})(\bar{d}_{js} - z_{js}))^2, \quad (4)$$

where $\bar{d}_{js} = f_j^{-1}(d_{js})$.

It was shown in [4] that the optimum of this alternatively loss function is the same (up to first order of a Taylor serie) as the one of the initial loss function L_j . Therefore, the optimum of the cost function $C_j^{(2)}$ is the same as the optimum of the initial cost function $C_j^{(1)}$. However, the advantage of using the function $C_j^{(2)}$ is that the weights and bias are not arguments of the nonlinear function (f_j) and thus, the optimum of this cost function can be obtained easier than the optimum of $C_j^{(1)}$. Also, it is guaranteed that there is a unique optimum (the minimum of the hyperparabola). This can be easily observed in the final expression of the proposed regularized cost function:

$$C_j^{(2)} = \sum_{s=1}^S \left(f'_j(\bar{d}_{js}) \left(f_j^{-1}(d_{js}) - b_j - \sum_{i=1}^I w_{ji}x_{is} \right) \right)^2 + \alpha \left(\sum_{i=1}^I w_{ji}^2 + b_j^2 \right). \quad (5)$$

The optimal weights and bias of the alternative cost function in equation (5) can be obtained by deriving it with respect to the weights and bias of the network and equating the partial derivatives to zero:

$$\begin{aligned} \frac{\partial C_j^{(2)}}{\partial w_{jp}} &= -2 \sum_{s=1}^S \left(f'_j(\bar{d}_{js}) \left(f_j^{-1}(d_{js}) - b_j - \sum_{i=1}^I w_{ji}x_{is} \right) \right) x_{ps} f'_j(\bar{d}_{js}) \\ &\quad + 2\alpha w_{jp} = 0; \quad p = 1, 2, \dots, I, \\ \frac{\partial C_j^{(2)}}{\partial b_j} &= -2 \sum_{s=1}^S \left(f'_j(\bar{d}_{js}) \left(f_j^{-1}(d_{js}) - b_j - \sum_{i=1}^I w_{ji}x_{is} \right) \right) f'_j(\bar{d}_{js}) \\ &\quad + 2\alpha b_j = 0. \end{aligned} \quad (6)$$

The previous system of equations can be rewritten as follows:

$$\begin{aligned} \sum_{s=1}^S f_j'^2(\bar{d}_{js}) f_j^{-1}(d_{js}) x_{ps} &= b_j \sum_{s=1}^S f_j'^2(\bar{d}_{js}) x_{ps} + \sum_{i=1}^I w_{ji} \sum_{s=1}^S f_j'^2(\bar{d}_{js}) x_{ps} x_{is} + \alpha w_{jp}, \\ p &= 1, 2, \dots, I, \\ \sum_{s=1}^S f_j'^2(\bar{d}_{js}) f_j^{-1}(d_{js}) &= \left(\alpha + \sum_{s=1}^S f_j'^2(\bar{d}_{js}) \right) b_j + \sum_{i=1}^I w_{ji} \sum_{s=1}^S f_j'^2(\bar{d}_{js}) x_{is} \end{aligned} \quad (7)$$

The system of linear equations in (7) contains $(I + 1) \times (I + 1)$ equations and variables. Therefore it has an unique solution (except for degenerate systems) that it is the global optimum of the cost function in (5).

The proposed method, based on the linear system of equations in (7), has two additional advantages. The first one is evident and it is related with the computational cost of the method. In this case, the optimum can be obtained in an analytical way instead of using an iterative algorithm; therefore the time required to obtain the solution is much lesser. The second advantage is that the learning is incremental. This can be easily observed in (7) where the variables of the system of equations (b_j and w_{ji}) are weighted for a sum of terms over the sample index (s). Therefore, due to the associative and commutative properties of the sum operator the learning can be increased storing only the current coefficients of the system of equations and not the samples used in a previous training.

3 Experimental Results

In this section, the performance of the proposed learning method is evaluated. Also, the influence of the regularization term on the test error, as the size of the training data increases, is checked. This study was made using two kinds of data examples sets. The first case corresponds to a classification problem, while the second is a regression one. In both cases, the desired outputs were normalized to the interval [0.05, 0.95]. Also, the regularization parameter α has been constrained to the interval [0, 1]. Due to the normalization of the desired signal and the small number of training data, as it is expected that the neural network can obtain an acceptable solution, we may assume that the R_j term should be much greater than the \bar{L}_j term in the equation (3). Therefore, in order to maintain an equilibrium between R_j and \bar{L}_j , α values above 1 should not be needed. The neural network described in figure 1 and the learning method described by the system of equations in (7) were used. The transfer functions employed in all the neurons was the logistic function defined as

$$f(x) = \frac{1}{1 + \exp(x)}.$$

3.1 The Intrusion Detection problem

Our first example uses the Intrusion Detection data from the KDD'99 Classifier Learning Contest¹. This is a two-class classification problem that consists on distinguishing between legitimate and illegitimate connections (attacks) in a computer network. Each example is formed by 41 high-level numerical and symbolic features derived from the connection records. For our purposes, we have used 2500 examples for training and 4996 for testing. These examples were randomly chosen from the original database maintaining the proportion between

¹ Data obtained from <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>

the two classes, that is, 25% of attacks and 75% of non-attacks. Several single layer neural networks were trained by increasing the size of the training set from 100 to 2500 samples, adding in each step 100 new elements. Also, for each one of these 25 training subsets, simulations were made varying the regularization parameter α from 0 to 1 in steps of 5×10^{-3} . In order to better estimate the true error rate, this process was repeated 12 times using 12 different and disjunct training sets. In all cases, the test set remained the same.

Figure 2(a) shows the mean error curve for the 12 simulations. To obtain this curve, the regularization parameters $\alpha \in (0, 1]$ that produce the minimum test classification errors for each simulation and training set were chosen. Also in this figure, it is shown the mean error curve when no regularization term is used ($\alpha = 0$). In order to establish whether there is a significant difference between the two error curves, a Wilcoxon rank sum test was performed. Results indicated that, with a 95% confidence level, the null hypothesis *there is no difference between the two errors* can be rejected. Also, this test has been used to estimate the minimum training set size at which the error is stabilized. Results indicated that 400 samples are needed to stabilize the error when a regularization term is used, whereas this size increases to 700 samples when no regularization term is used. Figure 2(b) shows the boxplot and the mean optimal value for the regularization parameter over the 12 simulations for each size of the training set. This boxplot shows the median α so as their interquartile range. The maximum length of each whisker is 1.5 times the interquartile range.

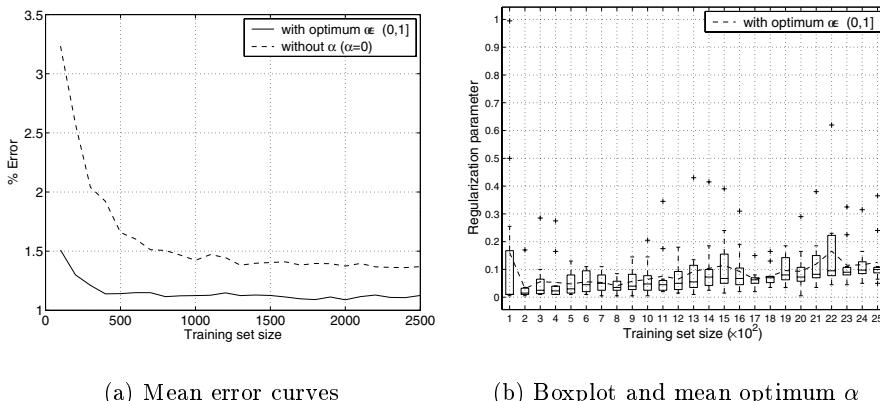


Fig. 2. Results for the Intrusion Detection problem

3.2 The Box-Jenkins time series

Our second example uses the Box-Jenkins time series [5] represented in figure 3(a). This is a regression problem where the system to be modeled is a gas

furnace that combines air with methane to obtain a gas mixture which contains CO_2 . There are originally 296 data points $\{y(t), u(t)\}, t = 1, \dots, 296$, where $y(t)$ is the furnace output CO_2 concentration and $u(t)$ is the input methane flow rate. In this case, we are trying to predict $y(t)$ based on $\{y(t-1), y(t-2), y(t-3), y(t-4), u(t-1), u(t-2), u(t-3), u(t-4), u(t-5), u(t-6)\}$. Thus, the number of effective data points is reduced to 290. Due to the size of the available data set 10-fold cross-validation was employed as the method to estimate the true error rate, so we have 261 examples for training and 29 for testing for each simulation. However, in order to study the influence of the regularization term for different training set sizes, in each of the 10 validation rounds the size of the training set was increased from 9 to 261 samples, adding in each step 9 new elements. Also, for each of these 29 training subsets the regularization parameter α was varied from 0 to 1 in steps of 10^{-3} . Finally, in order to improve the estimation of the error when small training sets are employed, the 10-fold cross-validation process described above was repeated 10 times using different composition of the training sets.

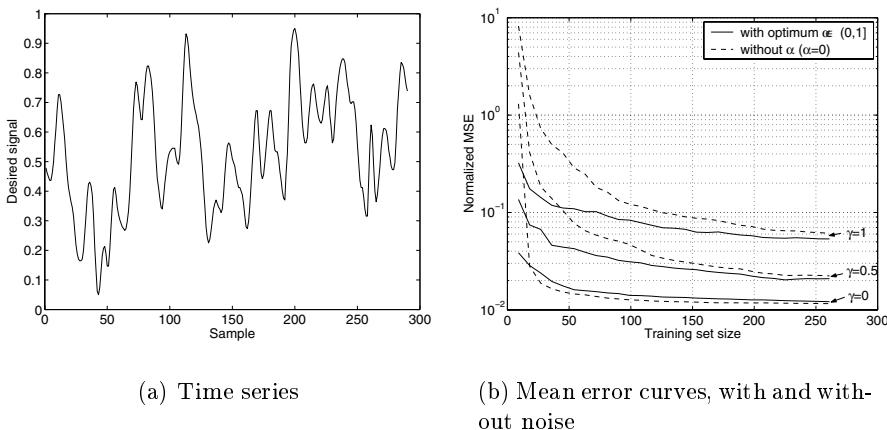


Fig. 3. Results for the Box-Jenkins problem

Figure 3(b) shows the mean error curve for the 10×10 simulations. To obtain this curve, the regularization parameter $\alpha \in (0, 1]$ that produces the minimum normalized mean squared error for each simulation and training set were chosen. Also in this figure, it is shown the mean error curve when no regularization term (dashed line) is used ($\alpha = 0$). As it can be observed, only for small training set sizes (below 45 examples) the use of the regularization term improves the error. For bigger data sets, there is no significant difference between the errors obtained whether the regularization term is used or not. This can be explained by the noise-free nature of the signal and also by the training set size that is

sufficient for the signal to be modeled by the network. Results are different when noise is added to the signal. Subsequently, a normal random noise was added to the time series. The standard deviation of the noise was $\sigma = \gamma\sigma_t$, where σ_t is the standard deviation of the noiseless time series and $\gamma \in \{0.5, 1\}$. Figure 3(b) also shows the corresponding mean error curves.

In order to establish whether there were significant differences between the two error curves when noise is added to the signal a Wilcoxon rank sum test was performed. Being the null hypothesis *there is no difference between the two errors*, results indicated that, with a 95% confidence level, this hypothesis can be rejected for $\gamma = 1$, while for $\gamma = 0.5$ it can be rejected only for training set sizes lesser than 225 examples. Also, this test has been used to estimate the minimum training set size at which the error stabilized. Results are shown in Table 1.

Table 1. Needed samples to stabilize the error

γ	training set size	
	optimum $\alpha \in (0, 1]$	$\alpha = 0$
0.5	198	198
1	189	207

4 Discussion

The linear learning method described in section 2 obtains always the global minimum of the error surface for the available training set. In the case of $\alpha = 0$, the training algorithm will adapt the weights with only the restriction of fitting the input data. Although the optimal solution obtained in this case is the desired goal, in some situations, there are some cases where this is not so. Generally speaking, when the training set sizes are too small to sufficiently represent the system to be modeled, neural networks will adapt to the training set and will fail when new inputs are presented. This problem is more noticeable when the linear learning method with $\alpha = 0$ is used, as it will guarantee the solution that best adapt to the available training data. On the other hand, when the signal is noisy, the network will learn this noise. Thus, in both cases the will be a loss in the generalization capability of the network. Adding the regularization term to the error function partially solves this problem, as it can be seen in figures 2(a) and 3(b). It can be observed that, when the problem is complex enough, as it is the Intrusion Detection problem, the inclusion of regularization term improves the test error (see figure 2(a)), that it is more evident for training set sizes below 700 elements. The opposite can be observed when the system is simple enough, as it can be seen in figure 3(b) when no noise is added to the Box-Jenkins signal ($\gamma = 0$) where the regularization term has practically no statistical influence in the test error. However, the influence of the regularization term augments as the noise magnitude γ increases, as it can be observed also in figure 3(b) and

in table 1, where the differences between the test errors obtained with $\alpha = 0$ and $\alpha \neq 0$ are higher as the noise is enhanced. Also it has been observed that, when the system is complex or noise is added ($\gamma = 1$), the number of training samples needed to obtain a certain error decreases when the regularization term is used. For the Intrusion Detection problem, only 400 samples are needed to obtain a similar error to that obtained using the whole training set, whereas this size increases to 700 when the regularization term is not used. In the case of the Box-Jenkins data with $\gamma = 1$ these sizes are 189 and 207, respectively. Finally, experimental results shown in figure 2(b) support the hypothesis about the optimal value of α being in the interval $[0, 1]$. Although this value presents a slightly increasing trend it can be seen that the median and mean values of α are concentrated, with a small deviation, in the interval $[0, 0.2]$.

5 Conclusions and Future Work

In this work, a new supervised learning for single layer neural networks has been proposed. The method, based on a regularized cost function, obtains the solution using a system of linear equations. The main advantages of this method are a) that it always obtains the global optimum, b) it is a very fast procedure, and c) it allows incremental learning. To validate the performance of the method several experiments were accomplished and the optimal regularization parameter was chosen through some simulations. As future work, an analytical method to obtain the regularization parameter (α) is being analyzed.

6 Acknowledgements

This research has been supported partially by the Pre-Doctoral Grants Programme 2002/2003 of the Xunta de Galicia and University of A Coruña funds.

References

1. Girosi, F., Jones, M., Poggio, T.: Regularization theory and neural networks architectures. *Neural Computation* **7** (1995) 219–269
2. Bishop, C.M.: *Neural Networks for Pattern Recognition*. Oxford University Press, New York (1995)
3. Hinton, G.E.: Learning translation invariant recognition in massively parallel networks. In Nijman, A.J., de Bakker, J., Treleaven, P.C., eds.: *PARLE Conference on Parallel Architectures and Languages Europe*, Berlin, Springer-Verlag (1987) 1–13
4. Fontenla-Romero, O., Erdogmus, D., Principe, J.C., Alonso-Betanzos, A., Castillo, E.: Accelerating the convergence speed of neural networks learning methods using least squares. In: *11th European Symposium on Artificial Neural Networks (ESANN)*. (2003) (in press)
5. Box, G.E.P., Jenkins, G.M.: *Time series analysis, forecasting and control*. Holden Day, San Francisco (1970)

A better selection of patterns in lazy learning radial basis neural networks

P. Isasi¹, J.M. Valls² and I.M. Galván²

Carlos III University of Madrid, Computer Science Department. Avd Universidad, 30,
28911, Leganés, Madrid

¹isasi@ia.uc3m.es

²{jvalls, igalvan}@inf.uc3m.es

Abstract. Lazy learning methods have been proved useful when dealing with problems in which the learning examples have multiple local functions. These methods are related with the selection, for training purposes, of a subset of examples, and making some linear combination to generate the output. On the other hand, neural network are eager learning methods that have a high nonlinear behavior. In this work, a lazy method is proposed for Radial Basis Neural Networks in order to improve both, the generalization capability of those networks for some specific domains, and the performance of classical lazy learning methods. A comparison with some lazy methods, and RBNN trained as usual is made, and the new approach shows good results in two test domains, a real life problem and an artificial domain.

1 Introduction

Lazy learning methods [1,2,3] are conceptually straightforward approaches to approximating real-valued or discrete-valued target functions. These learning algorithms defer the decision of how generalize beyond the training data until a new sample or instance is encountered. When a new sample is received, a set of similar related patters is retrieved from the available training patters and used to approximate the new query sample. Similar patterns are chosen using a distance measured with nearby points having high relevance.

Lazy methods that appear in the literature generally work by selecting the k least distant input patters from the novel samples, often in terms of Euclidean distance. Afterwards a local approximation using the selected samples is carried out with the purpose of generalize the new sample. That local approximation can be constructed using different strategies. The most basic form is the k - nearest neighbor method [4]. In this case, the approximation of the new sample is just the most common output value among the k selected examples. A refinement of this method, called weighted k -nearest neighbor [4], can be also used, which consists of weighting the contribution of each of the k neighbors according to the distance to the new sample, giving greater weight to closer neighbors. Other strategy to determine the approximation of the new sample is the locally weighted linear regression [2] that constructs an explicit and

linear approximation of the target function over a region around the new sample. The coefficients regression is based on the k nearest input patterns to new sample.

When lazy learning techniques are used, the target function is represented by a combination of many local approximations constructed in the neighborhood of the new samples. On the other hand, eager learning methods construct global approximations and the generalization is carried out beyond the training data before observing the new sample. That global approximation over the training data representing the domain could lead to poor generalization properties, mainly if the target function is complex. In these cases, lazy methods could be appropriate because the complex target function could be described by a collection of less complex local approximations.

Artificial neural networks can be considered as eager learning methods because they construct a global approximation that covers the entire sample space and all future samples. Although Radial Basis Neural networks (RBNN) [5,6] use multiple local approximations, they are also eager leaning methods because the network must commit to the hypothesis before the query point is known. The local approximations they create are not specially targeted to the query point to the same degree as in a lazy learning methods. Instead, RBN networks are built eagerly from local approximations centered around the training samples or around clusters of training samples, but not around the unknown future query point. That could contribute to poor generalization properties of RBNN.

The goal of this study is to improve the generalization capabilities of RBNN using a lazy learning strategy. In this context, the most basic strategy should consist of constructing local non-linear regression based on RBNN, this is, on approximating the target function in the neighborhood surrounding the new sample using a RBNN. In this case, parameters of the network –centers, width and weights- should be determined using the k nearest training patterns. However, the idea of selecting the k nearest patterns might not be the most appropriate mainly because of one factor: the network will always be trained with the same number of training data for each new sample. That may be a disadvantage in the context of artificial neural networks because each new sample could require different training data.

In this paper a lazy learning strategy is proposed to improve the generalization capability of RBNN. The main idea is to recognize from the whole training data set, the most similar patterns to each new pattern to be processed. This subset of useful patterns is used to train a RBNN and the training is deferred until a test pattern is received. The patterns retrieved from the available training data set to train the RBNN and posterior approximation of the new sample are determined by using the inverse of the Euclidean distance and a threshold distance or cut, which determines the extension of the neighborhood of the novel pattern. The number of retrieved patters will depend on the localization of the new sample in the input space.

The proposed lazy learning strategy to train RBNNs is validated in different domains and compared with other lazy methods, as k-nearest neighbor, weighted k-nearest neighbor, locally weighted linear regression and locally non-linear regression based on RBNN. The proposed method is also compared against the traditional way of training RBNN that uses the complete training data and construct a global approximation of the target function.

2 Lazy learning for Radial Basis Neural Networks

The lazy learning method studied in this work to train RBNN consists of selecting, from the whole training data, an appropriate subset of patterns to improve the answer of the RBNN for a novel sample. The general idea for the selection of patterns is to only include, and one or more times, those near patterns -in terms of Euclidean distance- to the novel sample. This way, the network is trained with the most useful information, discarding those patterns that not only provide no knowledge to the network, but, besides, can confuse the learning process.

The amount of training data to be selected to approximate the new sample encountered is given by a relative n -dimensional volume, named V_r , surrounding the test pattern, where n is the dimension of the input space. This relative volume V_r is a fraction of the total volume from where training patterns are selected. In order to determine the training patterns included in that relative volume, a threshold distance or cut r_s (radius of the sphere) must be calculated before the learning algorithm is applied. The relative n -dimensional volume can be written as:

$$V_r = \frac{V_s}{V_{\max}}$$

where V_s is the volume of the sphere centered in the test pattern and radius r_s – this sphere contains the patterns that will be selected- and V_{\max} is the volume of the sphere centered in the test pattern and radius equals to the maximum distance, r_{\max} . Since $V=kr^n$ where r is the radius of the sphere, n is the dimension of the space and k is a constant, we can write:

$$V_r = \frac{kr_s^n}{kr_{\max}^n} = r_r^n \quad \text{Hence:} \quad r_r = \sqrt[n]{V_r}$$

The relative threshold distance, r_r , calculated as the n -th root of the relative volume will be used to select patterns in the fraction volume around the test pattern.

Given a test pattern \mathbf{q} , described by a n -dimensional vector, $\mathbf{q}=(q_1, \dots, q_n)$, the steps to select the training set, named X_q , associated to the patterns, are the following:

Step 1. A real value, d_k , is associated to each training pattern (x_k, y_k) . That value is defined in terms of the standard Euclidean distance as:

$$d_k = d(x_k, \mathbf{q}) = \sqrt{\sum_{i=1}^n (x_{ki} - q_i)^2}$$

Step 2. A relative distance, d_{nk} , is calculated for each training pattern. Let d_{\max} be the maximum distance to the novel pattern \mathbf{q} , this is $d_{\max} = \text{Max}(d_1, d_2, \dots, d_N)$. Then, the relative distance is given by:

$$d_{nk} = d_k/d_{\max}$$

Step 3. A new real value, $f_k = 1/d_{nk}$, where $k=1, \dots, N$ is associated to each training pattern (x_k, y_k) . These values f_k are normalized in such a way that its sum is equal to the number of training patterns in X . The relative values, named as fn_k , are obtained by:

$$fn_k = \frac{f_k}{S} \quad \text{where} \quad S = \frac{1}{N} \sum_{k=1}^N f_k \quad \text{Thus:} \quad \sum_{k=1}^N fn_k = N$$

Step 4. The relative distance, d_{ik} , calculated in step 2 and the relative threshold distance, r_r , previously calculated as the n-th root of the relative volume V_r , are used to decide if the training pattern (x_k, y_k) is selected:

If $d_{ik} < r_r$, then the pattern (x_k, y_k) is included in the training subset.

Value fn_k calculated in step 4 is used to indicate how many times the training pattern (x_k, y_k) is going to be repeated in the new training subset. Hence, they are transformed into natural numbers as:

$$n_k = \text{int}(fn_k) + 1$$

At this point, each training pattern in X that has been selected has an associated natural number, n_k , which indicates how many times the pattern (x_k, y_k) has been used to train the RBNN when the new instance q is reached.

Step 5. Once the training pattern subset associated to the test pattern q , X_q , is built up, the RBNN is trained with this new subset. As usual, training a RBNN involves to determine the centers, the dilations or widths, and the weights. The centers are calculated in an unsupervised way using the K-means algorithm presented in [7]. After that, the dilations coefficients are calculated as the square root of the product of the distances to their two neighbors. Finally, the weights of the RBNN are estimated in a supervised way to minimize the mean square error measured in the training subset X_q .

3 Experimental Validation

The lazy learning method described in section 2 has been applied to RBNN and the generalization capability of the network has been measured in terms of the mean error over the test data set. Different domains have been used with that purpose. The results obtained with that lazy strategy have been compared, firstly, with the results provided by the network when a global approximation over the whole training data set is constructed, this is when the network is trained as usual. Secondly, the results are also compared with other lazy methods, as k-nearest neighbour, weighted k-nearest neighbour, the weighted local linear regression and a weighted local nonlinear regression methods.

In this section, the features of different domains and the conditions of the experiments carried out are described. Finally, the results obtained with the different lazy learning methods are presented and compared.

3.1 Experimental definition

Different domains have been used to compare the different lazy strategies: one-dimensional theoretical approximation problem, a piecewise-defined function, and a n-dimensional ($n > 1$) real life problem, defined by means of a time-series describing the behaviour of the water level at Venice Lagoon. In the next, the characteristics of both of them are presented.

- **Theoretical problem: A piecewise-defined function approximation**

The function is given by the equation:

$$f(x) = \begin{cases} -2.186x - 12.864 & \text{if } -10 \leq x < -2 \\ 4.246x & \text{if } -2 \leq x < 0 \\ 10e^{(-0.05x-0.5)} \sin[(0.03x + 0.7)x] & \text{if } 0 \leq x \leq 10 \end{cases}$$

The original training set is composed by 120 input-output points randomly generated by an uniform distribution in the interval [-10,10]. The test set is composed by 80 input-output points generated in the same way as the points in the training set. Both sets have been normalized in the interval [0,1]

- **Real life problem: Prediction of water level at Venice Lagoon**

Unusually high tides, result from a combination of chaotic climatic elements in conjunction with the more normal, periodic, tidal systems associated with a particular area. The prediction of such events have always been subjects of high interest. The water level of Venice Lagoon is a clear example of these events. That phenomenon is known as "high water". Different approaches have been developed for the purpose of predicting the behavior of sea level at Venice Lagoon [8].

In this work, a training data set of 3000 points, corresponding to the level of water measured each hour has been extracted from available data in such a way that both stable situations and high water situations appear represented in the set. The test set has also been extracted from the available data and it is formed by 50 samples including the high water phenomenon. A nonlinear model using the six previous sampling times seems appropriate because the goal is to predict only the next sampling time.

3.2 Experimental conditions

As it has been previously mentioned, different lazy learning strategies have been used to deal with different problems. Now, the conditions of the experiment run are described. The k-nearest neighbour, the weighted k-nearest neighbour and the local weighted linear regression methods [2] have been run for different values of k parameter (number of patterns selected). For the piecewise-defined function, k is varied from 1 to 23 and for the prediction of the water level at Lagoon Venice k is varied from 1 to 75, because more data are available.

A local nonlinear regression method based on RBNN is also tested. In this case, when the test sample is encountered the k nearest input training patterns are retrieved and a RBNN is trained with those patterns. The initial centers of the RBNN are fixed around the geometric medium of the k training patters selected, and the training of the network is carried out as usual. The value of k is also varied in the same range that for the other lazy methods, but in this case the k value is incremented by 5 units. RBNNs with different number of hidden neurons have been proven. After several experiments it has been verified that the number of hidden neurons depends on the value of k, for instance $\text{int}(k/r)+1$, where $r=2,3,4,5,\dots$. In this work, the best results are obtained using $\text{int}(k/2)+1$ hidden neurons, although significant differences do not exist.

The lazy learning method described in section 2 is used to train RBNNs. In this case different relative volumes (V_r) have been used to run the experimental simulations. As in the previous case, the results do not depend significantly on the number of

hidden neurones. The results shown in the next section are obtained using 19 hidden neurones for the piecewise-defined function and 15 hidden neurones for the time series prediction domain. Finally, by comparative reasons, the RBNN have also been trained as usual, this is, the network is trained using the whole training data set. When the lazy strategy involves the use of a RBNN, the training is carried out until the convergence of the network is reached, that is, when the derivative of the training error is near zero.

3.3 Experimental results

Figure 1 shows the behaviour of the different lazy strategies in both studied domains. The mean error over the test data sets for each value of k is represented. In figure 2, the behaviour of the lazy strategy proposed in section 2 to train RBNN is shown. In this case, the mean error over the test data is evaluated for every value of relative volume.

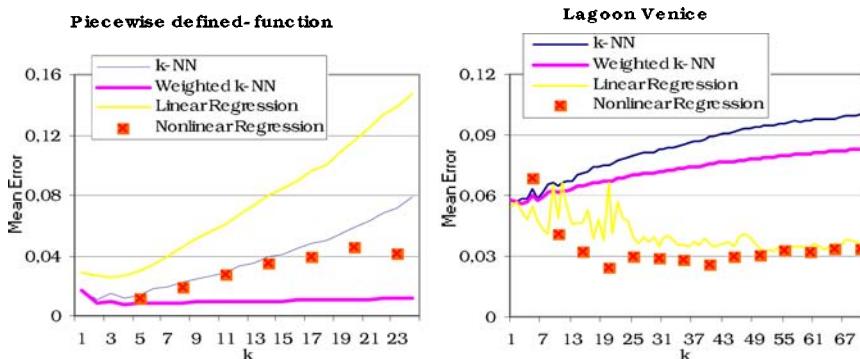


Figure 1. Evolution of the test mean errors for k-NN Weighted k-NN, linear and nonlinear local regression methods

In figure 1 it is observed that for the piecewise-defined function, the performance of classical lazy strategies is in general influenced by the value of k , increasing the error as the k value increases, although when a nonlinear local approximation is made, that influence is smaller. However when a weighted method is used, weighted k-NN, the influence of the parameter k almost disappears, as expected. Moreover the error decreases reaching the best values of all traditional methods.

For the prediction problem, the behaviour of the traditional methods is very different. On the one hand, there is a stabilization of the error after a certain value of k , but only when using regression methods. On the other hand, however for the k-NN algorithms the error has a worse behaviour. It increases when k increases, and even the best results, in k , are very bad. This is due to the local influence of data in the Venice lagoon domain. Any way, it seems clear that the influence of the domains in the results when using traditional lazy approaches is very high.

The performance of the lazy method proposed in this work (see figure 2) does not depends significantly on the value of relative volume for both domains. In this case, the mean error is more or less the same for every relative volume, once a certain

amount of data are selected. It can also be seen that similar results can be found in both domains, the proposed method seems to be less domain dependent. The new method takes advantage of the property of selecting a different set, in number and elements, of learning patterns. It is able to discover the most appropriate set of examples for each new test pattern.

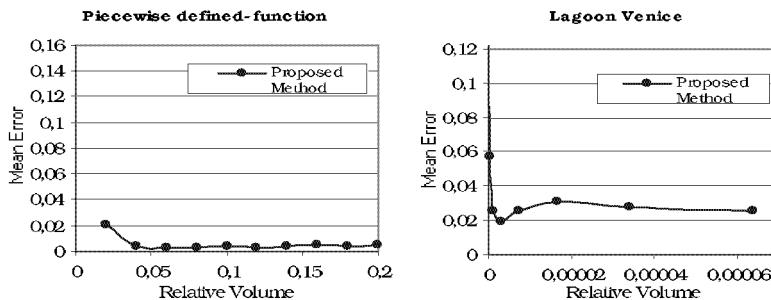


Figure 2. Evolution of the test mean errors for the proposed lazy method to RBNN

The best mean errors in test obtained by the different methods and the different domains are shown in tables 1 and 2. Table 2 shows also the mean error obtained when a global approximation of the target function using RBNN is made. As it is observed in both tables, the mean error over the test set in all domains is significantly reduced when the lazy strategy proposed in this work is used. When k patterns are selected and a nonlinear local regression is made, the results do not improve with respect to linear regression. However, when the training patterns are selected based on the new test sample received, better generalization capabilities are obtained. In addition, the RBNN has poor generalization capabilities if the network is trained with the whole training patterns, that is when a global approximation is built up.

Table 1. Best mean error for k-NN, Weighted k-NN, linear and nonlinear local methods

Mean Error (k value)	k-nearest neighbor	Weighted k-nearest neighbor	Local linear regression	Local nonlinear regression based on RBNN
Piecewise-defined function	0.01113 (k=2)	0.00793 (k=4)	0.02513 (k=3)	0.01120 (k=5)
Lagoon Venice	0.05671 (k=2)	0.05610 (k=3)	0.0323 $k \in [50, 70]$	0.02334 (k=19)

Table 2. Best mean error for the proposed lazy method to RBNN

Mean Error (relative volume)	Lazy method	Traditional method
Piecewise-defined function	0.002085	0.0396
Lagoon Venice	0.019	0.055

4 Conclusions

The idea of representing the target function by a combination of many local approximations constructed in the neighbour of the new sample could provide better performance of those learning methods than construct global approximation, mainly if the target function is complex. However, the performance of lazy methods is influenced by the criterion of selecting the patterns that determine each local approximation. When local approximation are built up using the lazy strategies based on the selection of k patterns -as the k-nearest neighbour, weighted k-nearest neighbour and locally weighted regression- for every test sample, the same amount of patterns are selected. The results presented in the previous section show that those lazy strategies have poor generalization capabilities when approximation and prediction problems are formulated, even if nonlinear local approximations are made.

The lazy method presented in this work makes an automatic selection of training patterns for each test samples allowing that number of training patterns is variable depending on the position in the input space. In addition, the performance of the lazy method presented in this work is higher than those lazy strategies selecting the k nearest patterns.

On the other hand, the lazy strategy presented in that work to train RBNN improve the generalization capabilities of those type of artificial neural network. In this paper is also shown that the selection of the most relevant training patterns, the neighbours of the novel pattern, helps to obtain RBNN's able to better approximate complex functions.

5 References

- [1] Aha D., D. Kibler and M. Albert. Instanced-based learning algorithms. *Machine Learning*, 6, (1991), 37-66.
- [2] Atkeson C. G., A. W. Moore and S. Schaal. Locally Weighted Learning. *Artificial Intelligence Review* 11, (1997), 11-73.
- [3] Wettschereck D., D.W. Aha and T. Mohri: A review and Empirical Evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review* 11, (1997), 273-314.
- [4] Dasarathy B.V. (Ed.). Nearest neighbor(NN) norms: NN pattern classification techniques. Los Alamitos, CA: IEEE Computer Society Press. (1991)
- [5] Moody J.E. and Darken C.J.: Fast Learning in Networks of Locally-Tuned Processing Units. *Neural Computation* 1, 281-294, 1989.
- [6] Poggio T. and Girosi F.: Networks for approximation and learning. *Proceedings of the IEEE*, 78, 1481-1497, 1990.
- [7] J. M. Valls, P. Isasi and I. M. Galván: Deferring the learning for better generalization in Radial Basis Neural Networks. *Lecture Notes in Computer Science* 2130. *Artificial Neural Networks – ICANN 2001*, 189-195
- [8] Vittori G.: On the Chaotic features of tide elevation in the lagoon Venice. In Proc. of the ICCE-92, 23rd International Conference on Coastal Engineering, 4-9 (Venice 1992), 361-362.

An Iterative Fuzzy Prototype Induction Algorithm

Inés González Rodríguez¹, Jonathan Lawry², and Jim F. Baldwin²

¹ Dept. de Ciencias Experimentales e Ingeniería, Universidad Rey Juan Carlos,
Avenida Tulipán s/n, 28339 Móstoles (Madrid), Spain
ines.gonzalez@universia.es

² A.I. Group, Dept. of Engineering Mathematics, University of Bristol
University Walk, BS8 1TR Bristol, UK
{j.lawry, jim.baldwin}@bris.ac.uk

Abstract. This paper is concerned with the automated induction of prototypes to represent a database in a way that combines transparency and accuracy. A clustering algorithm will be described which learns fuzzy prototypes from a set of data. The potential of the resulting method will be illustrated by its application to classification problems and comparing its performance with that of previous approaches in the literature.

1 Introduction

In many of the emerging information technologies, there is a clear need for automated learning from databases. In general, it is necessary to somehow summarise the overwhelming amount of information contained in collected data and learn transparent models which allow for insight into the underlying nature of the considered system. It must also be the goal of any learning process to obtain a model that is efficient at predicting the behaviour of the system. This requirement is in some sense contrary to that of transparency since for a model to be easy to understand it must be relatively simple and yet the system being modelled is often highly complex.

It is the need for automated learning that motivates our clustering algorithm, which tries to learn fuzzy prototypes to represent data sets and also decides the number of prototypes needed. We believe that the fuzzy prototype framework together with the proposed learning algorithm allow for the induction of transparent models that are competitive with other methods in terms of predictive accuracy.

2 Mass Assignment Theory and Linguistic Variables

Modelling real world problems typically involves processing uncertainty, either due to a lack of knowledge relating to concepts or due to inherent vagueness in concepts themselves. The notion of fuzzy set, introduced by Zadeh [1], tries to formalise the concept of gradeness in class membership, in connection with the

representation of human knowledge. Since Zadeh's foundational paper, fuzzy set theory has been widely developed and proved to be a successful tool in many fields. However, there is as yet no uniformity in the interpretation of the meaning of membership grades (usually but not necessarily taking values in the unit interval $[0, 1]$).

Mass assignment theory is an interpretation of possibility theory [2] which aims to provide a semantics for membership functions of fuzzy sets using the *voting model* [3][4]. Essentially the idea is that a fuzzy (or a vague) concept is simply a concept for which the definition is uncertain or variable (across, say, a population of voters)³. This theory allows us, in a sense, to use fuzzy sets as descriptions of probability distributions. Indeed, conditioning a variable X relative to a fuzzy constraint ' X is f ' we obtain a probability distribution, referred to as the *least prejudiced distribution* of f and denoted by lp_f . Furthermore, given a probability distribution p on a finite universe Ω there is a unique fuzzy set f conditioning on which yields this distribution. We shall refer to this unique fuzzy set as the *fuzzy description* of p and it will be denoted as $fd(p)$.

The above is only applicable in finite universes, but most real-world problems involve continuous variables. We propose then to discretise infinite universes using *linguistic variables*. More precisely, we use fuzzy sets to divide continuous universes into *information granules*, a group of points drawn together by similarity which can be viewed as corresponding to the meaning of words from natural language [6]. Then, a *linguistic variable* can be defined which is associated with the original continuous universe and takes as its values the words. This linguistic variable allows us to rewrite instances of a continuous variable X using linguistic labels. The object obtained in this way is referred to as *linguistic description* and it is a fuzzy set on words encoding the applicability of each word w as a label for the original numerical value of X [7].

3 Fuzzy Prototypes

One interpretation of a prototype representing a set of data is that of an amalgam of objects belonging to the set which are in some way similar. Clearly, if such a prototype is to be at all representative of this group of points, it cannot be precisely defined. Hence, we propose to define prototypes as tuples of fuzzy sets over attribute universes, where each fuzzy set can be interpreted as an elastic constraint on the values of the attribute [8].

Definition 1. A fuzzy prototype in $\Omega_1 \times \cdots \times \Omega_n$ is a n -tuple of fuzzy sets $f = \langle f_1, \dots, f_n \rangle$ where f_i is a fuzzy subset of Ω_i for $i = 1, \dots, n$.

A fuzzy prototype should be determined by the collection of points represented by it. In particular, a group of objects infer a probability distribution on each attribute universe. As data are often far from precise and certain, we

³ This is a somewhat non-standard definition and is sometimes referred to as the *epistemic* view of vagueness [5].

approximate this probability distribution by means of a possibility distribution. Indeed, each probability distribution is uniquely associated with a fuzzy set, its fuzzy description, which can be identified with a possibility distribution on the values of the attribute. We obtain in this manner a special type of prototype [8].

Definition 2. Given a set D of objects in $\Omega_1 \times \dots \times \Omega_n$ where Ω_i is a finite universe, let r_i be the distribution on Ω_i generated by D so that

$$\forall x \in \Omega_i \quad r_i(x) = \frac{|\{x = \langle x_1, \dots, x_n \rangle \in D : x_i = x\}|}{|D|}. \quad (1)$$

The tuple formed by fuzzy descriptions of r_i , $\langle fd(r_1), \dots, fd(r_n) \rangle$, is referred to as the fuzzy mean of D and is denoted by $fm(D)$.

Now, if we are interested in grouping similar elements, we need to define a notion of distance or similarity between vectors. Furthermore, data vectors can be viewed as a special case of prototypes, the fuzzy mean of the set containing a single element. Hence, we adopt the following notion of prototype distance metric [9], based on averaging distances of elements sampled independently according to distributions obtained by the fuzzy constraints represented by the prototypes. Notice that, given a fuzzy prototype f , it is possible that $d(f, f) \neq 0$. In fact, the distance between a prototype and itself quantifies the variability of objects within the prototype.

Definition 3. Let d_i be a distance metric on the finite universe Ω_i , $i = 1, \dots, n$, and let Π be the class of all prototypes in $\Omega_1 \times \dots \times \Omega_n$. Then, the fuzzy prototype distance measure based on $\{d_i\}_{i=1}^n$ is a function $d : \Pi \times \Pi \mapsto \mathbb{R}$ such that, for any two prototypes $f = \langle f_1, \dots, f_n \rangle$ and $g = \langle g_1, \dots, g_n \rangle$,

$$d(f, g) = \sum_{i=1}^n E_{lp_{f_i} \times lp_{g_i}}(d_i) = \sum_{i=1}^n \sum_{x \in \Omega_i} \sum_{y \in \Omega_i} lp_{f_i}(x)lp_{g_i}(y)d_i(x, y) \quad (2)$$

Another problem that needs to be addressed is that of how to make inferences from fuzzy prototypes in order to determine the behaviour of a new object. In this context, we need to determine a degree of matching between a prototype and a given point or, indeed, between two fuzzy prototypes. This can be done using the probability of fuzzy events [4] [8]:

Definition 4. Let $f = \langle f_1, \dots, f_n \rangle$ be a fuzzy prototype and $x = \langle x_1, \dots, x_n \rangle$ a tuple of values in $\Omega_1 \times \dots \times \Omega_n$. The support for x belonging to f is given by:

$$supp(f|x) = \prod_{i=1}^n \Pr(f_i|x_i) \quad (3)$$

The above definition is useful in the case of supervised learning, where the data points are classified as belonging to one of k classes, $\{C_1, \dots, C_k\}$. If we have a set of fuzzy prototypes representing each of the classes and we are given an object x whose class is unknown, we can determine the support for the tuple x belonging to each particular fuzzy prototype f . Then, the vector is classified as belonging to the class associated with the prototype with highest support.

4 An Error Measure for Fuzzy Prototypes

Having learnt a set of fuzzy prototypes from a dataset D , we may want to somehow assess the quality of these prototypes as descriptors of the data. In particular, for supervised learning we may be interested in knowing if a prototype is a good descriptor of a certain class. An obvious choice in this case is to use *predictive accuracy*, i.e., run a classifier through the data set and take the proportion of misclassified elements in a class as an error measure of the prototypes for this class and the proportion of correctly classified elements as an overall quality measure. Although quite straightforward, we feel this method has some strong disadvantages. For this reason, we propose another heuristic to assess the quality of prototypes based on conditional probabilities of fuzzy events.

Definition 5. Let $D \subseteq \Omega_1 \times \dots \times \Omega_n$ be a data set partitioned into classes and let C_i be one of these classes. The quality with respect to class C_i of a prototype f in $\Omega_1 \times \dots \times \Omega_n$ is the probability of C_i given f :

$$Q_i(f) = \Pr(C_i|f) = \frac{\sum_{x \in D_i} \Pr(f|x)}{\sum_{x \in D} \Pr(f|x)} \quad (4)$$

and the error with respect to class C_i of prototype f is the probability of “not class C_i ” given f :

$$E_i(f) = \Pr(\neg C_i|f) = 1 - Q_i(f) \quad (5)$$

The weighted quality with respect to class C_i of a set of prototypes $\{f_j\}_{j=1}^n$ is defined as:

$$WQ_i\left(\{f_j\}_{j=1}^m\right) = \sum_{j=1}^m w_j Q_i(f_j) \quad (6)$$

where w_j is the weight for prototype f_j

$$w_j = \frac{\sum_{x \in D} \Pr(f_j|x)}{\sum_{l=1}^m \sum_{x \in D} \Pr(f_l|x)}. \quad (7)$$

Similarly, the weighted error with respect to class C_i of this set of prototypes is defined as:

$$WE_i\left(\{f_j\}_{j=1}^m\right) = \sum_{j=1}^m w_j E_i(f_j) \quad (8)$$

It is interesting to see that $\sum_{j=1}^m w_j = 1$ and, in consequence, we have that

$$WE_i\left(\{f_j\}_{j=1}^m\right) = 1 - WQ_i\left(\{f_j\}_{j=1}^m\right). \quad (9)$$

We believe that these definitions overcome some of the problems of predictive accuracy: they are independent of the size of the classes, they take into account false positives and, most importantly, the error of a given prototype does not depend on other prototypes and hence is not affected by changes taking place during the clustering process.

5 Learning Prototypes from Data

Having proposed a framework of fuzzy prototypes, we need a method to automatically learn such prototypes from a given set of data. In other words, we need a means of finding groupings of similar points and, for each grouping, generate a prototype that constitutes a description of the elements contained therein.

A widely used clustering method is *c-means*, also known as *k-means*, which partitions a set of data into c clusters via the optimisation of an objective function and where each cluster is represented by its centre [10]. *Fuzzy c-means* is a fuzzy version of this method, which partitions the data set into c fuzzy clusters, each cluster still being represented by its centre [11].

An alternative version of *c-means* is *c-fuzzy means*, which tries to find c *crisp* clusters so as to minimise the average distance from the vectors in D to the fuzzy means of these clusters [8]. This method differs greatly from the fuzzy *c-means* and is, instead, a more straightforward extension of classical *c-means* algorithm. Here, clusters are crisp and fuzziness is introduced with the use of fuzzy prototypes. This might be considered an advantage over fuzzy *c-means*, as it learns tuples of fuzzy sets rather than multidimensional fuzzy sets and the resulting model is therefore easier to interpret and offers a better insight into the data. Also, fuzzy prototypes might result in better descriptions of the data than single data points, as more information is preserved by storing distributions rather than just instances.

The *c-fuzzy means* algorithm has proved to be useful both for unsupervised and supervised learning. However, it has the common disadvantage that the number of prototypes c must be specified. This is problematic, since we do not usually know beforehand what will be the optimum number of prototypes to describe a dataset. To overcome this problem, we propose to extend *c-fuzzy means* into a new clustering method, the *Iterative Prototype Induction Algorithm* or *IPI*. This method tries to learn the number of prototypes needed as part of the induction process by iteratively applying *c-fuzzy means*. It starts with $c = 1$ and then increases the number of clusters until a number c_0 is found such that the prototypes obtained from the *c-fuzzy means* with $c = c_0$ are “optimum” to describe the data set.

The idea of “optimality” is central to the clustering method. Let us suppose that we have *WE* an error measure for a set of prototypes (an example of *WE* for supervised learning was described in Section 4). We might then measure optimality in terms of *WE*, so that the *optimum* set of prototypes is the set with the smallest cardinality which has an error below a given threshold $\alpha \in [0, 1]$. This definition is based on the idea that by increasing the number of prototypes, we improve the description of the data and, in consequence, the error *WE* decreases. Nevertheless, there are many cases where we would expect there to be an optimum number of prototypes so that increasing the value of c beyond this would not result in a significantly smaller error. Hence, we propose to introduce a second parameter $\delta \in [0, 1]$ to monitor changes in the error and stop the clustering process whenever the difference between two consecutive errors

is less than δ , even if the threshold α has not yet been reached. A detailed description of the algorithm using both parameters α and δ is as follows:

```

c=0 {we start with 0 prototypes}
WE(F(0))=1 {the error of the set of 0 prototypes is maximum}
repeat
    c=c+1 {try to find 1 more prototype}
    F(c)= set of prototypes obtained using c-fuzzy means
until ((WE(F(c))<=ALPHA) or (WE(F(c-1))-WE(F(c))<=DELTA))
if (WE(F(c))<=ALPHA) then {the error is low enough}
    the "optimum" number of prototypes is c
    and the "optimum" set of prototypes is in F(c)
else {the error did not improve enough}
    the "optimum" number of prototypes is c-1
    and the "optimum" set of prototypes is in F(c-1)
end if

```

In supervised learning, where the data points are labelled with one of k classes, $\{C_1, \dots, C_k\}$ and the data set is partitioned according to these classes, IPI should be applied to the subsets of the data formed by each of the classes, D_1, \dots, D_k . Clearly, it is an improvement on standard c -fuzzy means, since only two parameters, α and δ , must be specified, instead of k parameters (i.e., the value of c for each class). Later, a classifier can be built according to Section 3.

6 Application to a Model Classification Problem

To illustrate the above algorithm, we consider a toy problem where a figure eight shape is generated and points in $[-1.6, 1.6]^2$ are labelled as *legal* if they lie within the figure and *illegal* if they lie outside. A data set of 961 points from a regular grid is used, with 241 elements classified as *legal* and 720 as *illegal*. Since attributes take numerical values, we discretise each attribute's universe using five trapezoidal fuzzy sets, which could be seen as encoding the meaning of *very small*, *small*, *medium*, *large* and *very large* in the universe $[-1.6, 1.6]$. To make IPI stop at the first local minimum of WE , we set $\alpha = \delta = 0$. In these conditions, IPI starts finding one fuzzy means for the *legal* elements. The error for this first prototype is $0.57919 > \alpha$, hence IPI applies c -fuzzy means again with $c = 2$. The error, equal to 0.48823, has thus decreased but is still greater than α . IPI finds then three fuzzy means and the error, equal to 0.489848, is greater than the error of two prototypes. Therefore, IPI stops and considers that the two prototypes from the second iteration are the optimal ones. This coincides with our intuition, which would be to have a prototype for each half of the eight shape. For the *illegal* points, IPI finds a single prototype. When a classifier is run on a test set of 2116 points from a regular grid on $[-1.6, 1.6]^2$, the classification accuracy is equal to 90.64%. Most importantly, the learnt prototypes manage to capture the underlying geometrical structure of the data. This can be seen in Figure 1, which shows the learnt shape compared with the original figure of eight, together with the histograms corresponding to the learnt *legal* prototypes

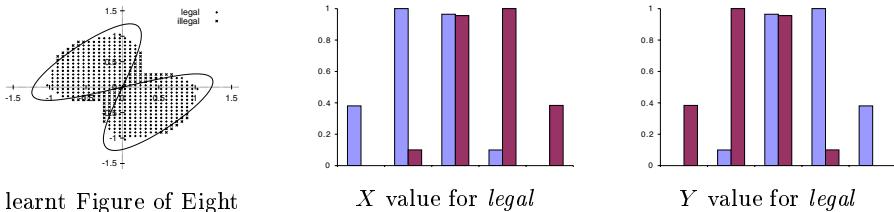


Fig. 1. Learnt Figure of Eight and prototypes for the *legal* class

7 Application to a Benchmark Classification Problem

We consider now a benchmark problem from the UCI machine learning repository [12] regarding the classification of glass fragments. These data originate from a project carried out by the Home Office Forensic Science Service Central Research Establishment on the identification of glass fragments found at crime scenes, motivated by the fact that in a criminal investigation the glass found at the scene of the crime can only be used as evidence if it is correctly identified. Glass fragments are divided into 7 possible classes, although the database only contains examples of six (there are no instances of the fourth class). These are: building windows (float and non float processed), vehicle windows (float and non float processed), containers, tableware and headlamps. The classification is to be made on the basis of the 9 attributes, relating to certain chemical properties of the glass. Following past usage in the literature, the database of 214 instances was split into a training and test set of 107 instances each in such a way that the instances of each class were divided equally between the two sets. Using five trapezoidal fuzzy sets to discretise the attribute universes, the results obtained by IPI compare favourably with other methods. For instance, a previous mass assignment based prototype induction algorithm gave a classification accuracy of 71% on the test set, a mass assignment ID3 gave an accuracy of 68% on the test set, a semi-naïve Bayes classifier obtained an accuracy of 71.03% and a neural network with topology 9–6–6 gave 72% on a smaller test set where the network was trained on 50% of the data and validated on 25% and tested on 25% [9] [7] [13]. Table 1 shows the predictive accuracy on the training and test set and the number of prototypes found per class given different values of the thresholds. It is interesting to see that, on the one hand, clustering improves the predictive accuracy but, on the other hand, an increasing number of prototypes does not always improve the predictive accuracy.

8 Conclusions

A clustering algorithm has been described which learns fuzzy prototypes as well as the number of prototypes needed to represent a data set adequately. In order to develop this algorithm, a heuristic measure for prototype validation has been defined. Finally, the framework of fuzzy prototypes and the use of linguistic

Table 1. Results of IPI for the Glass Data using 5 fuzzy sets

Thresholds		Predictive Accuracy		no. of prots					
α	δ	training set	test set	1	2	3	5	6	7
1	0.1	0.682243	0.579439	1	1	1	1	1	1
0.25	0.1	0.869159	0.803738	2	3	3	1	1	1
0.1	0.01	0.878505	0.766355	3	3	3	2	1	1
0	0	0.878505	0.757009	3	3	3	7	1	1

variables allows for the learnt model to be reasonably transparent, while the model's accuracy makes it comparable to other approaches in the literature.

Acknowledgements

Most of this work was carried out while the first author was a PhD student in the University of Bristol, under the supervision of the other two authors, with funding from the EPSRC.

References

1. Zadeh, L.A.: Fuzzy sets. *Information Control* **8** (1965) 338–353
2. Dubois, D., Prade, H.: Possibility Theory: An Approach to Computerized Processing of Uncertainty. Plenum Press, New York (USA) (1986)
3. Gaines, B.R.: Fuzzy and probability uncertainty logics. *Journal of Information Control* **38** (1978) 154–169
4. Baldwin, J.F., Lawry, J., Martin, T.P.: Mass assignment theory of the probability of fuzzy events. *Fuzzy Sets and Systems* **83** (1996) 353–367
5. Williamson, T.: Vagueness. Routledge, London (UK) (1994)
6. Zadeh, L.A.: The concept of a linguistic variable and its applications to approximate reasoning. *Information Sciences Part I:8; Part II:8, Part III:9* (1976) Part I:199–249; Part II:301–357, Part III:43–80
7. Baldwin, J.F., Lawry, J., Martin, T.P.: A mass assignment method for prototype induction. *International Journal of Intelligent Systems* **14** (1999) 1041–1070
8. Baldwin, J.F., Lawry, J.: A *c*-fuzzy means algorithm for prototype induction. In: Proceedings of FUZZ-IEEE2000. Volume 1. (2000) 164–169
9. Baldwin, J.F.: Logic programming with uncertainty and computing with words. In Martin, T.P., Arcelli Fontana, F., eds.: *Logic Programming and Soft Computing*. Research Studies Press (1998) 19–51
10. Duda, R.O., Hart, P.E., Stork, D.G.: *Pattern Classification*. 2nd edn. John Wiley & Sons, Inc., New York (USA) (2000)
11. Bezdek, J.C.: *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press (1981)
12. : UCI machine learning repository. (in world wide web)
<http://www.ics.uci.edu/~mlearn/MLRepository.html>.
13. Randon, N.J., Lawry, J.: Linguistic modelling using a semi-naïve bayes framework. In: Proceedings of IPMU2002. (2002)

A Recurrent Multivalued Neural Network for codebook generation in Vector Quantization

Benítez-Rochel, R.¹, Muñoz-Pérez, J.¹, and Mérida-Casermeiro, E.²

¹ Departamento de Lenguajes y Ciencias de la Computación,

² Departamento de Matemática Aplicada,

Universidad de Málaga

Complejo Tecnológico, Campus Universitario de Teatinos s/n, 29079 Málaga

munozp@lcc.uma.es, benitez@lcc.uma.es, merida@cetima.uma.es

Abstract. In this paper we propose a multivaluated recurrent neural network for vector quantization where the synaptic potential is given by a weighted sum of values of a function that evaluates the consensus between the states of the process units. Each process unit presents the state with the largest activation potential, that is, it depends on the state of the nearest process units (more strongly connected according to the synaptic weights). Like Hopfield network, it uses a computational energy function that always decreases (or remains constant) as the system evolves according to its dynamical rule based on an energy function that is equivalent to the distortion function of the vector quantization problem. It does not use tuning parameters and so it attains computational efficiency.

1 Introduction

Vector Quantization (VQ) is a well known technique that has been studied in a variety of contexts but most prominently for signal coding. In particular for speech coding and for image and video coding. In VQ the input space is divided into a number of distinct regions, and for each region a *reproduction* (reconstruction or prototype) *vector* is defined [5]. When the Euclidean distance is used as a similarity measure to decide on the region to which that input belongs, the quantizer is called *Voronoi quantizer*. However, practical use of VQ techniques has been limited because of the prohibitive amount of computation associated with existing algorithms. Artificial Neural Networks (ANN) with unsupervised learning have been successfully applied to pattern recognition and signal detection problems. One objective of using unsupervised learning is to define the classes or categories of the input data. These categories have to be discovered by the network on the basis of correlations or similarity measures. A number of competitive learning (CL) algorithms have been proposed for constructing VQ, [1, 3, 7, 9, 11]. The major objective of the CL algorithms is to effectively utilize the neural units as much as possible so that the average distortion for quantizing the input data can be minimized.

On the other hand, clustering is defined as the partitioning of data into classes with similar characteristics. This is done by allowing data samples with common attributes to be grouped into the same class. In [10] the relationship between clustering and vector quantization is shown; the problem of selecting a clustering based on the least sum of squares becomes the problem of selecting the reproduction codebook.

When the Euclidean distance is used then competitive neural networks can be used for clustering and the synaptic vectors give us the prototypes (centroids).

In this paper, we propose a recurrent neural networks that could be used with any similarity measure. Moreover, it evolves according to a dynamical rule so that the distortion function (computational energy function) always decreases (or remains constant).

This paper is organized as follows; in section 2 we briefly describe the basic theory of Vector Quantization. In section 3 we present our multivalued recurrent neural network in order to be applied to Vector Quantization. The effectiveness of the proposed model for a synthetic data set, real-world data (Anderson's IRIS data) and for uniformly generated data is shown in section 4. Finally, the conclusions are given in section 5.

2 Vector Quantization

Next a basic definition of VQ and the structural properties are presented. They are independent of any statistical considerations or distortion measures.

Definition 1

A vector quantizer Q of dimension k and size N is a mapping from a vector in a k -dimensional Euclidean space, \mathbb{R}^k , into a finite set $C = (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N)$ containing N outputs or reproduction points, called *code vectors* $\mathbf{y}_i \in \mathbb{R}^k$ for each $i \in \{1, 2, \dots, N\}$. The set C is called the *codebook*.

Given a sample $\mathbf{x}_j \in \mathbb{R}^k$, $j \in \{1, 2, \dots, n\}$, a quantizer with size N is optimal if it minimizes the distortion function between those input vectors \mathbf{x}_j and the reproduction vectors \mathbf{y}_i , $i \in \{1, 2, \dots, N\}$.

The problem of finding an optimal quantizer can be expressed as

$$\text{Minimize} \quad E(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N) = \sum_{i=1}^N \sum_{j \in R_i} \|\mathbf{x}_j - \mathbf{y}_i\|^2 \quad (1)$$

Each N points vector quantizer has associated a partition of \mathbb{R}^k into N disjoint and exhaustive regions or *cells*, R_i for $i \in \{1, 2, \dots, N\}$. The i th cell is defined by

$$R_i = \{\mathbf{x} \in \mathbb{R}^k : Q(\mathbf{x}) = \mathbf{y}_i\}, \quad (2)$$

For a given partition $\{R_i; i = 1, 2, \dots, N\}$, two necessary conditions to optimum code vector are (see [5]):

- C1) *Centroid condition*:

$$\mathbf{y}_i = \frac{\sum_{j \in R_i} \mathbf{x}_j}{|R_i|}$$

– C2) *Voronoi partition (competitiveness):*

For a fixed representative vectors, $(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N)$, the optimal partition should be constructed in such a manner that

$$Q(\mathbf{x}) = \mathbf{y}_i \Leftrightarrow \|\mathbf{x} - \mathbf{y}_i\| \leq \|\mathbf{x} - \mathbf{y}_r\|, \forall r \neq i$$

The problem (1) can be formulated in an alternative way and that is established in the following proposition:

Proposition 1

The problem (1) is equivalent to find the partition $\{R_1, R_2, \dots, R_N\}$ of the input vector set $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ that minimizes the expression

$$E = \sum_{i=1}^N \frac{1}{|R_i|} \sum_{\substack{r,s \in R_i \\ r < s}} \|\mathbf{x}_r - \mathbf{x}_s\|^2 \quad (3)$$

Proof:

It is easy to see that

$$\frac{1}{|R_i|} \sum_{\substack{r,s \in R_i \\ r < s}} \|\mathbf{x}_r - \mathbf{x}_s\|^2 = \sum_{j \in R_i} \|\mathbf{x}_j - \mathbf{y}_i\|^2.$$

In the following section a recurrent neural network is proposed in order to solve the above problem.

3 Multivalued Neural Network

3.1 Topology

The principal characteristics of our multivalued neural model \mathcal{H} are

- The state of the neuron i is characterized by its output. So, the global state of the network with n neurons is determined by its state vector (x_1, x_2, \dots, x_n) .
- The neurons outputs belong to a given set \mathcal{M} where \mathcal{M} can be \mathbb{R} , \mathbb{R}^L , $\{1, 2, \dots, N\}$ or even a symbolic set.
- The network is fully connected and a weight $w_{ij} \in \mathbb{R}$ is associated to each connection. The matrix of weights $\mathbf{W} = (w_{ij})$ is symmetric.
- Each state of the network has an associated energy given by

$$E = -\frac{1}{2} \sum_{i=1}^n \frac{1}{\sum_{j=1}^n S(x_i, x_j)} \sum_{j=1}^n w_{ij} S(x_i, x_j) \quad (4)$$

where w_{ij} measures the influence of neuron i into neuron j and the application $S : \mathcal{M} \times \mathcal{M} \rightarrow \mathbb{R}^+$ measures the matching between the outputs of neurons i and j .

The state of a neuron i indicates the group where the vector \mathbf{x}_i is assigned. So a configuration of the network $(x_1, x_2, \dots, x_i, \dots, x_n)$ reflects the group where each vector $\mathbf{x}_i \in \Re^k$ in the sample is assigned.

The outputs of the neurons belongs to the set $\mathcal{M} = \{1, 2, \dots, N\}$ where N is the size of the quantizer. It means that when $x_i = j$, $j \in \mathcal{M}$, then the network assigns the sample vector \mathbf{x}_i to the group j .

Each state of the network has an associated energy given by (4) where the function S is given by (5), it establishes the consensus between the state of the neurons.

$$S(x_i, x_j) = \begin{cases} 1 & \text{if } x_i = x_j \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

In next section a computational dynamics is proposed where the computational energy function decreases. The network will evolve using that computational dynamics in order to get a maximum decrease in the energy associated to the configuration of the network at each step.

If each configuration of the network is associated with a possible solution for the Vector Quantization problem then the network will look for a configuration that minimizes the expression (3). So, the synaptic weights are identified as $w_{ij} = -\|\mathbf{x}_i - \mathbf{x}_j\|^2$.

3.2 Computational Dynamics

Let $x_r(t)$ be the state of neuron r at time t , $x_r(t) \in \{1, 2, \dots, N\}$, and let $\mathbf{x}(t) = (x_1(t), x_2(t), \dots, x_n(t))$ be the global state of the recurrent network at time t . In order to define a dynamics two concepts are used: the synaptic potential h_r and the activation potential h_r^* of neuron r .

Definition 2

The *synaptic potential* h_r of neuron r is defined by the expression

$$h_r(x_1(t), x_2(t), \dots, x_n(t)) = \sum_{j=1}^n w_{rj} S(x_r(t), x_j(t))$$

where a pair of neurons r and j in the network are connected by a synaptic weight, w_{rj} , which specifies the contribution of the output signal $x_r(t)$ of neuron r to the synaptic potential acting on neuron j , and the function S is a function that measures the consensus between the states of the neurons.

Definition 3

The *activation potential* Δh_r of neuron r when $x_r(t) = b$, is defined by

$$\Delta h_r^b(t) = \frac{1}{1 + |C_b|} [h_r(t) - \theta_b]$$

where

$$\theta_b = \frac{1}{|C_b|} \sum_{i \in C_b} \frac{1}{2} h_i(t) \quad \text{and} \quad C_b = \{i \in \{1, 2, \dots, N\} : x_i(t) = b, i \neq r\}$$

Note that θ_b is a half of the mean synaptic potential of neurons in state b .

If neuron r is selected at time t , its state will be modified according to the deterministic rule.

$$x_r(t+1) = a \quad \text{si} \quad h_r^a(x_1(t), \dots, x_{r-1}(t), a, x_{r+1}(t), \dots, x_n(t)) = \quad (6)$$

$$\max_{b \in \mathcal{M}} h_r^b(x_1(t), \dots, x_{r-1}(t), b, x_{r+1}(t), \dots, x_n(t))$$

this rule that updates the unit is called *computational dynamics*.

The particular state of neuron r that satisfies the condition (6) is called the best matching for the input vector \mathbf{x}_r .

The selection of a neuron to perform updating is done randomly. The asynchronous (serial) updating procedure is continued until there no further changes to report. The state $\mathbf{x}(t) = (x_1(t), x_2(t), \dots, x_n(t))$ that satisfies the condition $x_i(t+k) = x_i(t)$, $\forall k > 1$, is called a *stable state* or *fixed point* of the phase space of the system.

3.3 Convergence

If the network is updated according to (6), the new configuration adopted will be a new state with less energy than the previous one. The network will evolve until a minimum of energy function is reached. It means that when the network is stabilized, then any change in one neuron will augment the value of the associated energy or will not cause any change in the previous energy.

Theorem 1

If the synaptic weight matrix is symmetric with null self-connections and S is given by (5) then the computational energy function decreases in each iteration when the network evolves according to the computational dynamics (6).

Proof:

If neuron r is updated at $t + 1$, then the energy associated to the new state of the network given in (4) can be rewritten as

$$\begin{aligned} -2E(t+1) &= \sum_{\substack{i=1 \\ i \neq r}}^n \sum_{\substack{j=1 \\ j \neq r}}^n w_{ij} \frac{S(x_i(t+1), x_j(t+1))}{\sum_{k=1}^n S(x_i(t+1), x_k(t+1))} + \\ &\sum_{\substack{i=1 \\ i \neq r}}^n w_{ir} \frac{S(x_i(t+1), x_r(t+1))}{\sum_{k=1}^n S(x_i(t+1), x_k(t+1))} + \sum_{j=1}^n w_{rj} \frac{S(x_r(t+1), x_j(t+1))}{\sum_{k=1}^n S(x_r(t+1), x_k(t+1))} \end{aligned}$$

Since $w_{ii} = 0 \ \forall i$ and $w_{ij} = w_{ji} \ \forall i, j$, the second and third terms at the right are equals. It can be noticed that when $S(x_i(t+1), x_r(t+1)) = 1$ then $\sum_{k=1}^n S(x_i(t+1), x_k(t+1)) = \sum_{k=1}^n S(x_r(t+1), x_k(t+1))$ (that is, denominators have same values in the above expression). So we have

$$-2E(t+1) = \sum_{\substack{i=1 \\ i \neq r}}^n \sum_{\substack{j=1 \\ j \neq r}}^n w_{ij} \frac{S(x_i(t+1), x_j(t+1))}{\sum_{k=1}^n S(x_i(t+1), x_k(t+1))}$$

$$+2 \sum_{j=1}^n w_{rj} \frac{S(x_r(t+1), x_j(t+1))}{\sum_{k=1}^n S(x_r(t+1), x_k(t+1))}$$

Moreover, as $x_i(t+1) = x_i(t)$, $\forall i \neq r$, then the first term at the right can be written as

$$\sum_{\substack{i=1 \\ i \neq r}}^n \sum_{\substack{j=1 \\ j \neq r}}^n w_{ij} \frac{S(x_i(t+1), x_j(t+1))}{\sum_{k=1}^n S(x_i(t+1), x_k(t+1))} = \sum_{\substack{i=1 \\ i \neq r}}^n \sum_{\substack{j=1 \\ j \neq r}}^n w_{ij} \frac{S(x_i(t), x_j(t))}{\sum_{k=1}^n S(x_i(t+1), x_k(t+1))}$$

Let C_a be the set defined by $C_a = \{i \neq r : x_i(t) = a\}$ and $C_b = \{i \neq r : x_i(t) = b\}$. Suppose that $x_r(t) = a$ and $x_r(t+1) = b$. Then we have that the only terms in the above expression updated from t to $t+1$ are

$$\begin{aligned} & \sum_{i \in C_a} \sum_{\substack{j=1 \\ j \neq r}}^n w_{ij} \frac{S(x_i(t), x_j(t))}{\sum_{k=1}^n S(x_i(t+1), x_k(t+1))} + \sum_{i \in C_b} \sum_{\substack{j=1 \\ j \neq r}}^n w_{ij} \frac{S(x_i(t), x_j(t))}{\sum_{k=1}^n S(x_i(t+1), x_k(t+1))} = \\ & \sum_{i \in C_a} \sum_{j \in C_a} w_{ij} \frac{S(x_i(t), x_j(t))}{|C_a|} + \sum_{i \in C_b} \sum_{j \in C_b} w_{ij} \frac{S(x_i(t), x_j(t))}{|C_b| + 1} \end{aligned}$$

Thus,

$$\begin{aligned} -2[E(t+1) - E(t)] &= \sum_{i \in C_a} \sum_{j \in C_a} w_{ij} \frac{S(x_i(t), x_j(t))}{|C_a|} + \sum_{i \in C_b} \sum_{j \in C_b} w_{ij} \frac{S(x_i(t), x_j(t))}{|C_b| + 1} \\ &\quad + 2 \sum_{j \in C_b} w_{rj} \frac{S(x_r(t+1), x_j(t+1))}{|C_b| + 1} \\ &\quad - [\sum_{i \in C_a} \sum_{j \in C_a} w_{ij} \frac{S(x_i(t), x_j(t))}{|C_a| + 1} + \sum_{i \in C_b} \sum_{j \in C_b} w_{ij} \frac{S(x_i(t), x_j(t))}{|C_b|} \\ &\quad \quad + 2 \sum_{j \in C_a} w_{rj} \frac{S(x_r(t), x_j(t))}{|C_a| + 1}] \\ &= \frac{1}{|C_a| + 1} \sum_{i \in C_a} \sum_{j \in C_a} w_{ij} \frac{S(x_i(t), x_j(t))}{|C_a|} - \frac{1}{|C_b| + 1} \sum_{i \in C_b} \sum_{j \in C_b} w_{ij} \frac{S(x_i(t), x_j(t))}{|C_b|} + \\ &\quad \frac{2}{|C_b| + 1} \sum_{j \in C_b} w_{rj} S(x_r(t+1), x_j(t+1)) - \frac{2}{|C_a| + 1} \sum_{j \in C_a} w_{rj} S(x_r(t), x_j(t+1)) \end{aligned}$$

Hence $\Delta E = E(t+1) - E(t) =$

$$= \frac{1}{|C_a| + 1} \left[h_r(t) - \frac{1}{2|C_a|} \sum_{i \in C_a} h_i(t) \right] - \frac{1}{|C_b| + 1} \left[h_r(t+1) - \frac{1}{2|C_b|} \sum_{i \in C_b} h_i(t+1) \right]$$

$$= \frac{1}{2}[\Delta h_r^a(t) - \Delta h_r^b(t+1)] \leq 0$$

Theorem 2

The recurrent network is stable and the stable states of the network are the local minima of the computational energy function.

Proof:

This recurrent network could oscillate between adjacent states with equal synaptic potential. However, if $x_i(t) = b, x_i(t+1) = a, b \neq a$ only when $\Delta h_i(t+1) > \Delta h_i(t)$, then the recurrent network is stable since the computational energy function can take only a finite number of values, N^n , at most, and decreases in each iteration. Moreover, if $(x_1(t), x_2(t), \dots, x_j(t), \dots, x_n(t))$ is a stable state but it is not a local minima then there exists an adjacent state $(x_1(t), x_2(t), \dots, x_j^*(t), \dots, x_n(t))$ with energy $E^*(t), E^*(t) < E(t)$. From theorem 1, we have $\Delta h_j^*(t) > \Delta h_j(t)$ and so the state $(x_1(t), x_2(t), \dots, x_j(t), \dots, x_n(t))$ is not stable, a contradiction.

4 Experimental results.

4.1 Iris data

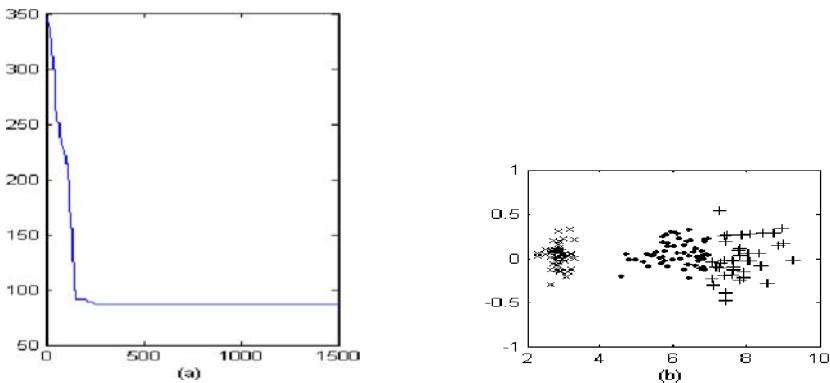


Fig. 1. (a)Evolution of the energy function. (b)IRIS data clustering with the network.

We use Anderson's IRIS data [2] as an experimental data set. Properties of the data are well known [4] and has been used in many papers to illustrate various clustering (unsupervised). In figure 1(b) we present the clustering obtained for IRIS data with a network constituted by 150 process units and 3 states. Typical error rates for unsupervised designs are around 15 'mistakes'. The network finds a clustering with 13 classification errors. Note that our network does not use learning parameters or prototypes (centroids). The algorithms with unsupervised learning such as k-means, Lloyd [8] or LBG [7] find clustering between 13 and 17 misclassifications. All errors occur in the overlapping region between Versicolor

and Virginica (see [12]) . In figure 1(a) we can see the decreasing energy function associated. It is interesting notice that the minimum energy is obtained after a few iterations that is because the considered dynamics provides the maximum diminution of the energy at each step.

5 Conclusions

We have proposed a multivaluated recurrent neural network for vector quantization where the synaptic potential is given by a weighted sum of synaptic weights of process units with the same value. This synaptic potential is used to define the activation potential associated to the neural units. The synaptic weights are the opposite values of the distance between the sample patterns. Each process unit presents the state with maximum activation potential and the system evolves according to this dynamical rule so that the computational energy function (distortion function) always decreases (or remains constant) and it does not use tuning parameters. The network has n process units where n is the sample size, and eventually reaches a stable state at a local minimum of the energy function, that is a local minimum of the distortion function in Vector Quantization problem. Moreover, the solution attained here forms the clusters or *cells* using the distances between the sample vectors, it does not use centroids.

References

1. Ahalt S.C., Krishnamurphy A.C., Chen P., and Melton D.E. Competitive learning algorithms for vector quantization. *Neural Networks*, **3** (1990) 277–290.
2. Anderson, E., The IRISes of the Gaspe Peninsula. *Bulletin of the American IRIS Society*, **59** (1939) 2–5
3. Dony R.D., and Haykin S. Neural Networks approaches to image compression. *Proceeding of the IEEE*, **83**(2) (1995) 288–303.
4. R. Duda and P. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.
5. Gersko A. and Gray R.M. *Vector Quantization*. Kluwer Academic Publishers (1992).
6. Gray R.M. Vector Quantization and signal compression *IEEE ASSP Magazine*, **1** (1980) 4–29.
7. Linde Y., Buzo A., and Gray R.M. An algorithm for vector quantizer design. *IEEE Trans. on Communication*, **28**(1) (1980) 84–95.
8. Lloyd, S.P. Least squares quantization in PCM's. *Bell Telephone Laboratories Paper*. Murray Hill, NJ, (1957).
9. Mao J. and Jain A.K. A self-organizing network for hyperellipsoidal clustering (HEC). *IEEE Trans. Neural Networks*, **7** (1996) 16–29.
10. Uchiyama T. and Arbib M.A. Color image segmentation using competitive learning. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **16**(12) (1994) 1197–1206.
11. Ueda N., and Nakano R. A new competitive learning approach based on a equidistortion principle for designing optimal vector quantizers. *Neural Networks*, **7**(8) (1980) 1211–1227.
12. Zahid, N., Abouelala, O., Limouri, M. Essaid, A. Unsupervised fuzzy clustering. *Pattern Recognition Letters*, **20** (1999) 123–129.

The recurrent IML-network

Joern Fischer

Fraunhofer Gesellschaft (AIS.INDY), Autonomous Intelligent Systems,
D-53754 Sankt Augustin, Germany
joern.fischer@ais.fhg.de

Abstract. Recurrent neural networks are still a challenge in neural investigation. Most commonly used methods have to deal with several problems like local minima, slow convergence or bad learning results because of bifurcations through which the learning system is driven. The following approach, which is inspired by Echo State networks [1], overcomes those problems and enables learning of complex dynamical signals and tasks.

1 Introduction

Learning in convergent recurrent neural networks, which is usually done by gradient descent, comes with various problems as there are local minima, slow convergence and unwanted bifurcations [2, 3]. Like the Echo State approach [1] the recurrent IML-network overcomes these problems, but it comes with some new properties: The hidden layer construction is simplified and does not have to be adjusted to prevent unwanted dynamical behaviour, the information flow inside of the hidden layer is easier to investigate and an efficient output learning with the Delta learning rule is enabled.

At first a constructive algorithm is presented in analogy to Echo State networks. Based on this approach a hidden layer adaptation is introduced which accelerates the learning process with the Delta rule significantly. Finally some examples are presented.

2 The networks architecture

The recurrent IML-network is a three layered network with a large number of hidden layer neurons (IML = Infinite Middle Layer) with sigmoid transfer function, fixed random input weights, and unidirectional lateral connections (fig.1). The first neuron may have connections to all others, the second one to all others but the first, the third to all others but the first and the second, etc. which means that only the weight values above the diagonal of the weight matrix may be different from zero (no self connections). The only recurrent connections are those from the linear output neurons back to the hidden neurons of the network. The hidden layer itself has no recurrent connections and therefore may not have any periodic or chaotic behavior.

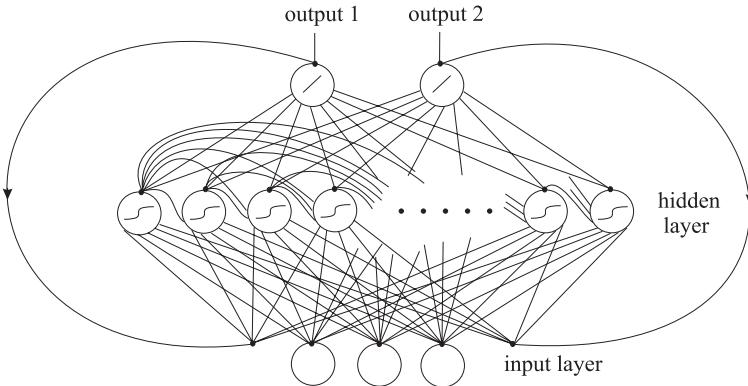


Fig. 1. a) The architecture of the recurrent IML-network. The hidden layer neurons have lateral connections only in one direction: The first hidden layer neuron may be connected to all others, the second one to all others but the first etc.. With this connectivity structure the information of an incoming signal is kept for some time in the hidden layer and past information may be used to make future predictions. The only recurrent connections of the IML-network are those from the output back onto the hidden layer. Using these connections an output may be generated without applying an input to the network.

The hidden layer neurons are activated synchronously according to the following equation:

$$\varphi_j^{t+1} = \varphi_j^t + \frac{1}{\tau_j} \left(\sum_{i=1}^n w_{ij}^t o_i^t - \varphi_j^t \right) \quad (1)$$

$n \in \mathbb{N}$ the number of synapses

$w_{ij} \in \mathbb{R}$ connection strength between neuron i to neuron j at time $t \in \mathbb{N}$

$o_i^t \in \mathbb{R}$ is the output activity of the presynaptic neuron i at time $t \in \mathbb{N}$

$\tau_j \in \mathbb{R}$ is the time constant of neuron j

The output activity at the axon as a function of φ_j^t is given by the transfer function:

$$o_j^t(\varphi_j^t) = \frac{2}{1 + e^{-2(\varphi_j^t - \theta_j)}} - 1 = \tanh(\varphi_j^t - \theta_j) \quad (2)$$

$o_j^t \in \mathbb{R}$ is the output of the postsynaptic Neuron j at time t

$\theta_j \in \mathbb{R}$ is the neurons threshold (const.)

The time constant τ_j is problem dependent and should be chosen with respect to the task the network should fulfill. If it is not explicitly given in the following examples it is set to $\tau_j = 1.0 \forall j$. All thresholds and all weights of the hidden layer neurons, are random valued, but fixed. As in Echo State networks the lateral connectivity should be sparse to get good results. In the following examples only about 5% of the lateral connections are different from zero. If there are too few

connections the incoming information may not be kept inside the hidden layer, while if there are too many connections statistical behaviour disturbs the lateral information flow. With a sparsely connected hidden layer the information of the actual input as well as the information of some past input activations may be kept inside the network and serves as a kind of information reservoir on which an output may be learned.

3 Training and optimal weights

If the hidden layer is seen as a fixed structure and the output is learned on the activation of this structure local minima do not exist. Each input episode has its specific episode of hidden layer activations which is a trajectory in the high dimensional activation space. On this space a linear mapping, as it is performed by our linear output neurons, may be calculated with the method of linear regression known as the "least mean squares" algorithm which works as follows: For each time step of an input episode the n hidden layer neurons have a defined output activity. For each activity a target output is claimed. To minimize the sum of the square errors $\sum_t (o_j^t - \text{target}_j^t)^2$ between the real output of an output neuron j and the target value for all times t the derivative of the sum of the square error functions is calculated and set equal to zero as follows:

$$\frac{\partial \sum_t (\sum_i o_i^t w_{ij} - \text{target}_j^t)^2}{\partial w_{kj}} = \sum_t 2o_k^t (\sum_i o_i^t w_{ij} - \text{target}_j^t) = 0 \quad (3)$$

Where k and i are the hidden layer neuron number. For $k = 1..n$ a system of n linear equations results which may be solved e.g. with the Gauss elimination algorithm [4]. It is not necessary to calculate the second derivative to decide whether the weights define a sattle point a minimum or a maximum of the sum of the error functions. The sum of the error functions has no sattle point, because it is of parabolic shape, it has no maxima, because it has a positive sign, so the result must be a minimum. This algorithm always adapts the weights with an expense of less than $O(n^3)$ to an absolute minimum of the calculated square errors. Networks with very large hidden layers lead to relatively high calculation times. On the other hand the existence of a global minimum is guaranteed and may be found directly.

There are two further improvements to consider to get good results. The first one is that during the training, instead of feeding back the real output to the hidden layer, one feeds back the corresponding target output. Anyway, the real output of the network is not known before the weights to the output neurons are calculated. The second improvement is that adding noise to the recurrent connection helps to stabilize the trajectory which should be learned.

4 A "sine function generator" example

In the first example the network should learn to produce a sine function. The hidden layer contains 50 neurons. The network has no input nodes and one

output node. The fixed input and lateral weights are random valued with values of the range $[-1..1]$. During the training of 5000 time steps (least mean squares algorithm) a noise signal of amplitude $\pm 1/200$ is induced to enable a convergent dynamics. Fig. 2 shows the resulting signal after the training of a sine function of the period length $p = 30$ time steps. The signal is stable over the whole tested time of 5000 time steps. Sine functions with other period lengths also have been tested successfully.

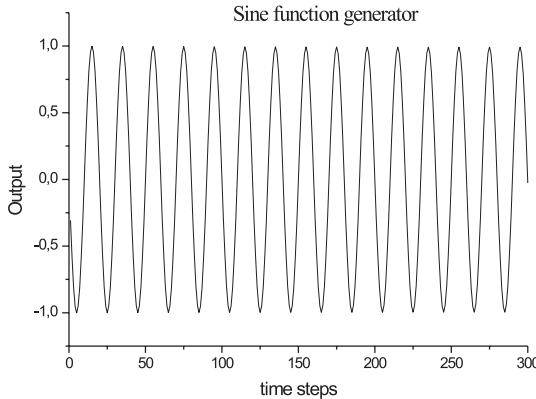


Fig. 2. A sine function of period length $p = 30$ time steps generated by the recurrent IML-network. The signal is stable over the whole tested time of 5000 time steps with a prediction mean error of $\langle e \rangle = 5.011110^{-5}$ and a mean square error of $\langle e^2 \rangle = 4.988410^{-7}$ per time step.

5 The "Lorenz attractor" example

To learn a chaotic attractor dynamics is a goal which is reached in the following example, where the network should learn the commonly known Lorenz attractor. The network with 50 hidden layer neurons has no input nodes, but three output nodes with recurrent connections back into the hidden layer. The fixed hidden layer weights are like before random valued of the range $[-1..1]$.

The Lorenz-equations are the following:

$$\frac{dx}{dt} = -\sigma(x - y) \quad (4)$$

$$\frac{dy}{dt} = rx - y - xz \quad (5)$$

$$\frac{dz}{dt} = b(xy - z) \quad (6)$$

The constants are $\sigma = 10$, $b = 8/3$, $r = 50.3$, while x , y and z are used as desired output. The training data of 5000 points was produced with the Euler-method with a step width of 0.01. For training white noise of the amplitude $\pm 1/6$ was added to the "target" value which is used for training instead of the recurrent connection to prevent the network from leaving the area around the attractor. The result of the network trained with the least mean squares algorithm is shown in fig.3.

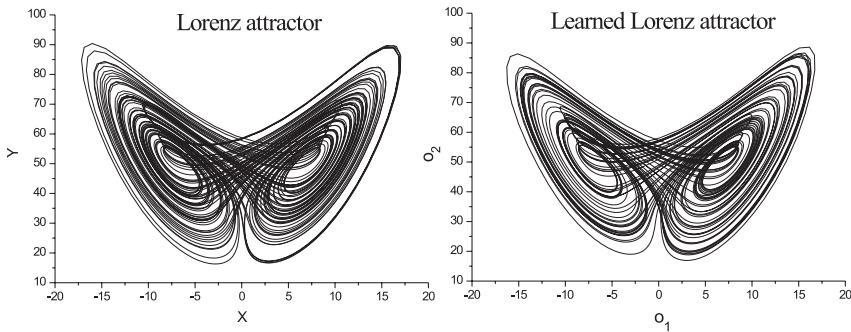


Fig. 3. Left: the Lorenz attractor in the X-Z-plane, right: the attractor produced with the recurrent IML-network. The trajectory is similar to the original chaotic Lorenz attractor. Although the starting point is the same, small errors of the learned attractor basin lead to different trajectories. The prediction for one time step has a mean error of $\langle e \rangle = 1.465210^{-3}$ and a mean square error of $\langle e^2 \rangle = 7.158610^{-4}$.

6 Introducing a hidden layer adaptation

The same global error minimum as calculated with the least mean squares algorithm may be reached by using the Delta learning rule also known as the Widrow-Hoff rule [5]:

$$\Delta w_{ij} = \eta o_i (o_{target} - o_j) \quad (7)$$

$\eta \in [0..1]$ is the learning rate

The problem with this rule is that its convergence properties are highly dependent on the dynamics of the hidden layer on which the output is learned. A general assertion says, that the spectrum of the eigenvalues of the covariance matrix of the hidden layer activity should be flat to obtain a fast convergence with the delta learning rule [6]. A typical spectrum of the eigenvalues has an

exponential distribution which implies that the gradient descent with a constant learning rate has different velocities in different directions. A network with flat eigenvalue spectrum adapts the weights on a direct way towards the optimum.

A network with a neural activation which is widely uncorrelated and whose neurons have nearly the same variances would fulfill the constraint of a flat eigenvalue spectrum. To make the neurons activity uncorrelated we introduce two things: An adaptation of the lateral weights with anti-hebbian learning like in [7] ($\Delta w_{ij} = -\eta o_i o_j$) and different time constants for all hidden layer neurons, because signals with different frequencies are in general uncorrelated. In our example the first neuron, which is connected to all others, has the lowest time constant and the last one has the highest time constant and the time constant τ falls with the neuron number $n \in [1..m]$ with $\tau_m = m + 1 - n$.

To obtain a flat eigenvalue distribution there are two things left to do: The threshold of each hidden layer neuron has to be adapted to get the same mean value for all neural outputs and the input weights should be adapted to equalize the variances of all hidden layer neurons. The mathematical formulation for the threshold adaptation may be written as follows:

$$\Delta \Theta_j = \begin{cases} +\eta & : o_j < 0 \\ -\eta & : o_j \geq 0 \end{cases} \quad (8)$$

The input weight adaptation for positive synapses may be written as

$$\Delta W_{ij+} = \begin{cases} \eta o_j & : -0.5 < o_j < 0.5 \\ -\eta o_j & : \text{else} \end{cases} \quad (9)$$

and for negative synapses as

$$\Delta W_{ij-} = \begin{cases} -\eta o_j & : -0.5 < o_j < 0.5 \\ \eta o_i & : \text{else} \end{cases} \quad (10)$$

The advantage of such a hidden layer adaptation is that it is a self organizing process which enhances learning capabilities without being specialized on a specific target. The whole adaptation only uses the presented input signals and tends to find a basis on which any target function may be learned much faster. The following figure (Fig.4) shows the spectra of the eigenvalues of the covariance matrix of the hidden layer activity with and without weight adaptation for different input function parameters. The lateral hidden layer connections and their input weights were trained during 5000 time steps with decreasing learning rate $\eta = 10/(t + 1000)$, where $t \in [1..5000]$ denotes the time step. Independent from the frequency of the input signal the eigenvalue spread is much smaller after training the network. The performance of the network with adaptable hidden layer weights is shown below on the sine function generator example.

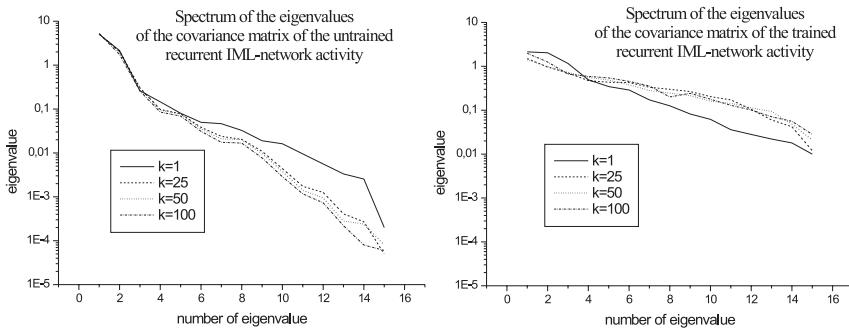


Fig. 4. Typical spectra of the eigenvalues of the covariance matrix of the networks activity without (left) and with learning (right) the lateral connections. The network consists of 15 hidden layer neurons and is trained with the input function $f(t) = \sin(t/k)$.

7 The "sine generator" with hidden layer adaptation

The following example shows the sine function generator as shown in section 4 but by using the Delta rule to adapt the weights to the output neuron. The hidden layer contains only 15 hidden layer neurons and is adapted to accelerate the target learning process. Fig.5 shows the networks output after a training of 10000 time steps. The target functions are two sine functions with different period lengths. The recurrent IML-network with hidden layer adaptation is compared to the same network without hidden layer adaptation and it is shown, that the network with adaptation produces a stable sine function over the whole test period of 5000 time steps, while the one without hidden layer adaptation fails in producing a stable sine function. The mean square error of a prediction after the training for the network with adapted hidden layer is $\langle e^2 \rangle \approx 0.0075$ for the period length $p = 10$ and $\langle e^2 \rangle \approx 0.0039$ for $p = 30$, while the network without hidden layer adaptation has a higher mean square error of $\langle e^2 \rangle \approx 0.0954$ for $p = 10$ and $\langle e^2 \rangle \approx 0.0436$ for $p = 30$.

8 Conclusion

The recurrent IML-network enables a learning of time dependent signals and tasks. It performs extremely well with the least mean squares algorithm which calculates the optimal output weights. With the Delta rule the same optimum is reached, but it is much worse in performance. To make the network able to converge faster, the lateral weight adaptation and the adaptation of the hidden layer weights lead to a hidden layer activation with a covariance matrix with flat eigenvalue distribution which induces a boost in the networks learning performance.

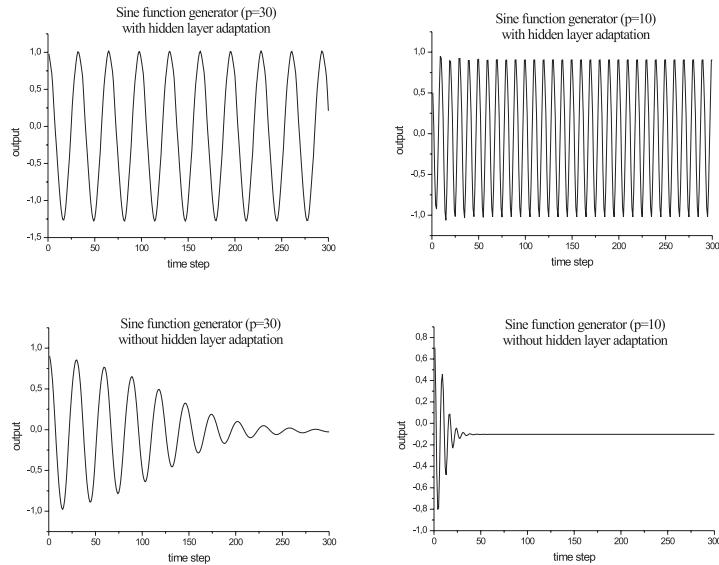


Fig. 5. The recurrent IML-network as a sine function generator. Top: the recurrent networks output after a hidden layer adaptation and a target training of 10000 time steps with a learning rate of $\eta = 0.2$. Bottom: the recurrent networks output without hidden layer adaptation after a training of 10000 time steps. The signal of the network without hidden layer adaptation is unstable, while the one with hidden layer adaptation is stable over the whole test period of 5000 time steps. At time step 0 the recurrent input connection is switched directly from the target values to the output values the network produces itself. This may lead to irregularities at the beginning of the produced sine functions.

References

1. Jäger, Herbert, "Output-only learning in recurrent neural networks", GMD- techreport, Schloss Birlinghoven, Sankt Augustin, Germany, October 14, 2000
2. Atiya A.F., Parlos A.G., "New results on recurrent neural network training: Unifying the algorithms and accelerating convergence", IEEE Trans. Neural Networks, 11(3): 697-709, 2000
3. Doya K., "Recurrent neural networks: Supervised learning", The Handbook of Brain Theory and Neural Networks, pages 796-800. MIT Press - Bradford Books, 1995
4. Sedgewick R., "Algorithmen in C++", Addison-Wesley, München, 1992
5. Rumelhart J.L., "Parallel Distributed Processing: Explorations in the Microstructure of cognition, Volume 1: Foundations", MIT Press, 1986
6. Fahrhang-Boroujeny B., "Adaptive Filters Theory and Application", National University of Singapore, John Wiley & Sons, Chichester, England, 1998
7. J.Rubner, K.Schulten, P.Tavan, "A Self-Organizing Network for complete Feature Extraction", Parallel Processing in Neural Systems and Computers, pages 365-368. Elsevier, Amsterdam, 1990

Estimation of Multidimensional Regression Model with Multilayer Perceptrons

Joseph Rynkiewicz

Université de Paris I,
SAMOS-MATISSE, 90 rue de tolbiac,
75013 Paris, France

Abstract. This work concerns estimation of multidimensional nonlinear regression models using multilayer perceptron (MLP). For unidimensional data, the ordinary least squares estimator matches with the Gaussian maximum likelihood estimator. However, in the multidimensional case, the Gaussian maximum likelihood estimator minimize the determinant of the empirical error's covariance matrix. This paper is devoted to the study of this estimator using a MLP. In particular, we show how to modify the backpropagation algorithm to minimize such cost function and we give heuristic explanations in favor of the use of such function in the multidimensional case.

1 Introduction

Consider a sequence $(Y_t, Z_t)_{t \in \mathbb{N}}$ of random vectors, where $Y_t \in \mathbb{R}^d$, and $Z_t \in \mathbb{R}^{d'}$ (d and d' are positive integer) verifying

$$Y_t = F_{W_0}(Z_t) + \varepsilon_t \quad (1)$$

where

- F_{W_0} is a function represented by a MLP with parameters or weights W_0 .
- (ε_t) is an i.i.d. centered noise with unknown invertible covariance matrix Γ_0 .

Our goal is to estimate the true parameter by minimizing an appropriate cost function. This model is called a regression model and a popular choice for the associated cost function is the mean squares error :

$$V_n(W) := \frac{1}{n} \sum_{t=1}^n \|Y_t - F_W(Z_t)\|^2 \quad (2)$$

where $\|\cdot\|$ denote the Euclidean norm on \mathbb{R}^d . The weights minimizing this cost function : the ordinary least squares estimator, had been widely studied and if the observations $(Y_t)_{t \in \mathbb{N}}$ are scalar, this estimator matches with the Gaussian maximum likelihood estimator. However, this is not the case if the observations Y_t are d -dimensional with $d \geq 2$.

Indeed, when F_W is a linear function it is well known that the ordinary least square error is a sub-optimal estimator since the best linear unbiased estimator¹ is

$$\bar{W}_n = \arg \min_W \frac{1}{n} \sum_{t=1}^n (Y_t - F_W(Z_t))^T \Gamma_0^{-1} (Y_t - F_W(Z_t)) \quad (3)$$

where X^T denote the transpose of vector X and Γ_0^{-1} the inverse of Γ_0 . In general, the covariance matrix Γ_0 is unknown and we have to estimate this matrix in order to get a better estimator of the weights. For example, Gallant [2] considers the generalized least squares :

$$G_n(W, \Gamma) := \frac{1}{n} \sum_{t=1}^n (Y_t - F_W(Z_t))^T \Gamma^{-1} (Y_t - F_W(Z_t)), \quad (4)$$

assuming that Γ is a good approximation of the true covariance matrix of the noise Γ_0 . A possible way to construct a sequence of $(\Gamma_k)_{k \in \mathbb{N}^*}$ yielding a good approximation of Γ_0 is the following : using the ordinary least squares estimator \hat{W}_n , the noise covariance can be approximated by

$$\Gamma_1 := \Gamma(\hat{W}_n) := \frac{1}{n} \sum_{t=1}^n (Y_t - F_{\hat{W}_n}(Z_t))(Y_t - F_{\hat{W}_n}(Z_t))^T.$$

then, we can use this new covariance matrix to find a generalized least squares estimate \hat{W}_n^2 :

$$\hat{W}_n^2 = \arg \min_W \frac{1}{n} \sum_{t=1}^n (Y_t - F_W(Z_t))^T (\Gamma_1)^{-1} (Y_t - F_W(Z_t))$$

and calculate again a new covariance matrix

$$\Gamma_2 := \Gamma(\hat{W}_n^2) = \frac{1}{n} \sum_{t=1}^n (Y_t - F_{\hat{W}_n^2}(Z_t))(Y_t - F_{\hat{W}_n^2}(Z_t))^T.$$

It can be shown that this procedure gives a sequence of parameters

$$\hat{W}_n \rightarrow \Gamma_1 \rightarrow \hat{W}_n^2 \rightarrow \Gamma_2 \rightarrow \dots \quad (5)$$

achieving a local maximum of the Gaussian log-likelihood.

However, if we consider the whole parameter (W, Γ) , such procedure will be useless because we can directly maximize the Gaussian log-likelihood by minimizing the logarithm of the determinant of the empirical covariance matrix :

$$T_n(W) := \log \det \left(\frac{1}{n} \sum_{t=1}^n (Y_t - F_W(Z_t))(Y_t - F_W(Z_t))^T \right). \quad (6)$$

¹ This estimator is called BLUE

$T_n(W)$ is called the concentrated Gaussian log-likelihood but, naturally, it can be used even if the noise is non-Gaussian.

This paper is devoted to the study of this cost function in the framework of the MLP models. It is organized as follow :

In the second section, we introduce $W_n^* := \arg \min_W T_n(W)$, the weights minimizing the cost function $T_n(W)$. We show how to construct a numerical algorithm to approximate this estimator thanks a modification of the backpropagation algorithm.

In the third section we give heuristic arguments in favor of the use of this estimator when the covariance matrix of the noise is not the identity

In the fourth section we compare the performance of this estimator with the ordinary least square estimator on a simulated example.

2 Minimization of $T_n(W)$

We introduce first some notations :

1. For a $d \times d$ matrix Γ , let $(\Gamma_{ij})_{1 \leq i,j \leq d}$ be the vector $(\Gamma_{11}, \Gamma_{12}, \dots, \Gamma_{1d}, \Gamma_{21}, \dots, \Gamma_{2d}, \Gamma_{31}, \dots, \Gamma_{dd})$.
2. If X is a multidimensional vector, let $X(i)$ be the i th element.
3. If A is an non singular matrix and A^{-1} its inverse, let a_{ij}^{-1} be the coefficients of A^{-1} .
4. let $tr(A)$ be the sum of diagonal element of the matrix A .

The observations are the data $(y_t, z_t)_{1 \leq t \leq n}$, and we want estimate the parameters \hat{W}_n^* minimizing $T_n(W) = \log \det(\frac{1}{n} \sum_{t=1}^n (y_t - F_W(z_t))(y_t - F_W(z_t))^T)$. As usual, we cannot find exact solution to such problem. However, we can get a good approximation of the solution with differential optimization. This involve the calculus of the gradient with respect to the weights of the MLP of the cost function which is performed thanks the following modified backpropagation algorithm.

2.1 Calculus of the derivative of $W \mapsto T_n(W)$:

If $T_n(W)$ is a matrix depending of the parameter vector W , we get From Magnus and Neudecker [5]

$$\frac{\partial}{\partial W_k} \ln \det(\Gamma_n(W)) = tr \left(\Gamma^{-1} \frac{\partial}{\partial W_k} \Gamma_n(W) \right)$$

Hence, if $\Gamma_n(W) = \sum_{t=1}^n (y_t - F_W(z_t))(y_t - F_W(z_t))^T$, the derivative of $\ln(\det(\Gamma_n(W)))$ with respect to the weight W_k is :

$$\frac{\partial}{\partial W_k} (\ln(\det(\Gamma_n(W)))) = (\Gamma_{ij}^{-1})_{1 \leq i,j \leq d}^T \left(\frac{\Gamma_{ij}}{\partial W_k} \right)_{1 \leq i,j \leq d}$$

with

$$\frac{\partial \Gamma_{ij}}{\partial W_k} =$$

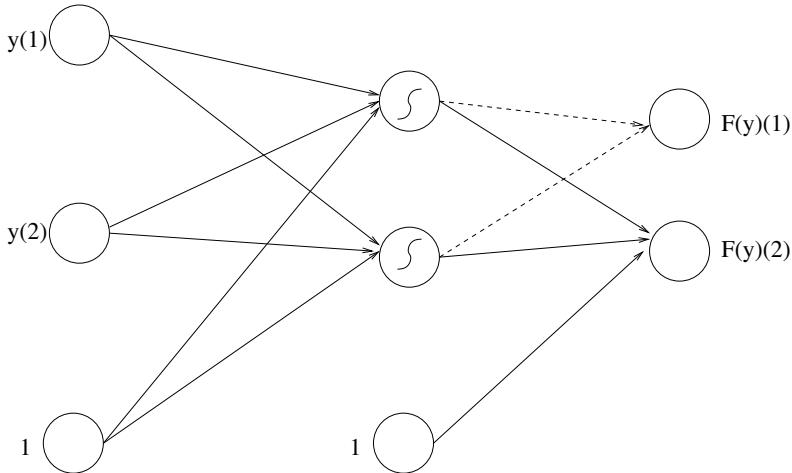
$$\frac{1}{n} \sum_{t=1}^n \left[-\frac{\partial F_W(z_t)(i)}{\partial W_k} \times (y_t - F_W(z_t))(j) - \frac{\partial F_W(z_t)(j)}{\partial W_k} (y_t - F_W(z_t))(i) \right] \quad (7)$$

so

$$\begin{aligned} \frac{\partial}{\partial W_k} (\ln(\det(T_n(W)))) &= \frac{1}{n} (\Gamma_{ij}^{-1})_{1 \leq i,j \leq d}^T \times \\ \left(\sum_{t=1}^n -\frac{\partial F_W(z_t)(i)}{\partial W_k} \times (y_t - F_W(z_t))(j) - \frac{\partial F_W(z_t)(j)}{\partial W_k} (y_t - F_W(z_t))(i) \right)_{1 \leq i,j \leq d} . \end{aligned} \quad (8)$$

The quantity $\frac{\partial F_W(z_t)(i)}{\partial W_k}$ is computed by backpropagating the constant 1 for the MLP restricted to the output i . The figure 1 give a example of a MLP restricted to the output 2.

Fig. 1. MLP restricted to the output 2 : the plain lines



Hence, the calculus of the gradient of $T_n(W)$ with respect to the parameters of the MLP is straightforward. We have to compute the derivative with respect to the weights of each single output MLP extracted from our MLP by backpropagating the constant value 1. Then, according to the formula (7), we can easily compute the derivative of each terms of the empirical covariance matrix of the noise. Finally the gradient is obtained by the sum of all the derivative terms of this empirical covariance matrix pondered by the terms of it's inverse as in formula (8).

2.2 Differential optimization

Using the previous calculus, we can apply one of the numerous techniques of differential optimization to find a local minimum of the cost function $T_n(W)$. We can find a comprehensive review of such techniques in Press et al. [6] and we recommend especially the BFGS algorithm, which is a very fast quasi-newton algorithm.

We note that the calculus of the gradient of $T_n(W)$ is more complex than the calculus of derivatives with the classical mean square criteria, but, in general, optimization algorithms are associated with minimization along a line like Brent's method and the derivatives of the cost function are less evaluated than the function itself. So, finally, the minimization of $T_n(W)$ is only slightly more costly than the minimization of the ordinary mean square criteria.

3 Heuristics on the efficiency of \hat{W}_n^*

Under suitable assumptions, see for example Sussmann [7], the model admit a theoretical MLP with an optimal parameter W_0 defined up to a permutation of the weights. Under weak conditions the estimator \hat{W}_n^* converges almost surely to the true parameter, see for example Gourieroux et al. [3]. This result of consistency holds also for the ordinary least square estimator \hat{W}_n , see for example White [8].

The differences between \hat{W}_n^* and \hat{W}_n is the speed of convergence or more precisely the variance of the two estimator in function of the number of observation. The better estimator is the estimator with the smallest variance.

First of all, we have to remark that, if the density of the noise is really Gaussian, the estimator \hat{W}_n^* is asymptotically efficient as maximum likelihood estimator. This properties imply that no other consistent estimator can achieve a better variance asymptotically. Gaussian assumption of the noise can appear to be a strong assumption but it is justified by the theorem central limit and the fact that the noise is generally the sum of a lot of random effects.

Moreover, as we have seen in the introduction, maximum Gaussian likelihood estimator matches with the limit estimator of the iterated generalized least square (cf equation (5)). So it matches with the generalized least square estimator with the best approximation of Γ_0 we can get from the observations $(y_t, z_t)_{1 \leq t \leq n}$.

If the regression function is linear the generalized least square estimator using Γ_0 is optimal (BLUE), in the non linear case we have the same property but only asymptotically.

Indeed, if $\psi(W_0)$ is the Jacobian matrix of the MLP function with respect to the true weights, E is the expectation with respect to the true distribution of the data (Y, Z) and \hat{W}_n^Γ are the weights minimizing cost function $G_n(W, \Gamma)$ (see equation (4)), we get from Yao [9] :

$$\lim_{n \rightarrow \infty} \sqrt{n} (\hat{W}_n^\Gamma - W_0) \xrightarrow{n \rightarrow \infty} \mathcal{N}(0, \Phi_\Gamma)$$

with

$$\begin{aligned}\Phi_{\Gamma} &= [E(\psi(W_0) \Gamma^{-1} \psi^T(W_0))]^{-1} \times [E(\psi(W_0) \Gamma^{-1} \Gamma_0 \Gamma^{-1} \psi^T(W_0))] \\ &\quad \times [E(\psi(W_0) \Gamma^{-1} \psi^T(W_0))]^{-1}\end{aligned}$$

Now it is straightforward to establish (see for example Ljung [4]), that

$$\forall \Gamma : \Phi_{\Gamma} \geq \Phi_{\Gamma_0}$$

where the inequality is the standard inequality for definite positive matrices².

So the less asymptotically variant estimator is obtained when we use the true covariance of the noise to compute the generalized least square criterion. It seems finally natural to use the best estimation of the this true covariance matrix that we can achieve from the data, and so to use the cost function $T_n(W)$.

4 Simulated example

Although the estimator associated to the cost function $T_n(W)$, is theoretically better than the ordinary mean least square estimator we have to confirm this fact on simulation. Indeed, their are some pitfalls in practical situations with MLP.

The first point is that we have no guaranty to reach the global minimum of the cost function, we can only hope to find a good local minimum especially if we are using many estimations with different initials weights.

The second point, is the fact that MLP are black box, it means that it is difficult to give an interpretation of their parameters and it is almost impossible to compare MLP by comparing their parameters even if we try to take into account the possible permutations of the weights.

All these reasons explain why we choose to compare the estimated covariance matrices of the noise instead of compare directly the estimated parameters of MLP.

4.1 The model

To simulate our data, we use a MLP with 2 inputs, 3 hidden units, and 2 outputs. We choose to simulate a time series because it is very easy task as the outputs at time t are the inputs for the time $t+1$. Moreover, with MLP, the statistical properties of such model are the same than with independent identically distributed (i.i.d.) data.

The equation of the model is the following

$$Y_{t+1} = F_{W_0}(Y_t) + \varepsilon_{t+1}$$

where

² We say $\Phi_{\Gamma} \geq \Phi_{\Gamma_0}$ if and only if $\Phi_{\Gamma} - \Phi_{\Gamma_0}$ is a positive semidefinite matrix

- $Y_0 = (0, 0)$.
- $(Y_t)_{1 \leq t \leq 1000}$, $Y_t \in \mathbb{R}^2$, is the bidimensional simulated random process
- F_{W_0} is a MLP function with weights W_0 chosen randomly between –2 and 2.
- (ε_t) is an i.i.d. centered noise with covariance matrix $\Gamma_0 = \begin{pmatrix} 1.81 & 1.8 \\ 1.8 & 1.81 \end{pmatrix}$.

In order to study empirically the statistical properties of our estimator we make 10 independent simulations of the bidimensional times series of length 1000.

On each time series we estimate the weights of the MLP using the cost function $T_n(W)$ and the ordinary least square estimator (*MCO*). The estimations have been done using the second order algorithm BFGS, and for each estimation we choose the best results obtained after 20 random initializations of the weights. Doing so, we avoid to plague our learning with poor local minima.

We show here the mean of estimated covariance matrices of the noise for the different estimators:

$$T_n(W) : \begin{pmatrix} 1.793 & 1.785 \\ 1.785 & 1.797 \end{pmatrix} \text{ and } MCO : \begin{pmatrix} 1.779 & 1.767 \\ 1.767 & 1.783 \end{pmatrix}$$

the estimated standard deviation of the terms of the matrices are all equal to 0.003, so the differences observed between the two matrices are statistically significant. We can see that the estimated covariance of the noise is in mean better with the estimator associated to the cost function $T_n(W)$, in particular it seems that there is slightly less overfitting with this estimator, and the non diagonal terms are greater than with the estimator associated with the *MCO*. Indeed, as expected, the determinant of the mean matrix associated to $T_n(W)$ is 0.036 instead of 0.050 for the matrix associated to the *MCO*.

5 Conclusion

In the multidimensional case the ordinary least square estimator are often sub-optimal if the covariance matrix of the noise is not the identity matrix. In seeking to take into account the covariance matrix of the noise we find that it is natural to use the concentrated log-likelihood as cost function. We have shown that the differential minimization of this cost function is easy with MLP, since we can compute the gradient of this function tanks a modification of the backpropagation algorithm. Finally the theoretical advantages of this estimator have been verified on a simulation and we can expect a amelioration of the learning process in using this cost function. Even if this amelioration is small, it can be very important to improve the variance of the parameter especially when we are using pruning techniques based on this variance like the SSM algorithm of Cottrell et al. [1].

References

1. Cottrell, M., Girard, B., and Y., Mangeas, M., Muller, C.: Neural Modeling for Time Series : a Statistical Stepwise Method for Weight Elimination. *IEEE Trans on Neural Networks* **6:6** (1995) 1355–1364
2. Gallant, R.: Non linear statistical models. J. Wiley and Sons (1987)
3. Gourieroux, C., Monfort, A., Trognon, A.: Pseudo maximum likelihood methods: Theory. *Econometrica* **52:3** (1984) 681–700
4. Ljung, L.: System identification : Theory for the user. Prentice Hall (1999)
5. Magnus, J., Neudecker, H.: Matrix differential calculus with applications in statistics and econometrics. J. Wiley and Sons (1988)
6. Press, W., Flannery, B., Teukolsky, S., Vetterling, W.: Numerical recipes in C : The art of scientific computing. Cambridge University Press (1992)
7. Sussmann, H.: Uniqueness of the wieghts for minimal feedforward nets with a given input-output map. *Neural Networks*, **5** (1992) 589–593
8. White, H.: Artificial neural networks. Blackwell (1992)
9. Yao, J.F.: On least square estimation for stable nonlinear AR processes. *The Annals of Institut of Mathematical Statistics* **52** (2000) 316-331

Principal Components Analysis Competitive Learning

Ezequiel López-Rubio, José Muñoz-Pérez, José Antonio Gómez-Ruiz

Department of Computer Science and Artificial Intelligence. University of Málaga.
Campus de Teatinos, s/n. 29071 Málaga.
SPAIN
`{ezeqlr, munozp, janto}@lcc.uma.es`

Abstract. We present a new neural model, which extends the classical competitive learning (CL) by performing a Principal Components Analysis (PCA) at each neuron. This model represents an improvement with respect to known local PCA methods, because it is not needed to present the entire data set to the network on each computing step. This allows a fast execution, while retaining the dimensionality reduction properties of the PCA. Furthermore, every neuron is able to modify its behaviour to adapt to the local dimensionality of the input distribution. Hence our model has a dimensionality estimation capability. Experimental results are presented, which show the dimensionality reduction capabilities of the model with multisensor images.

1 Introduction

Two of the best known techniques for multidimensional data analysis are vector quantization (VQ) and Principal Components Analysis. Our aim here is to combine them to obtain a new neural model.

Vector quantization is a method for approximating a multidimensional signal space with a finite number of representative vectors (*code vectors*). The aim of competitive neural networks is to cluster or categorize the input data, and they can be used for data coding and compression through vector quantization (see [1], [2]). This is achieved by adapting the *weight vector* of each neuron so that it approximates the mean or *centroid* of a data cluster. On each computing step, only the *winning neuron* is updated, i.e., the neuron with the weight vector which is closest to the current input vector. The Self-Organizing Feature Map (SOFM) [3] can be regarded as an evolution of this model where the *neighbour neurons* of the winner are also updated, according to a *network topology*.

However, data analysis with competitive networks has severe limitations. This is because the model only provides information about the mean of each cluster. The Principal Components Analysis methods overcome this problem by obtaining the *principal directions* of the data, i. e., the maximum variance directions (see [4], [5]). It has been proved that PCA is an optimal linear technique for dimensionality reduction, in the mean sense (see [4]).

The original method, sometimes called *Karhunen-Loëve (KL) transform* or *global PCA*, considers the entire input distribution as a whole. A number of *local PCA*

methods have been proposed to partition the distribution into meaningful clusters (see [6], [7]). These methods have been widely used to compress multispectral and multilayer images (see [8]).

This paper is organized as follows. Section 2 presents our new neural model. In Section 3 we prove some important properties of the model. We make a short discussion of the advantages of our model in Section 4. Sections 5 and 6 are devoted to experimental results and conclusions, respectively.

2 The PCACL model

2.1 Neuron weights updating

The PCA method uses the *covariance matrix* \mathbf{R} to analyze the input distribution. If we have M input samples, $\mathbf{x}_1, \dots, \mathbf{x}_M$, we can make the following approximation:

$$\mathbf{R} \approx \mathbf{R}^I = \frac{1}{M-1} \sum_{i=1}^M (\mathbf{x}_i - E[\mathbf{x}])(\mathbf{x}_i - E[\mathbf{x}])^T, \quad (1)$$

where $E[\cdot]$ is the mathematical expectation operator. Note that this is the best approximation that we can obtain with this information (it is an unbiased estimator with minimum variance). Now, if we obtain N new input samples, $\mathbf{x}_{M+1}, \dots, \mathbf{x}_{M+N}$, we may write:

$$\mathbf{A} = \frac{1}{N} \sum_{i=M+1}^{M+N} (\mathbf{x}_i - E[\mathbf{x}])(\mathbf{x}_i - E[\mathbf{x}])^T, \quad \mathbf{R} \approx \mathbf{R}^{II} = \frac{1}{M+N-1} ((M-1)\mathbf{R}^I + N\mathbf{A}). \quad (2)$$

Now we need to approximate the expectation of the input vector $E[\mathbf{x}]$. We may use a similar approach:

$$E[\mathbf{x}] \approx \mathbf{e}^I = \frac{1}{M} \sum_{i=1}^M \mathbf{x}_i, \quad \mathbf{e}^{II} = \frac{1}{N} \sum_{i=M+1}^{M+N} \mathbf{x}_i, \quad (3)$$

$$E[\mathbf{x}] \approx \mathbf{e}^{III} = \frac{1}{M+N} \left(\sum_{i=1}^M \mathbf{x}_i + \sum_{i=M+1}^{M+N} \mathbf{x}_i \right) = \frac{1}{M+N} (M\mathbf{e}^I + N\mathbf{e}^{II}). \quad (4)$$

We use \mathbf{e}^I to approximate $E[\mathbf{x}]$ in (1), and \mathbf{e}^{III} to approximate $E[\mathbf{x}]$ in (2). Note that we do not use \mathbf{e}^{II} in (2) because it is a poor approximation of $E[\mathbf{x}]$ while \mathbf{e}^{III} is better.

Every processing unit of our model stores approximations to the matrix \mathbf{R} and the vector $E[\mathbf{x}]$. The above equations are used to update these approximations. So, in the time instant t the unit j stores the matrix $\mathbf{R}_j(t)$ and the vector $\mathbf{e}_j(t)$.

Let the *learning rate for the covariance matrix* η_R and the *learning rate for the mean* η_e be

$$\eta_R = \frac{N}{N+M-1}, \quad \eta_e = \frac{N}{N+M}. \quad (5)$$

Then we may rewrite (4) and (2) as

$$\mathbf{e}^{III} = (1 - \eta_e) \mathbf{e}^I + \eta_e \mathbf{e}^{II}, \quad \mathbf{R}^{II} = (1 - \eta_R) \mathbf{R}^I + \eta_R \mathbf{A}. \quad (6)$$

Note that these expressions are analogous to the weight update equations of the competitive learning and the self-organizing feature map (SOFM). Equations (5) correspond to a hyperbolic decay of the learning rates, but a linear decay may be used as an alternative.

2.2 Competition among neurons

When an input sample is presented to the network, a competition is held among the neurons. We note the *orthogonal projection* of a vector \mathbf{x} on an orthonormal vector basis $B = \{\mathbf{b}_h \mid h=1, \dots, K\}$ as $\hat{\mathbf{x}} = \text{Orth}(\mathbf{x}, B)$. The vector \mathbf{x} can be decomposed in two vectors, the orthogonal projection and the *projection error*, i. e., $\mathbf{x} = \hat{\mathbf{x}} + \tilde{\mathbf{x}}$. Every unit (say, j) of our network has an associated vector basis $B^j(t)$ at every time instant t . It is formed by the $K_j(t)$ eigenvectors corresponding to the $K_j(t)$ largest eigenvalues of $\mathbf{R}_j(t)$. This is rooted in the Principal Components Analysis (PCA). Note that $B^j(t)$ must be orthonormalized in order for the system to operate correctly. The difference among input vectors $\mathbf{x}^i(t), i=1, \dots, N$ and estimated means are projected onto the vector bases of all the neurons. The neuron c that has the minimum sum of projection errors is the winner:

$$\mathbf{z}_j^i(t) = \text{Orth}\left(\mathbf{x}^i(t) - \mathbf{e}_j(t), B^j(t)\right), \quad c = \arg \min_j \left(\sum_{i=1}^N \|\mathbf{x}^i(t) - \mathbf{e}_j(t) - \mathbf{z}_j^i(t)\|^2 \right). \quad (7)$$

2.3 Vector bases size updating

Our model considers a variable number of basis vectors $K_j(t)$, which is computed independently for each neuron j . This number reflects the intrinsic dimensionality of the data in the receptive field of neuron j (i. e., the set of inputs for which the neuron j is the winner). There are several algorithms to approximate the intrinsic dimensionality of the data (see [9]). Most of them are based in the analysis of the eigenvalues $\lambda_j^p(t)$ of the covariance matrix $\mathbf{R}_j(t)$, $p=1, \dots, D$. The quotient between $\lambda_j^p(t)$ and the trace of $\mathbf{R}_j(t)$ is the amount of variance explained by the p th principal direction of $\mathbf{R}_j(t)$. Our approach is based on the use of a parameter α to specify the amount of variance which we want the neurons to explain. Hence, we select $K_j(t)$ so that the amount of variance explained by the directions associated to the $K_j(t)$ largest eigenvalues is at least α :

$$K_j(t) = \min\{ k \in \{0, 1, \dots, D-1\} \mid \sum_{p=1}^k \lambda_j^p(t) \geq \alpha \sum_{q=1}^D \lambda_j^q(t) \} . \quad (8)$$

Note that the eigenvalues $\lambda_j^p(t)$ in (8) are sorted in decreasing order. As the model we have presented uses both PCA techniques and a competitive learning procedure, we call it *Principal Components Analysis Competitive Learning (PCACL)*.

2.4 Summary

The algorithm that summarizes this model can be stated as follows:

1. For every neuron j , obtain the initial covariance matrix $\mathbf{R}_j(0)$ as a random nonnegative symmetric matrix close to the unit matrix, and initial mean vector $\mathbf{e}_j(0)$ as a random vector with small components (either negative or positive).
2. At time instant t , select the input vectors $\mathbf{x}^i(t)$, $i=1, \dots, N$, with $N \geq l$, from the input distribution. Compute the winning neuron c according to (7).
3. For every neuron j , update the vector \mathbf{e}_j and the matrix \mathbf{R}_j by using the following equations:

$$\mathbf{e}_j(t+1) = \mathbf{e}_j(t) + \eta_e(t) \left(\frac{1}{N} \left(\sum_{i=1}^N \mathbf{x}_i \right) - \mathbf{e}_j(t) \right), \quad (9)$$

$$\mathbf{A}_j(t+1) = \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i - \mathbf{e}_j(t+1))(\mathbf{x}_i - \mathbf{e}_j(t+1))^T, \quad (10)$$

$$\mathbf{R}_j(t+1) = \mathbf{R}_j(t) + \eta_R(t) (\mathbf{A}_j(t+1) - \mathbf{R}_j(t)). \quad (11)$$

4. If convergence has been reached, stop the simulation. Otherwise, go to step 2.

3 Properties

Proposition 1: Let $z \in \{0, 1, \dots, D\}$. If $\alpha \in [0, z/D]$, then for every neuron j , it holds that $0 \leq K_j(t) \leq z$.

Proof: As the eigenvalues are sorted in decreasing order, it holds that:

$$\forall p \in \{1, \dots, z\} \forall q \in \{z+1, \dots, D\}, \lambda_j^p(t) \geq \lambda_j^q(t) \Rightarrow (D-z) \sum_{p=1}^z \lambda_j^p(t) \geq z \sum_{q=z+1}^D \lambda_j^q(t). \quad (12)$$

Next we add $z \sum_{p=1}^z \lambda_j^p(t)$ to both sides, and then we use the hypothesis $\alpha \in [0, z/D]$:

$$\Rightarrow D \sum_{p=1}^z \lambda_j^p(t) \geq z \sum_{q=1}^D \lambda_j^q(t) \Rightarrow \sum_{p=1}^z \lambda_j^p(t) \geq \frac{z}{D} \sum_{q=1}^D \lambda_j^q(t) \Rightarrow \sum_{p=1}^z \lambda_j^p(t) \geq \alpha \sum_{q=1}^D \lambda_j^q(t) . \quad (13)$$

Then by (8) and (13) we have $0 \leq K_j(t) \leq z$.

Corollary 1: If we take $\alpha=0$, the PCACL model reduces to classical competitive learning (CL).

Proof: We use $z=0$ in the previous proposition, and we get $\alpha=0 \Rightarrow K_j(t)=0$ for every neuron j . Then the covariance matrices are not used in the competitive phase, so it reduces to that of CL. The estimated mean vectors $\mathbf{e}_j(t)$ play the role of the weight vectors $\mathbf{w}_j(t)$ of CL.

4 Discussion

The PCACL algorithm is quite different from known local PCA methods. The PCACL model has the following advantages:

a) It is a generalization of the competitive learning. The learning rates have the same meaning as in the CL.

b) It allows any number of inputs to be presented to the network at every computation step. Hence, there is no need to present the entire data set to the network, while in some local PCA methods this is mandatory [7]. PCACL supports batch mode, but it is optional.

c) The neurons are able to adapt to the local dimensionality of the data. Many local PCA methods do not have this capability (again, see [7]). These methods rely on user's experience and knowledge to select the appropriate number of basis vectors, which is the same for all neurons. This leads to severe problems if the local dimensionality of the data varies significantly from one region of the input space to another. Hence, we may expect that the PCACL has a better dimensionality reduction than those methods, because of its flexibility.

5 Experimental results

A set of experiments has been designed to test the performance of the PCACL model. We have selected the local PCA approach of Kambhatla and Leen [7] to make a comparison with our method. This local PCA network uses a fixed number of basis vectors K for all neurons. It is aimed to reduce the projection error, as PCACL model. Hence we use the *mean squared projection error (MSPE)* E to measure the performance of both systems:

$$E = \frac{1}{N} \sum_{i=1}^N \|\tilde{\mathbf{x}}_i\|^2 . \quad (14)$$

In order to reduce the dimensionality of the input distribution, the projection error minimization must be achieved with the minimum possible number of basis vectors. So, we are interested in the relation between these two values.

Our experiments have used a freely-accessible data set from the NASA Earth Observatory [10]. This data set is made up of images taken every month by satellites, which represent the values of several atmospheric parameters over the surface of our planet. We have selected 6 months (from January 1988 to June 1988), and 14 parameters for each month. So we have 14 images per month, each with 360x180 pixels. These 14 images are combined to form a single multisensor image, where every pixel is a vector with 14 components. The values of the components are real numbers in the interval [0,1]. Each of the vectors is an input sample. Every experiment starts by presenting the input samples of a month to a network. When the training is finished, the projection error is computed for all the input samples of the month. Then these projection errors are used to compute the mean error of the month. Finally, the mean errors from the 6 months are averaged to yield the final mean error.

Table 1. Average CPU time used to train the PCACL network, in seconds

# neurons	$\alpha=0.5$	$\alpha=0.6$	$\alpha=0.7$	$\alpha=0.8$	$\alpha=0.9$
2	1329	1494	1637	1598	1856
4	1663	1055	1718	1678	2346
8	2934	3405	2154	2122	2341
16	3498	3490	3146	3123	3393

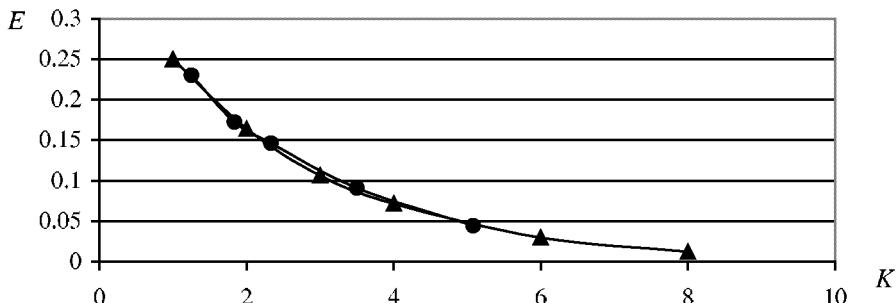
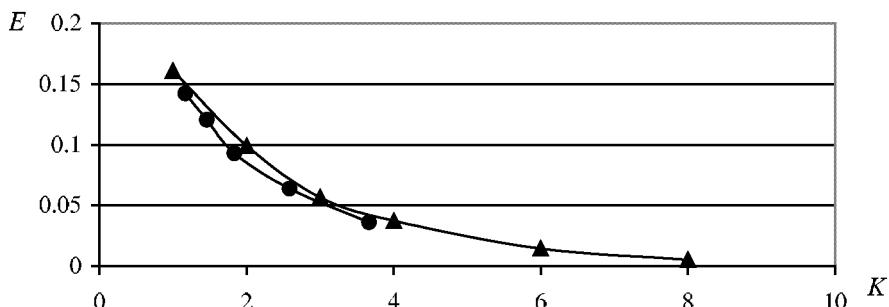
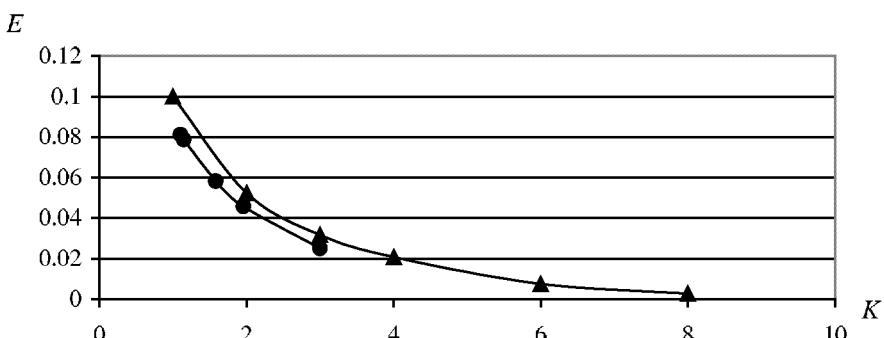
Table 2. Average CPU time used to train the local PCA network, in seconds

# neurons	K=1	K=2	K=3	K=4	K=6	K=8
2	6070	7762	8822	7928	9063	9325
4	7520	8358	8695	10172	11625	13166
8	10347	12304	14686	16336	19058	21875
16	18343	22056	26096	33357	35891	41261

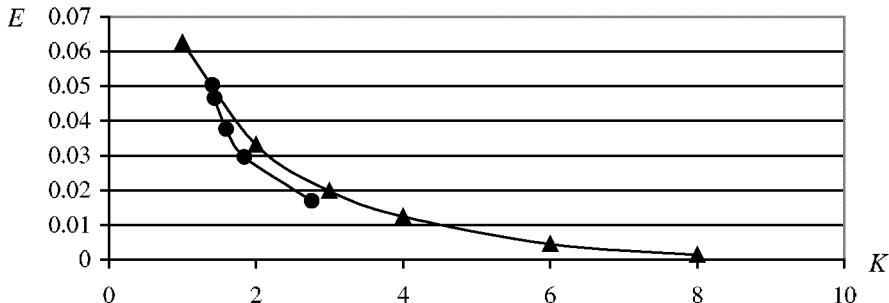
The parameters of the local PCA network have been selected as follows. We have run 20 iterations of the method, i.e., we have presented the data of the considered month 20 times to the network. As the algorithm needs the number of basis vectors K to be specified, we have carried out experiments with $K=1, 2, 3, 4, 6$ and 8 . All these values of K have been used with 2, 4, 8 and 16 neurons.

For the PCACL network, we have used different values of the parameter α : 0.5, 0.6, 0.7, 0.8 and 0.9. All these values of α have been tested with the same numbers of neurons as local PCA. We have used a linear decay of the learning rates, with initial values $\eta_R(0)=\eta_c(0)=1$ and final values $\eta_R(t_{final})=\eta_c(t_{final})=0$. The input samples have been presented to the network one at a time (i.e., $N=1$). In order to train the network adequately, every input sample has been presented two times.

The plots of the error E versus the number of basis vectors K is shown in Figs. 1, 2, 3 and 4. Note that for the PCACL network, the results corresponding to lower α appear to the left, because a smaller amount of variance can be explained with fewer basis vectors.

Fig. 1. MSPE with 2 neurons. Local PCA=triangles, PCACL=circles.**Fig. 2.** MSPE with 4 neurons. Local PCA=triangles, PCACL=circles.**Fig. 3.** MSPE with 8 neurons. Local PCA=triangles, PCACL=circles.

We can see that the PCACL network has better performance than local PCA for all the parameters used, except with 2 neurons, where both algorithms have equal performance. Please note that nearer to the coordinate origin is better. The CPU time required for both methods is shown in Tables 1 and 2. It can be seen that PCACL has a faster execution in all cases. The speedup factor with respect to local PCA is always higher than 5.

Fig. 4. MSPE with 16 neurons. Local PCA=triangles, PCACL=circles.

6 Conclusions

We have presented a new neural network, which combines competitive learning and Principal Components Analysis. It does not require to present the entire data set to the network at a time. This allows a fast execution, while retaining the dimensionality reduction properties of the PCA. Furthermore, every neuron is able to adapt to the local dimensionality. Hence our model has a dimensionality estimation capability, and a greater plasticity. This feature is controlled by a parameter which specifies the desired amount of explained variance. Experimental results have been presented, which show the dimensionality reduction capabilities of the model with multisensor images, and its advantages with respect to a well known local PCA network.

References

1. Ahalt, S.C., Krishnamurthy, A.K., Chen, P., Melton, D.E.: Competitive Learning Algorithms for Vector Quantization. *Neural Networks* 3 (1990), 277–290.
2. Yair, E.K., Zeger, K., Gersho, A.: Competitive Learning and Soft Competition for Vector Quantizer Design. *IEEE Trans. Signal Processing* 40(2) (1992), 294–308.
3. Kohonen, T.: The Self-Organizing Map. *Proc. IEEE* 78 (1990), 1464–1480.
4. Jolliffe, I.T.: *Principal Component Analysis*. Springer-Verlag, Berlin (1986).
5. Kendall, M.: *Multivariate Analysis*. Charles Griffin&Co, London (1975).
6. Weingessel, A., Hornik, K.: Local PCA Methods. *IEEE Trans. Neural Networks* 11(6) (2000), 1242–1250.
7. Kambhatla, N., Leen, T.K.: Dimension Reduction by Local Principal Component Analysis. *Neural Computation* 9(7) (1997), 1493–1516.
8. Tretter, D., Bouman, C.A.: Optimal Transforms for Multispectral and Multilayer Image Coding. *IEEE Trans. Image Processing* 4(3) (1995), 296–308.
9. Verveer, P.J., Duin, R.P.W.: An Evaluation of Intrinsic Dimensionality Estimators. *IEEE Trans. Pattern Analysis and Machine Intelligence* 17(1) (1995), 81–86.
10. NASA Earth Observatory, Data & Images, Internet: <http://earthobservatory.nasa.gov/Observatory/datasets.html>. Date of access: July 2002.

Progressive Concept Formation in Self-organising Maps

Emilio Corchado and Colin Fyfe

School of Information and Communication Technologies,
The University of Paisley,
Scotland.

Abstract. We review a technique for creating Self-organising Maps (SOMs) in a Feature space which is nonlinearly related to the original data space. We show that convergence is remarkably fast for this method. The resulting map has two properties which are interesting from a biological perspective: first, the learning forms topology preserving mappings extremely quickly; second, the learning is most refined for those parts of the feature space which is learned first and which have most data. By considering the linear feature space, we show that it is the interaction between the overcomplete basis in which learning takes place and the mixture of one-shot and incremental learning which comprises the method that gives the method its power. Finally, as an engineering application, we show that maps representing time series data are able to successfully extract the time-dependent structure in the series.

1 Introduction

Topographic map formation is to be found in those parts of the cortex which deal with early sensory processing [3]. Examples are to be found in the somatosensory cortex, the visual cortex and the auditory cortex. The Self-organising Map (SOM) is one of the most widely used unsupervised learning techniques in the field of artificial neural networks [5] and is known to provide a low dimensional representation of a data set which captures local topographical relations. Nearby neurons quantise similar parts of the input space and similar inputs are quantised to the same or similar outputs.

We have recently [1, 2, 6] developed an extension of the SOM which works in Kernel space. Not only is learning very fast and the topology preserving mapping very robust [1] but also the learning shows properties similar to human learning. In particular, those concepts learned first are refined most as more data becomes available while those properties learned somewhat later are less refined. This is because of an interaction between one-shot (or immediate) learning (which is how people tend to learn) and continuous decay of the first and possibly roughly defined concept as new information of a similar type becomes available. This mirrors how a child may initially categorise all animals together, then subsequently be able to differentiate between cats and dogs, and then to differentiate between different types of dogs etc.

The paper is structured as follows: in section 2, we review k-means clustering in feature spaces; in section 3, we review the Kernel Self Organising Map and show how it converges on a standard data set. We then analyse the linear version to uncover why it converges so quickly and how it models human learning. We then use the method on a time series which we have previously used for forecasting.

2 Kernel K-means Clustering

We will follow the derivation of [7] who have shown that the k means algorithm can be performed in Kernel space. The basic idea of the set of methods known as kernel methods is that the data set is transformed into a nonlinear feature space ($\phi : \mathbf{x} \rightarrow \phi(\mathbf{x})$). Any linear operation now performed in this feature space is equivalent to a nonlinear operation in the original space.

The aim is to find k means, \mathbf{m}_μ so that each point is close to one of the means. First we note that each mean may be described as lying in the manifold spanned by the observations, $\phi(\mathbf{x}_i)$ i.e. $\mathbf{m}_\mu = \sum_i w_{\mu i} \phi(\mathbf{x}_i)$. Now the k means algorithm chooses the means, \mathbf{m}_μ , to minimise the Euclidean distance between the points and the closest mean

$$\begin{aligned} \|\phi(\mathbf{x}) - \mathbf{m}_\mu\|^2 &= \|\phi(\mathbf{x}) - \sum_i w_{\mu i} \phi(\mathbf{x}_i)\|^2 \\ &= k(\mathbf{x}, \mathbf{x}) - 2 \sum_i w_{\mu i} k(\mathbf{x}, \mathbf{x}_i) + \sum_{i,j} w_{\mu i} w_{\mu j} k(\mathbf{x}_i, \mathbf{x}_j) \end{aligned}$$

i.e. the distance calculation can be accomplished in Kernel space by means of the K matrix alone.

Let $M_{i\mu}$ be the cluster assignment variable. i.e. $M_{i\mu} = 1$ if $\phi(\mathbf{x}_i)$ is in the μ^{th} cluster and is 0 otherwise. [7] initialise the means to the first training patterns and then each new training point, $\phi(\mathbf{x}_{t+1})$, $t+1 > k$, is assigned to the closest mean and its cluster assignment variable calculated using

$$M_{t+1,\alpha} = \begin{cases} 1 & \text{if } \|\phi(\mathbf{x}_{t+1}) - \mathbf{m}_\alpha\| < \|\phi(\mathbf{x}_{t+1}) - \mathbf{m}_\mu\|, \forall \mu \neq \alpha \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

In terms of the kernel function (noting that $k(\mathbf{x}, \mathbf{x})$ is common to all calculations) we have

$$M_{t+1,\alpha} = \begin{cases} 1 & \text{if } \sum_{i,j} w_{\alpha i} w_{\alpha j} k(\mathbf{x}_i, \mathbf{x}_j) - 2 \sum_i w_{\alpha i} k(\mathbf{x}, \mathbf{x}_i) \\ & < \sum_{i,j} w_{\mu i} w_{\mu j} k(\mathbf{x}_i, \mathbf{x}_j) - 2 \sum_i w_{\mu i} k(\mathbf{x}, \mathbf{x}_i), \forall \mu \neq \alpha \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

We must then update the mean, \mathbf{m}_α to take account of the $(t+1)^{th}$ data point

$$\mathbf{m}_\alpha^{t+1} = \mathbf{m}_\alpha^t + \zeta (\phi(\mathbf{x}_{t+1}) - \mathbf{m}_\alpha^t) \quad (3)$$

where we have used the term \mathbf{m}_α^{t+1} to designate the updated mean which takes into account the new data point and

$$\zeta = \frac{M_{t+1,\alpha}}{\sum_{i=1}^{t+1} M_{i,\alpha}} \quad (4)$$

Now (3) may be written as

$$\sum_i w_{\alpha i}^{t+1} \phi(\mathbf{x}_i) = \sum_i w_{\alpha i}^t \phi(\mathbf{x}_i) + \zeta(\phi(\mathbf{x}_{t+1}) - \sum_i w_{\alpha i}^t \phi(\mathbf{x}_i)) \quad (5)$$

which leads to an update equation of

$$w_{\alpha i}^{t+1} = \begin{cases} w_{\alpha i}^t(1 - \zeta) & \text{for } i \neq t + 1 \\ \zeta & \text{for } i = t + 1 \end{cases} \quad (6)$$

3 The Kernel Self Organising Map

We have previously used the above analysis to derive a Self Organising Map [6] in Kernel space. The SOM algorithm is a k means algorithm with an attempt to distribute the means in an organised manner and so the first change to the above algorithm is to update the closest neuron's weights and those of its neighbours. Thus we find the winning neuron (the closest in feature space) as above but now instead of (2), we use

$$M_{t+1,\mu} = \Lambda(\alpha, \mu), \forall \mu \quad (7)$$

where α is the identifier of the closest neuron and $\Lambda(\alpha, \mu)$ is a neighbourhood function which in the experiments reported herein was a gaussian. Thus the winning neuron has a value of $M=1$ while the value of M for other neurons decreases monotonically with distance (in neuron space) away from the winning neuron. For the experiments reported in this paper, we used a one dimensional vector of output neurons numbered 1 to 20 or 30. The remainder of the algorithm is exactly as reported in the previous section.

The Kernel SOM was reported in [6] to have very fast convergence and results in that paper were based on Gaussian kernels. We now report that equally fast convergence can be found with linear kernels and so use these to investigate the reasons for this speed of convergence.

Let us illustrate its learning on a standard data set which is composed of 90 points drawn uniformly from $[0,1]*[0,1]$. The effect of this learning on artificial data is illustrated in Figure 1 in which we show the weights learned after each of the first five data points are presented. In Figure 2, we show the weights after all 90 data points were presented. The converged weights can be translated back into the data space, of course, by simply using the weights found and multiplying by the respective data points. A converged map is shown in Figure 3. Now, in Figure 2, we see that the top part of the figure is different in character to the bottom part of the figure. The top part of the figure was that part of the SOM's weights which responded first to the data set (Figure 1) while the bottom part

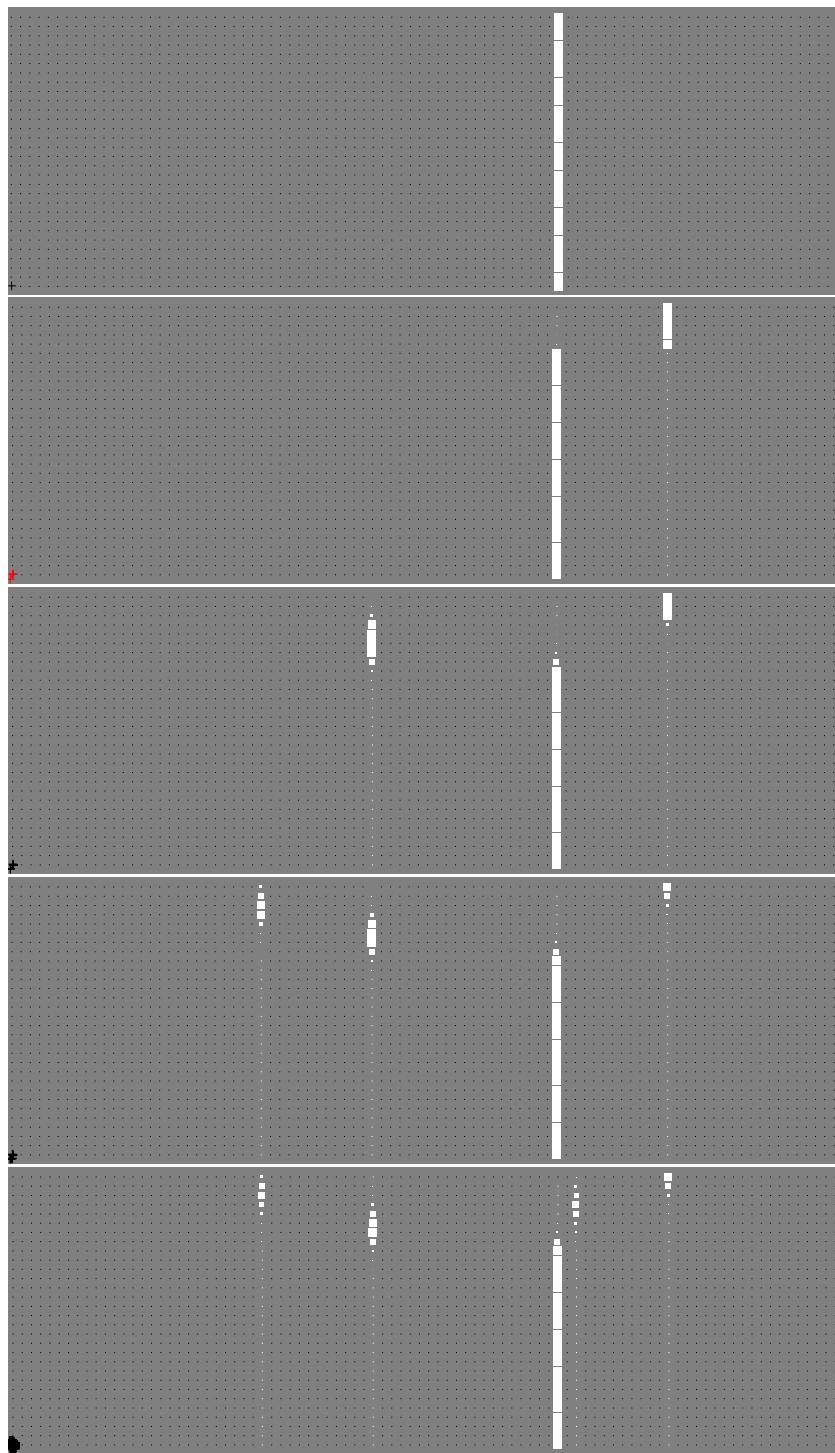


Fig. 1. Each diagram is 90 by 30 which is the number of data points by the number of centres. The top diagram shows that all centres respond totally to the first presented data point; the second shows a group of centres changing (one-shot) to a second data point; the third shows a second group of centres responding to a third data point. This is continued with the fourth and fifth data point presentation.

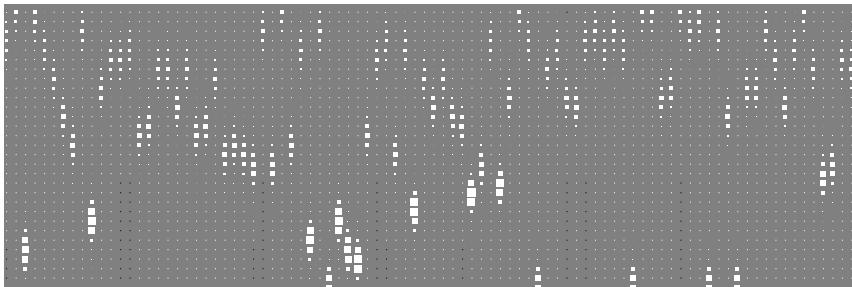


Fig. 2. The map of the previous figure after all 90 data points have been presented.

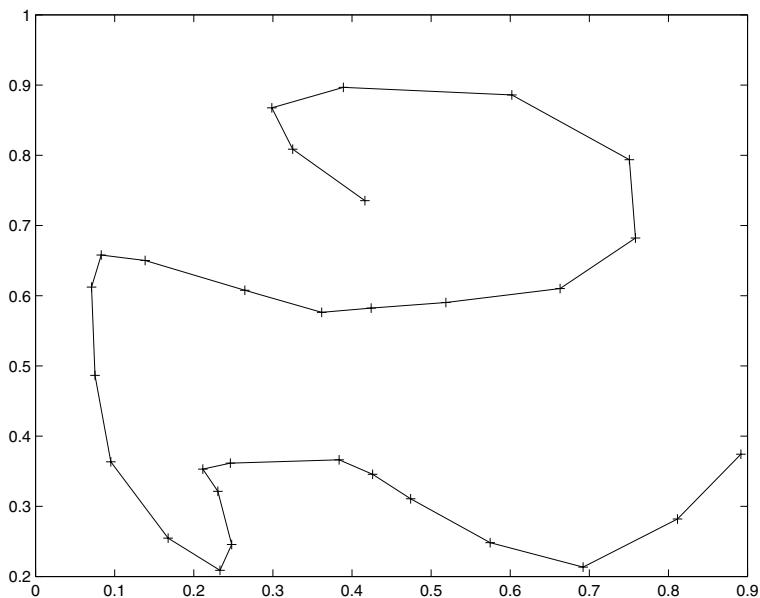


Fig. 3. The converged centres of the SOM (in data space) found by the new training algorithm on one pass through artificial data drawn uniformly from $[0,1]^* \times [0,1]$.

of the figure shows the weights which were learned last. We see that the top neurons' centres are defined by far more weights than the bottom neurons i.e. they depend on more samples or instances drawn from the distribution. This mirrors human learning in which any class met frequently is more precisely defined and determined by more instances than one just recently met.

3.1 Discussion

To explain why the learning takes this form, in this section we will consider the linear Kernel SOM in order to investigate its very fast convergence. Note that the learning

$$w_{\alpha i}^{t+1} = \begin{cases} w_{\alpha i}^t(1 - \zeta) & \text{for } i \neq t + 1 \\ \zeta & \text{for } i = t + 1 \end{cases} \quad (8)$$

has two modes:

1. The first mode is one shot learning for the current input; the weights from all outputs to the current input are set to the exponential of the negative squared distance in output space of the output neurons from the winning neuron.
2. The second mode is decay from the values set in the first mode; note that we need only consider this decay subsequent to the one shot learning because the one shot learning removes any previously learned values and the algorithm ensures that each node is selected exactly once for the one shot learning.

The second thing to emphasise is that while we are working in the space of input variables we are using an unusual basis in treating every data point as the end point of a basis vector. This will typically give us a very overcomplete representation.

Let the first time neuron α wins a competition be the competition for the i^{th} input, \mathbf{x}_i . Then the weight $w_{\alpha i}$ is set to $\frac{e^0}{e^0} = 1$ where the denominator is the history to date of its winnings to date. For the time being, we will ignore the interaction with other neurons in the environment of α . Let the neuron α now win the competition for the input, \mathbf{x}_j . Then the weight $w_{\alpha j}$ is set to $\frac{e^0}{\sum_{i,j} e^0} = \frac{1}{2}$.

Also the weight $w_{\alpha i}$ decays to $w_{\alpha i} * (1 - \frac{1}{2})$, so that now the neuron's centre is the mean of the two data points for which it has won the competition. Similarly if the same neuron wins the competition to represent \mathbf{x}_k , the weight $w_{\alpha k}$ is set to $\frac{e^0}{\sum_{i,j,k} e^0} = \frac{1}{3}$, and the weights $w_{\alpha i}$ and $w_{\alpha j}$ decay to $\frac{1}{3}$, which again means the centre of the neuron is the mean of the three data points for which it has won the competition. A recursive argument shows that this will always be the case.

Now consider the effect of these three competitions on a neuron, μ in the neighbourhood of α . When neuron α wins the competition be for the i^{th} input, \mathbf{x}_i , the weight, $w_{\mu i}$ is set to $\frac{A(\alpha, \mu)}{A(\alpha, \mu)} = 1$. When neuron α wins the competition for the input, \mathbf{x}_j , the weight, $w_{\mu j}$ is set to $\frac{A(\alpha, \mu)}{\sum_{i,j} A(\alpha, \mu)} = \frac{1}{2}$ and the weight $w_{\mu i}$

decays to $\frac{1}{2}$. Note the strength of the effect of the neighbourhood function which explains the strong grouping effect of the method.

Thus initially all neurons will respond to the first inputs met during training. But the above argument ignores the interaction between different responses. Consider the effect on neuron α 's weight vector when neuron μ wins the competition for the l^{th} , input, \mathbf{x}_l . Now

$$\zeta_\mu = \frac{\Lambda(\mu, \alpha)}{\Lambda(\mu, \alpha) + \sum_{i,j,k} e^0} = \frac{\Lambda(\mu, \alpha)}{\Lambda(\mu, \alpha) + 3} \quad (9)$$

where we have used ζ_μ to designate that part of the ζ vector devoted to μ . For μ sufficiently far from α , this will be rather small and both of the learning effects will be negligible as a response to this input. Note that quite the opposite effect happens at μ and vectors close to it. Their previous history has given very small values of $\sum \Lambda(\alpha, \mu)$ and so now a value of $\Lambda(\mu, \mu) = 1$ will easily dominate the learning of these neurons and a type of phase change will take place as these neurons respond to a new input region due to the one-shot learning.

4 Modelling Time-Dependent Data

The fact that the order of presentation of the data affects the learning in a data set can be used to create a quantisation of time-dependent data that is useful for further processing. We also wish to illustrate the effect of continuing to learn on the same data set and believe that we can show a refinement of concepts as the learning proceeds.

We thus choose to illustrate the quantisation on a financial time series - the U.S. dollar- Japanese yen rate. We have previously [4] used a variety of methods to find the underlying factors in this data set and then used a standard multilayered perceptron using backpropagation to predict each factor separately. To illustrate the method, we show the quantisation of the time series in Figure 4. Each figure is 90 (the number of data points we used) by 50 (the number of centres). We see that we can easily identify the structure of the time series and days which are similar rates to each other are similarly quantised. The left diagram shows the result after 6 iterations; the right one after 10 iterations. We see that the right quantisation is finer than the left.

5 Conclusion

We have reviewed the Kernel Self-organising Map and highlighted two properties of its learning:

1. The learning is very fast.
2. Those features which are learned first and for which similar data is subsequently learned are more refined in that they depend on more data points than those learned early.

These two facets mirror very closely what happens in human learning and we consider the model worthy of future investigation.

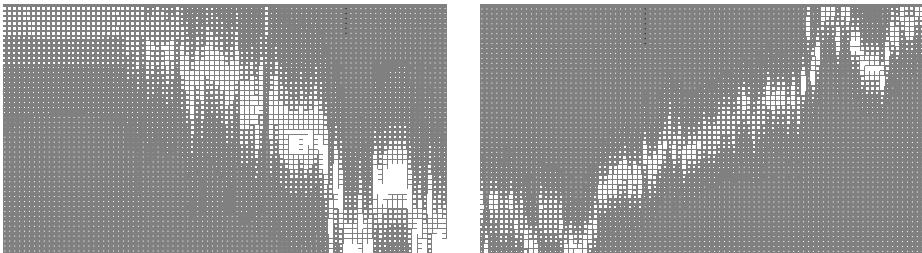


Fig. 4. Two quantisations of samples from the dollar-yen exchange rate. The left one is from 6 iterations of the algorithm; the right after 10 iterations.

References

1. E. Corchado and C. Fyfe. Initialising self-organising maps. In *Fourth International Conference on Intelligent Data Engineering and Automated Learning, IDEAL2003*, 2003. (submitted).
2. E. Corchado and C. Fyfe. Relevance and kernel self-organising maps. In *International Conference on Artificial Neural Networks, ICANN2003*, 2003. (submitted).
3. Hubel D. H. and Wiesel T. N. Receptive fields, binocular interaction and functional architecture in the cats visual cortex,. *ournal of Physiology (London)*, 160:106–154, 1962.
4. Y. Han and C. Fyfe. Finding underlying factors in time series. *Cybernetics and Systems: An International Journal*, 33:297–323, March 2002.
5. Tuevo Kohonen. *Self-Organising Maps*. Springer, 1995.
6. D. MacDonald and C. Fyfe. The kernel self-organising map. In R.J. Howlett and L. C. Jain, editors, *Fourth International Conference on Knowledge-based Intelligent Engineering Systems and Allied Technologies, KES2000*, 2000.
7. B. Scholkopf, A. Smola, and K.-R. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

Supervised Classification with Associative SOM

Rafael del-Hoyo¹, David Buldain², Alvaro Marco

¹ Instituto Tecnológico de Aragón, C/ María Luna nº6,
50005 Zaragoza, Spain
rdelhoyo@ita.es
<http://www.ita.es>

² Departamento de Electrónica y Comunicaciones, University of Zaragoza,
C/ María Luna 1, 50005 Zaragoza, Spain
buldain@posta.unizar.es

Abstract. This paper presents an extension of the Self Organizing Map model called Associative SOM that is able to process different types of input data in separated data-paths. The ASOM model can easily deal with situations of incomplete data-patterns and incorporate class labels for supervisory purposes. The ASOM is successfully compared with Multilayer Perceptrons in the incremental classification of six erythematо-squamous diseases, where only partial data is available in successive steps.

1 Introduction

Supervised neural models as Multilayer Perceptrons (MLP) and Radial Basis Functions (RBF) have demonstrated in many applications their capability for resolving classification problems. However in a real situation, the necessary information for a classification task often is not completely obtained or is not acquired at the same time. The incomplete data and the great heterogeneity of information are usual situations in real data, a very common problem found in medical diagnosis. The correct diagnosis is a classification problem done in two steps of incremental data acquisition: a compilation of symptoms and, if it is necessary, several laboratory analyses. The first step is critical in many diseases, and physicians usually can decide the medication using only the clinical inspection. However, the second step sometimes is necessary to obtain a correct diagnosis. This second data acquisition is modified when new laboratory tests are discovered and old ones are abandoned. This is a good example of what we call an incremental classification problem.

This variability and heterogeneity in the data sources is not well handled by the most known neural models as MLP and RBF networks supervised with the mean squared error (MSE). An automated classification system must present high flexibility in its structural implementation to deal with the continued apparition of new data sources or the deletion of obsolete ones. A neural classification system should be able to deal with the different data sources in an associative process, where each data source could be easily included or excluded during the classification process. The solution consists

in including separated data-paths for the different data sources with certain modulator mechanism to select the information incoming to the neural system.

A well-known neural model with associative behavior is the SOM network [3]. In this paper the multi-path data structure is implemented in the SOM model by extending the description of the SOMPACK available in [7]. We call it the Associative SOM (ASOM). In Sect. 2, we describe the problem selected to represent an incremental classification task. It is a real dermatology database of the differential diagnosis of six erythemato-squamous diseases. Sect. 3 contains an outlook of the ASOM and its learning schemes. Finally, the learning experiments with the ASOM and the MLP are presented in Sect. 4.

2 Differential diagnosis of erythemato-squamous disease

The differential diagnosis of erythemato-squamous diseases is a difficult problem in dermatology. It has been studied using several IA algorithms [2]. Usually a biopsy is necessary for the correct and definitive diagnosis. Patients are evaluated by the physician in two steps: first the clinical inspection of the degree of erythema, scaling and the compilation of historical data that configure 12 features. In the second step, skin samples have to be taken for the evaluation of 22 histopathological features determined by an analysis under a microscope. This is an expensive process, not always necessary to have a correct diagnosis, and the patient is not medicated until the results are obtained.

The database available in [6] contains 34 features: the ‘family history’ has the value 1 if any of these diseases has been observed in the family, and 0 otherwise. The ‘age’ represents the age of the patient. The rest of the features are separated in two groups: 10 clinical features and 22 histopathological features, all of them with numeric values in the range of 0 to 3. Here, 0 indicates that the feature was not present, 3 indicates the largest amount possible, and 1, 2 indicate the relative intermediate values.

We separated the features in three groups: 12 clinical features, 22 histopathological features and 6 features with the class labeling. The class labels codification contains 6 binary input components assigning one-class-to-one-component. The rest of the features were normalized to mean zero and variance scaled to value one. This preprocess ensures that all input components present a similar importance for the neural classifiers. Also, in the case of learning simulations with the MLP, this preprocess allows to manage the situations where some input data is not presented, by assigning null values to the missing input components (their mean values).

The results found by Güvenir et al. [2] were obtained using 10-fold cross-validation evaluation. They claim that the VF15 algorithm (Voting Features Intervals) achieves 96.2 % accuracy on the dermatology dataset and the 99.2% when VF15 weights are selected with a genetic algorithm.

3 Associative Self-Organized Map (ASOM)

The ASOM model is a SOM extension that allows the incremental classification and introduces a certain associative supervision during the learning process by means of the inputs, instead of using the class labels for supervising the outputs. The map receives different paths of information, that we call data-paths, each one of them processing features from similar sources. Another data-paths can introduce information about the correct classification of the pattern. With such an input structure, the neurons in the map generate prototypes associating all the incoming information from different data sources and are able of managing situations with missing data by a simple modulator method of the gains assigned to the input-paths. This idea was originally proposed in [1] using Radial Basis Units with a competitive algorithm.

3.1 The ASOM description

Let's denote the neuron label by 'i' and the weights or prototypes associated to it by ' \mathbf{w}_i '. Consider the input data divided in a number of sub-patterns D, in such a form that the different data sub-patterns 'd' would be processed by the corresponding data-path 'd', denoted as super indexes. The symbol N^d denotes the number of components or data inputs of the data-path 'd'. The sub-weights $w_i^{(d)}$, represent the fraction of weights of the neuron 'i' assigned to the data-path 'd'. Each data-path processes similar information and calculates the Euclidean distance between sub-patterns and sub-weights like the SOM model does. So we calculate the distance in the data-path 'd' by:

$$dist(\mathbf{w}_i^{(d)}, \mathbf{x}^{(d)}) = \sum_{j=1}^{N^d} (w_{ij}^{(d)} - x_j^{(d)})^2 \quad (1)$$

The path-excitation is calculated by multiplying the path-distance by a particular path-gain coefficient, $g^{(d)}$ (eq. 2). The sum of the path-excitations gives the whole excitation of the neuron 'Exc_i' (eq. 3). Notice that if all the path-gains present value one, the whole excitation is the Euclidean distance between the complete data pattern and the weights of the neuron, as in the SOM model.

$$exc_i^{(d)} = g^{(d)} dist(\mathbf{w}_i^{(d)}, \mathbf{x}^{(d)}) \quad (2)$$

$$Exc_i = \sum_{d=1}^D exc_i^{(d)} \quad (3)$$

If a certain path-gain is set to zero, the data introduced by the corresponding path provide a low contribution in the whole excitation of the neurons, and does not decide which neuron is the winner. However, if the path-gain has a high value, its data-path predominates in the selection of the winner neuron and, therefore leads the selection of the winner neuron in both the recall and the learning processes. During the recall phase, the class-gain must be annulled and the rest of path-gains can be modulated to process situations with incomplete data patterns, assigning null values to the gains in paths with missing data.

Another difference with the SOM model is the output response. The output of the SOM is the index of the winner neuron, but the ASOM outputs the winner's sub-weights in the class-path. If the winner prototype corresponds to a data region with a clear classification result, the output of the network should only present a value one in the corresponding class-component and the rest with value zero. When the neuron prototype is positioned in a data region between two classes, the output will present two high responses (near value 0.5) for the corresponding classes, as the result of the interpolation between two class labels. These special neurons must be recognized as neurons associated to map regions where there exist a high indetermination about the class. The output response with the sub-weights of the class-path also permits the calculation of error measures like the MSE, so the comparison with other neural models based in the MSE is straightforward.

The most common features of a disease can also be identified by recognizing the winner neuron among the neurons of the same class. This best neuron of the class corresponds to the best matching neuron when we exclusively evaluate the map with the class labels. The ASOM also presents all the interesting properties of the SOM model like the possibility of the visual inspection of the topological map representation.

However we must realize that the path-gains only influence in the choice of the winner neuron, but the training algorithm is the original SOM algorithm, that remains essentially like an unsupervised learning process. The ASOM is like an associative memory [4] providing association of data patterns and labels.

We used, for training the maps in batch mode and for the weight initialization, the functions available in the SOMPACK [7]. The map size chosen along all the experiments was 6x12 neurons, because this map size was suited to the two first PCA projections.

4. Experiments

The original dataset contains 366 patterns and 6 classes of diseases. We applied a 10-fold cross validation method for evaluating the performance of the maps, because the number of samples is quite low in certain classes. Ten data groups were generated randomly by separating the dataset in a training set with the 95% of the samples and the evaluation set with the 5%, avoiding duplication of the samples in the datasets.

For the evaluation of the neural classifiers, we formulate three possibilities: use exclusively clinical data, use only histopathological data, or use both of them. This scheme is suited for real diagnosis, where the physician obtains first the clinical information and later can access to the histopathological information. At the first step only the clinical information is processed, so if the classification result is clear, the extraction of the histopathological data can be avoided. However if the clinical data classification were unclear between two classes, the physician would decide if either is possible to find medication focused in both diseases or to obtain extra information from a histopathological analysis that would refine the response of the classifier. This is a clear example of that we call the incremental classification problem.

4.1 Experiments with ASOM

The maps must face with three types of data: clinical, histopathological and class labels, therefore their input structure presents three data paths. The maps were trained in two fixed number of cycles (600 and 1200 cycles), although better methods based in the evaluation of the MSE will be considered in future studies. We defined three learning schemes for the path-gains. The first scheme, called the unbalanced-gain training (UG), assigns value 1 for the three gains along the training process. The second scheme, called balanced-gain training (BG), tries to balance the excitations of the data-paths assigning values to the gains proportional to the inverse of the number of input components in the path. The values selected were: value 1 for the class-path, value 0.25 for the histopathological path and value 0.5 for the clinical path.

The third scheme, called balanced increasing training (BIG), presents path-gains varying during the training process in order to lead the associative learning of the map with the classification labels as the predominant influence. The previous results obtained in the UG and BG schemes were better for the second one, so we decided to balance the evolving gains. A possible training strategy gives initially a high value to the class-gain (value 1) and low values to the path-gains of the rest of the data-paths (for example value 0.1). As training progresses, the value of the class-gain is maintained constant, but the rest of the path-gains are increased exponentially to promote an increasing importance of the data pattern in the development of the map. With this scheme, in the beginning of the learning process, the map neurons are mainly clustered by the class labels, and the topological ordering of the map is established by the data-pattern that have a less influence in the choice of the winners. As the training progresses, the gains of the data are higher and the pattern gets more influence in the result of the competition process.

The BIG learning scheme needs a step-wise increase in the gains. If gains are continuously increased, the subspaces of data are expanded quicker than the development of the map, and the resulting map remains contracted after many cycles with the training sets. This behavior reveals that the map needs several cycles with constant gains to get expanded over the data subspaces. The solution for increasing the gains, is to maintain them stable during a number of cycles that we call step. Several values of steps were tested taking the values: 3, 6, 10, 60 and 120. The best MSE was obtained with the map trained 100 steps of 6 cycles (600 cycles). The main results are resumed in table 2. The ASOM in the UG and BG schemes presented the same errors whether trained 600 or 1200 cycles, while the BIG scheme showed different errors depending of the training cycles. The results of the ASOM for two BIG schemes with step value 6 during 600 cycles (case 100x6) and 1200 (case 200x6) cycles are also presented in table 2.

4.2 Experiments with Multilayer Perceptrons (MLP)

The MLP model was simulated with the same data groups in a 10-fold cross validation, separating in each data group a 5% of the training data for early stopping evaluation. The chosen learning algorithm was the Levenberg-Marquardt algorithm

implemented in the Neural-Toolbox of Matlab. Two different types of architectures were simulated: the first network architecture with one hidden layer and the output layer with six units associated to the six class labels (represented by 34-X-6). The second network architecture consisted in six separated networks including one hidden layer and only one output-class unit (represented by 34-X-1). Each one of the six networks was assigned to recognize only one disease-class and reject the rest of the classes. Both architectures received the 34 input components. In the evaluations with partial data, the unavailable components were assigned value zero.

As the numbers of samples in the classes were so uneven, in the first architecture the training sets were augmented by replicating the samples of classes with less representation to equilibrate the number of samples in all classes. In the second architecture, the number of samples of the class recognized by the network was also augmented to equilibrate the sum of samples in the rest of the classes.

The number of hidden units in the MLP networks were estimated with an previous learning tests with the whole dataset during a fixed number of cycles (50 was enough in all cases). The first network was estimated using only even numbers of units from 6 to 30 (13 numbers), and the second networks were estimated using even numbers of units from 4 to 24 (11 numbers). Each network was simulated five times (in total 120 simulations) with a target error of value 10^{-12} (with this target all networks were trained till the cycle 50). The MSE evaluated in the 50th cycle were averaged in the five exemplars of the networks. The numbers of hidden units of the networks with the lowest MSE were selected for the training in the cross validation method (see table 1).

Table 1. Number of hidden units selected in the MLP networks

MLP Networks	Number of hidden units
MLP with 6 outputs	24
MLP-Psoriasis	22
MLP-seboreric dermatitis	18
MLP-lichen planus	12
MLP-pityriasis rosea	10
MLP-cronic dermatitis	12
MLP-pityriasis rubra pilaris	12

The selected architectures were trained during 50 cycles with the training data and each cycle the resulting network was stored. The exemplar with the minimum MSE in the 5% of the training data (separated for this early stopping method) was selected for the evaluation with the test dataset. The same combinations of incomplete data that were evaluated with the maps were performed during the evaluation of the MLP classifiers, and the resulting classification errors are presented in table 2.

The graphs in the figure 1 represent a detailed comparison between the performance of the networks in the best MLP classifier (figure 1b) and the ASOM networks trained with the BIG scheme (figure 1a).

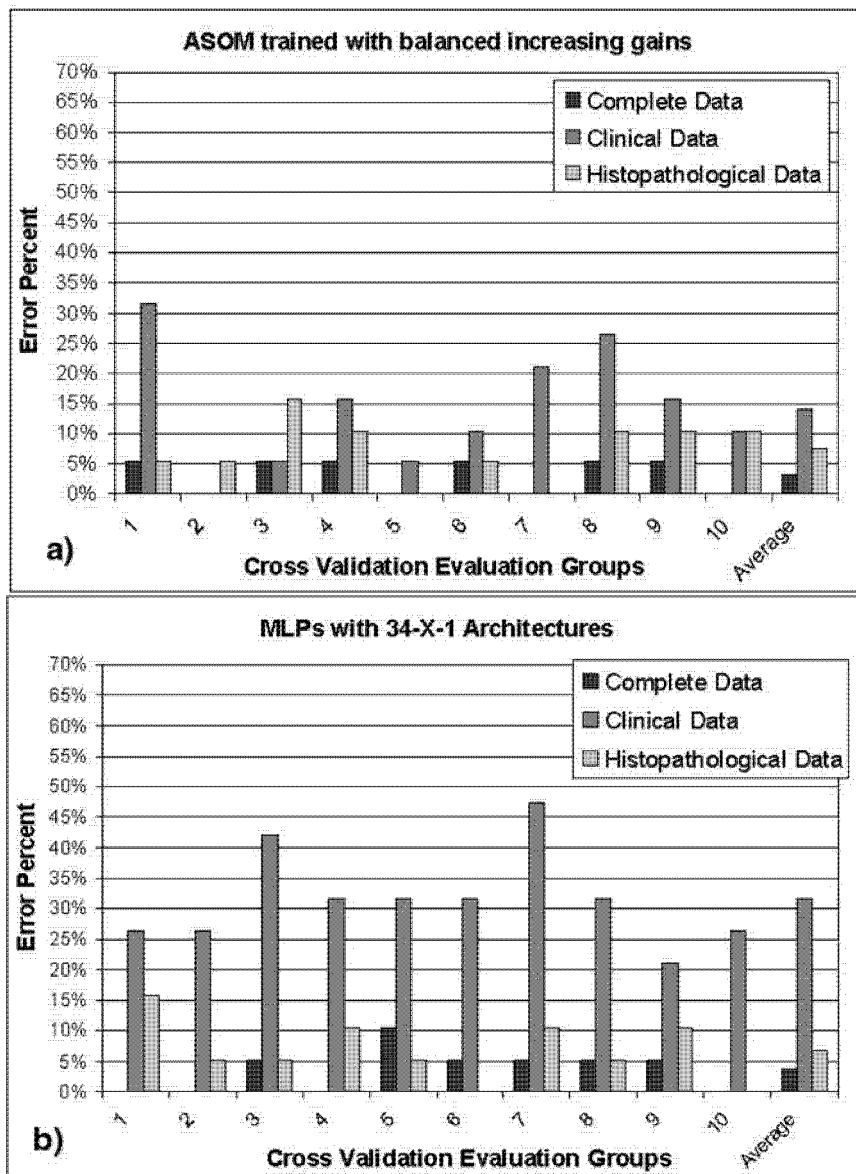


Fig. 1. Both graphs resume the classification percent errors measured in the 10 validation groups and the resulting average. The bars represent the resulting errors with only the clinical data, only the histopathological data and the complete data patterns. Graph a) presents the results obtained in the ASOM network trained with balanced increasing gains with step value 6 (ASOM-BIG-100x6 in table 2). Graph b) presents the results in the six MLP networks with one class output (MLP-34-X-1 architecture in table 2)

Table 2. Average classification errors from 10-fold cross validation in the ASOM and MLP networks, for the three types of evaluation with the two data groups.

Neural Network Model	Only Clinical Data	Only Histopathological Data	Complete Data
ASOM – UG(600)	13,68%	6,31%	4,21%
ASOM – BG(600)	12,10%	7,36%	3,68%
ASOM-BIG(200x6)	15,78%	6,31%	3,15%
ASOM-BIG(100x6)	14,21%	5,78%	3,15%
MLP-34-X-6	45%	8%	6%
6 MLP-34-X-1	32%	7%	4%

5. Conclusions

The paper presents a SOM extension that allows introducing different information paths to the maps. These data-paths can be processed separately by modulating their corresponding gains. This extension turns the non-supervised SOM model into a supervised one by means of the associative supervision with the class-labels used as inputs in a certain class-path. It was tested with the differential diagnosis of six erythemato-squamous diseases. This classification problem is suited for evaluations of incomplete data information as it deals with the combination of two kinds of information attributes: the clinical and the histopathological information. The MLP classifiers and the Güvenir results with the VF15 algorithm have been compared with those of ASOM and all of them obtained for the complete data classification a similar classification error around 4%. However if we compare those situations where incomplete data is presented to the classifier, only clinical or histopathological data, the ASOM performs quite better than the MLP networks.

References

1. J.D.Buldain, A.Roy: Association with Multi-Dendritic Radial Basis Units. IWANN 99, Lecture Notes in Computer Science 1606, Foundations and Tools for Neural Modeling (1999) 573-581
2. H.A.Güvenir, G. Demiröz and N. İlter,: Learning Differential Diagnosis of Erythemato-Squamous Diseases using Voting Feature Intervals. Artificial Intelligence in Medicine, Vol. 13, No. 3 (1998) 147-165
3. T.Kohonen. The self-organizing map. In Proc. IEEE, volumen 78, pages 1464-1480,1990.
4. T.Kohonen. Self-Organization and Associative Memory, Springer Series In Information Sciences 8. Springer, Heidelberg, 1984.
5. H.Ritter. Parametrized selft-organizing maps. In S. Gielen and B. Kappen, editors, ICANN93-Proceddings, Amsterdam, pages 568-575. Springer Verlag, Berlin, 1993.
6. <ftp://ftp.ics.uci.edu/pub/machine-learning-databases/dermatology/>
7. <http://www.cis.hut.fi/projects/somtoolbox/>

Neural Implementation of Dijkstra's Algorithm.

Mérida-Casermeiro, E.¹, Muñoz-Pérez J.² and Benítez-Rochel, R.²

¹Dept. Matemática Aplicada, ²Dept. Ciencias de la Computación,
Universidad de Málaga.*

Abstract. In this paper a new neural network model that generalizes Hopfield model is proposed. It allows to implement an algorithm similar to the Dijkstra's one in order to solve the shortest path problem on a weighted graph. With this neural and parallel implementation, the presented model is adapted to possible modifications in the graph. Moreover, it is possible to solve other related problems with the same structure.

1 Introduction

A great number of problems in the real world (tasks sequencing, networks of computers, roads or railways, etc.) can be represented by means of graphs. Shortest path problem (SPP) instances are the most fundamental and commonly encountered in the study of transportation and communication networks. In 1959 Edsger Wybe Dijkstra [4] developed a greedy algorithm that solves the problem in polynomial time.

Neural networks (NN) are an interesting alternative for solving combinatorial optimization (CO) problems in real time because of their parallelism. Most CO problems are NP-hard, so NN are the best alternative for obtaining solutions close to the optimum in a reasonable time. The advantage of NN must not be looked for in the computational complexity of the parallel architecture that realizes it, their potential is assessed in terms of the inherent parallelism and implementation of the nodes and components that greatly affect the effectiveness of the solution. Neural implementation of a classical algorithm results in a fast and reliable implementation that improves its efficiency.

Neural models for the SPP have been studied by several authors [14, 3]. They employ both Hopfield models. Although they uses different expressions of energy, all of them conclude that there exists local minima that correspond with non feasible solutions and that the number of non-solution local minima increases with the graph size.

A new NN is presented in this paper, its convergence in a finite number of step to optimum solution is ensured with synchronous and asynchronous dynamics. It solves both the single-source SPP and the all-pairs one. The dynamics is strongly inspired in the Dijkstra's algorithm. Additionally the NN can solve some related problems as the shortest path between any two subgraphs.

There exists some parallel no neural implementations [6, 10, 11, 2] and due to the convergence in a finite number of steps of SPP, these algorithms quickly obtains the optimum solution, however the model proposed has an additional advantage, it is capable of adapting itself to all possible variations in the graph. This variations can be

* Email addresses: merida@ctima.uma.es, munozp@lcc.uma.es, benitez@lcc.uma.es

additions or eliminations of edges and/or nodes and modifications in the cost matrix. This property allows the algorithm can be indefinitely executed looking on-line for the optimum when the graph is dynamically updated.

In the section 2, the SPP is formulated. The model is presented and applied to the SPP in section 3, whilst its performance is analyzed in section 4. Other related problems are studied in section 5, finally, we present our conclusions in the last section.

2 Shortest Path Problem (SPP)

2.1 Statement of shortest path problem

Let $G(V, A)$ be a weighted graph with $\text{card}(V) = N$ nodes (or vertices), we denote the weight (cost or length) of the arc $(i, j) \in A$ by w_{ij} . We assume that the cost of each arc is non negative and the cost of arcs (i, i) is zero for all i , so the entries of the matrix $W = (w_{ij})$ are all non negative and its diagonal components are zero.

A sequence of nodes $C = \{x_i\}, i = 0, 1, \dots, n_c$ in the considered graph G is a path from node a to b if the following conditions are satisfied:

- 1) $x_0 = a$,
- 2) $x_{n_c} = b$,
- 3) For $k \in \{0, 1, \dots, n_c - 1\}$, the arc $(x_k, x_{k+1}) \in A$.

Each path C in the graph has an associated length $L(C)$ and it can be calculated as:

$$L(C) = \sum_{i=0}^{n_c-1} w_{x_i, x_{i+1}}$$

Given any two nodes a and b , the objective of the single source SPP is to find a path C between a y b with minimum value of $L(C)$. If there exist different paths with the same length, we will find only one of them.

Dijkstra's algorithm is a classical algorithm that efficiently solves the SPP. It's a good example of greedy algorithm where locally optimum solutions provides the global optimum solution, so it is reasonable a parallel implementation.

Proposition 1 Suppose $a \in V$ the source node and $b \in V$ a node in the graph G , the length of the path from a to b verifies $L(a, b) = \min_{(i,b) \in A} \{L(a, i) + w_{ib}\}$ where i is a neighbour node to b and $L(a, i)$ is the length of a path from a to i .

The initial values are $L(a, i) = \infty, i \neq a$ and $L(a, a) = 0$, and updates according to:

$$L(a, b) = \min_{(i,b) \in A} \{L(a, b), L(a, i) + w_{ib}\} \quad (1)$$

He showed that the algorithm converges to the optimum length path in $N - 1$ steps [4]. Moreover, in each step a list S with length N , is updated in such a way that the element $S(i)$ represents the previous node to i in the best found path from a to i . It can be easily observed that this algorithm simultaneously ends up at the solution of the single source SPP to any node b in the graph.

There exists some works that parallelize this algorithm by updating the values of $L(a, i)$ and $S(i)$ in parallel.

2.2 Neural models for the shortest path problem

Many attempts of using NN have been made to solve the SPP. First approach was made by Rauch and Winarske [13], their method has some important limitations, one of them

is the need to know the number of nodes of the shortest path in advance. It was improved by Zhang and Thomopoulos [15] that uses the energy:

$$E = \frac{A}{2} \sum_{k=1}^{N-1} \sum_{i=1}^N \sum_{j=1}^N V_{ik} C_{ij} V_{jk+1} + \frac{B}{2} \sum_{k=1}^N \sum_{i=1}^N \sum_{j=1}^N V_{ik} V_{jk} + \frac{C}{2} \left(\sum_{i=1}^N \sum_{j=1}^N V_{ij} - N \right)^2 \quad (2)$$

where V_{ik} represents the output of the neuron (i, k) and $V_{i,k} = 1$ indicates that k is the i th node to be visited in the path and otherwise $V_{ik} = 0$. The convergence depends on the correct choice of constants A , B and C . In a similar way the convergence to the solution of model proposed in [3] depends on fine tuning of parameters.

There exists some NN approaches [1, 12] that propose new methods capable of adapting to external varying conditions. However, these methods have three major drawbacks: firstly, the methods fail to converge towards a valid solution in many runs; secondly, this problem worsens with N ; finally, the method finds poor solutions. These methods use N^2 neurons and their energy contains parameters that need to be adjusted (see eq. 2).

3 Multivalued neural model for SPP

We propose a multivalued network to solve the SPP. This model has been successfully applied to other CO problems [7–9]. Its principal characteristics are:

- The neurons have outputs in a given set \mathcal{M} , where \mathcal{M} can be \mathbb{R} , \mathbb{R}^n , $\{1, 2, \dots, n\}$ or even a symbolic set as $\{\text{green}, \text{red}, \text{blue}\}$.
- The state of the neuron i is characterized by its output V_i . So, the global state of the network is determined by its state vector $\mathbf{V} = (V_1, V_2, \dots, V_N) \in \mathcal{M}^N$.
- Each state of the network has an associated energy function given by:

$$E = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N f(w_{ij}, V_i, V_j, i, j) \text{ where } W \text{ and } f \text{ depend on the problem.}$$

For the SPP, we propose an energy function that corresponds to the sum of all paths from a to any node i : $E = \sum_i L(a, i)$ where $L(a, a) = 0$ and $L(a, i) = \infty$ (or an arbitrary large value) if there is no path from a to i .

3.1 Mapping the shortest path problem into the multivalued model

Let consider a NN with as fully connected neurons as nodes N in the graph. The output of neuron i , at time t , is given by $V_i(t) = (V_{i1}(t), V_{i2}(t)) \in \mathcal{M} = \{1, 2, \dots, N\} \times \mathbb{R}$. The first component is interpreted as a pointer to the precedent node in the path and the second one the cost of that path. So the energy is:

$$E(t) = \sum_{i=1}^N V_{i2}(t) \quad (3)$$

Definition 1. Given two neurons i and $j \neq i$, we say that j is a neighbour neuron of i if and only if there exists an arc $(j, i) \in A$. The set of all neighbour neurons, of a given one i , will be denoted by $\mathcal{W}(i)$.

At time t , a neuron i changes its states accordingly to the states of its neighbour neurons. Since any path from a to i has to include a last arc from a neuron in $\mathcal{W}(i)$ to

i , the optimum path from a to i must verify: $L(a, i) = \min_{j \in \mathcal{W}(i)} \{L(a, j) + w_{ji}\}$ that justifies the next dynamics for the NN:

$$V_{i2}(t+1) = \min_{j \in \mathcal{W}(i)} \{V_{j2}(t) + w_{ji}\} = \min_{1 \leq j \leq N} \{V_{j2}(t) + w_{ji}\} \quad (4)$$

$$V_{i1}(t+1) = k \Leftrightarrow V_{k2}(t) + w_{ki} \leq V_{j2}(t) + w_{ji} \quad \forall j \in \{1, 2, \dots, N\}$$

where $V_{i1}(t+1) = k$ means that the last arc of the path from a to i is (k, i) .

Definition 2. The decrement of energy derived from the output of neuron i when the path arrives to it from $j \in \mathcal{W}(i)$ will be called increment of potential of the neuron i when it takes the value j ,

$$U_{ij}(t) = -(V_{j2}(t) + w_{ji}) \quad (5)$$

The dynamics given by equation 4 is now expressed:

$$V_{i1}(t+1) = k \Leftrightarrow U_{ik}(t) = \max_j U_{ij}(t) \quad V_{i2}(t+1) = -U_{ik}(t) \quad (6)$$

The neuron associated to source node a will be clamped with $V_a(0) = (a, 0)$ at any time, whilst any other neuron $i \neq a$ will be initialized with $V_i(0) = (i, \infty)$. Where ∞ can be replaced by a value that can not be exceeded by the length of any path (without cycles) in the graph, for example $S = \sum_{i=1}^N \sum_{j=1}^N w_{ij}$.

Neural Algorithm (Static version)

- (1) Initialize neurons with $V_a(0) = (a, 0)$ and $V_i(0) = (i, \infty)$ if $i \neq a$, and $t = 0$.
- (2) Compute increment of potential $\forall i \neq a$ by equation 5 and find k by equation 6.
- (3) Update in a synchronic way, the states of the neurons $i \neq a$ by:

$$V_i(t+1) = (k, V_{k2} + w_{ki}).$$
- (4) When $\forall i, V_i(t+1) = V_i(t)$, or $t = N - 2$ then stop; else $t = t + 1$ and go to (2).

Convergence: The convergence of Dijkstra's algorithm [5], supports the validity of the neural algorithm here presented. It proves that at least one neuron i is stabilized at each step, or equivalently, the shortest path from a to i is found in each step. So, the shortest path from a to the rest of nodes is reached at most in $N - 1$ steps.

At end, $V_{i,2}$ indicates the shortest path length from a to i , while V_{i1} specifies the precedent node in that path, so the node sequence is available. It could happen, for not fully connected graph, that $V_{i,2}(N - 1) = \infty$ indicating that node i is not accessible.

Next version has also a good performance when matrix W is dynamically modified.

Neural algorithm (Dynamical version)

- (1) Initialize neurons with $V_a(0) = (a, 0)$ and $V_i(0) = (i, \infty)$ if $i \neq a$, and $t = 0$.
- (2) Compute increment of potential $\forall i \neq a$ by next equations:

$$U_{ij}(t) = -(V_{j2}(t) + w_{ji}), j \neq i \quad \text{and} \quad V_{i1}(t+1) = k \Leftrightarrow U_{ik} = \max_j U_{ij}$$
- (3) In parallel, the states of neurons $i \neq a$ are updated by: $V_i(t+1) = (k, V_{k2} + w_{ki})$.
- (4) Go to step 2.

The new algorithm has only two differences from the static version:

- 1) The algorithm does not finish, so it always is looking for a better path.
- 2) The node i must be excluded in the search for the previous node to itself.

The algorithm will find the optimum path in $N - 1$ step, since the proof of the convergence of the static version is still valid. In section four will be shown the performance of the network when the matrix W is changed during the process.

3.2 Neural network architecture

Figures 1 and 2 show the architecture and a generic neuron of the proposed model.

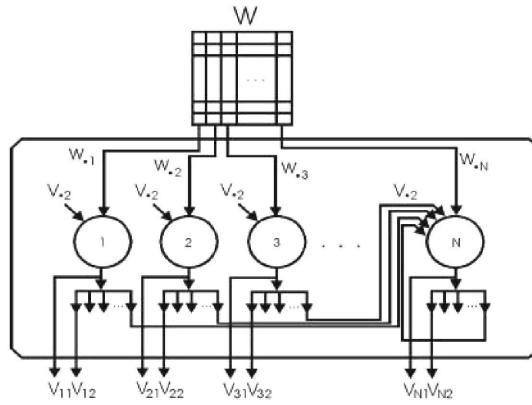


Fig. 1. Architecture of the proposed NN. It consists of N neurons, receives the cost matrix W and produces the matrix V . The vector V_2 is used as feedback.

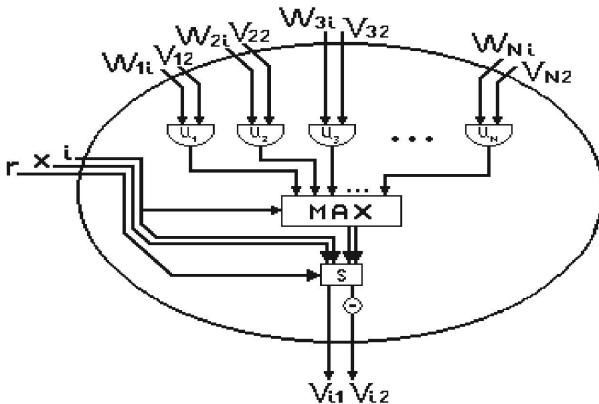


Fig. 2. A generic processing element. The input signals are the i th column of the cost matrix and the vector formed by all the second components of the neuron outputs. Three extra signals are also necessary to exclude the i th component, a reset r allows to fix the neuron output with x .

The neuron receives $W_{\cdot i} = (w_{1i}, w_{2i}, \dots, w_{Ni})$ and the feedback signals $V_{\cdot 2} = (V_{12}, V_{22}, \dots, V_{N2})$ as input signals. Firstly, the sums $w_{ji} + V_{j2}$ are calculated, then the k -th component verifying $-(w_{ki} + V_{k2}) = \max_j \{-(w_{ji} + V_{j2})\}$ is chosen. So, $V'_i = (k, V_{k2} + w_{ki})$ is the new neuron output.

4 Adaptability of the neural network at real time

Next we will prove that the NN reaches the optimum path from any initial state V , so it does not need be initialized adapting to variations.

4.1 Modifications of elements of the cost matrix

Suppose that new circumstances concerning the problem have appeared (opening a new road, works in other road, . . .), so, the matrix W changes to another one W' .

Let V^* be the equilibrium state of NN for the matrix W . In that point, matrix W is replaced by W' , so the initial state of the NN is V^* when the cost matrix is replaced. Next theorem proves the network state will converges towards a new equilibrium.

Theorem 1. *When the cost matrix W is replaced by another one W' with the same dimensions, the network state will converge towards a next equilibrium state V'^* that corresponds with the new optimum path of the graph.*

Proof: Let $\mathcal{F}(t)$ be the set formed by neurons with optimum output at time t . At time t , a neuron i verifies some of the next cases

- $i \in \mathcal{F}(t)$ and there is not any neuron $j \notin \mathcal{F}(t)$ with $V_{j2} < V_{i2}$.
- $i \in \mathcal{F}(t)$ and there is a neuron $j \notin \mathcal{F}(t)$ with $V_{j2} < V_{i2}$.
- $i \notin \mathcal{F}(t)$.

Let $m(t)$ be the minimum of $V_{j2}(t)$ for neurons $j \notin \mathcal{F}(t)$ and $w_0 = \min_{i \neq j} w_{ij}$.

A neuron of the first group is stable because does not exist any better path formed with neurons in $\mathcal{F}(t)$ and any other path through neuron $j \notin \mathcal{F}(t)$ has a larger cost. All neurons with $V_{i2} \leq m(t)$ are included in the first group and they are stables.

It is easy prove that $m(t+1) \geq m(t) + w_0$, since any neuron j verifying $V_{j2}(t+1) = \min_k \{V_{k2}(t) + w_{kj}\} < m(t) + w_0$ will be in $\mathcal{F}(t+1)$ because only stable neurons have $V_{k2}(t) < m(t)$ and $w_{kj} \geq w_0$. Hence, if an optimum path from a to i exists then m will reach its length in a finite number of steps and the node will be stable while for non connected graphs V_{j2} can increase indefinitely but if $V_{j2} > \sum_{j \neq a} \text{fmax}_k w_{kj}$, we will know an optimum path does not exist. Note: $\text{fmax}_k w_{kj}$ means that the maximum is only obtained over finite values of w_{kj} , infinite values will be ignored. \square

Moreover when arcs only reduce their cost or new arcs are introduced then the NN obtains the optimum at the most in $N-1$ steps.

Example 1: Consider the graph in figure 3-a. Table 1 shows that the network reaches a stable state after 3 iterations giving the shortest paths from node $a = 1$ to any other.

When the network is stabilized, the following modifications are generated: $w'_{61} = w'_{16} = w'_{45} = w'_{54} = 1$. The network is newly stabilizes by steps 1' y 2'.

Let now consider the matrix $W' = W$ excepts $w'_{1,10} = w'_{10,1} = 10$. The initial state is 0'' and the NN reaches, through iterations 1'' and 2'', the new equilibrium.

4.2 Elimination or addition of nodes

To eliminate a node consists on finding a path with minimum length which does not pass through it, or, in terms of the cost matrix W , on eliminating all arcs related with it $w'_{ij} = w'_{ji} = \infty$, $\forall j$. To add a new node, some arcs related with that new node must be added also. So both cases can be observed as a modification on weight of arcs.

Example 2: It corresponds with the graph of fig. 3-b. When the net is stabilized, node 2 is eliminated ($w'_{i2} = w'_{2i} = \infty$, $\forall i \neq 2$). Table 4.1 shows that the new equilibrium is obtained after 3 steps. In that point, node 2 is eliminated, the optimum is obtained after 9 steps. It is less than the upper bound for it, that is $M = \frac{1}{w_0} \sum_{k \neq a} \text{fmax}_k = \frac{22}{2} = 11$.

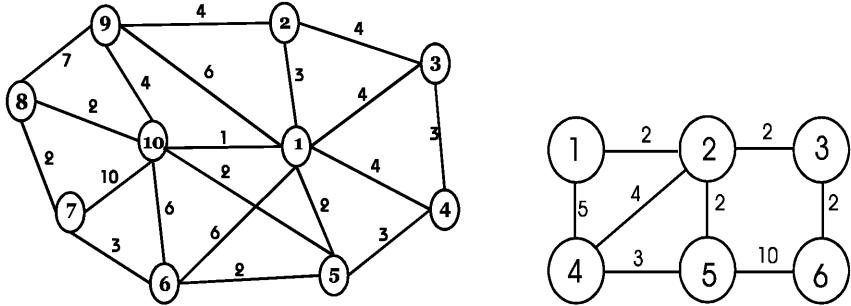


Fig. 3. Structure of graphs used as examples.

Table 1. Evolution of network outputs V_i . The output V_1 has been fixed to $(1, 0)$.

Step	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9	V_{10}
0	$(2, \infty)$	$(3, \infty)$	$(4, \infty)$	$(5, \infty)$	$(6, \infty)$	$(7, \infty)$	$(8, \infty)$	$(9, \infty)$	$(10, \infty)$
1	$(1, 3)$	$(1, 4)$	$(1, 4)$	$(1, 2)$	$(1, 6)$	$(7, \infty)$	$(8, \infty)$	$(1, 6)$	$(1, 1)$
2	$(1, 3)$	$(1, 4)$	$(1, 4)$	$(1, 2)$	$(5, 4)$	$(6, 9)$	$(10, 3)$	$(10, 5)$	$(1, 1)$
3	$(1, 3)$	$(1, 4)$	$(1, 4)$	$(1, 2)$	$(5, 4)$	$(8, 5)$	$(10, 3)$	$(10, 5)$	$(1, 1)$
...
∞	$(1, 3)$	$(1, 4)$	$(1, 4)$	$(1, 2)$	$(5, 4)$	$(8, 5)$	$(10, 3)$	$(10, 5)$	$(1, 1)$
$0'$	$(1, 3)$	$(1, 4)$	$(1, 4)$	$(1, 2)$	$(5, 4)$	$(8, 5)$	$(10, 3)$	$(10, 5)$	$(1, 1)$
$1'$	$(1, 3)$	$(1, 4)$	$(5, 3)$	$(1, 2)$	$(1, 1)$	$(8, 5)$	$(10, 3)$	$(10, 5)$	$(1, 1)$
$2'$	$(1, 3)$	$(1, 4)$	$(5, 3)$	$(1, 2)$	$(1, 1)$	$(6, 4)$	$(10, 3)$	$(10, 5)$	$(1, 1)$
$0''$	$(1, 3)$	$(1, 4)$	$(1, 4)$	$(1, 2)$	$(5, 4)$	$(8, 5)$	$(10, 3)$	$(10, 5)$	$(1, 1)$
$1''$	$(1, 3)$	$(1, 4)$	$(1, 4)$	$(1, 2)$	$(5, 4)$	$(8, 5)$	$(10, 6)$	$(1, 6)$	$(5, 4)$
$2''$	$(1, 3)$	$(1, 4)$	$(1, 4)$	$(1, 2)$	$(5, 4)$	$(6, 7)$	$(10, 6)$	$(1, 6)$	$(5, 4)$

Table 2. Evolution of network outputs V_i when the node 2 is eliminated.

Step	V_1	V_2	V_3	V_4	V_5	V_6
0	$(1, 0)$	$(2, \infty)$	$(3, \infty)$	$(4, \infty)$	$(5, \infty)$	$(6, \infty)$
1	$(1, 0)$	$(1, 2)$	$(3, \infty)$	$(1, 5)$	$(5, \infty)$	$(6, \infty)$
2	$(1, 0)$	$(1, 2)$	$(2, 3)$	$(1, 5)$	$(2, 4)$	$(6, \infty)$
3	$(1, 0)$	$(1, 2)$	$(2, 3)$	$(1, 5)$	$(2, 4)$	$(3, 5)$
...
∞	$(1, 0)$	$(1, 2)$	$(2, 3)$	$(1, 5)$	$(2, 4)$	$(3, 5)$
$0'$	$(1, 0)$	$(1, 2)$	$(2, 3)$	$(1, 5)$	$(2, 4)$	$(3, 5)$
$1'$	$(1, 0)$	$(2, \infty)$	$(6, 7)$	$(1, 5)$	$(4, 8)$	$(3, 5)$
$2'$	$(1, 0)$	$(2, \infty)$	$(6, 7)$	$(1, 5)$	$(4, 8)$	$(3, 9)$
$3'$	$(1, 0)$	$(2, \infty)$	$(6, 11)$	$(1, 5)$	$(4, 8)$	$(3, 9)$
$4'$	$(1, 0)$	$(2, \infty)$	$(6, 11)$	$(1, 5)$	$(4, 8)$	$(3, 13)$
$5'$	$(1, 0)$	$(2, \infty)$	$(6, 15)$	$(1, 5)$	$(4, 8)$	$(3, 13)$
$6'$	$(1, 0)$	$(2, \infty)$	$(6, 15)$	$(1, 5)$	$(4, 8)$	$(3, 17)$
$7'$	$(1, 0)$	$(2, \infty)$	$(6, 19)$	$(1, 5)$	$(4, 8)$	$(3, 17)$
$8'$	$(1, 0)$	$(2, \infty)$	$(6, 19)$	$(1, 5)$	$(4, 8)$	$(3, 18)$
$9'$	$(1, 0)$	$(2, \infty)$	$(6, 20)$	$(1, 5)$	$(4, 8)$	$(3, 18)$
...
∞	$(1, 0)$	$(2, \infty)$	$(6, 20)$	$(1, 5)$	$(4, 8)$	$(3, 18)$

5 Other related problems.

Some other problems related can be studied in a similar way.

1) Shortest path between each pair of nodes in a graph: It can be easily solved by a NN with N^2 neurons making up N groups of N neurons, where group i obtains the path from node i to the rest.

2) Shortest path between two subgraphs: It can be solved by a NN with N neurons where all nodes of the first subgraph are clamped with $V_{i,2} = 0$.

6 Conclusions.

In this paper a classic problem in graph theory (SPP) has been studied. We present two neural algorithms inspired in Dijkstra's algorithm that exactly and efficiently solves it.

This work shows that NN models are a good alternative to implement classical algorithms, they are parallel and can be implemented in hardware easily. Hopfield network is the neural model widely used, but it sometimes has given poor results, so other neural models must be proposed.

A new NN has been introduced for the one source SPP and two dynamics have been defined, both properly solve that problem in parallel, but the dynamic version has the advantage of its adaptability to modifications. Moreover, several other related problems can easily be solved with this model.

References

1. Ali M.K.M. and Kamoun F. Neural Networks for shortest path computation and routing in computer networks. *IEEE Trans. N.N.* **4**, 941-54, (1993).
2. Brodal, G. S., Träff, J. L. & Zaroliagis, C. D. A parallel priority data structure with applications. *Proc. 11th Int. Par. Processing Symp.*, 689-93, (1997).
3. Cavalieri S., Di Stefano A. and Mirabella O., Optimal Path Determination in a Graph by Hopfield Neural Network. *Neural Networks* **7-2**, 397-404, (1994).
4. Dijkstra, E. W. A note on two problems in connection with graphs. *Numerische Mathematik*, **1**, 269-71, (1959).
5. Grimaldi, Ralph. *Matemática Discreta y Combinatoria*. Cap 13.1, (1999).
6. Mateti, P. & Deo N., Parallel Alg. for Single Source SPP. *Computing* **29**, 31-49, (1982).
7. Mérida Casermeiro, E. Red Neuronal Recurrente Multivaluada para el rec. patrones y la optimiz. comb. Ph.D. dissertation. University of Málaga, Spain, (in spanish), 2000.
8. Mérida Casermeiro, E., Galán Marín, G. & Muñoz Pérez, J. An efficient multiv. Hopfield N. for the T.S.P. *Neural Procc. Letters*. **14**, 203-16, (2001).
9. Mérida Casermeiro, E., Muñoz Pérez, J. and Benítez Rochel, R. A recurrent multivalued N.N. for N-Queens Problem. *LNCS* **2084**, 522-29, (2001).
10. Mohr T. and Pasche C., A Parallel Shortest Path Alg., *Computing* **40** 767-86, (1988).
11. Paige, R. and Kruskal, C. Parallel algorithms for shortest paths problems. *Proceedings International Conf. on Parallel Processing*, 14-19, (1989).
12. Park D-C. and Choi S-E. A N.N. based multi-destination routing algorithm for communication network. *IEEE Int. Joint Conf. N.N.* **2**, 1673-78, (1998).
13. Rauch H.E. and Winarske T. Neural Networks for routing communication traffic. *IEEE Cont. Syst. Mag.*, 26-30, (1988).
14. Serpen, Gursen and Parvin, Azadeh. On the performance of Hopfield network for graph search problem. *Neurocomputing* **14**, 365-81, (1997).
15. Zhang L. and Thomopoulos S. C. A., N.N. implem. of the shortest path alg. for traffic routing in comm. networks. *Proc. Int. Conf. N.N.* **II**, 591, (1989).

Spurious minima and basins of attraction in higher-order Hopfield networks*

M. Atencia¹, G. Joya², and F. Sandoval²

¹ Departamento de Matemática Aplicada. E.T.S.I.Informática

² Departamento de Tecnología Electrónica. E.T.S.I.Telecomunicación
Universidad de Málaga, Campus de Teatinos, 29071 Málaga, Spain
matencia@ctima.uma.es

Abstract. A theoretical investigation of the dynamics of continuous Hopfield networks, in a modified formulation proposed by Abe, is undertaken. The fixed points are classified according to whether they lie inside or they are vertices of the unit hypercube. It is proved that interior equilibria are saddle points. Besides, a procedure is sketched that determines the basins of attraction of stable vertices. The calculations are completed for the two-neuron network. These results contribute to a solid foundation for these systems, needed for the study of practical problems such as local minima or convergence speed.

1 Introduction

In this work, a dynamical analysis of higher-order continuous Hopfield networks in the Abe formulation, is performed. Hopfield [1] proved the stability of these systems by defining a Lyapunov function for first order networks, and the higher order generalization -e.g. [2, 3]- is straightforward. An important application of the Hopfield model is combinatorial optimization [4], which is accomplished by, on one hand, identifying the Lyapunov function with the target function, and on the other hand, representing the attained solution by the stable equilibrium reached by the network. Usually, the objective function is multilinear and $\{1, -1\}$ are the only permitted values for valid solutions [5]. Hence, stable fixed points that lie inside the unit hypercube -i.e. they are not vertices- are spurious solutions that do not represent any valid configuration of the optimization problem. However, the Lyapunov function of Hopfield networks contains an integral term that does not match the multilinear target, thus producing stable interior equilibria. Several approaches have addressed this problem, but it is not completely solved. In contrast, Abe proposed a closely related formulation [6], which has the advantage that the Lyapunov function exactly matches a multilinear target. This fact explains that the Abe network is used in many applications, although this is not always conveniently detailed. Besides, theoretical studies of the Abe dynamics are scarce, they are often problem-dependent [7] or limited to first

* This work has been partially supported by the Spanish Ministerio de Ciencia y Tecnología (MCYT), Project No. TIC2001-1758.

order networks [8]. This is in contrast to the abundance of recent contributions on the Hopfield formulation, e.g. [9] and references therein. Thus, Abe networks deserve a deep theoretical investigation in order to clarify the existence and stability of interior fixed points, as well as the boundaries of the basins of attraction of stable vertices. Even if general results can not be obtained due to the complexity of these nonlinear systems, results on two-neuron networks uncover many of the interesting qualitative properties of higher dimensional networks. This simplified study has been undertaken for the Hopfield dynamics [10] where it was proven worthwhile, but it is still lacking for the Abe model.

The Abe formulation of Hopfield continuous networks is defined by the following system of ordinary differential equations (ODEs):

$$\begin{aligned} \frac{du_i}{dt} &= \text{net}_i; \quad s_i(t) = \tanh\left(\frac{u_i(t)}{\beta}\right) \\ \text{net}_i &= \sum_{q=1}^r \sum_{\substack{(i_1, i_2, \dots, i_q) \in C_q^n \\ i \neq i_1, i_2, \dots, i_q}} w_{i i_1, i_2, \dots, i_q} s_{i_1} s_{i_2} \dots s_{i_q} - b_i \end{aligned} \quad (1)$$

where n is the number of neurons, r is the order of the network, $w_{i i_1, i_2, \dots, i_q}$ is the weight of the q -th order connection from neurons $i_1 \dots i_q$ to neuron i , b_i is the bias of neuron i , and C_q^n represents combinations of q elements among the first n natural numbers. This model has the following Lyapunov function:

$$V(\mathbf{s}) = - \sum_{q=1}^r \sum_i \sum_{\substack{(i_1, i_2, \dots, i_q) \in C_q^n \\ i \neq i_1, i_2, \dots, i_q}} w_{i i_1, i_2, \dots, i_q} s_i s_{i_1} s_{i_2} \dots s_{i_q} + \sum_i b_i s_i \quad (2)$$

The main aim of this contribution is twofold. Firstly, in Section 2 the fixed points of the network are determined and classified into vertices and interior points. A proof of instability of interior fixed points is provided, which includes the more particular first order result by Abe [8]. Secondly, based upon these results, in Section 3 a procedure is sketched in order to determine the basins of attraction of stable vertices, since the boundary between two basins crosses the unstable interior equilibrium. The detailed calculations are performed for the two-neuron network, leading to an approximate expression of the boundary. Finally, Section 4 summarizes the main contributions of the paper and offers some open fields for further research.

2 Stability

In this section, some results on stability of the Abe model are obtained by means of linearization and a local eigenvalue analysis. This aim first requires formulating the model equation (1) as a single ODE:

$$\frac{ds_i}{dt} = \frac{ds_i}{du_i} \frac{du_i}{dt} = \left(\frac{1}{\beta}\right) (1 - s_i^2) \text{net}_i \triangleq f_i(\mathbf{s}) \quad (3)$$

The fixed points of this system can be classified into three classes:

- The vertices of the unit hypercube: $\mathbf{s} = \{-1, 1\}^n$.
- The interior fixed points, that lead to $\mathbf{net} = \mathbf{0}$.
- The points such that $|s_i| = 1 \forall i \in I$ and $\text{net}_i = 0 \forall i \notin I$ for some subset $I \subset \{1 \dots n\}$. These points, which are a mixture of the two previous classes, lie on sides of the hypercube.

Since combinatorial optimization is intended, valid results should be obtained at the vertices and stable fixed points inside the hypercube are undesirable. In order to determine whether interior equilibria are unstable, we calculate the jacobian of the function f , taking into account $\partial \text{net}_i / \partial s_i = 0$, since there are no self weights:

$$\mathbf{J}(\mathbf{s}) = \left(\frac{\partial f_i}{\partial s_j} \right)_{ij}$$

$$\frac{\partial f_i}{\partial s_i} = -\frac{2}{\beta} s_i \text{net}_i; \quad \frac{\partial f_i}{\partial s_j} = \frac{1}{\beta} (1 - s_i^2) \frac{\partial \text{net}_i}{\partial s_j} \quad j \neq i \quad (4)$$

Next we prove the main theorem of this section, preceded by an auxiliary lemma:

Lemma 1. *The jacobian of f at a non vertex fixed point \mathbf{s} is a matrix with zero diagonal, and it is similar to a symmetric matrix.*

Proof. If \mathbf{s} is a fixed point, but $|s_i| \neq 1 \forall i$, then $\text{net}_i = 0 \forall i$ must hold. Then, $\partial f_i / \partial s_i = 0$ and the jacobian $\mathbf{J}(\mathbf{s}) = \mathbf{D} \mathbf{H}$ is a zero diagonal matrix, that results from multiplying the i -th row of $\mathbf{H} = (\partial \text{net}_i / \partial s_j)_{ij}$ by $(1/\beta)(1 - s_i^2)$, which is the i -th diagonal element of \mathbf{D} . Next we prove the symmetry of \mathbf{H} , observing that the Lyapunov function defined in Equation (2) fulfills $\partial V / \partial s_i = -\text{net}_i$ so that:

$$h_{ij} = \frac{\partial \text{net}_i}{\partial s_j} = -\frac{\partial^2 V}{\partial s_i \partial s_j} \quad (5)$$

and the symmetry of \mathbf{H} stems from the interchangeability of the order of partial derivation. The final assertion results from the similarity transformation [11]:

$$\mathbf{J}(\mathbf{s}) \sim D^{-\frac{1}{2}} \mathbf{J}(\mathbf{s}) D^{\frac{1}{2}} = D^{-\frac{1}{2}} \mathbf{D} \mathbf{H} D^{\frac{1}{2}} = D^{\frac{1}{2}} \mathbf{H} D^{\frac{1}{2}} = \mathbf{A} \quad (6)$$

which is possible because \mathbf{D} is positive definite as long as \mathbf{s} is not a vertex. The final matrix \mathbf{A} is symmetric, since so is \mathbf{H} .

Theorem 1. *Let V be the target function of an optimization problem and let Equation (1) define the dynamics of the neural solver. Given a non vertex fixed point \mathbf{s} , if the hessian of $V(\mathbf{s})$ is not the zero matrix, then \mathbf{s} is unstable.*

Proof. From the previous lemma, the jacobian evaluated at \mathbf{s} is a zero diagonal matrix. Now, recall that for any matrix, the sum of its eigenvalues coincides with its trace, e.g. [12]. Hence the sum of the eigenvalues of the jacobian is zero and, either there is at least one positive eigenvalue, or all eigenvalues are

zero. In the former case, the point \mathbf{s} is unstable and the proof is finished. But the later case leads to a contradiction. It is well known that every symmetric matrix is diagonalizable and all its eigenvalues are real. Since $\mathbf{J}(\mathbf{s})$ is similar to a symmetric matrix, it is also similar to a real diagonal matrix \mathbf{A} whose diagonal elements are its eigenvalues. If all eigenvalues of $\mathbf{J}(\mathbf{s})$ are null, so are the eigenvalues of \mathbf{A} , which is then the zero matrix and, by similarity, $\mathbf{J}(\mathbf{s})$ is also zero. In turn, since \mathbf{D} is nonsingular outside vertices, \mathbf{H} also vanishes. Finally, observe in the proof of the lemma that the hessian of V at \mathbf{s} is the matrix $-\mathbf{H}$ that, by hypothesis, is nonzero, and a contradiction results.

The condition on the hessian in Theorem 1 seems rather technical and difficult to prove in the general case. This is easily explained since $\mathbf{H} = \mathbf{0}$ implies that \mathbf{s} is a non hyperbolic fixed point, whose analysis is considerably more complicated than in the hyperbolic case. A detailed study of this possibility, via the center manifold theory, is left for further research. Anyway, in practical applications, \mathbf{H} is usually nonsingular due to the way the target function is constructed.

Next we analyse the mixed fixed points, given by $|s_i| = 1 \forall i \in I$ and $\text{net}_i = 0 \forall i \notin I$. These points can be studied as interior points within the side they lie on, which is an hypercube of reduced dimensionality:

Lemma 2. *Consider a fixed point \mathbf{s} that lies on a side of the hypercube: $\mathcal{S} = \{|s_i| = 1 \forall i \in I\}$. If the hessian of $V(\mathbf{s})$ is nonsingular, then \mathbf{s} is unstable.*

Proof. The point \mathbf{s} is an interior point within \mathcal{S} . Apply Theorem 1 restricted to the hypercube \mathcal{S} , and conclude that either \mathbf{s} is unstable or a principal submatrix of \mathbf{H} is the zero matrix, concretely $h_{ij} = 0 \forall i, j \in I$. Also, from Equation (4), $h_{ij} = 0 \forall i \notin I \forall j \in I$, since the rows $i \notin I$ correspond to $|s_i| = 1$. Thus, if \mathbf{s} is stable, then \mathbf{H} has at least one zero column, hence it is singular, which contradicts the hypothesis.

The results on first order networks that appear in [8] are easy corollaries of the statements presented in this section, but the proofs are here conceptually simpler. To the best of our knowledge, the proofs for higher order networks are novel. Furthermore, the fact that interior fixed points are saddle points is emphasized, since it suggests the situation of the boundaries between basins of attraction of stable vertices.

3 Basins of attraction

In this section we calculate an approximation to the basins of attraction of stable vertices, adapting standard techniques from the theory of dynamical systems - see [13] for background. Since interior fixed points are unstable, they could be considered unimportant to any practical extent. However, their importance stems from the fact they help determine the basins of attraction of stable vertices: it is known that in many dynamical systems the boundary of the basin of attraction of a stable fixed point is the stable manifold of a saddle point. Note in Theorem

1 that interior equilibria are saddle points, since the jacobian possesses both positive and negative eigenvalues. We perform the detailed calculations for the simplest Hopfield network, which is a first order network with only two neurons, although in principle the method is applicable to any system. The procedure for finding the stable manifold of the saddle point begins with formulating an hypothetical function g that defines the manifold. The lowest order coefficient in g is determined because the manifold must cross the saddle. Besides, the first order coefficients are fixed, since the stable manifold must be tangent to the stable linear subspace. The stable subspace is spanned by the eigenvectors that correspond to negative eigenvalues of the jacobian. Finally, higher order coefficients can be calculated by imposing the condition that the stable manifold must be invariant. In this section we assume $\beta = 1$ in order to simplify notation, since this leads to the same qualitative dynamical behaviour.

Firstly, we calculate and diagonalize the jacobian. The Lyapunov function of the general two-neuron network has the form $V = -w s_1 s_2 + b_1 s_1 + b_2 s_2$, so that the linear terms of the dynamical equation are $net_1 = w s_2 - b_1$ and $net_2 = w s_1 - b_2$. The condition $net_i = 0 \forall i$ that defines an interior fixed point $\tilde{\mathbf{s}}$ leads to $\tilde{s}_1 = b_2/w$ and $\tilde{s}_2 = b_1/w$. Hence the jacobian reduces to:

$$\mathbf{J}(\tilde{\mathbf{s}}) = \begin{pmatrix} 0 & (1 - \tilde{s}_1^2) w \\ (1 - \tilde{s}_2^2) w & 0 \end{pmatrix} \quad (7)$$

The eigenvalues λ of this simple matrix can be explicitly calculated and also the matrix \mathbf{B} of the base that diagonalizes \mathbf{J} has a compact expression:

$$\lambda = \pm \sqrt{(1 - \tilde{s}_1^2)(1 - \tilde{s}_2^2)} w; \quad \mathbf{B} = \begin{pmatrix} \sqrt{1 - \tilde{s}_1^2} & -\sqrt{1 - \tilde{s}_1^2} \\ \sqrt{1 - \tilde{s}_2^2} & \sqrt{1 - \tilde{s}_2^2} \end{pmatrix} \quad (8)$$

Assume, to fix ideas, that $w > 0$ and the second column \mathbf{B}_2 of \mathbf{B} is the eigenvector corresponding to the negative eigenvalue. So the stable manifold \mathcal{M}^s of the saddle point $\tilde{\mathbf{s}}$ is tangent to the vector \mathbf{B}_2 .

Next, we obtain an approximate expression of the stable manifold. Intuition suggests that the role played by both variables s_1 and s_2 should be symmetric so that, rather than a standard asymptotic expansion, we propose the form $\mathcal{M}^s = \{(s_1, s_2) \text{ such that } g(s_1, s_2) = 0\}$ where the function g is given by:

$$g(s_1, s_2) = \sum_i \sum_j c_{ij} (s_1 - \tilde{s}_1)^i (s_2 - \tilde{s}_2)^j \quad (9)$$

and the coefficients c_{ij} must be chosen appropriately. For this expression to be consistent, its zero-th order term must be consistent with \mathcal{M}^s crossing the fixed point $\tilde{\mathbf{s}}$. The notation given in (9) has been chosen in order to give the simple condition $c_{00} = 0$. Furthermore, the first order terms are forced by \mathcal{M}^s being tangent to the stable subspace, which is the line spanned by the vector \mathbf{B}_2 , i.e. $\sqrt{1 - \tilde{s}_2^2} s_1 + \sqrt{1 - \tilde{s}_1^2} s_2 = 0$, hence this line and g must coincide up to first order, which results in the coefficients $c_{10} = \sqrt{1 - \tilde{s}_2^2}$ and $c_{01} = \sqrt{1 - \tilde{s}_1^2}$. The remaining coefficients can be calculated by forcing the necessary condition for

\mathcal{M}^s being the stable manifold of $\tilde{\mathbf{s}}$: it must be an invariant set. Hence, $d g / dt$ must vanish at any point that belongs to \mathcal{M}^s . The procedure for calculating the coefficients c_{ij} is an adaptation to our case of the standard asymptotic techniques and it consists of the following steps: the derivative $d g / dt$ is expanded; we arrange the terms so that we obtain $d g / dt = k g + h$; finally, the coefficients are chosen to annihilate all terms in h , up to certain order. Since this order can be arbitrarily chosen, this technique allows the determination of g to any degree of accuracy. For the sake of brevity, we perform the calculations for $i + j \leq 3$, $i, j \leq 2$. Hence the partial derivatives of g are:

$$\begin{aligned}\frac{\partial g}{\partial s_1} &= c_{10} + 2c_{20}T_{10} + c_{11}T_{01} + 2c_{21}T_{11} + c_{12}T_{02} \\ \frac{\partial g}{\partial s_2} &= c_{01} + 2c_{02}T_{01} + c_{11}T_{10} + 2c_{12}T_{11} + c_{21}T_{20}\end{aligned}\quad (10)$$

where the simplifying notation $T_{ij} = (s_1 - \tilde{s}_1)^i (s_2 - \tilde{s}_2)^j$ has been adopted. Also, the dynamical equation of the system can be simplified using the already obtained values c_{10} and c_{01} , resulting, after some algebra, in:

$$\begin{aligned}\frac{1}{w} \frac{ds_1}{dt} &= c_{01}^2 T_{01} - 2\tilde{s}_1 T_{11} - T_{21} \\ \frac{1}{w} \frac{ds_2}{dt} &= c_{10}^2 T_{10} - 2\tilde{s}_2 T_{11} - T_{12}\end{aligned}\quad (11)$$

hence the time derivative of g is obtained:

$$\begin{aligned}\frac{1}{w} \frac{dg}{dt} &= \frac{1}{w} \left(\frac{\partial g}{\partial s_1} \frac{ds_1}{dt} + \frac{\partial g}{\partial s_2} \frac{ds_2}{dt} \right) = c_{10}c_{01}^2 T_{01} + c_{01}c_{10}^2 T_{10} \\ &+ (2c_{20}c_{01}^2 + 2c_{02}c_{10}^2 - 2\tilde{s}_1 c_{10} - 2\tilde{s}_2 c_{01}) T_{11} + c_{01}^2 c_{11} T_{02} + c_{10}^2 c_{11} T_{20} \\ &+ c_{01}^2 c_{12} T_{03} + (-c_{10} - 4c_{20}\tilde{s}_1 + 2c_{12}c_{10}^2 - 2\tilde{s}_2 c_{11}) T_{21} \\ &+ c_{10}^2 c_{21} T_{30} + (-c_{01} - 4c_{02}\tilde{s}_2 + 2c_{21}c_{01}^2 - 2\tilde{s}_1 c_{11}) T_{12} \\ &+ (-2c_{20} - 2c_{21}\tilde{s}_2) T_{31} + (-2c_{02} - 2c_{12}\tilde{s}_1) T_{13} \\ &+ (-4\tilde{s}_1 c_{21} - 4\tilde{s}_2 c_{12} - c_{11} - c_{11}) T_{22} - 3c_{21} T_{32} + -3c_{12} T_{23}\end{aligned}\quad (12)$$

where the first two terms coincide with those of g , multiplied by the constant $c_{10}c_{01}$. Thus, this expression can be cast into the form $d g / dt = kg + h$:

$$\begin{aligned}\frac{1}{w} \frac{dg}{dt} &= c_{10}c_{01}g + (2c_{20}c_{01}^2 + 2c_{02}c_{10}^2 - 2\tilde{s}_1 c_{10} - 2\tilde{s}_2 c_{01} - c_{10}c_{01}c_{11}) T_{11} \\ &+ (c_{01}^2 c_{11} - c_{10}c_{01}c_{20}) T_{02} + (c_{10}^2 c_{11} - c_{10}c_{01}c_{02}) T_{20} \\ &+ (-c_{10} - 4c_{20}\tilde{s}_1 + 2c_{12}c_{10}^2 - 2\tilde{s}_2 c_{11} - c_{10}c_{01}c_{21}) T_{21} \\ &+ (-c_{01} - 4c_{02}\tilde{s}_2 + 2c_{21}c_{01}^2 - 2\tilde{s}_1 c_{11} - c_{10}c_{01}c_{12}) T_{12}\end{aligned}\quad (13)$$

where higher order terms have been neglected, since they can not be annihilated without including additional terms. Finally, we obtain the formulas for the

coefficients, by solving the system that results if we force each term to vanish:

$$\begin{aligned} c_{20} &= \frac{c_{10}}{c_{01}} c_{11}; \quad c_{02} = \frac{c_{01}}{c_{10}} c_{11}; \quad c_{11} = \frac{2}{3} \left(\frac{\tilde{s}_1}{c_{01}} + \frac{\tilde{s}_2}{c_{10}} \right) \\ c_{12} &= \frac{1}{c_{10}} \left(1 + \frac{8}{3} \frac{c_{11}\tilde{s}_2}{c_{10}} + \frac{10}{3} \frac{c_{11}\tilde{s}_1}{c_{01}} \right); \quad c_{21} = \frac{1}{c_{01}} \left(1 + \frac{8}{3} \frac{c_{11}\tilde{s}_1}{c_{01}} + \frac{10}{3} \frac{c_{11}\tilde{s}_2}{c_{10}} \right) \end{aligned} \quad (14)$$

In Figure 1 the basins of attraction for two networks, with different values of the bias vector \mathbf{b} , are represented. They have been numerically calculated by simulating the networks for a large number of initial conditions. Also, the approximation obtained by the proposed method up to first, second and third order is drawn. It is observed that even the second order approximation is fairly good when the saddle is near the main antidiagonal of the unit square. Then, the stable manifold is almost coincident with the stable linear subspace. However, when the stable manifold is highly nonlinear, the second order approximation does not even separate the phase space in two regions. Although the third order approximation loses accuracy when it is far from the saddle, at least it separates the space in two domains, thus providing some insight on the relative size of the basins. Note that this loss of accuracy is a logical consequence of local methods, and it is common to most techniques in dynamical systems theory.

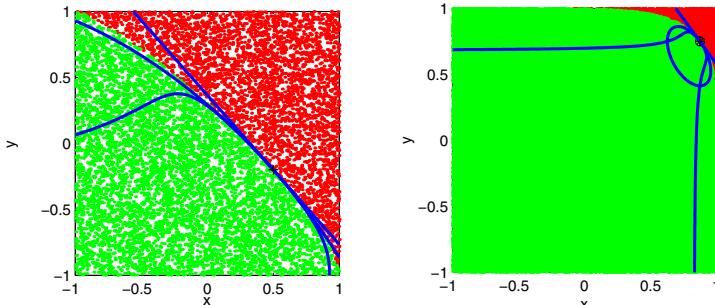


Fig. 1. Numerically obtained basins of attraction and three different approximations to the stable manifold of the saddle points: $\tilde{\mathbf{s}} = (0.5, -0.2)$ for the first network and $\tilde{\mathbf{s}} = (0.875, 0.75)$ for the second one. (A larger number of simulations have been performed for the second, highly nonlinear network)

4 Conclusions and lines of future research

This contribution aims at filling the gap that exists in the study of the Abe formulation of Hopfield networks. A theoretical analysis of these networks is accomplished, with the tools of dynamical systems theory. As a result, it is proved that fixed points that lie inside the unit hypercube are unstable. Furthermore,

they are saddle points, i.e. they possess non trivial stable and unstable manifolds. This fact sheds some light on the basins of attraction of stable vertices: the boundary between basins is the stable manifold of the saddle. Consequently, we explain a method to calculate this stable manifold. The detailed calculations are performed for two-neuron networks.

This work opens several paths for further research. The calculations are being implemented on a symbolic programming language, so as to obtain a high number of terms of the stable manifold, thus improving the accuracy of the basin approximation. Besides, we are studying the practical consequences of the presented results, aimed at improving the optimization performance of Hopfield networks. In particular, the relation between the basins of attraction and the network parameters is being determined, to explore the possibility of reducing the size of basins of local minima. Finally, we are extending these results to higher-order, higher-dimensional networks.

References

1. Hopfield, J.: Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Natl. Acad. Sci. USA* **81** (1984) 3088–3092
2. Samad, T., Harper, P.: High-order Hopfield and Tank optimization networks. *Parallel Computing* **16** (1990) 287–292
3. Joya, G., Atencia, M.A., Sandoval, F.: Hopfield neural networks for optimization: Study of the different dynamics. *Neurocomputing* **43** (2002) 219–237
4. Tank, D., Hopfield, J.: 'Neural' computation of decisions in optimization problems. *Biological Cybernetics* **52** (1985) 141–152
5. Vidyasagar, M.: Minimum-seeking properties of analog neural networks with multi-linear objective functions. *IEEE Trans. on Automatic Control* **40** (1995) 1359–1375
6. Abe, S.: Theories on the Hopfield neural networks. In: *Proc. IEE International Joint Conference on Neural Networks*. Volume I. (1989) 557–564
7. Takefuji, Y., Lee, K.C.: Artificial neural networks for four-coloring map problems and k-colorability problems. *IEEE Trans. On Circuits and Systems* **38** (1991) 326–333
8. Abe, S.: Global convergence and suppression of spurious states of the Hopfield neural networks. *IEEE Trans. on Circuits and Systems-I* **40** (1993) 246–257
9. Chen, T., Amari, S.I.: New theorems on global convergence of some dynamical systems. *Neural Networks* **14** (2001) 251–255
10. Tino, P., Horne, B., Giles, C.: Attractive periodic sets in discrete-time recurrent networks (with emphasis on fixed-point stability and bifurcations in two-neuron networks). *Neural Computation* **13** (2001) 1379–1414
11. Demmel, J.: Hermitian eigenproblems. In: Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, H. van der Vorst, editors. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM (2000)
12. Golub, G.H., van Loan, C.F.: Matrix computations. The Johns Hopkins University press (1996)
13. Guckenheimer, J., Holmes, P.: *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*. Springer Verlag (1997)

Cooperative Co-evolution of Multilayer Perceptrons

P.A. Castillo, M.G. Arenas, J.J. Merelo, and G. Romero

Department of Architecture and Computer Technology
University of Granada. Campus de Fuentenueva. E. 18071 Granada (Spain)

e-mail: pedro@atc.ugr.es

Abstract. Co-evolution is a possible solution to the problem of simultaneous optimization of artificial neural network and training algorithm parameters, due to its ability to deal with vast search spaces. Moreover, this scheme is recommendable when the optimization problem is decomposable in subcomponents.

In this paper an approach to cooperative co-evolutionary optimisation of multilayer perceptrons, that improves the G-Prop genetic back-propagation algorithm, is presented.

Obtained results show that this co-evolutionary version of G-Prop obtains similar or better results needing much fewer training epochs and thus using much less time than the sequential versions.

1 Introduction

Using artificial neural networks (ANN) requires establishing the structure in layers and connections between them, the parameters (such as initial weights) and a set of learning constants. This is followed by a training method, which is usually an iterative gradient descent algorithm. However, gradient descent methods, successful as they are in many fields, do encounter certain difficulties in practice: 1) the convergence tends to be extremely slow; 2) convergence to the global optimum is not guaranteed; 3) learning constants and other parameters must be found heuristically.

These latter two problems, premature convergence and parameter setting, have been approached using several optimization procedures, such as incremental/decremental methods [14] or evolutionary algorithms [20].

Evolutionary neural networks are an efficient way of searching, however, the main problem with these methods is that they usually concentrate on optimizing the architecture and the weights, disregarding the learning constants (number of training epochs and learning parameter), in spite of their importance: a low *number of training epochs* could not be enough to learn the training set, while a number of training epochs higher than needed could lead to over-training; on the other hand, a small *value of the learning parameter* can make the convergence too slow and the gradient descent algorithm may be trapped in a local minima, while a value higher than needed could make it step over the optimum.

As optimization method, evolutionary algorithms (EA) use a single genetic encoding for finding the solution to the problem at hand (i.e., a single “species”). The method proposed in [9] tries to avoid overfitting encoding the number of training epochs as a bit string in the individual. However, this representation can lead to a lack of precision.

Co-evolutionary algorithms, employing more than one interacting “species” evolving under different evaluation functions, can be used to solve hard optimisation problems in a more efficient way than single species EA [6]. Moreover, the cooperative model is suitable when the problem solution is decomposable in subcomponents and there are strong interdependencies among them.

In this work we propose using SOAP for implementing a co-evolutionary algorithm. SOAP is a standard protocol proposed by the W3C ([19]) that extends the remote procedure call, to allow the remote access to objects. Nevertheless, SOAP is a high level protocol, which makes easy the task of distributing objects between different servers, without having to worry about the message formats, nor the explicit call to remote servers. In this paper we intend to explore abilities of SOAP, implementing a co-evolutionary algorithm using Perl and SOAP, to tune learning parameters and to set the initial weights and hidden layer size of a MLP, based on an EA and Quick Propagation [4] (QP).

This paper continues the research on evolutionary optimisation of MLP (*G-Prop* method) presented in [1, 3]. G-Prop leverages the capabilities of two classes of algorithms: the ability of EA to find a solution close to the global optimum, and the ability of the back-propagation algorithm (BP) to tune a solution and reach the nearest local minimum by means of local search from the solution found by the EA.

We propose a co-evolutionary system in which a population of QPs evolves in parallel with a population of MLPs; thus both network architecture (network structure and weights) and training parameters (number of training epochs and learning coefficient) are optimized.

The remainder is structured as follows: section 2 makes a short overview of the methods based on co-evolutionary algorithms. Section 3 describes the proposed model, section 4 describes the experiments, and section 5 presents results obtained, followed by a brief conclusion in Section 6.

2 Co-evolutionary Algorithms: Related Research

EAs [11] are global optimization methods, based on the theory of the natural evolution. Each generation the best solutions mate, while the worst, disappear. In order to distinguish between good and bad solutions, an objective function (evaluation or “fitness” function) that measures the quality of the solution (the closer an individual is to the optimal solution) is used.

Most EAs involve a single “species”, that is, a single genetic encoding aimed at finding solutions to a problem. Co-evolutionary algorithms [13] involve more than one “species” (populations) interacting among them. Each popula-

tion evolves separately in an EA, and to obtain the fitness of an individual of a population, some individuals of the other populations are taken into account.

According to the dependency between species (interactions between populations), following classification could be done:

- competitive co-evolutionary algorithms [17], where the fitness of an individual depend on competition with other individuals from other species (each species competes with the remainder).
- cooperative co-evolutionary algorithms [15], where the goal is to find individuals from which better systems can be constructed. The fitness of an individual depends on its ability to cooperate with individuals from other species to solve a target problem.

Most authors propose cooperative models: Smalz and Conrad [18] propose to use two populations evolved separately: a population of nodes, divided into clusters, and a population of networks that are combinations of neurons, one from each cluster. This method neither carries out competition among the neurons of the same cluster nor enforces cooperation among the different clusters of neurons.

Moriarty and Miikkulainen [12] developed a method for designing ANN based on two EAs: a population of nodes and another of networks (different ways of combining nodes of the first population). The nodes are coded using floating point vectors that represent weights.

Zhao [21] propose to decompose a pattern recognition problem in several functions (one per class); then assign a module-network to each function. Thus, the whole classifier (network) consists of N sub-systems, and these sub-systems (module-networks) are searched by evolution using several EAs.

Hallinan and Jackway [5] propose a cooperative feature selection algorithm which utilizes a genetic algorithm to select a feature subset in conjunction with the weights of a ANN. Each network was encoded as a binary string. This coding might lead to a lack of precision, and good solutions could be lost due to the limitations of the representation.

As commented, not only the network architecture and weights should be optimized, but also the learning constants. However, these methods concentrate on optimizing the network structure, despite the importance of training parameters.

3 The cooperative co-evolutionary model

In this work, we are interested on optimizing the QP training parameters, so that an EA that searches for those values will be used. The fitness of an individual-QP depends on the MLPs classification ability and on the number of training epochs (the higher the number of epochs, the higher the simulation time).

We propose a model where two species are evolving in a cooperative way. Both the MLP structure (architecture and weights) and the training parameters (number of training epochs and learning coefficient) should be optimized, since both determine the simulation time and the obtained classification ability. Thus, our model consists on:

- **EA with a population of QP algorithms** : in this EA, each individual is a QP algorithm that codes the number of training epochs and learning coefficient. Each individual fitness is calculated using the number of training epochs and the average classification ability (obtained by those MLPs that are trained with that QP -see below for details-).
- **EA with a population of MLP**: this EA optimizes the MLP (each individual is a MLP). In order to obtain the MLP-individual fitness, a QP (taken from the other population, with its number of training epochs and learning coefficient) is used to train that MLP on the training set; its classification ability is obtained on the validation set.

Instead of using an EA with a QP population and several MLPs to obtain the fitness, it would be advisable to use a MLP population that evolves along with the QP population, so that a good MLP to solve that problem is obtained.

3.1 Proposed model operation

Proposed model consists of two parts (see figure 1):

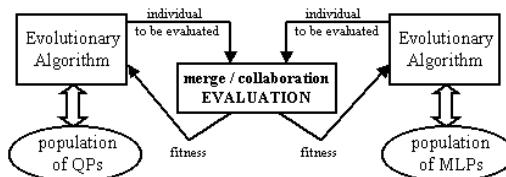


Fig. 1. Proposed model representation: EA that optimizes MLPs and uses individuals (QP) of a second EA to train the networks and thus to obtain the fitness.

EA to optimize QP algorithm parameters.

Each individual codes, using floating point vectors, the QP algorithm parameters (number of training times and learning coefficient). That QP, using those parameters, will be applied to train several MLPs: each generation, the EA that optimizes MLPs needs to evaluate the new generated MLPs; thus an individual-QP (with no fitness assigned) is sent to train those MLPs, obtaining the fitness value for those MLPs and for that QP. Fitness function uses as first criteria the optimization of the classification ability obtained with those MLPs (the average value is used), and as second criterium, the number of training epochs times (to minimize the simulation time).

A steady state algorithm was used because it was empirically found to be faster at obtaining solutions than other selection algorithms. For each generation, the best 30% individuals of the population are selected to mate using the genetic operators.

EA to optimize MLP (architecture and weights).

Proposed co-evolutionary model is based on the *G-Prop* method (optimization of MLPs using an evolutionary algorithm); since *G-Prop* has been described and analysed out in previous papers, thus we refer the reader to [1–3] for a full description. The EA optimises the classification ability of the MLP, and at the same time it searches for the number of hidden units (architecture) and the initial weight setting.

Implementation was carried out using the `SOAP::Lite` module [8] for the Perl programming language. In addition, servers are easy to implement using the computer infrastructure that exists in our department. The co-evolutionary algorithm has been implemented using the OPEAL library [10], available at <http://opeal.sourceforge.net> under GPL license. SOAP was included from the beginning in the OPEAL EC library, and so far, several distributed evolutionary algorithms configurations have been tested on EC benchmark problems [7].

4 Experiments

In these experiments, the **Glass** problem was used. This problem consists of the classification of glass types [16]. The results of a chemical analysis of glass splinters (percent content of 8 different elements) plus the refractive index are used to classify the sample. The data set contains 214 instances. Each sample has 9 attributes plus the class attribute.

The main data set was divided into three disjoint parts, for training, validating and testing. In order to obtain the fitness of an individual, the MLP is trained with the training set and its fitness is established from the classification error with the validating set. Once the EA is finished (when it reaches the limit of generations), the classification error with the testing set is calculated: this is the result shown.

We propose to study this classification problem using the following models:

- **sequential version** of G-Prop: the EA optimizes the MLP, although the training parameters values are fixed (they are obtained using statistical techniques) [3].
- **parallel version** of G-Prop: several machines work in parallel searching for the solution, interchanging (sending/receiving) some population individuals after some generations. The parallel case uses a *ring migration scheme*: sending some individuals to the next island (computer), and getting the best individuals from the previous island [2].
- **co-evolutionary model** to optimize both the MLP (classification ability and network structure) and the QP learning parameters (number of training epochs and learning coefficient) using two EAs.

Figure 2 shows the studied models in these experiments.

The co-evolutionary algorithm was executed using only “default” parameters (100 individual population, 0.3 as selection rate, and hidden layer sizes ranging from 2 to 90), genetic operators were applied using the same application rate. All these represent default parameter values which need not be fine-tuned for

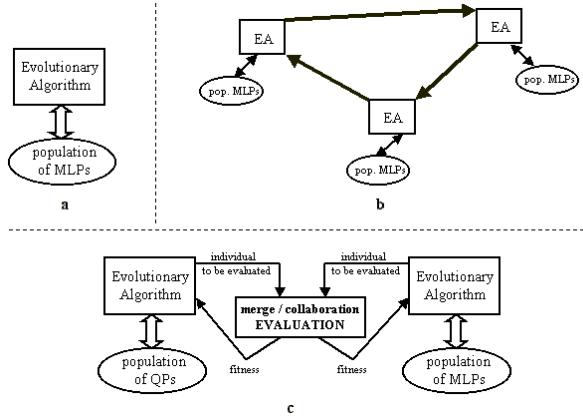


Fig. 2. Experiments studied in this work: a) EA for MLP optimization (G-Prop); b) parallel distributed EA for MLP optimization; c) Co-evolution of MLP and QP.

achieving the results presented here. The sequential and parallel versions were run using 300 training epochs and 0.1 as learning parameter, while in co-evolutive version, these parameters are optimised by the EA (see [1–3] for a detailed description of the sequential and parallel versions of the method).

5 Results

Results obtained using the sequential version are shown in Table 1:

Generations	Error (%)	Time (min.)	Training epochs
100	35 ± 1	20 ± 2	300
200	35 ± 1	30 ± 5	300
300	33 ± 2	44 ± 8	300
400	32 ± 2	70 ± 9	300

Table 1. Results (error % and time) obtained using the **sequential version**. In this experiment, the number of training epochs was fixed to 300 (see [1] for details).

Results obtained using the parallel version are shown in Table 2. As can be seen, better results in time and classification error are obtained dividing the problem between several computers.

Results obtained using the co-evolutive model are shown in Table 3. Classification ability obtained is similar to the obtained using other models (MLP optimization is carried out by the EA). At the time, simulation time is lower due to the optimization of the number of training epochs. On average, better results are obtained using a small learning coefficient, close to 0.02 (a small value can make the convergence too slow, while a value higher than needed could make to step over the optimum).

Islands	Error (%)	Time (min.)	Speedup	Training epochs
1	35 ± 1	20 ± 2	1	300
2	33 ± 2	18 ± 1	1.6	300
3	33 ± 1	20 ± 1	2.2	300
4	32 ± 1	20 ± 2	3.5	300

Table 2. Results (error % and time) obtained using the **parallel version** of the method, and the speedup for each experiment. In this experiment, the number of training epochs was fixed to 300 (see [2] for details).

Generations	Error (%)	Time (min.)	Training epochs	Learning coefficient
100	35 ± 2	22 ± 5	175 ± 16	0.0939417 ± 0.0034522
200	35 ± 1	31 ± 5	139 ± 19	0.0875031 ± 0.0025649
300	33 ± 2	40 ± 4	126 ± 13	0.0673017 ± 0.0029725
400	32 ± 2	65 ± 6	114 ± 12	0.0303432 ± 0.0078888

Table 3. Results (error %, time and number of training epochs) obtained using the **co-evolutionary version** of the model. As can be seen, this approach obtains lower number of training epochs than used using other models.

Obtained classification ability using these methods is similar. On the other hand, simulation time in the co-evolutionary approach is lower due to the fact that optimizing the number of training epochs makes the fitness calculation less expensive on average. It can be seen that the co-evolutionary model obtains MLPs with classification ability similar to others where the main target is the network optimization.

6 Conclusions and Work in Progress

This paper presents a co-evolutionary method to optimize both the MLP architecture (number of hidden units and initial weights) and the learning parameters to apply the QP algorithm (number of training epochs and learning coefficient).

We can see the validity of the proposed model: as the individual-QPs improve, the number training epochs is reduced; that makes the MLP training faster, and thus, simulation time is reduced. At the same time QPs are optimized, MLPs are optimized too, and thus the classification error improves.

Also it can be shown that the parallel version of the method (presented in [2]) obtains similar results, needing less simulation time. It would be interesting to implement and study the performance of the distributed co-evolutionary approach: along with the QP population there are several MLP populations evolving in parallel. Thus the simulation time could be reduced substantially and at the time the space search could be better explored. Along with the implementation of this approach, it would be of interest to study the learning coefficient values through the evolution and how it affects the classification ability and network size.

References

1. P. A. Castillo, J. J. Merelo, V. Rivas, G. Romero, and A. Prieto. G-Prop: Global Optimization of Multilayer Perceptrons using GAs. *Neurocomputing*, Vol.35/1-4, pp.149-163, 2000.
2. P.A. Castillo, M.G. Arenas, J. J. Merelo, V. Rivas, and G. Romero. Optimisation of Multilayer Perceptrons Using a Distributed Evolutionary Algorithm with SOAP. *Lecture Notes in Computer Science*, Vol.2439, pp.676-685, Springer-Verlag, 2002.
3. P.A. Castillo, J.J. Merelo, G. Romero, A. Prieto, and I. Rojas. Statistical Analysis of the Parameters of a Neuro-Genetic Algorithm. in *IEEE Transactions on Neural Networks*, vol.13, no.6, pp.1374-1394, ISSN:1045-9227, november, 2002.
4. S. Fahlman. Faster-Learning Variations on Back-Propagation: An Empirical Study. *Proc. of the 1988 Connectionist Models Summer School*, Morgan Kaufmann, 1988.
5. Jennifer Hallinan and Paul Jackway. Co-operative evolution of a neural classifier and feature subset. *Lecture Notes in Computer Science*, 1585:397-404, 1999.
6. P. Husbands. Distributed coevolutionary genetic algorithms for multi-criteria and multi-constraint optimisation. *Evolutionary Computing, Lecture Notes in Computer Science*, Vol. 865, pp.150-165, T. Fogarty (Ed.), Springer-Verlag, 1994.
7. J.J.Merelo, J.G.Castellano, and P.A.Castillo. Algoritmos evolutivos P2P usando SOAP. pages 31-37. Universidad de Extremadura, Febrero 2002.
8. P. Kuchenko. SOAP::Lite. Available from <http://www.soaplite.com>.
9. H.A. Mayer, R. Schwaiget, and R. Huber. Evolving topologies of artificial neural networks adapted to image processing tasks. In *Proc. of 26th Int. Symp. on Remote Sensing of Environment*, pp.71-74, Vancouver, BC, Canada, 1996.
10. J. J. Merelo. OPEAL, una librería de algoritmos evolutivos. *Actas del Primer Congreso Español de Algoritmos Evolutivos y Bioinspirados*. ISBN:84-607-3913-9. pp.54-59. Mérida, Spain, febrero, 2002.
11. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs* , Third, Extended Edition. Springer-Verlag, 1996.
12. D.E. Moriarty and R. Miikkulainen. Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation*, vol.4, no.5, 1998.
13. J. Paredis. Coevolutionary computation. *Artificial Life*, 2:355-375, 1995.
14. R. Parekh, J. Yang, and V. Honavar. Constructive Neural Network Learning Algorithms for Pattern Classification. *IEEE Transactions on Neural Networks*. 11(2), pp. 436-451, 2000.
15. M.A. Potter and K.A. De Jong. Cooperative coevolution: an architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1-29, 2000.
16. L. Prechelt. PROBEN1 — A set of benchmarks and benchmarking rules for neural network training algorithms. Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, D-76128 Karlsruhe, Germany, September 1994.
17. C.D. Rosin and R.K. Belew. New methods for competitive coevolution. *Evolutionary Computation*, 5(1):1-29, 1997.
18. R. Smalz and M. Conrad. Combining evolution with credit apportionment: A new learning algorithm for neural nets. *Neural Networks*, vol.7, no.2, pp.341-351, 1994.
19. D. Box; D. Ehnebuske; G. Kakivaya; A. Layman; N. Mendelsohn; H.F. Nielsen; S. Thatte; D. Winer. Simple Object Access Protocol (SOAP) 1.1, W3C Note 08 May 2000. Available from <http://www.w3.org/TR/SOAP>.
20. X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423-1447, 1999.
21. Q. Zhao. Co-evolutionary learning of neural networks. *Journal of Intelligent and Fuzzy Systems* 6, pp.83-90. ISSN 1064-1246, 1998.

A statistical model of pollution-caused pulmonary crises

Daniel Rodríguez-Pérez, Jose L. Castillo, and J.C. Antoranz

Departamento de Física Matemática y Fluidos, UNED,
28080 Madrid, Spain

{daniel, castillo, antoranz}@apphs.uned.es
<http://apphs.uned.es>

Abstract. Many natural complex systems show peculiar behaviors that are reproduced by computational systems with simple microscopic evolution rules. Here a model that was firstly introduced for the simulation of submicronic particle deposition and deposit growth is adapted as a model for the pulmonary damage caused precisely by such particles. The model allows to specify the structure of the “pulmonary tree” by means of a dimensionless parameter and also the flow pattern of the inhaled particles (due to air drag and diffusion). The model may qualitatively reproduces particle inhalation profiles obtained through detailed simulations. A continuous decrease of pulmonary volume and oxygenation surface is found, presenting “crises” in both magnitudes.

1 Introduction

Many properties observed in natural complex systems are reproduced in simple simulations, being one of the more often mentioned Witten and Sander’s [1] DLA aggregation model, which has properties found [2] in aerosol coagulation, dendrite growth, viscous fingering, electrodeposition, bacterial colonies growth, ... The main properties of DLA are [2]: growth instability, and shadowing of inner zones of the aggregate by already aggregated particles. However, the rules followed to grow a DLA cluster are really simple.

We apply a model introduced by Tassopoulos, O’Brian & Rosner [3] and previously used by Hui & Lenormand [4] to perform a simple simulation of the structure of submicronic particle deposits formed over a cold wall. The structure of the growing deposit (surface width, growth rate, bulk density, etc) depends on the Péclet number, Pe . This dimensionless parameter, Pe , is a measure of the ratio between the air drag velocity and the particle diffusion coefficient.

For low Péclet numbers, branched deposits, resembling the structure of DLA, are obtained, while for high Pe we recover the widely known ballistic deposition model (see, for instance, [5]). In 2-D simulations, a deposit density decrease with Pe , is observed due to pore formation which is an extreme case of shadowing.

Based on these previous studies we use a simple model to represent the damage suffered by the respiratory airways in the lungs in a polluted environment (maybe due to the presence of submicronic particles).

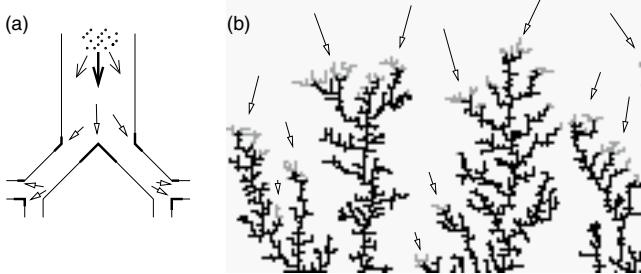
Several works have been published simulating this pollutant-bronchial tree interaction using complicated computational fluid dynamics simulations with precise geometric layout of the tracheo-bronchial region [6,7,8, and references therein], concluding

that most particles deposit near the bifurcations of the tree. Hence, these bifurcations play the same role as the tips in our model (see figure 1 for a pictorial representation).

The deposit growing algorithm is used to simulate not only particle deposition, but also the “rough” structure of the lung. Due to the anatomy of the bronchial tree, we will focus our attention on small Pe number grown, branched deposits.

We will seek for qualitative behaviors, matching those found in previously proposed models and clinical practice; however, we do not pretend to obtain really quantitative prediction, at this stage of our work.

Fig. 1. Analogy between bronchial tree deposition of particles and our deposit growth model. Triangle head arrows point to places of preferential deposition in each model. Their lengths represent probability.



2 Materials and methods.

The motion of a particle approaching a deposit can be approached by a mean velocity plus fluctuations in the particle velocity associated to the particle Brownian motion (diffusion). We use a two dimensional on-lattice Monte Carlo model with a dynamics described by a set of four jump probabilities. Time and space are discretized. Each time step a particle initially located at a given position moves to one of its four first neighbour sites with different probabilities for each site (see figure 2a). We choose their values such that, for long times, the position of a particle has the probability distribution expected for an ensemble of particles starting from the same point and fulfilling the corresponding diffusion-convection equation in a space of d dimensions. These probabilities for the displacement of the particle in the vertical direction are (see figure 2a for an explanation of the symbols)

$$p_+ = \frac{1}{2} [1 - 2(d-1)p_0 + \sqrt{1-2dp_0}]$$

$$p_- = \frac{1}{2} [1 - 2(d-1)p_0 - \sqrt{1-2dp_0}]$$

being p_0 , the lateral probabilities

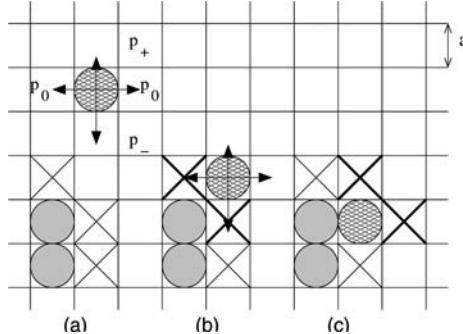
$$p_0 = \frac{\sqrt{d^2 + Pe^2} - d}{Pe^2}$$

and

$$Pe = \frac{av}{D}$$

where D is the diffusion coefficient, v the deterministic velocity and a the lattice constant, a , taken as the characteristic length in the process.

Fig. 2. Three steps in the model: a) particles (\odot) move according to a set of “jump” probabilities p_+ (up), p_- (down), and p_0 (left and right), b) particles are added to the deposit whenever they occupy an “active” site (\times), c) the non occupied neighbours of an attached particle (\odot) become active sites (\times).



Our simulations start with previously grown substrates generated by using a small Péclet number, Pe_0 . We have chosen a typical value of $Pe_0 = 0.1$, which gives a moderately branched structure, and a substrate 400 sites wide. We obtained saturated surfaces (i.e. whose structure, characterised with the r.m.s. of heights, does not change in time) after depositing 500 particle monolayers (1 monolayer = 400 particles, in this case). This initial substrate simulates the “clean” lung structure. A portion of the top of one of these substrates is represented on figure 1b.

Once the lung structure is generated, the particle laden air stream in the respiratory airways is simulated by the same model, but with a different Péclet number, Pe' , associated to the particle suspended in the stream. The time needed to deposit 400 particles will be taken as unit.

New arriving particles attach irreversibly to the substrate or to already attached particles as they reach a neighbouring site in their biased random walk (figure 2b,c). The height at which this occurs depends on both the structure of the deposit and the movement of the “incident” particle (that is, on Pe_0 and Pe'). We will characterise the generated deposit by the particle deposition histogram, that is, the fraction of deposited particles, $n(h)$, at a given height, h .

We define the lost surface of the deposit, ΔS , as the variation of the original deposit exposed surface due to particle adhesion normalized with the width of the simulation box (400 sites, in our case). This is the length (in our 2-D simulations) of the original deposit connected to the “exterior” (which is larger than the one accessible to new particles) and stands for the gas exchange surface of the lung. Its variation represents a decrease in the oxygen absorption rate by the blood. We expect these variations to be

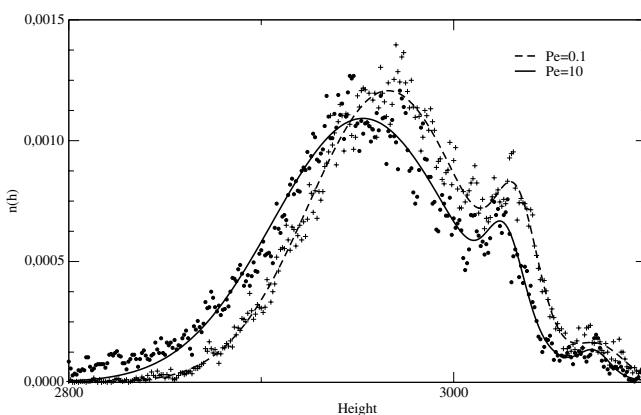
due to a series of discrete jumps occurring when a portion δS of the surface is isolated. We call these events “crises”.

We also define the “obstructed” volume, ΔV , as the area (again, in 2-D simulations) of the pores formed by “polluting” particles adhered to the “deposit”, normalized with the number of sites in the base (400 sites). We expect a correspondence to exist between ΔV and the reduction of lung “tidal volume” (the air volume moved during either the inspiratory or expiratory phase of each breathing cycle). In a similar way, pulmonary crises will appear as “instantaneous” variations, δV , of the blocked volume.

3 Results

Particle deposition occurs mainly in an interval of heights (pulmonary region). We calculate the attachment probability as a function of the deposit height for several values of Pe' (from 0.1 to 10). After depositing 4×400 particles and averaging the results for 50 deposits, distributions as those shown in figure 3, are obtained. For $Pe' = 0.1$ several preferred deposition heights become apparent and this profile is maintained for higher Pe' , converging to a distribution similar to that shown for $Pe' = 10$. These distributions have been fitted to a sum of three gaussians (centered in each maxima) for better representation because, in spite of the averages, the data are affected by a large dispersion.

Fig. 3. Particle arrival probability distribution for a deposit grown with $Pe_0 = 0.1$ (branched structure) and particles arriving with $Pe' = 0.1$ and $Pe' = 10$. Also shown are regression lines obtained by fitting the data as the sum of three gaussians.



ΔV and ΔS time evolutions are represented in figures 4 and 5. These evolution depicts the appearance of characteristic transients with a low and almost constant reduction rate of both magnitudes, due to incoming particles. These transients are separated by “crises” of intensities, δV and δS (the step sizes), that were calculated as forward time differences of ΔV and ΔS , respectively. The “crises” correspond to the closure of “pores” during the deposition process.

Fig. 4. Temporal evolution of “obstructed” volume ΔV (isolated in pores formed during deposition). Particles incide with $Pe' = 1.0$ (dashed line) and $Pe' = 10$ (solid line) on a deposit built with a $Pe_0 = 0.1$.

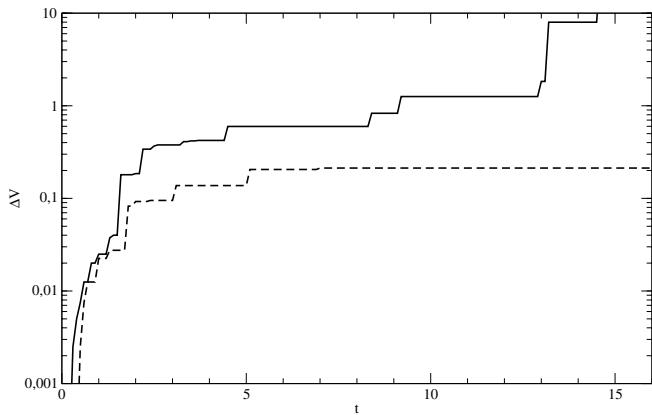
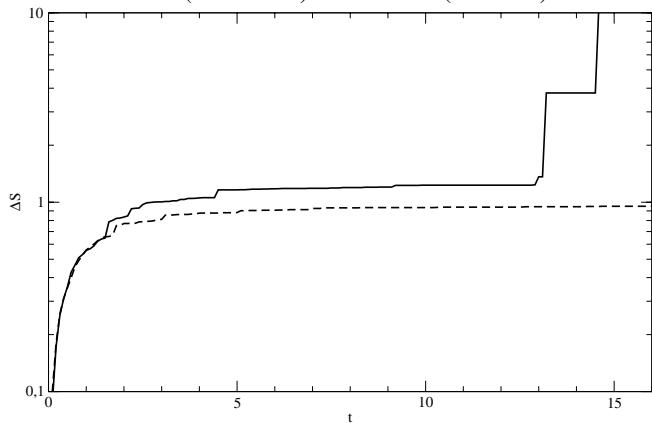


Fig. 5. Temporal evolution of the covered surface, ΔS , of a deposit grown with $Pe_0 = 0.1$ for incident particles with $Pe' = 1.0$ (dashed line) and $Pe' = 10$ (solid line).



4 Discussion

It can be seen from figure [3] that particles attach the deposit at heights around a value which (for a fixed Pe_0) deeps into the substrate as Pe' increases. This has a simple interpretation: profound inspirations will lead particles deeper in the bronchial tree. The region of deposition becomes narrower with increasing Pe' , but tends to a limit distribution (for simulations at $Pe' > 10$, the profiles do not change).

Several deposition maxima appear at higher positions (“in the tracheobronchial region”) which behave in a similar fashion and are more apparent for lower Pe' , making the total deposition interval wider.

Figures are similar to those reported by Salma et al. [8] for a simulation of several particle distributions (derived from experimentally measured ones) and various breathing regimes, using anatomical models for adult males and females. Our model reproduces the same behavior for resting (sleeping), sitting activity, and light exercise, corresponding to low (< 0.1), moderate (~ 1) and high (> 10) Péclet numbers.

In these regimes, the air flow is mainly laminar and particles can be characterised by their diffusion coefficient (independent of fluid velocity). In the heavy exercise regime, Salma et al. [8] report a splitting of the preferred deposition region in two: a proximal deposition, in the tracheobronchial region, and a deep deposition in the acinar region. This is found in our model only for low Pe' , and can be interpreted as an increase in the “effective” particle diffusion coefficient, due to the appearance of turbulence effect.

The importance of this coincidence is that the assumptions made to obtain these results in our model are very general: the presence of a branched interface (with a fractal structure) and the deposition of particles whose motion is characterized by a drift component (our deterministic velocity, v) and a diffusion component (D , in our model) that can be linked through a dimensionless Péclet number.

For ΔV and ΔS , a discontinuous time evolution is observed, due to the formation of pores with areas δV that isolate “gas exchange” lengths with size δS . These “crises” are more frequent at the beginning of the deposition for $Pe' > 1$. This can be interpreted as the beginning of a pulmonary injury, consisting of many small episodes. Larger crises occur as time passes and “pulmonary tidal volume” diminishes. We have disregarded those atypical cases when large crises occur at the beginning of the simulation, as they do not have any “physiological meaning”.

Cumulative frequency histograms for $\log \delta V$ s and $\log \delta S$ (see figures 6 and 7) show an almost linear dependence. These distributions describe a higher “crises” frequency the smaller their severity. According to our model, this sort of profile should appear in clinical statistics of the number of respiratory problems with a given severity (asthmatic crises, for example) caused by suspended particles. Once again, however, we cannot claim this distribution to be the “real” one, but only a qualitative feature appearing in all models having those characteristics described above.

5 Conclusions

We have studied the differential deposition profile for an on-lattice analog of the respiratory tree, as a function of tree structure and particle Pe' number, relating diffusive

Fig. 6. An accumulated frequency histogram for “volume crises” occurring in a deposit grown with $Pe_0 = 0.1$ due to particles approaching with $Pe' = 1.0$ (\times and dashed line of equation $a + 0.126 \log \delta V$) and $Pe' = 10$ (+ and continuous line of equation $a + 0.137 \log \delta V$).

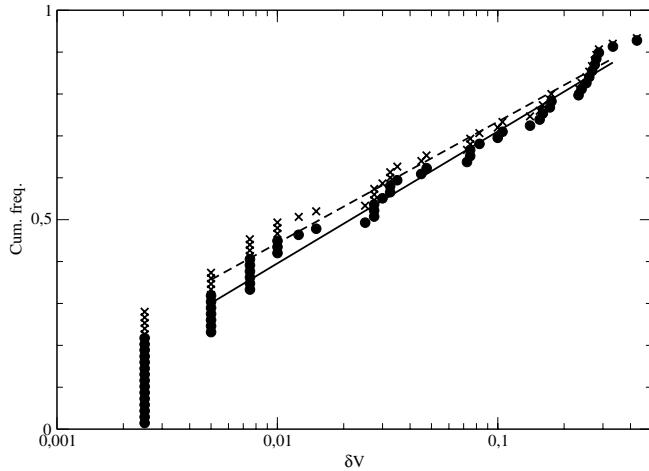
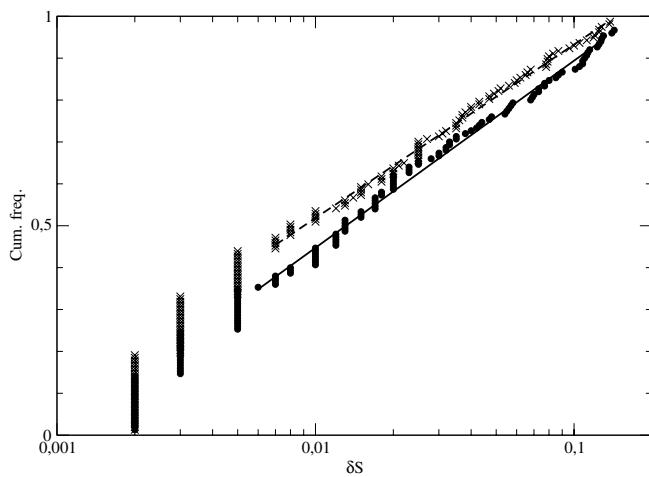


Fig. 7. An accumulated frequency histogram for “surface crises” occurring in a deposit grown with $Pe_0 = 0.1$ due to particles approaching with $Pe' = 1.0$ (\times and dashed line of equation, $a + 1.65 \log \delta S$) and $Pe' = 10$ (+ and solid line of equation $a + 1.79 \log \delta S$).



and convective (inspiration depth) effects. We find that this model reproduces qualitatively the same profiles found in more detailed simulations [8] and predicts chains of small obstructive pulmonary crises due to the exposure to air pollution.

Properties reproduced by this model should appear in systems where a flux of particles, with an associated *Pe* number (that is, having both diffusion and deterministic velocity), interact with a fractal rough structure. Whenever this interaction can result in the formation of pores (regions isolated from the environment), their size distribution should behave as the “respiratory crises” described above.

Acknowledgements

This work has been funded by *Ministerio de Ciencia y Tecnología* of Spain through project BFM2001-1314-C0302, by the SCIT 2000-2003 Contract–Program from the *Comunidad de Madrid*, Spain.

References

1. T.A. Witten, L.M. Sander: Diffusion-limited aggregation, a kinetic critical phenomenon. *Phys. Rev. Lett.* **47** (1981) 1400-1403
2. L.M. Sander: Diffusion-limited aggregation, a kinetic critical phenomenon?. *Contemporary Physics* **41** (2000) 203-218
3. M. Tassopoulos, J.A. O'Brien, D.E. Rosner: Simulation of microstructure/mechanism relationships in particle deposition. *AIChE Journal* **35** (1989) 967-980
4. D. Hui, R. Lenormand: Particle deposition on a filter medium. In: Family, Landau (eds): *Kinetics of aggregation and gelation*. Elsevier Science Publishers (1984) 173-176
5. Barabási, Stanley: *Fractal concepts in surface growth*. Cambridge University Press (1995)
6. Z. Zhang, C. Kleinstreuer, C.S. Kim: Cyclic micron-size particle inhalation and deposition in a triple bifurcation lung airway model. *Journal of Aerosol Science* **33** (2002) 257-281
7. C. Darquenne: A realistic two-dimensional model of aerosol transport and deposition in the alveolar zone of the human lung. *Journal of Aerosol Science* **32** (2001) 1161-1174
8. I. Salma, I. Balásházy, W. Hofmann, G. Záray: Effect of physical exertion on the deposition of urban aerosols in the human respiratory system. *Journal of Aerosol Science* **33** (2002) 983-997

A new penalty-based criterion for model selection in regularized nonlinear models

Elisa Guerrero¹, Joaquín Pizarro¹, Andrés Yáñez¹, Pedro Galindo¹

¹ University of Cádiz, Lenguajes y Sistemas Informáticos

11510 Puerto Real, Cádiz, Spain

{elisa.guerrero, joaquin.pizarro, andres.yanez, pedro.galindo}@uca.es

Abstract. In this paper we describe a new penalty-based model selection criterion for nonlinear models which is based on the influence of the noise in the fitting. According to Occam's razor we should seek simpler models over complex ones and optimize the trade-off between model complexity and the accuracy of a model's description to the training data. An empirical derivation is developed and computer simulations for multilayer perceptron with weight decay regularization are made in order to show the efficiency and robustness of the method in comparison with other well-known criteria for nonlinear systems.

1 Introduction

We consider nonlinear systems of the form:

$$y = g(x) + \xi \quad (1)$$

where y is the output signal, x denotes the multivariate vector input signal, g constitutes a lineal or nonlinear mapping and ξ is the inherent noise, which is supposed to be independent random variables with zero mean and variance σ_ξ^2 .

Let F be a set of nonlinear functions which are parameterized by an m -dimensional vector $w = \{w_1, w_2, \dots, w_m\}^T$ such as:

$$\hat{y} = f(x, w) \quad (2)$$

where f is an estimator of the underlying function g .

The generalization performance of the estimator can be measured by the prediction risk which is the expected error on the entire input domain:

$$PR(g) = \int L(y, f(x, w)) p(x, y) dx dy \quad (3)$$

where L is the discrepancy function, $p(x, y)$ is the unknown joint probability distribution. $PR(g)$ can be calculated only when exact knowledge of g , p , and σ_ξ^2 is available. Actually, one can at best obtain an estimate of $PR(g)$ since for a finite set of n observations $(x_i, y_i)_{i=1..n}$ we can obtain a set of parameters $\hat{w} = \{\hat{w}_1, \dots, \hat{w}_d\}$ which minimizes the corresponding cost function.

For the case of quadratic loss and a large number of observations, a significant number of penalty-based methods have been proposed such as these solutions minimize functionals of the form:

$$PR_{Gen} = PR_{Emp} * T(d, n) \quad (4)$$

where n is the sample size, PR_{Emp} is the minimum of the empirical risk when training with a model of size d (achieved by the function f_d), and $T(d, n)$ is the penalty term, which grows with the complexity of the model [16].

For linear models and unbiased nonlinear models different model selection criteria have appeared in the statistics literature such as Mallow's CP estimate [8], the Generalized Cross-Validation (GCV) formula [5], Akaike's Final Prediction Error (FPE) [1] and Akaike's Information Criteria (AIC) [2], etc. These criteria are based on the asymptotic properties of the estimates, a generalization of these estimates with better statistical properties is Barron's Predicted Squared Error (PSE) [3]. On the other hand, Vapnik-Chervonenkis VC-theory also provides analytical generalization bounds that can be used for estimating prediction risk [15], however it cannot be rigorously applied to nonlinear estimators (such as neural networks) where the VC-dimension cannot be accurately estimated and the empirical risk cannot be reliably minimized [4]. Alternatively, data resampling methods, such as k-fold Cross-validation (kCV) [14] or bootstrap estimation [6] make maximally efficient use of available data, but they can be very CPU time consuming for neural networks.

In this paper we address algebraic model selection criteria. For general nonlinear learning which may be biased and may include weight decay or other regularizers Moody [10] introduced the Generalized Prediction Error (GPE), which for a data sample of size n is:

$$GPE(\lambda) = PR_{Emp} + 2\hat{\sigma}_{\xi}^2 \frac{p_{eff}(\lambda)}{n} \quad (5)$$

where $\hat{\sigma}_{\xi}^2$ is an estimate of σ_{ξ}^2 (the noise variance on the data) and the regularization parameter λ controls the effective number of parameters $p_{eff}(\lambda)$ of the solution.

$$\hat{\sigma}_{\xi}^2 = \frac{SSE}{n - p_{eff}(\lambda)} \quad (6)$$

The effective number of parameters usually differs from the true number of model parameters (p_{true}). It depends upon the number of training samples, the regularization parameter and the nonlinearity of the problem. It can be estimated by the following expression:

$$p_{eff}(\lambda) \equiv \text{trace}(H_T H_R^{-1}) \quad (7)$$

where H_T and H_R the Hessians of the quadratic and regularized cost respectively [9].

Murata et al. [11] proposed the Network Information Criterion (NIC), which underlying idea is to estimate the deviance for a data set of size n , compensating for the fact that the weights were chosen to fit the training set:

$$NIC = PR_{Emp} + \frac{2}{n} p_{eff}(\lambda) \quad (8)$$

2 The Noise Derived Information Criterion for Regularized Nonlinear models

In [12] we derived a penalty term for linear models without regularization which was independent of the noise variance and based on the relation between the expected and generalization error of pure noise samples (T_ε). Let be ε a Gaussian zero mean, i.i.d. sequence, which is independent of the input and with variance $\sigma_\varepsilon^2 = 1$, i.e. $\varepsilon \sim N(0,1)$ and recall ξ is the inherent Gaussian noise $N(0, \sigma_\xi^2)$ of the available data. We defined the Noise Derived Information Criterion as

$$NDIC_k = PR_{Emp}(y) * T(p_{true}) \quad (19)$$

$$T(p_{true}) = T_\xi = \frac{\langle PR_{Gen}(\xi) \rangle}{\langle PR_{Emp}(\xi) \rangle} \approx T_c = \frac{\langle PR_{Gen}(\varepsilon) \rangle}{\langle PR_{Emp}(\varepsilon) \rangle}$$

Thus, we are able to compute $T(p_{true})$ analysing the behaviour of candidate models under a noise distribution $\varepsilon \sim N(0,1)$. This is a very important result, because it allows us to determine the penalty term without knowing the true variance of noise. This criterion shows better performance than other criteria reported in the literature in small sample scenarios and similar performance in large sample scenarios, specially in the polynomial cases. However, for the case of nonlinear regularized models, the regularization term is part of the cost function, thus the relation T_ξ for linear models does not hold since the optimal error does not exactly equal the inherent noise [7].

As we have seen, the general idea behind regularized nonlinear model selection criteria based on the estimation of the Prediction Risk, is to express the correction factor as a functional which depends on the effective number of parameters. GPE in addition gives an estimate of the noise variance while NIC just takes into account the number of samples.

Our method expresses the correction factor as a functional which is proportional to the effective number of parameters and it depends on the inherent noise. But we aim at determining this correction factor in such a way that it can be independent of the noise variance. In order to control the complexity of the model in regularized nonlinear systems, the effective number of parameters of the training data must be taken into account. Then, T_ξ for regularized nonlinear models can be estimated such as for $\varepsilon \sim N(0,1)$:

$$T_\xi(p_{eff}(\xi)) = \frac{\langle PR_{Gen}(\xi) \rangle}{\langle PR_{Emp}(\xi) \rangle \langle p_{eff}(\xi) \rangle} \approx T_c(p_{eff}(\varepsilon)) = \frac{\langle PR_{Gen}(\varepsilon) \rangle}{\langle PR_{Emp}(\varepsilon) \rangle \langle p_{eff}(\varepsilon) \rangle} \quad (9)$$

This ratio will give an estimate of the change in the error estimates based on the available sample. Thus, we define the Nonlinear Noise Derived Information Criterion (NNDIC) as:

$$PR_{Gen}(y) \approx NNDIC(y) = PR_{Emp}(y) * T(p_{eff}(y)) \quad (10)$$

where

$$T(p_{\text{eff}}(y)) = p_{\text{eff}}(y) * T_{\xi}(p_{\text{eff}}(\xi)) \approx p_{\text{eff}}(y) * T_{\epsilon}(p_{\text{eff}}(\epsilon)) \quad (11)$$

and $p_{\text{eff}}(y)$ is the effective number of parameters of the set of available data $(x_i, y_i)_{i=1..n}$ and $p_{\text{eff}}(\epsilon)$ is the effective number of parameters when training a noise sample $(x_i, \epsilon_i)_{i=1..n}$ $\epsilon \sim N(0,1)$. Note that this expression reduces to NDIC when regularization is not used since the effective number of parameters is the true number of parameters: $p_{\text{eff}}(y) = p_{\text{eff}}(\xi) = p_{\text{eff}}(\epsilon) = p_{\text{true}}$

If the model is too simple, in terms of the effective number of parameters, it will give a large value for the criterion because the residual training error is large, while a model which is too complex will have also a large value because the complexity term is large. The minimum value represents a trade-off between bias and variance.

In order to illustrate the relation between the training cost and the generalization error in regularized nonlinear models we have chosen an artificial task. Let us consider the problem of fitting a sum of three hyperbolic tangent functions:

$$y = 1.8 * \tanh(3.2 * x + 0.8) + 2.5 * \tanh(2.1 * x + 1.2) + 0.2 * \tanh(0.1 * x - 0.5) + \xi \quad (12)$$

where $x \in [-2, 2]$ and ξ is a Gaussian zero mean, i.i.d. sequence which is independent of the input with variance σ_{ξ}^2 .

Experimentally Q ($Q=1000$) independent training sets of size 100 and Q independent test sets of size 5000 have been generated whereupon the training cost and generalization error are calculated for each realization. The noise standard deviation values σ_{ξ} were taken as a percentage of the standard deviation (σ) of the output values of the true underlying function: [20%* σ , 40%* σ , 60%* σ].

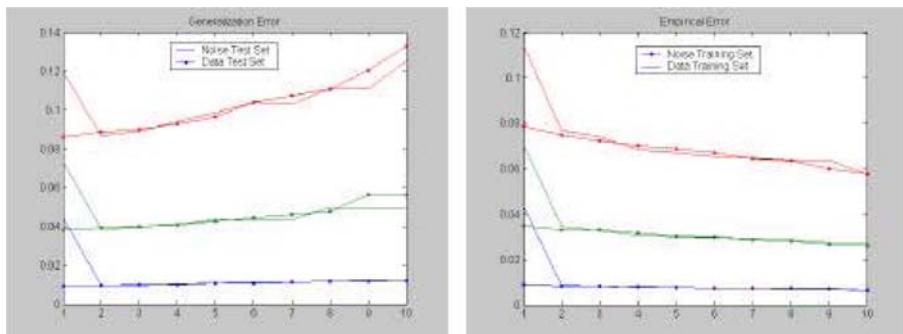


Fig. 1. For the different noise variance values, the generalization error of the data set (x, y) and the noise data set (x, ξ) are computed

Fig. 2. For the different noise variance values, the training error of the data set (x, y) and the noise data set (x, ξ) are computed

The weight decay parameter was fixed to $\lambda = 1e-5$ and the set of MLP candidate models ranged from 1 to 10 hidden units. The following empirical observations were made in order to derive the criterion:

First. We empirically observed the fact that for the case of the quadratic loss with a regularization term, the optimal error does not exactly equal the inherent noise, instead it depends on both the inherent noise and the inputs [7]. Let us suppose that we know the noise in each data point. For a given model k , we consider the fitting of two different data sets, the sample (x, y) and the noise (x, ξ) . Figures 1 and 2 show the relation between the generalization error and empirical error estimates for both, the data and the noise sample ($PR_{Gen}(y)$, $PR_{Emp}(y)$, $PR_{Gen}(\xi)$ and $PR_{Emp}(\xi)$).

Second. Now we study just the noise samples: we empirically observed the relation between the training and the generalization error estimates of the noise samples. Figure 3 shows the median of the noise error estimates for different noise variances. As the complexity of the model grows generalization error estimates tend to increase while the empirical error tend to decrease. We take into account the effective number of parameters because it constitutes a penalty to the complexity of the model. When regularization is used it is different from the true number of parameters and it must be taken into account in the formula of the optimal error. We can state that when the problem is reduced to fitting noise samples the relation between empirical and generalization error using regularization might be expressed as:

$$T_\xi = \frac{\langle PR_{Gen}(\xi) \rangle}{\langle PR_{Emp}(\xi) \rangle \langle p_{eff}(\xi) \rangle} \quad (13)$$

where $\langle \cdot \rangle$ denotes the median of the Q realizations of each measure.

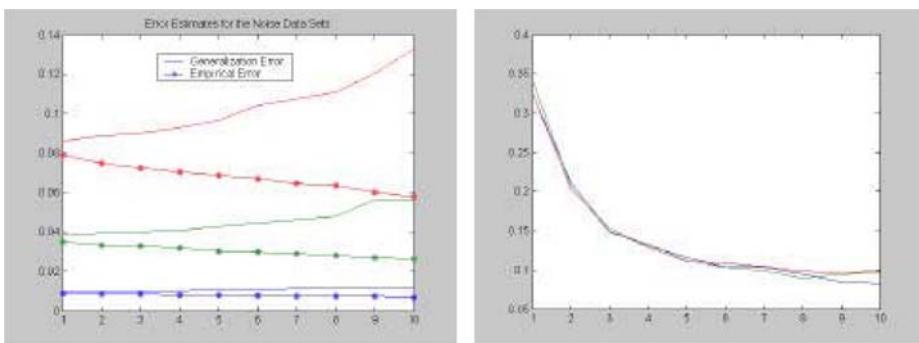


Fig. 3. For different noise variance values, the training and generalization error estimates for each noise data set are computed

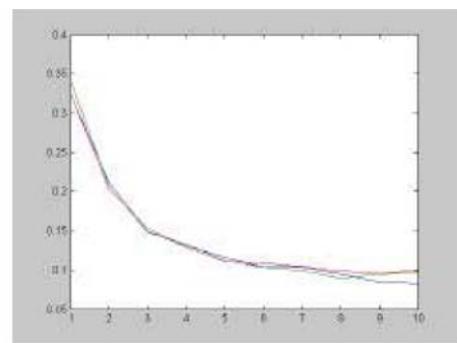


Fig. 4. The correction factor for different noise variance values

Third. We empirically observed (figure 4) that for different noise variance values this relation is independent of the noise variance. This is an important observation because it allows to obtain a correction factor which can be independent of the data sample, since in real situations the noise distribution is unknown and we just have a set of data (x,y) which underlying distribution is also unknown: $T_\xi \approx T_\varepsilon$.

3 Experimental Setting and Simulation Results

The experiments focus on feedforward neural networks with a single layer of units with hyperbolic tangent activation functions, trained by ordinary least-squares using Levenberg-Marquardt algorithm, and $\lambda=0.00001$. The fitting experiment was repeated 1000 times, each time generating a new different data set of a certain size N . The target functions were (12) and the block function from Donoho-Johnstone benchmarks [13], where $x \in [0,1]$ and ξ is a Gaussian zero mean and standard deviation σ_ξ .

Table 1. Results for the block function, $n=30$ and $\sigma_\xi = 70\% * \sigma$

Hidden Units	NNDIC	NIC	GPE	PR
1	10	0	0	14
2	23	9	3	30
3	18	10	1	18
4	14	16	7	17
5	3	15	5	8
6	6	17	5	8
7	1	9	7	0
8	1	8	5	3
9	0	4	9	0
10	2	3	5	1
11	0	5	5	0
12	3	2	10	0
13	4	2	10	0
14	7	0	12	0
15	8	0	16	1

Efficiency 0.601 0.513 0.302 1.0
Ranking 1 2 3

Table 2. Results for the block function, $n=30$ and $\sigma_\xi=20\% * \sigma$

Hidden Units	NNDIC	NIC	GPE	PR
1	0	0	0	4
2	0	11	0	12
3	3	16	0	23
4	12	30	1	18
5	8	17	4	16
6	21	18	5	9
7	5	6	6	5
8	6	1	7	3
9	13	1	6	3
10	7	0	11	1
11	4	0	10	0
12	4	0	10	2
13	8	0	16	2
14	6	0	6	2
15	3	0	18	0

Efficiency 0.596 0.637 0.391 1
Ranking 2 1 3

For a given training sample and different noise standard deviations σ_ξ we computed the series of best fit models M_1, \dots, M_k , with number of hidden units $H=1, \dots, k$, respectively. For a network with H hidden units, the weights for the previously trained network were used to initialize $H-1$ of the hidden units, while the weights for the H^{th} hidden unit were generated from a pseudorandom normal distribution. We used a Monte Carlo approach to calculate T_e , using artificially generated noise samples from a Gaussian distribution $N(0,1)$.

As a measure of performance we used the observed efficiency, which is defined as the ratio that compares the prediction risk estimate between the closest candidate model ($\text{PR}(M_e)$) to the true model in terms of the expected prediction risk, and the model selected by some criterion ($\text{PR}(M_k)$) ($\text{PR}(M_e) / \text{PR}(M_k)$). For each of the 1000 realizations each criterion selected a model and the observed efficiency of this selection was recorded, where higher observed efficiency denotes better performance.

Tables from 1 to 4 show the results for different data sets and noise standard deviation values. The first column shows the number of hidden units, the next ones the

percentage of times each criterion selected a model, and the last column the prediction risk estimate (PR). The last two rows show the mean observed efficiency and the ranking for each criterion in terms of the observed efficiency.

Table 1 shows the results for the block function with $n=30$ observations and standard deviation $\sigma_\xi = 70\% * \sigma$. In this case, NNDIC has the first position in the observed efficiency ranking. NNDIC favors more underfitted models than NIC or GPE. GPE favors overfitted models. Table 2 shows the results for $\sigma_\xi = 20\% * \sigma$, in this case for a lower standard deviation value, NNDIC slightly tends to more overfitted models, but always less than GPE.

Table 3. Results for the hyperbolic tangent function, for $n=30$ $\sigma_\xi = 10\% * \sigma$

Hidden Units	NNDIC	NIC	GPE	PR
1	4	100	1	2
2	57	0	18	54
3	27	0	10	25
4	6	0	12	10
5	2	0	10	2
6	2	0	9	2
7	2	0	8	2
8	0	0	9	3
9	0	0	4	0
10	0	0	19	0
<i>Efficiency</i>	0.7208	0.1658	0.4133	
<i>Ranking</i>	1	2	3	

Table 4. Results for the hyperbolic tangent function, for $n=100$ and $\sigma_\xi = 70\% * \sigma$

Hidden Units	NNDIC	NIC	GPE	PR
1	1	88	0	1
2	35	12	53	84
3	33	0	16	10
4	7	0	9	4
5	12	0	12	1
6	8	0	6	0
7	1	0	1	0
8	1	0	1	0
9	2	0	1	0
10	0	0	1	0
<i>Efficiency</i>	0.9343	0.8256	0.9344	
<i>Ranking</i>	2	3	1	

Table 3 and 4 show the results for the experimental function (12). In contrast to previous results, in this case for 30 samples and a low value of the standard deviation the observed efficiency of NIC is better in NNDIC than NIC. NIC tend to select always a model with 1 hidden unit, while GPE tends to more overfitted models. While for 100 observations even, when a high degree of noise is present all criteria give good results.

From the experiments we can observe that in all the cases our criterion gives reliable results, and depending on the particular problem our method can outperform GPE or NIC. When N is large, all methods give reasonable efficiency results, although our criterion tends to be more efficient as the noise variance grows. In higher nonlinear problems NIC could outperform NNDIC and GPE when the noise variance is not too large. For small sample sizes and higher nonlinear problems GPE tends to more overfitted models, while our criterion tends to more underfitted models.

4 Conclusions and Future Work

We have derived a new penalty term for model selection in nonlinear regularized models. The derivation of this term is obtained by considering that the main

contribution to the generalization error estimation in overfitted models is due to the noise data points. This expression indicates the ratio of change in function of the generalization error estimate, the empirical error estimate and the effective number of parameters. The main advantage lies on the independency of the noise variance and the reliability of the model selection results in small samples scenarios and at different noise variance values in comparison to other well known criteria such as GPE and NIC. A complete theoretical derivation is being prepared, which is based on a Taylor series expansion of the cost function. As all the algebraic estimates of prediction risk, NNDIC is very attractive from the computational perspective since the computation of the correction factor is independent of the noise variance.

References

1. Akaike, H. Statistical predictor identification. *Ann. Inst. Statist. Math.* 22 (1970) 203-217
2. Akaike, H. Information theory and an extension of the maximum likelihood principle. In B.N. Petrov and Csaki ed. 2nd Intl. Symp. Inform. Theory (1973) 267-281 , Budapest.
3. Barron, A. Predicted squared error: a criterion for automatic model selection. *Self-Organizing Methods in Modeling*, S. Farlow, ed., Marcel Dekker, New York (1984)
4. Cherkassky V., Mulier F.: Learning from data: concepts, theory and methods. New York, Wiley (1998)
5. Craven P., Wahba G. Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross-validation. *Numer. Math.*, 31:377-403 (1979)
6. Efron B. & Tibshirani R. J., An Introduction to the Bootstrap. London, U.K.: Chapman & Hall (1993)
7. Larsen J., Hansen L.K.: Generalization performance of regularized neural network models. Proc. IEEE Workshop: Neural Networks for Signal Processing IV, Piscataway, New Jersey (1994) 42-51
8. Mallows, C. L. Some comments on CP . *Technometrics* 15 (1973):661--675
9. Moody, J.: The effective number of parameters: an analysis of generalization and regularization in nonlinear learning systems. *NIPS* (1992) 847-854
10. Moody, J.: Prediction Risk and Architecture Selection for Neural Networks. In Cherkassky, V., et al. editors, *From Statistics to Neural Networks: Theory and Pattern Recognition Applications*, NATO ASI Series F. Springer-Verlag (1994)
11. Murata N., Yoshizawa S., Amari S.: Network Information Criterion – Determining the Number of Hidden Units for an Artificial Neural Network Model. *IEEE Transactions on Neural Networks* (1994) 5, 865-872
12. Pizarro J., et al. : Noise derived information criterion for model selection. *Proceedings of European Symposium on Artificial Neural Networks, ESANN* (2002) pp. 155-160
13. Sarle W.: Donojo-Jonhstone benchmarks: neural nets results. <ftp://ftp.sas.com/pub/neural/dojo/dojo.html> (1999)
14. Stone, M.. Cross-validation: a review. *Math. Operations for Sch. Statist.*, ser. Statistic, (1978) 9, 1.
15. Vapnik, V. *Statistical Learning Theory* John Wiley , New York (1995)
16. Zapranis A., Refenes A.: *Principles of Neural Model Identification, Selection and Adequacy: with applications to financial economics. (Perspectives in neural computing)*. Springer-Verlag London (1999)

Data Driven Multiple Neural Network Models Generator Based on a Tree-like Scheduler

Kurosh Madani, Abdennasser Chebira, Mariusz Rybnik

Intelligence in Instrumentation and Systems Laboratory (I²S)
Senart Institute of Technology, University PARIS XII
Av. Pierre Point, F-77127 Lieusaint, France
{madani, chebira, rybnik}@univ-paris12.fr
<http://www.univ-paris12.fr/>

Abstract. In large number of real world dilemmas and applications, especially in industrial areas, efficient processing of the data is a chief condition to solve problems. The constraints relative to the nature of data to be processed, difficult dilemma related to the choice of appropriated processing techniques and allied parameters make complexity reduction a key point on both data and processing levels. In this paper we present an ANN based data driven treelike Multiple Model generator, that we called T-DTS (Treelike Divide To Simplify), able to reduce complexity on both data and processing levels. The efficiency of such approach has been analyzed trough applications dealing with none-linear process identification. Experimental results validating our approach are reported and discussed.

1 Introduction

In a very large number of cases dealing with real world dilemmas and applications (system identification, industrial processes and manufacturing regulation and optimization, decision, pattern recognition, systems and plants safety, etc), information is available as data stored in files (databases etc.). So, the efficient data processing becomes a chief condition to solve problems related to above-mentioned areas. In the most of those cases, processing efficiency is closely related to several issues among which are:

- Data nature: including data complexity, data quality and data representiveness.
- Processing technique related issues: including model choice, processing complexity and intrinsic processing delay.

Data complexity, related to nonlinearity may affect the processing efficiency. Quality (noisy data, etc.) may influence processing success and expected results quality. Representativeness concerning scarcity of pertinent data could affect processing achievement. On the other hand, choice or availability of appropriated model describing the behavior related to the processed data is of major importance. Processing technique or algorithm complexity (designing, precision, etc) shapes the processing effec-

tiveness. Intrinsic processing delay or processing time related to the processing technique's implementation (software or hardware related issues).

One of the key points on which one can act is the complexity reduction. It concerns not only the problem solution level but also appears at processing procedure level. An issue could be model complexity reduction by splitting of a complex problem into a set of simpler problems: multi-modeling where a set of simple models is used to sculpt a complex behavior [1]. Another promising approach to reduce complexity takes advantage from hybridization [2].

Several ANN based approaches were suggested allowing complexity and computing time reduction. Among proposed approaches, one can note the Intelligent Hybrids Systems [2], Neural Network Ensemble concept [3], Models or experts mixture ([4], [5]), Dynamic Cell Structure architecture [6] and active learning approaches [7].

In this paper we present an ANN based data driven treelike Multiple Model generator, that we called T-DTS (Treelike Divide To Simplify), able to reduce complexity on both data and processing chain levels. The main idea of the T-DTS is based on the notion "Divide et impera"¹ (Julius Caesar), transformed here as "Divide To Simplify" (DTS) [8]. The purpose is based on the use a set of small and specialized mapping neural networks, called Neural Network based Models (NNM), supervised by a Scheduling Unit (SU). Scheduling Unit could be a prototype based neural network, Markovian decision process, etc.. Leafs of the obtained tree are Artificial Neural Networks (models). At the node's level, the input space is decomposed into a set of sub-spaces of smaller sizes. While, at the leaf's level the aim is to learn the relations between inputs and outputs relatives to one of sub-spaces, obtained from splitting.

The paper is organized in following way. We present first the T-DTS structure. Different parameters related to this structure will be presented and discussed. Then, we will analyze the efficiency of such approach trough the two-spiral classification dedicated benchmark. Experimental results will be given validating our approach.

2 T-DTS Based Multiple Neural Network Structure

Before presenting the T-DTS based multiple neural network structure and associated algorithm, let us introduce some terms and abbreviations that we will use: *Pattern* denotes the smallest inseparable amount of data, usually vector; *Prototype* denotes the generic pattern's group (subset) representative having the same properties as patterns belonging to that subset; *SU* (Scheduling Unit) will mean NN based decision or possibly another technique used to decide **if** and **how** to divide subset; *NNM* Neural Network based Model) designates NN based model used to perform subset processing; *D* for Data; *PD* for Pre-processed Data; *P* for Prototypes.

T-DTS (Treelike Divide To Simplify) is a data driven neural networks based Multiple Processing (multiple model) structure that is able to reduce complexity on both data and processing chain levels. T-DTS and associated algorithm construct a treelike

¹ "divide and rule".

evolutionary neural architecture automatically, where nodes (SU) are decision units and leafs correspond to neural based processing units (NNM).

The T-DTS includes two main operation modes. The first is the learning phase, when T-DTS system decomposes the input data and provides processing substructures and tools for decomposed sets of data. The second phase is the operation phase (usage the system to process unlearned data). There could be also a pre-processing phase at the beginning, which arranges (prepare) data to be processed. Pre-processing phase could include several steps (conventional or neural stages). Figure 1 gives the general bloc diagram of T-DTS concept. Pre-processing is expected to ease the processing of data. During pre-processing several operations such as data normalizing, data scaling, data dimensionality reduction could be performed. Pre-processing could also include other kind of operations, as removing outliers or Principal Component Analysis ([9]) to enhance input data quality or eliminate redundancy in data.

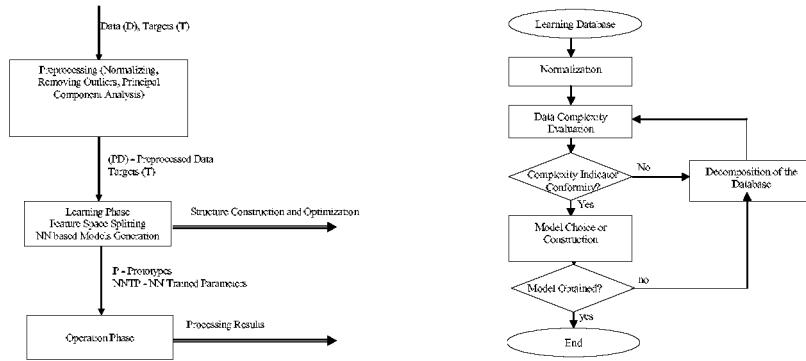


Fig. 1. General bloc diagram of DTS, presenting main operation levels (left). Bloc diagram of DTS decomposition and models generation algorithm (right).

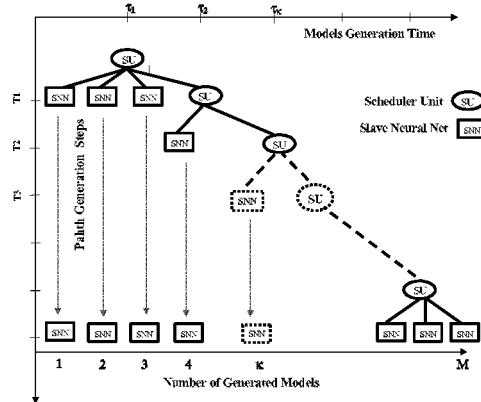


Fig. 2. General bloc diagram of T-DTS splitting and NNM learning process.

The learning phase is an important phase during which T-DTS performs several key operations: splitting the learning database into several sub-databases, constructing

(dynamically) a treelike Supervision/Scheduling Unit (SSU) and building a set of sub-models (NNM) corresponding to each sub-database.

Figure 2 represents the division and NNM construction process bloc diagrams. As this figure shows, after the learning phase, a set of neural network based models (trained from sub-databases) are available and cover (model) the behavior region-by-region in the problem's feature space. In this way, a complex problem is decomposed recursively into a set of simpler sub-problems: the initial feature space is divided into M sub-spaces. For each subspace k , T-DTS constructs a neural based model describing the relations between inputs and outputs. If a neural based model cannot be built for an obtained sub-database, then, a new decomposition will be performed on the concerned sub-space, dividing it into several other sub-spaces.

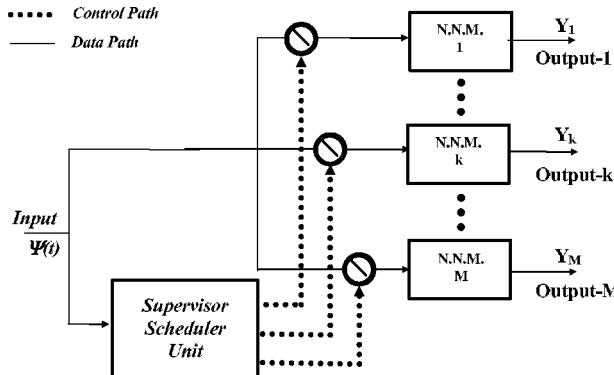


Fig. 3. Bloc diagram of T-DTS Operation phase.

The second operation mode corresponds to the use of the constructed neural based Multi-model system for processing unlearned (work) data. The Operation Phase is depicted by figure 3. The Supervisor/Scheduler Unit (SSU), constructed during the learning phase, receives data (unlearned input vector) and classifies that data (pattern) as corresponding to one of the processing subset. Then, the most appropriated neural processing unit (NNM) is authorized (activated) to process that pattern.

Let $\Psi(t)$ be the input ($\Psi(t) \in \mathbb{R}^{n_\Psi}$), a n_Ψ -Dimensional vector and $Y_k(t) \in \mathbb{R}^{n_Y}$ be the k -th ($k \in \{1, \dots, M\}$) model's output vector of dimension n_Y . Let $F_k(\cdot): \mathbb{R}^{n_\Psi} \rightarrow \mathbb{R}^{n_Y}$ be the k -th NNM's transfer function. Let $S(\Psi(t), p, \xi) \in B^M$, where $B = \{0,1\}$, be the Supervisor/Scheduler Unit's (SSU) output, called also Scheduling Function, which depends on $\Psi(t)$, but which may also depend on some parameters p and/or conditions ξ . p_k represents some particular values of parameter p and ξ_k denotes some particular value of condition ξ , respectively, obtained from learning phase process for the k -th sub-dataset.

$$S(\Psi(t), p, \xi) = (s_1 \ \dots \ s_k \ \dots \ s_M)^T \text{ with } \begin{cases} s_k = 1 & \text{if } p = p_k \text{ and } \xi = \xi_k \\ s_k = 0 & \text{else} \end{cases} \quad (1)$$

The scheduling vector $S(\Psi, p_k, C_k)$ will activate (select) the k -th NNM, and so the processing of an unlearned input data conform to parameter p_k and condition C_k will be given by the output of the selected NNM:

$$Y(\Psi, t) = Y_k(t) = F_k(\Psi(t)) \quad (2)$$

The splitting (during the learning phase) could lead to two general cases. The first one corresponds to the situation where the splitting process doesn't modify the feature space dimension. That means that the initial problem's decomposition into M sub-problems divides the initial complex model into M easier models describing behaviour in each related sub-problem. The second case, correspond to the situation where the splitting process divided the initial feature space into M feature spaces with smaller dimensions (that doesn't mean that the obtained feature sub-spaces will be orthogonal). So, in this case, the activation of appropriated NNM will not depend to the complete input vector but to some partial input vector $\Phi(t) \in \mathcal{R}^{n_p}$, with $\Phi(t) \in \mathcal{R}^{n_p} \subseteq \mathcal{R}^{n_f}$.

2.1 Case of a Probabilistic Rule based SSU

In this case, the activation of an appropriate NNM is given in term of probability of activation the k -th NMM among M neural network based models. If the k -th NNM has been obtained with respect to the k -th learning sub-database, then the probability of activation of the corresponding NNM, $P_k(\Psi(t))$, could be expressed by relation (3), where $\rho_k(\Psi)$ is some Gaussian approximation of k -th learning sub-database density.

$$P_k(\Psi) = \frac{\rho_k(\Psi)}{\sum_{k=1}^M \rho_k(\Psi)} \quad \text{with} \quad \rho_k(\Psi) = \text{Exp}\left[-\frac{(\Psi - \mu_k)^T (\Psi - \mu_k)}{\sigma_k^2}\right] \quad (3)$$

where μ_k represents the average prototype's center and σ_k denoting the learned prototypes standard deviation. Then the Scheduling vector (SSU output) will be expressed according to relation (1), where $s_k(\Psi, P_k)$ is given by (4) with $l \in \{1, \dots, M\}$.

$$s_k(\Psi, P_k) = \begin{cases} 1 & \text{if } P_k = \text{Max}(P_l) \\ 0 & \text{Else} \end{cases} \quad (4)$$

2.2 Case of a Similarity Matching based SSU

2.2.1 Kohonen Self Organizing Map (SOM) based SSU

In this case, splitting process dividing the initial complex problem into M reduced sub-problem takes advantage from Kohonen SOM ([10]) properties. As the splitting has been performed on the basis of *Similarity Matching*, the activation of an appropriate NNM will be issued from similarity measure between an unlearned input vector $\Psi(t)$ and the k -th SOM cluster representative (W_k). As previously, it is supposed that the initial feature space has been decomposed to M clusters by a Kohonen like SOM

process. The Scheduling vector (SSU output) will be conform to relation (1), with $s_k(\Psi, W_k)$ given by (5).

$$s_k(\Psi, W_k) = \begin{cases} 1 & \text{if } |\Psi(l) - W_k| = \min_M |\Psi(l) - W_k| \\ 0 & \text{Else} \end{cases} \quad (5)$$

2.2.2 Global Similarity based SUU

The global similarity is defined in term of some empirically determined resemblance threshold. The main parameter is MaxStd, which defines the standard deviation maximum value (in each dimension) in a given subset. This parameter should not exceed a pre-defined threshold to avoid dividing. In other words, during the splitting phase, if the data of a given sub-database is homogenous enough then it is not necessary to divide more.

The splitting process starts by evaluating the average and standard deviation of the learning database. If the obtained standard deviation is greater than the MaxStd, then, a two-clusters Kohonen SOM (or a distance based competitive NN) divides the learning database into two sub-databases. These operations are repeated until the standard deviation relative to each created sub-database doesn't exceed the MaxStd value.

If Ψ_j^k represents the j-th learning prototype of the k-th learning sub-database, if $\bar{\Psi}_k$ denotes average representative prototype of this sub-database and σ_k its standard deviation (obtained from relations (6)), then, the Scheduling vector (SSU output) components $s_k(\Psi, \bar{\Psi}_k, \sigma_k)$ could be expressed as presented by relation (7), where n_k is the number of prototypes in k-th sub-database.

$$\bar{\Psi}_k = \frac{1}{n_k} \sum_{j=1}^{n_k} \Psi_j^k \quad \text{and} \quad \sigma_k = \frac{1}{n_k} \sum_{j=1}^{n_k} (\Psi_j^k - \bar{\Psi}_k)^2 \quad (6)$$

$$s_k(\Psi, \bar{\Psi}_k, \sigma_k) = \begin{cases} 1 & \text{if } \|\Psi - \bar{\Psi}_k\| \leq \sigma_k \text{ with } \forall \sigma_k, \sigma_k \leq \text{MaxStd} \\ 0 & \text{Else} \end{cases} \quad (7)$$

3 Tree-DTS Implementation Example and Validation Results

In order to study performances of T-DTS, an academic classification problem, called the two-spiral problem [11], which is used as benchmark, has been considered. This benchmark problem, used for performances comparison, especially in classification problems, is equivalent to a generalized Exclusive Or. Construction of neural tree to treat the problem consists of decomposing the input space into a set of subspaces, then performing classification in each subspace by a specialized neural unit (at leaf's level). The database used evaluation and validation includes 1000 patterns. The testing protocol has been defined as following: two databases, including the same number of patterns (500 patterns each), are generated from initial database.

The T-DTS structure on which the validation has been performed includes two kinds of Supervisor/Scheduler Unit's (SSU): Competitive Network (CN) and Self Organization Map (SOM) [10]. Different possibilities, concerning number of neurones (for CN) and network's topology (different topologies for SOM as: 2x2, 3x2, 3x3, 4x4 or 5x5), have been implemented. In the case where Kohonen maps have a grid 2x1 topology, T-DTS builds a binary decision tree. The implemented splitting criterion corresponds to the 2.2.2 similarity matching based on MaxStd, which defines the standard deviation maximum value (in each dimension) in a given subset. Concerning Neural Networks based Models (processing units) several possibilities have been implemented: LVQ (Learning Vector Quantization, [12]), LN (Linear Neurone), RBF (Radial Basis Functions, [13]) and MLP (Multi-Layers Perceptron, [14]).

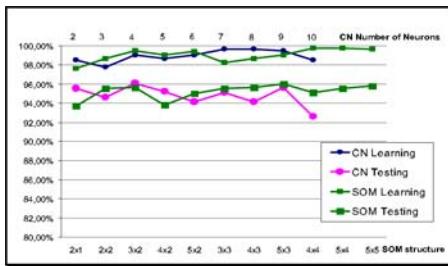


Fig. 4. “Classification Rate” as a function of number of neurons (for competitive SU) and as a function of topology (for Kohonen SU).

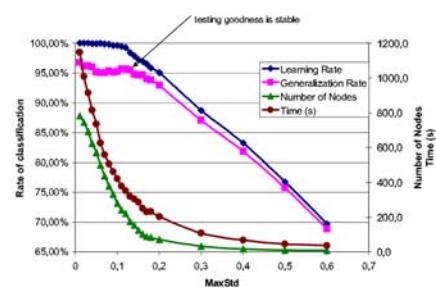


Fig. 5. Learning and generalization evolution rates according to the MaxStd value when LVQ-like SNN are used in the tree leafs level.

Figure 4 gives a comparative study, expressed as “Correct Classification Rate” as a function of number of neurons (for competitive SU) and as a function of considered topology (for Kohonen SU). Figure 5 gives learning and generalization performances, expressed in term of “Correct Classification Rate”, according to the maximum standard deviation threshold value’s evolution. Tables 1 compares the processing time for the case of competitive Supervisor/Scheduler based T-DTS and for different SNN structures. For the presented case, MAXSTD=0.12 leading to 107 sub-databases.

4 Conclusion

By dividing the initial database into several sub-databases and by a dedicated processing of each of those data subsets, The proposed ANN based data driven Multiple Model generator (T-DTS) reduces the initial problem’s complexity at several levels, especially at processing and modeling ones. De facto, dividing the initial problem into several sub-problems with reduced sizes, on the one hand, simplifies also the learning complexity and duration (learning of relations between inputs and outputs), and on the other hand reduces the processing procedure’s or unit’s complexity. Finally, it decreases globally implementation and parameters optimization constraints.

Table 1. Processing time for both SSU construction and NNM learning and generalization in the case of a Competitive-like NN based splitting (MaxStd=0.12 leading to 107 sub-databases).

Splitting (107 nodes)	SNN Type	Learning Database	Testing Database
159.88 s	LVQ	261.51 s	6.33 s
159.88 s	LN	4.77 s	4.84 s
159.88 s	RBF	37.67 s	5.84 s
159.88 s	MLP	110.91 s	8.46 s

An implementation example has been reported, implementing a standard deviation based splitting criterion. Very promising results, obtained for the two-spiral benchmark, show efficiency of such multiple model structure to enhance processing capability by reducing complexity on both processing and data levels. On the other hand, the neural character of the proposed structure makes it adaptable for different kinds of applications, offering solution to a wide range of complex processing problems. We are currently working on the splitting criterion based on advanced complexity measurement techniques.

References

1. Multiple Model Approaches to Modeling and Control, edited by R. Murray-Smith and T.A. Johansen, Taylor & Francis Publishers, 1997, ISBN 0-7484-0595-X.
2. Goonatilake S. and Khebbal S.: Issues, Classification and Future Directions. In Intelligent Hybrid Systems. John Wiley & Sons, pp 1-20, ISBN 0 471 94242 1.
3. Krogh A., Vedelsby J.: Neural Network Ensembles, Cross Validation, and Active Learning, in Advances in Neural Information Processing Systems 7, The MIT Press, Ed by G. Tesauro, pp 231-238, 1995.
4. Sridhar D.V., Bartlett E.B., Seagrave R.C., "An information theoretic approach for combining neural network process models", Neural Networks, Vol. 12, pp 915-926, Pergamon, Elsevier, 1999.
5. Jordan M. I. and Xu L., "Convergence Results for the EM Approach to Mixture of Experts Architectures", Neural Networks, Vol. 8, N° 9, pp 1409-1431, Pergamon, Elsevier, 1995.
6. Bruske J., Sommer G., Dynamic Cell Structure, Advances, in Neural Information Processing Systems 7, The MIT Press, Ed by G. Tesauro, pp 497-504, 1995.
7. Sang K. K. and Niyogi P., Active learning for function approximation, in Neural Information Processing Systems 7, The MIT Press, Ed by G. Tesauro, pp 497-504.
8. Madani K., Chebira A., "A Data Analysis Approach Based on a Neural Networks Data Sets Decomposition and it's Hardware Implementation", PKDD 2000, Lyon, France, 2000.
9. Jolliffe I.T., "Principle Component Analysis", New York, Springer Verlag 1986.
10. Kohonen T., "Self-Organization and Associative Memory", Springer-Verlag, 1984.
11. Lang K. J. and Witbrock M. J., Learning to tell two spirals apart. Proc. of the 1988 Connectionist Models Summer School, Morgan Kauffman, pp 52-59, (1988).
12. Desmartines P., Hérault J., "Representation of Non Linear Data Structures Trought a Fast VQP Neural Networks", Neuro-Nîmes'93 Proceedings, pp 411-424, Nîmes, France, 1993.
13. Reyneri L., "Weighted Radial Basis Function for Improved Pattern Recognition and Signal Processing", from Neural Processing Letters, Vol. 2, N°3, pp 2-6, 1995.
14. Rumelhart D., Hinton G., Williams R., "Learning Internal Representations by Error Propagation", "Parallel Distributed Processing ", I & II, MIT Press, Cambridge MA, 1986.

Performance-Enhancing Bifurcations in a Self-organising Neural Network

Terence Kwok and Kate A. Smith

School of Business Systems, Faculty of Information Technology, Monash University,
Clayton, Victoria 3168, Australia
{terence.kwok, kate.smith}@infotech.monash.edu.au

Abstract. The self-organising neural network with weight normalisation (SONN-WN) for solving combinatorial optimisation problems (COPs) is investigated in terms of its performance and dynamical characteristics. A simplified computational model of the weight normalisation process is constructed, which reveals symmetry-breaking bifurcations in a typical node outside the winning neighbourhood. Experimental results with the N -queen problem show that bifurcations can enhance solution qualities in a consistent manner. A mechanism based on the weights' transient trajectories is proposed to account for the neural network's capacity to escape local minima.

1 Introduction

Since the introduction of the self-organising feature map (SOFM) by Kohonen in the early eighties [1], a class of neural networks using the self-organising approach have been developed to solve NP-hard combinatorial optimisation problems (COPs) [2]. The travelling salesman problem (TSP) was often used as a test problem in the early attempts, which include the elastic net method [3] and other Kohonen-type algorithms [4, 5]. However, these models are limited to solving Euclidean problems only. For the broader class of “0-1” optimisation problems, a more general self-organising neural network (SONN) [6, 7] was developed to compute with feasible permutation matrices rather than the Euclidean space. Weight normalisation was later introduced to the SONN as an efficient means for constraint satisfactions, as well as controlling the convergence process [8]. The resulting self-organising neural network with weight normalisation (SONN-WN) employs a temperature parameter T in the normalisation function to control the extent of 0-1 convergence as the weight evolves. A similar use of the normalisation function has been applied to Hopfield-type networks to improve solution quality [9, 10]. Experiments of the SONN-WN with an “annealing” T showed that solution quality has a sensitive dependence on the annealing rate [11]. Since the updating-normalisation dynamics of the SONN-WN are nonlinear and spatio-temporal in nature, a simplified model has been constructed to describe such dynamics of the winning node [12]. The same study also measured experimentally the optimisation performance of the SONN-WN in the key parameter space of T and the Kohonen learning rate β when solving the N -queen problem. The results showed that a distinct parameter regime exists where both solution quality and efficiency are

significantly better than other parameter combinations. This strongly suggests a close coupling of the two parameters in the SONN-WN's convergence towards high quality solutions, and some bifurcation dynamics of the winning nodes have been identified as candidates for interpreting the measured optimisation performance [12]. In addition, a range of other nonlinear phenomena including cascades of bifurcations to chaos can also be observed in two related models of the winning node [12, 13].

It is the aim of this paper to show how bifurcations occurring in the SONN-WN can clearly improve its optimisation performance. While in previous research the focus has been on the dynamics of the winning node, here we look at nodes outside the winning neighbourhood. More importantly, we demonstrate that the self-organising process is operating at its best when the weights are near their bifurcation point. To achieve these, the N -queen problem is used as an example for illustration. Sect. 2 outlines the SONN-WN architecture and algorithm. In Sect. 3, experimental results on the optimisation performance and the model of non-winning nodes are presented. Discussions are concluded in Sect. 4.

2 Architecture and Algorithm of the SONN-WN

The SONN-WN is outlined here as an implementation to solve the N -queen problem. It should be noted that the SONN approach is general and has been used to solve a variety of COPs [6-8]. The N -queen problem is an example of NP-hard constraint satisfaction problems (CSPs). The aim is to place N queens onto an $N \times N$ chessboard without attacking each other. This is enforced by having 1) one queen in each row; 2) one queen in each column; 3) one queen on each diagonal (there are more than two diagonals), and 4) exactly N queens on the chessboard. Mathematically, the state of the chessboard can be represented by an $N \times N$ matrix \mathbf{X} with elements of 1 wherever a queen is placed, and 0 otherwise. This forms a 0-1 optimisation problem minimising the following cost (f) which measures violation of the constraints described above,

$$f = \frac{1}{2} \sum_{i,j} \sum_{l \neq j} x_{ij} x_{il} + \frac{1}{2} \sum_{i,j} \sum_{k \neq i} x_{ij} x_{kj} + \frac{1}{2} \left(\sum_{i,j} x_{ij} - N \right)^2 \quad (1)$$

$$+ \frac{1}{2} \left(\sum_{i,j} \sum_{\substack{p \neq 0 \\ 1 \leq i-p \leq N \\ 1 \leq j-p \leq N}} x_{ij} x_{i-p, j-p} + \sum_{i,j} \sum_{\substack{p \neq 0 \\ 1 \leq i-p \leq N \\ 1 \leq j+p \leq N}} x_{ij} x_{i-p, j+p} \right).$$

To solve the N -queen problem of minimising constraint violation by (1), we use a SONN-WN based on that used in [8]. The architecture of the network for this problem is shown in Fig. 1. The architecture of the SONN is a feed-forward neural network with an input layer of N nodes, and an output layer of N nodes, representing the N columns and N rows of the chessboard respectively. The weight between an input node j and an output node i is given by W_{ij} and represents the continuous relaxation of the decision variable x_{ij} in (1). The training set consists of one input pattern for each column (each queen in this example). The input pattern corresponding to a column j^* is a vector of N components where the j^* -th component is 1, and the remaining

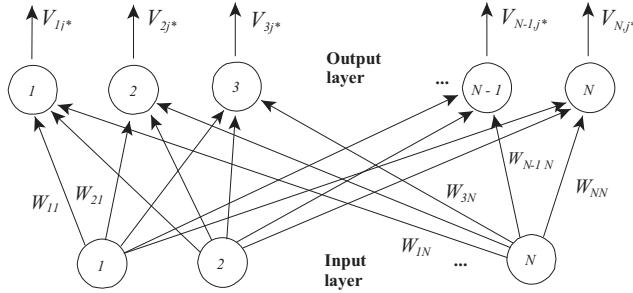


Fig. 1. Architecture of the SONN

components are 0. When the input pattern corresponding to a column j^* is presented, the net input to each node i of the output layer is the potential

$$V_{ij^*} = \sum_{k,l=1}^N (1 - \delta_{ik}) (\delta_{i+j^*, k+l} + \delta_{i-j^*, k-l}) W_{kl} + B \sum_{k=1}^N (1 - \delta_{ik}) W_{kj^*} \quad (2)$$

which is the diagonal and column contribution to the objective function f of placing a queen on column j^* of row i , and δ_{ik} is the Kronecker delta. The cost potential V_{ij^*} represents the partial cost of assigning a queen to position (i, j^*) , and is related to the partial derivative of the objective function (1). B is a penalty parameter which is arbitrarily chosen to be 1 in this paper, and determines the balance between violation of diagonal constraints and column constraints. The row constraints are enforced by the normalisation procedure described below during the weight updates.

The competition between the output nodes gives the winning node i_0 corresponding to the row with minimum potential

$$i_0 = \arg \min_i V_{ij^*} \quad (3)$$

For self-organisation to develop we define a neighbourhood of the winning nodes as those with the least potential

$$N(i_0, j^*) = \{i_0, i_1, i_2, \dots, i_\eta\} \quad (4)$$

where

$$V_{i_0, j^*} \leq V_{i_1, j^*} \leq V_{i_2, j^*} \leq \dots \leq V_{i_\eta, j^*} \leq \dots \leq V_{i_{N-1}, j^*} \quad (5)$$

and $\eta \geq 1$ is the neighbourhood size. Thus the neighbourhood is defined according to relative cost, rather than spatially as in Kohonen's SOFM [1].

Once both the winning node and its neighbourhood have been determined, the weights are modified according to a modification of the Kohonen's weight adaptation rule. In order to explicitly enforce the constraint of one queen in each row, the weight normalisation for each output node as described in [8] is used:

$$W_{ij} \leftarrow \frac{\exp\left(-\frac{(1-W_{ij})}{T}\right)}{\sum_{j=1}^N \exp\left(-\frac{(1-W_{ij})}{T}\right)} \quad (6)$$

where T is a parameter (called temperature), which may be lowered as the learning process proceeds. The normalisation operation guarantees that when convergence is completed, only one queen is assigned to each row. During the learning process, the neighbourhood size, the magnitude of the weight adaptations, and the temperature are gradually decreased. The complete algorithm follows:

1. Randomly initialise the weights around 0.5.
2. Randomly choose a column j^* and present its corresponding input pattern.
3. Compute the potential V_{ij^*} for each output node i according to (2).
4. Determine the winning node, i_0 , as well as its neighbouring nodes according to (3)-(5).
5. Update weights, W_{ij^*} , connecting input node j^* with every output node i :

$$\Delta W_{ij^*}(t) = \alpha(i, t)(1 - W_{ij^*}) \quad \forall i \in N(i_0, j^*) \quad (7)$$

$$\Delta W_{ij^*}(t) = 0 \quad \forall i \notin N(i_0, j^*) \quad (8)$$

where

$$\alpha(i, t) = \beta(t) \exp\left[-\frac{|V_{i_0j^*} - V_{ij^*}|}{|V_{i_0j^*} - V_{i_{N-1}j^*}|}\right] \quad (9)$$

The updated weights are:

$$W_{ij^*} \leftarrow W_{ij^*} + \Delta W_{ij^*} \quad (10)$$

6. Normalise W_{ij} 's to enforce the one-queen-per-row constraint using (6).
7. Repeat from Step 2 until all columns have been selected as input patterns. This is one training epoch. Repeat from Step 2-6 for Ω epochs, then anneal T to encourage 0-1 solution according to the following:

$$T(t + L_1) = rT(t) \quad (11)$$

where $T(t)$ represents the temperature at epoch t , and $0 < r \leq 1$ is the cooling rate. $\beta(t)$ also decays in a similar manner with r_β .

8. Repeat Step 7 until $|W_{ij}^{(t+1)} - W_{ij}^{(t)}| < \varepsilon \quad \forall i, j$, where ε is a small number. This condition signifies the convergence of the weights for a given neighbourhood size η . Reduce η linearly. Halt when $\eta < 0$. W_{ij} 's are then rounded to 0 or 1 to obtain the final solution:

$$W_{ij}^{final} = \begin{cases} 1 & \text{if } W_{ij} = \max_j W_{ij} \\ 0 & \text{otherwise} \end{cases} \quad \forall i \quad (12)$$

3 Experimental Results

First, the N -queen problem with $N=8$ is solved via computer simulation of the SONN-WN, and we measure the performance in terms of feasibility. Feasibility is defined here as the proportion of globally optimal solutions obtained with 100 different initial weight matrices. A global minimum solution satisfies all the N -queen constraints, and a local minimum is a solution with at least one constraint violated. Feasibility here thus measures the optimisation ability of the network. Here, $\Omega = 300$, $\varepsilon = 0.005$ and initial $\eta = 4$ are chosen. If the weights cannot converge within 300 epochs for the current η (Step 8 in Sect. 2), the program proceeds as if the convergence condition is satisfied. This is to avoid exceedingly long running times.

Fig. 2 shows the feasibility in the (β, T) parameter space. Each point represents the feasibility obtained with runs using constant (β, T) values, i.e. $r = r_\beta = 1$. Keeping (β, T) constant for a given run allows us to focus on the basic β and T interactions without the complications caused by parameter annealing. It can be seen that high feasibility (dark area) is favoured for high β with relatively small T . Feasibility is zero for around $T > 0.3$ no matter what β is used. Combinations of (β, T) yielding good performance form a largely horizontal band near $T = 0.2$. This performance band is darker for large β 's and becomes lighter for smaller β 's. The dark band extends to lower T 's when β is large, forming a roughly triangular dark region. This feasibility pattern clearly shows that β and T need to match one another for good results.

Next, we show how the observed feasibility pattern is in fact caused by the nonlinear phenomenon of bifurcation. This is done by constructing a simplified model of the network's weight dynamics. Previously, the typical winning node's updating-normalisation dynamics have been modelled in a similar attempt [12, 13]. Here, we model the weight dynamics of a typical node outside the winning neighbourhood.

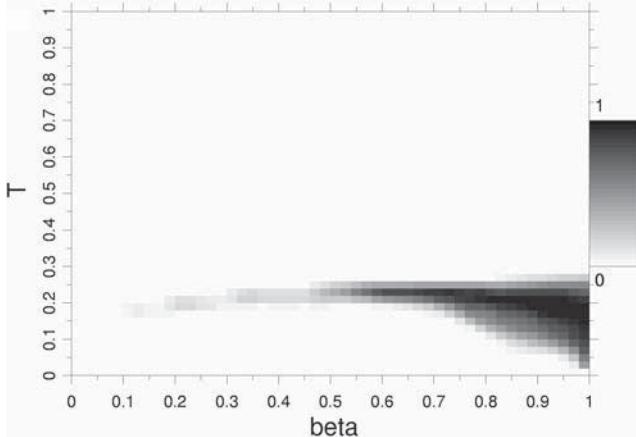


Fig. 2. Measured feasibility for the 8-queen problem. Darker shades represent higher feasibility

Since we are focusing on the weight dynamics within a typical non-winning node, the feature of SOFM-type competition among nodes is deliberately excluded from the model. The following is the computational model presented in pseudo-code format:

[1] initialisation: set $T \in (0, 1]$ and $N = 8$

[2] choose $w_j \in [0, 1]; j = 1, \dots, N$

[3] for (iteration = 1 to 100)

[4] for ($j = 1$ to N)

$$[5] \quad \text{normalisation: } w_j \leftarrow \frac{\exp[-(1 - w_j)/T]}{\sum_{k=1}^N \exp[-(1 - w_k)/T]}$$

The loop at line [3] above is to repeat the normalisation function for many iterations, so that we can examine the asymptotic states of the weights (see Step 6 in Sect. 2). Figure 3 shows the numerical result of the above model after 100 iterations, plotting the last 50 w values for each T . Because of the normalisation, the eight w_j values always add up to 1 for any T . It can be observed that for $T > 0.2$ (approx.), all the weights except w_5 converge to the mean value of 0.125. For $T < 0.2$ (approx.), w_5 converges towards 1 and all other w_j 's towards 0. Around $T = 0.2$, a symmetry-breaking bifurcation can be seen. Let's call the bifurcation temperature T^* . Initially, we set $w_5 = 0.957$, $w_3 = 0.95$, and other w_j 's = $(1 - 0.95) / 7 \approx 0.007$. Since w_5 is larger than all other w_j values, it breaks away from others when $T < T^*$. This particular set of initial w_j values is chosen to simulate the after-effect of an update to w_5 with $\beta = 0.95$ from a previously converged state of $\{w_3 = 0.95, \text{all other } w_j \text{'s } \approx 0.007\}$. In other words, we are testing how the weights respond to the repeated normalisation process *after* a previous updating. Note that the weight updating as in (10) is not included in the model as a process, but as an initial condition only. Furthermore, the value of T^* varies with the initial w_j settings, e.g. the initial condition of $\{w_5 = 0.957, w_3 = 0.51, \text{other } w_j \text{'s } \approx 0.007\}$ corresponds to $T^* = 0.235$ (not shown).

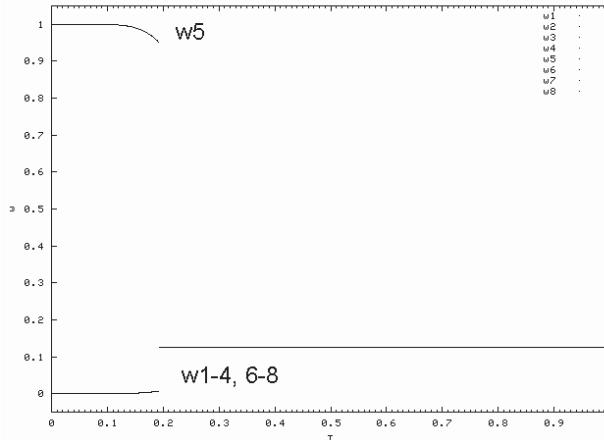


Fig. 3. Asymptotic states of w against T computed from the simplified model of normalisation dynamics for $N = 8$

From the asymptotic states shown in Fig. 3, some features of the feasibility pattern in Fig. 2 can now be explained. The poor feasibility region for $T > 0.3$ in Fig. 2 is a result of the SONN-WN having all the weights converged to the same value of 0.125 as shown Fig. 3. This kind of dynamics is clearly undesirable for the Kohonen-type competition process, as it tends to “overwrite” all previous updating history, rendering all W_{ij} ’s to 0.125, and hence no meaningful solution is obtained. For the high feasibility band in Fig. 2, it can be seen that it centres around the bifurcation temperature of $T^* \approx 0.2$ in Fig. 3. This strongly suggests that the weight dynamics near the bifurcation point as shown in Fig. 3 is particularly favourable for yielding high feasibility. Since in the simplified model T^* varies with different initial w_i settings, the dark band of good feasibility shown in Fig. 2 also covers a range of T depending on β . This reflects well the normal operating condition of the SONN-WN as there is a certain variability to the distribution of weight values among the nodes.

In order to understand why bifurcation dynamics are conducive to good optimisation, we need to look at the weights’ transient phase, i.e. before they reach the converged attractor as shown in Fig. 3. Since on average a node in the SONN-WN wins the competition once every N presentations of the input vector (see Steps 2-4 in Sect. 2), in Fig. 4a-b we plot the first 10 iterations in the transient convergence phase for two different temperatures around T^* . Figure 4a shows the case with T just above T^* . It can be seen that although w_5 is initially larger than w_3 , both are attracted to join the rest of the weights towards the mean value, with w_5 converging slightly slower. With a lower T of just below T^* as shown in Fig. 4b, a very different scenario appears. Initially, the slightly larger value of w_5 follows w_3 towards low values, but breaks apart after the second iteration and converges towards 1. This kind of “cautious” selectivity exhibited by w_5 is observed for a range of temperatures lower than, but close to T^* . Other experimental results (not shown) reveal that the “cautiousness” in w_5 becomes less pronounce for lower T ’s, with the initial drop becoming shallower. If we turn to Fig. 2 again in light of these observations, it is clear that the characteristic bifurcation dynamics is enhancing the feasibility dramatically, as evident from the horizontal dark band around T^* .

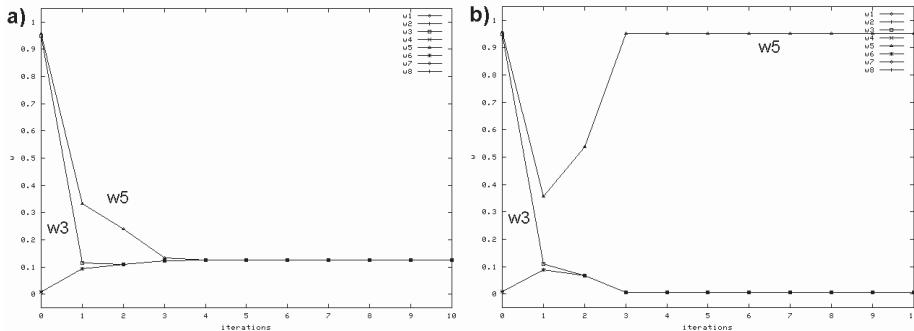


Fig. 4. Transient convergence phase of the weights w for the simplified model of normalisation. The weight values are plotted against the number of iterations, with a) $T = 0.193$ and b) $T = 0.192$. The first 10 iterations of the convergence are plotted with temperatures chosen around the bifurcation point in Fig. 3, using the same initial w values

4 Discussions and Conclusions

In the last section, we have illustrated the convergence properties of the normalisation process by means of a simplified model. The model reveals that the weights of a node that is not currently in the winning neighbourhood can exhibit two different kinds of attracted states: a mean-valued state and a bifurcated state. By matching the feasibility results with the modelled dynamics, it is evident that bifurcated states with $T < T^*$ near the bifurcation point are associated with good quality solutions. One possible underlying mechanism is hereby suggested: a recently updated weight follows a “cautious” attracting orbit before converging to its final state, dipping temporarily in its trajectory. This transient phase provides a time window for other weights to be updated into the high-value orbit near 1 when next time its node wins again. Thus this mechanism allows the self-organising process to revert its partially decided states when the need arises, thereby escaping from local minimum solutions. Future work should investigate the statistical properties of this mechanism towards feasibility.

References

1. Kohonen, T.: Self-organized Formation of Topologically Correct Feature Maps. *Biol. Cybern.*, Vol. 43. (1982) 59-69
2. Smith, K. A.: Neural Networks for Combinatorial Optimisation: A Review of More Than a Decade of Research. *INFORMS Journal on Computing*, Vol. 11, No. 1. (1999) 15-34
3. Durbin, R., Willshaw, D.: An Analogue Approach to the Travelling Salesman Problem Using an Elastic Net Method. *Nature*, Vol. 326. (1987) 689-691
4. Fort, J. C.: Solving a Combinatorial Problem via Self-Organizing Process: An Application of the Kohonen Algorithm to the Travelling Salesman Problem. *Biol. Cybern.*, Vol. 59. (1988) 33-40
5. Favata, F., Walker, R.: A Study of the Application of Kohonen-Type Neural Networks to the Travelling Salesman Problem. *Biol. Cybern.*, Vol. 64. (1991) 463-468
6. Smith, K. A., Palaniswami, M., Krishnamoorthy, M.: A Hybrid Neural Approach to Combinatorial Optimization. *Computers Ops. Res.*, Vol. 23. (1996) 597-610
7. _____: Neural Techniques for Combinatorial Optimization with Applications. *IEEE Trans. Neural Networks*, Vol. 9, No. 6. (1998) 1301-1318
8. Guerrero, F., Lozano, S., Smith, K. A., Canca, D., Kwok, T.: Manufacturing Cell Formation Using a New Self-Organizing Neural Network. *Computers & Industrial Engineering*, Vol. 42. (2002) 377-382
9. Van den Bout, D. E., Miller, T. K.: Improving the Performance of the Hopfield-Tank Neural Network Through Normalization and Annealing. *Biol. Cybern.*, Vol. 62. (1989) 129-139
10. Peterson, C., Söderberg, B.: A New Method for Mapping Optimization Problems onto Neural Networks. *Int. Jnl. Neural Systems*, Vol. 1, No. 1. (1989) 3-22
11. Kwok, T., Smith, K. A.: Improving the Optimisation Performance of a Self-Organising Neural Network with Weight Normalisation. Proc. Int. ICSC Congress on Intelligent Systems & Applications (ISA'2000), Paper no. 1513-285. (2000)
12. _____: Characteristic Updating-Normalisation Dynamics of a Self-Organising Neural Network for Enhanced Combinatorial Optimisation. Proc. 9th Int. Conf. Neural Information Processing, Vol. 3. (2002) 1146-1152
13. _____: Nonlinear System Dynamics in the Normalisation Process of a Self-Organising Neural Network for Combinatorial Optimisation. In: *Lecture Notes in Computer Science*, Vol. 2084. Springer-Verlag, Berlin (2001) 733-740

A Competitive Neural Network based on dipoles

M.A. García-Bernal, J. Muñoz-Pérez, J.A. Gómez-Ruiz and
I. Ladrón de Guevara-López

Dpto. Lenguajes y Ciencias de la Computación
E. T .S. Ingeniería Informática. Universidad de Málaga
Campus de Teatinos s/n
29071-Málaga, Spain.
e-mail : munozp@lcc.uma.es

Abstract. In a competitive neural network, a process unit (node) in the competitive layer is completely described by the vector of weight from the input node to it. Each such weight vector becomes the *centroid* of a cluster of inputs since the principal function of a competitive learning network is discovers cluster of overlapping input. In this paper we propose a competitive neural network where each process unit has a couple of weight vectors (dipoles) that becomes a line segment as representation of a cluster. A weight update is formulated such that the dipole associated with each process unit is as near as possible to all the input samples for which the node is the winner of the competition. This network allows the formation of groups or categories by means of unsupervised learning, where each class or category is identified by a line segment instead of a centroid. The line segment leads to a better representation of a group or class than a centroid that gives us only the position of the cluster. The network has been applied to the formation of groups or categories using the data IRIS, where the unsupervised learning algorithms reach between 12 and 17 incorrect classifications. However, while many partitional clustering algorithms and competitive neural networks are only suitable for detecting hyperspherical-shaped clusters, the proposed network gets only 5 incorrect classifications and is also suitable for detecting hyperspherical-shaped clusters.

1 Introduction

Neural Networks have provided good, if not the best, architectures for many traditional pattern recognition and data algorithms. The individual neurons of the network learn to specialize on sets of similar patterns, and thereby become feature detectors. In particular, each of the output neurons discovers a cluster of inputs by moving its synaptic weight vector to the center of gravity of the discovered cluster. It was introduced by Grossberg ([1] and [2]) and von der Malsburg ([3]) and developed by Amari *et al* ([4] and [5]), Bienstock *et al* ([6]) and Rumelhart *et al* ([7]).

The aim of competitive neural networks is to cluster or categorize the input data and it can be used for data coding and compression through vector quantization. Competitive learning is an appropriate algorithm for VQ of unlabelled data. Ahalt, Krishnamurthy and Chen [8] discussed the application of competitive learning neural networks to VQ and developed a new training algorithm for designing VQ codebooks which yields near-optimal results and can be used to develop adaptive vector quantizers. Yair, Zeger and Gersho [9] have proved convergence properties of the Kohonen algorithm for vector quantization design and obtained conditions on the step-size schedules to guarantee that the centroid and nearest neighbour rules are satisfied. Moreover, they have proposed a deterministic VQ design algorithm, called the soft competition scheme, that updates all the codevectors simultaneously with a step size that is proportional to its probability of winning. Pal, Bezdek and Tsao [10] proposed a generalization of learning vector quantization for clustering which avoids the necessity of defining an update neighbourhood scheme and the final centroids do not seem sensitive to initialization. Xu, Krzyzak and Oja [11] developed a new algorithm called rival penalized competitive learning which for each input not only the winner unit is modified to adapt itself to the input, but also its rival delearns with a smaller learning rate. Ueda and Nakano [12] presented a new competitive learning algorithm with a selection mechanism based on the equidistortion principle for designing optimal vector quantizers. The selection mechanism enables the system to escape from local minima. Uchiyama and Arbib [13] showed the relationship between clustering and vector quantization and presented a competitive learning algorithm which generates units where the density of input vectors is high and showed its efficiency as a tool for clustering colour space in colour image segmentation based on the least sum of squares criterion. Mao and Jain [14] have proposed a self-organizing network for hyperellipsoidal clustering which is applied to texture segmentation problems. Hwang *et al* [15] have presented an entropy-constrained competitive learning algorithm that can achieve a near-optimal performance subject to the average rate constraint and it is effective tool for designing a variable-rate VQ. Zhang and Liu [16] have developed a competitive learning algorithm that is able to find the natural number of clusters based the one-prototype-take-one-cluster paradigm and a self-splitting validity measure.

However, many partitional clustering algorithms and competitive neural networks for unsupervised learning are only suitable for detecting hyperspherical-shaped clusters, since Euclidean distance is most commonly used to compute the distance between a pattern and the assigned cluster center. Many competitive learning algorithms, like the K-means algorithm, have the undesirable property of splitting big and elongated cluster (hyperellipsoidal cluster). In this paper, we propose a competitive learning algorithm based on dipoles instead of centrodes that is suitable for detecting hyperspherical or hyperellipsoidal-shaped clusters. In section 2, we present the computational dynamics and the learning rule of the proposed networks. In section 3, the network is applied to clustering problem with hyperellipsoidal-shaped clusters (IRIS data). Finally, we present the conclusions in section 4.

2 A Dipolar Competitive Neural Networks

A competitive neuronal network consists of a single layer of M process units (neurons) fully connected to a same input $\mathbf{x} \in \mathbb{R}^N$ and producing an outputs $y_j \in \{0,1\}$, $j=1,2,\dots,M$. We say that process unit j is active if $y_j=1$. For each input (stimulus) there is only one active unit. This active unit is called the *winner* and is determined as the unit with largest *activation potential*. The activation potential of the process unit j is the inner products $\mathbf{w}_j^T \mathbf{x}$, where \mathbf{w}_j is the synaptic weight vector of process unit j and $\|\mathbf{w}_j\|=1$. Thus the winner is that process unit with the weight vector closest (in a Euclidean distance sense) to the input vector \mathbf{x} . That is, the best match of the input vector \mathbf{x} with the synaptic weight vectors. The synaptic weight vector of the winning process unit is updated according to the competitive learning rule,

$$\Delta \mathbf{w}_j = \eta (\mathbf{x} - \mathbf{w}_j). \quad (1)$$

In this way, the synaptic weight vectors will become the *centroids* of the M clusters or categories that are formed by the networks. However, a centroid is a simple representation of a cluster and does not give us information about shape of clusters. So, we consider now that each connecting link is characterized by a couple of weight vectors. The aim of such network is to discover clusters of inputs by moving its synaptic weight couple to line segment (dipole) that represents the discovered cluster. Note that the weight couples are viewed as one-dimensional objects.

Let $(\mathbf{w}_{il}, \mathbf{w}_{i2})$ denote the synaptic weight couple connecting input to neuron i . The activation potential h_i of neuron i is given by

$$h_i = \alpha_i(\mathbf{x})h_{il} + \bar{\alpha}_i(\mathbf{x})h_{i2} - \frac{1}{2}\alpha_i(\mathbf{x})\bar{\alpha}_i(\mathbf{x})\|\mathbf{w}_{il} - \mathbf{w}_{i2}\|^2, \quad i = 1, 2, \dots, M \quad (2)$$

where

$$h_{il} = \mathbf{w}_{il}^T \mathbf{x} - \frac{1}{2}\mathbf{w}_{il}^T \mathbf{w}_{il} \quad i = 1, 2, \dots, M \quad (3)$$

$$h_{i2} = \mathbf{w}_{i2}^T \mathbf{x} - \frac{1}{2}\mathbf{w}_{i2}^T \mathbf{w}_{i2} \quad i = 1, 2, \dots, M \quad (4)$$

$$\alpha_i(\mathbf{x}) = \frac{(\mathbf{w}_{il} - \mathbf{w}_{i2})^T (\mathbf{x} - \mathbf{w}_{i2})}{\|\mathbf{w}_{il} - \mathbf{w}_{i2}\|^2} \quad (5)$$

$$\alpha_i(\mathbf{x}) + \bar{\alpha}_i(\mathbf{x}) = 1 \quad (6)$$

Note that h_{i1} , h_{i2} and $\alpha_i(\mathbf{x})$ are lineal functions in \mathbf{x} .

The action potential has been defined in this way because the winner unit is the one whose weight vector couple (line segment) is nearest to the input vector. The line segment determined by the dipole, $(\mathbf{w}_{i1}, \mathbf{w}_{i2})$, of the neuron i is given by the set

$$S_{\mathbf{w}_{i1}\mathbf{w}_{i2}} = \left\{ \mathbf{s} \in R^N : \mathbf{s} = \alpha_i(\mathbf{x})\mathbf{w}_{i1} + (1 - \alpha_i(\mathbf{x}))\mathbf{w}_{i2}, \alpha_i(\mathbf{x}) \in [0,1] \right\} \quad (7)$$

The distance between an input pattern \mathbf{x} and the line segment $S_{\mathbf{w}_{i1}\mathbf{w}_{i2}}$ is given by expression following

$$D(\mathbf{x}, S_i) = \|\mathbf{x} - S_{\mathbf{w}_{i1}\mathbf{w}_{i2}}\| = \begin{cases} \|\mathbf{x} - (\alpha_i(\mathbf{x})\mathbf{w}_{i1} + (1 - \alpha_i(\mathbf{x}))\mathbf{w}_{i2})\| & \text{si } \alpha_i(\mathbf{x}) \in (0,1) \\ \|\mathbf{x} - \mathbf{w}_{i1}\| & \text{si } \alpha_i(\mathbf{x}) \geq 1 \\ \|\mathbf{x} - \mathbf{w}_{i2}\| & \text{si } \alpha_i(\mathbf{x}) \leq 0 \end{cases} \quad (8)$$

where $\|\cdot\|$ is the Euclidean norm. Note that when $\alpha_i(\mathbf{x}) \in (0,1)$ then $\mathbf{w}_{io} = \alpha_i(\mathbf{x})\mathbf{w}_{i1} + (1 - \alpha_i(\mathbf{x}))\mathbf{w}_{i2} \in S_{\mathbf{w}_{i1}\mathbf{w}_{i2}}$ is the nearest point to \mathbf{x} .

Given an input pattern \mathbf{x} , we say that neuron r is the winner (active) if

$$\|\mathbf{x} - S_r\| = \min_{1 \leq i \leq M} \|\mathbf{x} - S_i\|$$

Next, we characterize this property in terms of the action potential.

Teorema

$h_r \geq h_i$, $\forall i \neq r$ if and only if $\|\mathbf{x} - S_r\| \leq \|\mathbf{x} - S_i\|$, $\forall i \neq r$, $i = 1, 2, \dots, M$

Proof.

$$\begin{aligned} \|\mathbf{x} - S_i\|^2 &= \|\mathbf{x} - (\alpha_i(\mathbf{x})\mathbf{w}_{i1} + \bar{\alpha}_i(\mathbf{x})\mathbf{w}_{i2})\|^2 \\ &= \mathbf{x}^T \mathbf{x} - 2\alpha_i(\mathbf{x}) \mathbf{x}^T \mathbf{w}_{i1} - 2\bar{\alpha}_i(\mathbf{x}) \mathbf{x}^T \mathbf{w}_{i2} + \alpha_i^2(\mathbf{x}) \mathbf{w}_{i1}^T \mathbf{w}_{i1} \\ &\quad + 2\alpha_i(\mathbf{x}) \bar{\alpha}_i(\mathbf{x}) \mathbf{w}_{i1}^T \mathbf{w}_{i2} + \bar{\alpha}_i^2(\mathbf{x}) \mathbf{w}_{i2}^T \mathbf{w}_{i2} \end{aligned}$$

Moreover, as

$$\begin{aligned} \mathbf{w}_{i1}^T \mathbf{x} &= h_{i1} + \frac{1}{2} \mathbf{w}_{i1}^T \mathbf{w}_{i1} \\ \mathbf{w}_{i2}^T \mathbf{x} &= h_{i2} + \frac{1}{2} \mathbf{w}_{i2}^T \mathbf{w}_{i2}, \end{aligned}$$

we have

$$\begin{aligned} &= \mathbf{x}^T \mathbf{x} - 2\alpha_i(\mathbf{x})h_{i1} - 2\bar{\alpha}_i(\mathbf{x})h_{i2} - \alpha_i(\mathbf{x})\mathbf{w}_{i1}^T \mathbf{w}_{i1}(1 - \alpha_i(\mathbf{x})) \\ &\quad - \bar{\alpha}_i(\mathbf{x})\mathbf{w}_{i2}^T \mathbf{w}_{i2}(1 - \bar{\alpha}_i(\mathbf{x})) + 2\alpha_i(\mathbf{x})\bar{\alpha}_i(\mathbf{x})\mathbf{w}_{i1}^T \mathbf{w}_{i2} \\ &= \mathbf{x}^T \mathbf{x} - 2h_i, \end{aligned}$$

$$\begin{aligned} &\geq \mathbf{x}^T \mathbf{x} - 2h_r \\ &= \|\mathbf{x} - S_r\|^2 \quad \square \end{aligned}$$

Next, we establish the learning rule. The problem is to determine M line segment S_1, S_2, \dots, S_M such that the representation quadratic error

$$E = \sum_{i=1}^M \sum_{\mu=1}^p M_i^\mu \|\mathbf{x}^\mu - S_i\|^2$$

is minimized, where (M_i^μ) is the cluster membership matrix which specifies whether or not input pattern \mathbf{x}^μ activates neuron i as winner:

$$M_i^\mu = \begin{cases} 1 & \text{if } \mathbf{x}^\mu \text{ belong to cluster } i \\ 0 & \text{otherwise} \end{cases}$$

Gradient descent on this function yields the following learning rule:

$$\Delta \mathbf{w}_{r1}(k) = \begin{cases} \eta [\mathbf{x}(k) - \mathbf{w}_{r0}(k)] \alpha_r(\mathbf{x}(k)) & \text{si } 0 < \alpha_r(\mathbf{x}(k)) < 1 \\ \eta [\mathbf{x}(k) - \mathbf{w}_{r1}(k)] & \text{si } \alpha_r(\mathbf{x}(k)) \geq 1 \\ 0 & \text{si } \alpha_r(\mathbf{x}(k)) \leq 0 \end{cases}$$

$$\Delta \mathbf{w}_{r2}(k) = \begin{cases} \eta [\mathbf{x}(k) - \mathbf{w}_{r0}(k)] \bar{\alpha}_r(\mathbf{x}(k)) & \text{si } 0 < \alpha_r(\mathbf{x}(k)) < 1 \\ 0 & \text{si } \alpha_r(\mathbf{x}(k)) \geq 1 \\ \eta [\mathbf{x}(k) - \mathbf{w}_{r2}(k)] & \text{si } \alpha_r(\mathbf{x}(k)) \leq 0 \end{cases}$$

where η is the learning parameter. Like simple competitive learning, we can take $\eta(k) = \eta_o(1+k)^{-\alpha}$ (with $\alpha \leq 1$) or $\eta(k) = \eta_o(1-\alpha k)$.

3 Experimental Results

In this section we illustrate the performance of the proposed neural network on clustering problems. It was tested using the well-known Anderson's IRIS data set, which has been used for evaluating the performance of pattern clustering algorithms. This data set contains 150 feature vectors of dimension 4 which belong to three physical classes representing different IRIS subspecies: Virginica, Setosa and Versicolor. Although iris data are labeled, we only use that knowledge at end of clustering process, in order to compare the obtained results. The performance of the algorithm is evaluated by counting the number of crisp clustering errors, i.e., the number of feature vectors that are assigned to a wrong cluster. Figure 1 shows the two-dimensional projections of the IRIS data onto a plane spanned by the first two principal eigenvectors. There are two obvious cluster, but neither of them has a spherical shape. A clus-

ter is a mixture of two classes (iris versicolor and iris virginica). A number of graph-theoretic clustering methods, such as the minimal spanning tree approach, can handle nonspherical clusters. But, they all suffer from other difficulties such as definition of "inconsistent" edges. As a result, the squared-error clustering method with Euclidean distance is the most commonly used in partitional clustering methods. Many partitional clustering algorithms and competitive neural networks for unsupervised learning are suitable for detecting hyperspherical-shaped clusters, because Euclidean distance is most commonly used to compute the distance between a pattern and the assigned cluster center. Unsupervised clustering of the IRIS data typically results in 12-17 clustering errors [10]. Mao and Jain [10] have proposed a self-organizing network for hyperellipsoidal clustering (HEC) which can adaptively estimate the hyperellipsoidal shape of each cluster and use the estimated shape information in competitive learning.

Since the clustering solution of the K-means algorithm and the HEC network depend on initial weights, we repeat all experiment 50 times with different initializations and the best solutions obtains are reported in this paper. Since the second class (vesicolor) and the third class (virginica) are slightly overlapped and both are not spherically shaped as we can see from Figs. 1, the K-means algorithm produces a partition which does correctly group patterns from these two classes and this algorithm misclassifies 16 patterns (compared to the true classes). The HEC network adaptively estimates the hyperellipsoidal shape of each cluster by the covariance matrix (using the regularized Mahalanobis distance), and use the estimated shape information in competitive learning. It has misclassified only five patterns. The proposed network has misclassified also five patterns and needs less than 300 iterations (two learning epochs). Moreover, the proposed network finds three dipole segments that are a better representation of the clusters than the centroids. The dipole segments give us information about the shape of the cluster while the centroids only about the positions.

The following dipoles have been found:

$$\begin{array}{ll} \mathbf{w}_{11} = (4.8684, 4.1870), & \mathbf{w}_{12} = (7.2147, 5.9856) \\ \mathbf{w}_{21} = (2.5878, 4.5200), & \mathbf{w}_{22} = (3.0590, 6.5846) \\ \mathbf{w}_{31} = (6.5350, 4.9034), & \mathbf{w}_{32} = (9.1935, 6.3192) \end{array}$$

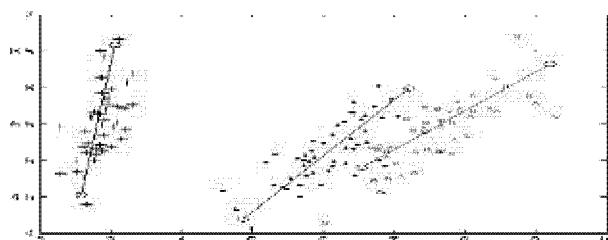


Figure 1. Two-dimensional projection maps (the first two principal components) of the IRIS data and the three dipole segments.

The confusion matrix C that correspondent to theses dipole segments is as following:

$$C = \begin{bmatrix} 50 & 0 & 0 \\ 0 & 46 & 4 \\ 0 & 1 & 49 \end{bmatrix}$$

This is, four patterns of the class 2 (subspecies) have been classified in class 3 and one pattern of the class 3 have been classified in class 2. The figure 2 shows as these dipoles are a good representation of the clusters.

4. Conclusions

Many partitional clustering algorithms and competitive neural networks are only suitable for detecting hyperspherical-shaped clusters. We have proposed a model of competitive neural network based in dipole segments instead of centroids where the synaptic weights of a process unit are given by a couple of vectors. The action potential has been defined so that the winner unit (maximum potential) is the one whose weight vector couple (line segment) is nearest to the input vector. Moreover, a weight update is formulated such that the dipole associated with each process unit is as near as possible to all the input samples for which the node is the winner of the competition. The network performs a partitional clustering using theses dipoles and finds dipole segments that are a better representation than the centroids used by competitive neural networks and so it is suitable for detecting hyperspherical-shaped clusters. Experiments on IRIS data have shown that the proposed network finds dipole segments that are a good representation of the clusters and has misclassified only five patterns.

5. References

1. Grossberg, S. Neural expectation: Cerebellar and retinal analogues of cells fired by unlearnable and learnable pattern classes, *Kybernetik*, 10: 49-57, 1972.
2. Grossberg, S. Adaptative pattern classification and universal recording: I. Parallel development and coding of neural detectors, *Biolog. Cybernetics*, 23: 121-134, 1976.
3. von der Malsburg, C. Self-organization of orientation sensitive cells in the striate cortex, *Kybernetik*, 14: 85-100, 1973.
4. Amari, S. and Takeuchi, M. Mathematical theory on formation of category detecting nerve cells, *Biological Cybernetics*, 29: 127-136, 1978.
5. Amari,S-I. Field theory of self-organizing neural nets, *IEEE Transactions on Systems, Man and Cybernetics, SMC-13*: 741-748, 1983.
6. Biensstock E., Cooper, E. & Munro, P. Theory for the development of neural selectivity: Orientation specificity and binocular interaction in visual cortex, *Journal of Neuroscience*, 2: 32-48, 1982.
7. Rumelhart, D. and Zipser, D. Feature discovery by competitive learning, *Cognitive Science*, 9: 75-112, 1985.
8. Ahalt, S.C., Krishnamurphy, A.K., Chen P. and Melton D.E. Competitive Learning Algorithms for Vector Quantization, *Neural Networks*, 3: 277-290, 1990.

9. Yair, E.K., Zeger K. and Gersho A. Competitive Learning and Soft Competition for Vector Quantizer Design, *IEEE Trans. Signal Processing*, 40 (2): 294-308, 1992.
10. Pal, N.R., Bezdek J.C. and Tsao E.C. Generalized Clustering Networks and Kohonen's Self-Organizing Scheme, *IEEE Trans. Neural Networks*, 4 (4): 549-557, 1993.
11. Xu, L., Krzyzak A. and Oja E. "Rival Penalized Competitive Learning for Clustering Analysis, RBF Net, and Curve Detection," *IEEE Trans. Neural Networks*, 4 (4): 636-649, 1993.
12. Ueda, N. and Nakano R. A New Competitive Learning Approach Based on an Equidistortion Principle for Designing Optimal Vector Quantizers, *Neural Networks*, 7 (8): 1211-1227, 1994.
13. Uchiyama, T. and Arbib M.A. Colour image segmentation using competitive learning, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16 (12): 1197-1206, 1994.
14. Mao, J. and Jain A.K. A Self-Organizing Network for Hyperellipsoidal Clustering (HEC), *IEEE Trans. Neural Networks*, 7(1): 16-29, 1996.
15. Hwang, W.J., Ye B.Y. and Liao S.C. A novel entropy-constrained competitive learning algorithm for vector quantization. *Neurocomputing*, 25: 133-146, 1999.
16. Zhang, Y.J. and Liu Z.Q. Self-splitting competitive learning: A New on-line clustering paradigm. *IEEE Trans. Neural Networks*, 13(2): 369-380, 2002.

An N-Parallel Multivalued Network: Applications to the Travelling Salesman Problem

Mérida-Casermeiro, E.¹, Muñoz-Pérez J.² and Domínguez-Merino, E.²

¹Dept. Matemática Aplicada, ²Dept. Ciencias de la Computación.
Universidad de Málaga.*

Abstract. In this paper, a new family of multivalued recurrent neural networks (MREM) is proposed. Its architecture, some computational properties and convergence is shown.

We also have generalized the function of energy of the Hopfield model by a new function of the outputs of neurons that we named “function of similarity” as it measures the resemblance between their outputs. When the function of similarity is the product function, the model proposed is identical to the binary Hopfield one.

This network shows a great versatility to represent, in an effective way, most of the combinatorial optimization problem [14–17] due to it usually incorporates some or all the restrictions of the problem generating only feasible states and avoiding the presence of parameters in the energy function, as other models do. When this interesting property is obtained, it also avoids the time-consuming task of fine tuning of parameters.

In order to prove its suitability, we have used as benchmark the symmetric Travelling Salesman Problem (TSP). The versatility of MREM allows to define some different updating rules based on effective heuristic algorithms that cannot be incorporated into others Hopfield models.

Keywords: Hopfield Network, Multivalued Network, Travelling Salesman Problem.

1 Introduction

In 1982, J.J. Hopfield introduced a powerful two-state neural network [8] for content addressable memory. The first neural network for combinatorial optimization (C.O.) problems was the analog Hopfield model [9, 10]. In general, the analogue version is superior to the binary one in terms of the local minima problem, however its convergence is slow and they do not always produce a valid state. On the other hand, the binary model has high speed for convergence but, usually, it has oscillatory behaviors and produces poor solutions. Some of these problems have been highlighted by Wilson et al. [20, 7].

Other powerful generalization of the binary model is the discrete model where the node outputs belong to a discrete set of values, $\{1, 2, \dots, N\}$. Multivalued type Hopfield neural networks have been used [12, 6] as content addressable

* Email addresses: merida@ctima.uma.es, munozp@lcc.uma.es, enrique@lcc.uma.es

memory. In [6] is introduced the MAREN model and used to solve C.O. problems, but it leads to a very complex representation and the obtained results are similar to standard models. The main problem with MAREN model is the expression of its energy: $E = \sum_{i=1}^P \sum_{j=1}^P w_{ij} \text{sgn}(V_i - V_j)$ where the use of the sign function produces drastic waste of information. Other problems with MAREN are:

- If W is a symmetric matrix then all states have the same energy.
- All states with the same ordering among their components have the same energy. So the states $(5, 3, 1, 2)$, $(6, 4, 1, 3)$, $(6, 4, 1, 2)$, $(4, 3, 1, 2)$, ..., and, in general, all states verifying $V_1 > V_2 > V_4 > V_3$ have the same energy.

These problems lead to more complicated expressions for the energy function than the multivalued recurrent model (MREM) proposed in this paper [14–17]. This model permits a good representation for most of C.O. problems and generates only feasible solutions in many problems, and so, the convergence to non feasible states is avoided and it does not demand fine-tuning of parameters, as most others models do.

The TSP is one of the most studied C.O. problem with many technical applications to scheduling and routing robots among them. It is a typical NP problem, so it is an important task to obtain an efficient approximate algorithm for finding a near optimum solution in reasonable time.

Neural networks have been used to solve C.O. problems since 1985 [10]. The advantages on using neural networks for C.O. problems can be seen in terms of speed, parallelism and hardware implementation. Usually type Hopfield neural networks have been used but the main problem on using the Hopfield model is the need to tune parameters to obtain both convergence to a valid state and good solutions (see [20]). This is not usually possible, so a significant amount of work has been reported to obtain some variants of the standard Hopfield model that improves their efficiency.

Some others neural models have been used to solve the TSP. Such as, Kohonen's self organizing map algorithm (SOM) [11, 1], elastic net algorithm [5] and Kohonen network incorporating explicit statistics (KNIES) [2]. Among them, the most accurate reported solutions have been obtained by KNIES. A focused to the problem preliminary version of MREM that improves the results obtained by KNIES has been recently published [15]. In this paper is given a general model for OC problems that improves all those results by using a N-parallel dynamic.

The MREM model is presented in section 2 and some of its properties are derived. Section 3 is focused to solve the TSP by MREM and finally we present our conclusions in section 4.

2 The MREM model of neural network

2.1 Network architecture

We consider a recurrent neural network, \mathcal{H} , formed by P nodes (neurons). The state of neuron i is defined by its output V_i , $i \in I = \{1, 2, \dots, P\}$, and it takes values in a set $\mathcal{M} = \{m_1, m_2, \dots, m_N\}$. This set can be a numerical set (e.g., $\mathcal{M} = \{1, 2, \dots, N\}$) or a non numerical one (e.g., $\mathcal{M} = \{\text{red}, \text{green}, \text{blue}\}$).

The state of the network, at time t , is given by a P -dimensional vector representing the outputs of the P neurons: $\mathbf{V}(t) = (V_1(t), V_2(t), \dots, V_P(t))$.

This network is fully connected and w_{ij} is the weight of the connection from j -th to the i -th node. The energy function is given by the following expression

$$E = -\frac{1}{2} \sum_{i=1}^P \sum_{j=1}^P w_{ij} f(V_i, V_j) + \sum_{i=1}^P \theta_i(V_i) \quad (1)$$

where $f : \mathcal{M} \times \mathcal{M} \rightarrow \mathbb{R}$ is a measure of similarity between the output neurons, $\theta_i : \mathcal{M} \rightarrow \mathbb{R}$ and \mathbb{R} is the set of real numbers.

The energy function in the Hopfield network is $E = -\frac{1}{2} \sum_{i=1}^P \sum_{j=1}^P w_{ij} V_i V_j + \sum_{i=1}^P \theta_i V_i$ where θ_i is a threshold. The energy function defined by (1) is more general than original Hopfield energy since bipolar Hopfield model is obtained taking $f(V_i, V_j) = V_i V_j$, $\theta_i(V_i) = \theta_i V_i$ and $\mathcal{M} = \{-1, 1\}$ (for $\mathcal{M} = \{0, 1\}$ the binary Hopfield model is obtained).

If \mathcal{M} is a discrete set, the similarity function is defined by an $N \times N$ matrix, F , where $F_{ij} = f(m_i, m_j) \in \mathbb{R}$. The function θ_i also can be defined by a vector of size $1 \times N$, so that all functions θ_i can be expressed by a $P \times N$ matrix, Θ , where θ_{ij} represents a threshold for the neuron i when it presents the state m_j .

In this way, the network can be represented by a quadruple $\mathcal{H} = (\mathcal{M}, W, F, \Theta)$ and the energy function (1) by

$$E = -\frac{1}{2} \sum_{i=1}^P \sum_{j=1}^P w_{ij} F_{V_i, V_j} + \sum_{i=1}^P \Theta_{i, V_i} \quad (2)$$

2.2 Dynamic of MREM network

We have considered discrete time and asynchronous dynamic, where only a neuron is updated at time t . Suppose that a is the index of the interrogated neuron at time t , then $V_a(t+1) = V_a(t)$, $\forall i \neq a$. In order to determine the next state of neuron a , we define the potential increment.

Definition 1. We define the potential increment of the a -th neuron, at time t , when it takes the state l , as:

$$U_{a,l}(t) = \frac{1}{2} \sum_{i=1}^{i=P} [w_{ai} F_{l, V_i(t)} + w_{ia} F_{V_i(t), l}] - \frac{1}{2} w_{aa} F_{ll} - \theta_{al} \quad (3)$$

This expression is formed by the terms in (2) that are affected by changes of a -th neuron, with different sign and it allows the following updating rule:

$$V_a(t+1) = l \Leftrightarrow U_{a,l}(t) \geq U_{a,i}(t), \forall i \in \mathcal{M} \quad (4)$$

Properties and Convergence Analysis: Next theorem and corollary establish that MREM, with the dynamic given by (4), converges in a finite number of steps to a stable state. Demonstrations can be seen in [14–17].

Theorem 1 (Convergence of the asynchronous dynamic). When at time t only the state of neuron a is updated by expression (4), then $E(t+1) \leq E(t)$.

Corollary 1. When $V_a(t) = k$ and $V_a(t+1) = l$ then $E(t+1) - E(t) = U_{a,k} - U_{a,l}$

To avoid cycles among states with the same energy the network will take that with minimal lexicographical order in \mathcal{M} .

3 Applications to the Travelling Salesman Problem.

In order to show the applications of any neural networks to C.O. problems, the symmetric TSP is widely used as a benchmark. Before to represent this problem by MREM, we will describe the problem:

Definition 2. The Travelling Salesman Problem (TSP).

Given a $N \times N$ symmetric matrix $D = (d_{ij})$ of distances between a set of N cities, ($i = 1, 2, \dots, N$), the objective is to find a minimum-length tour that visits each city exactly once.

In the classical Hopfield and Tank approach [10], the solution for the problem is described by an $N \times N$ matrix of binary elements, such that if $v_{ij} = 1$ the city i is visited in j th place in the tour. Preference of the obtained tour is expressed by the energy function:

$$E = \frac{A}{2} \sum_{i=1}^N \left(\sum_{j=1}^N v_{i,j} - 1 \right)^2 + \frac{A}{2} \sum_{j=1}^N \left(\sum_{i=1}^N v_{i,j} - 1 \right)^2 + \frac{B}{2} \sum_{i=1}^N \sum_{j=1}^N \sum_{k=1}^N d_{ij} v_{ik} (v_{j,k+1} + v_{j,k-1})$$

where the subscript are cyclic, that is $v_{i,N+1} = v_{i,1}$ and $v_{i,0} = v_{i,N}$.

The MAREN network approach [6] only uses N discrete multivalued neurons, where each neuron can take any value in $\{1, 2, \dots, N\}$. If $v_i = k$ it represents that the city i is visited in k th place. It was considered the next energy function:

$$E = A \sum_{i=1}^N \sum_{j=1}^N (sgn(v_i - v_j))^2 - \frac{B}{2} \sum_{i=1}^N \sum_{j=1}^N d_{ij} [1 - (sgn(abs(v_i - v_j) - 1))] - \frac{B}{2} \sum_{i=1}^N \sum_{j=1}^N d_{ij} [1 + (sgn(abs(v_i - v_j) - N + 1))]$$

where the first term penalizes that two different neurons take the same value, whilst the other terms evaluate the tour length.

The proposed MREM approach uses N neurons whose outputs take values in $\mathcal{M} = \{1, 2, \dots, N\}$, as in MAREN approach. The output $v_i = k$ means that the city k is visited in the i th place of the tour. So when the state vector \mathbf{V} is a permutation of $\{1, 2, \dots, N\}$ then it represents a valid tour.

Instead of applying a penalty method (as it is usual in Hopfield-type networks), we consider the search space as the set of valid tours. So we initialize the network in a valid state \mathbf{V}^0 and only transitions between feasible states are allowed. Therefore the network state is always a feasible one and the TSP can be solved by minimizing the simple energy function:

$$E = \sum_{i=1}^{i=N} D(v_i, v_{i+1}) \quad (5)$$

where the subscripts are cyclic, that is $v_{N+1} = v_1$ and $D(x, y)$ represents the distance function between cities x and y . This equation 5 can be obtained from expression of the energy of MREM given by equation 1, considering $f(x, y) = -2D(x, y)$, $\theta_i(x) = 0$, $\forall i$ and the weight matrix W with $w_{ij} = 1$ for $j = i + 1$, $w_{N,1} = 1$ and $w_{i,j} = 0$ in other case.

The most famous of the classical local optimization algorithms for the TSP are the 2-opt and 3-opt moves introduced by Croes [4]. In [14, 15] we proposed an focused to the TSP preliminary version of MREM that efficiently implements the 2-opt and 3-opt moves by an asynchronous dynamic that improves others existing neural approaches to TSP.

To implement the 2-opt move, let consider \mathbf{V} be a feasible state that represents a valid tour for the TSP, then two edges $[v_m, v_{m+1}]$ and $[v_n, v_{n+1}]$ are sequentially removed, it produces two paths that can be joined of two different forms to produce newly a valid tour, the original and that formed by reconnecting the two paths by the edges $[v_m, v_n]$ and $[v_{m+1}, v_{n+1}]$.

The network dynamics for MREM that implements the 2-opt moves, consists on selecting sequentially two neurons m and n , next we calculate the value of the potential increment by the expression:

$$-\Delta E = U = D(v_m, v_n) + D(v_{m+1}, v_{n+1}) - D(v_m, v_{m+1}) + D(v_n, v_{n+1}), \quad (6)$$

if $U < 0$ then we can obtain a new tour with less length by recombining the value of neurons between $m + 1$ and n . In a similar way in [15] the 3-opt move is implemented by considering 3 neurons m , n and p , extracting the three edges $[m, m+1]$, $[n, n+1]$ and $[p, p+1]$, it produces three paths that can be recombined of eight different forms to produce a valid tour. By analyzing which one produces the minimal length tour, we obtained a neural dynamics that produced better results in error terms but with a cost in time.

Now we present a new dynamic that implement the 2-opt and 3-opt moves but in a N-parallel way. Firstly, let we consider the 2-opt move. In step k , the process units calculates in parallel the increment of the length tour, so the unit i calculates the value of equation 6 for $m = i$ and $n = i + k$, where these values are cyclic, that is, if $i + k > N$ then $n = i + k - N$.

When the minimum value obtained among all neurons is negative, the index m_0 that obtains this value indicates the modification to be made in the tour. Finally the neuron states are updated in parallel by the rule:

$$v_i(t+1) = \begin{cases} v_i(t) & i \notin \{m_0 + 1, m_0 + 2, \dots, n_0\} \\ v_{m_0+n_0+1-i}(t) & i \in \{m_0 + 1, m_0 + 2, \dots, n_0\} \end{cases}$$

where newly the subscripts are cyclic and $n_0 = m_0 + k$.

Since two neurons can be selected of $\binom{N}{2}$ forms, the asynchronous dynamics needs $\frac{N(N-1)}{2}$ steps to consider all possible moves, whilst the N-parallel version only needs $\lceil \frac{N}{2} \rceil$ steps, where $\lceil x \rceil$ represents the highest integer value minor or equal than x .

In a similar way, the 3-opt move can be implemented by an N-parallel dynamics. In each step two values k and l indicate to process unit i that $m = i$, $n = i + k$ and $p = n + l$ where these indices are cyclic. So each process unit calculates the optimal recombination among the three paths formed by eliminating the edges $[m, m+1]$, $[n, n+1]$ and $[p, p+1]$. There exists seven possible recombination of those 3 paths different from the original tour. The best recombination obtained from the $7N + 1$ possible tours, indicates the new state of the network.

Simulations

The presented MREM model with the N-parallel dynamic has been tested with other existing neural approaches and its asynchronous version over instances taken from TSPLIB, the well-known data library collected by Reinelt [19]. The selected instances range between 51 and 532 cities. All experiments for MREM were run on a 1.8 Ghz Pentium IV PC with 64 Mbytes RAM by MATLAB, whilst simulations for others models have been reported on a HP7000 workstation.

The N-parallel routine has two parts. Firstly the neurons calculates in parallel the potential increments, later in sequential form their minimum and if it proceeds a better recombination is performed. To be fair we show the summa of the time that a neuron spends to obtain the potential increment and the sequential part, though it is simulated in a sequential computer.

Table 1. Comparison among several neural approaches for eil51 instance of TSPLIB.

	Hopf+SA	LME	LME-A	MR-A2	MR-P2	MR-A3	MR-P3
Av. error(%)	101.8	25.1	12.4	7.18	6.71	2.7	2.29
Av. time	16 h.	4.5 h.	187.3 s.	0.5 s.	0.22 s	22.29 s.	1.53 s

Table 2. Comparisons of the results of different dynamics of MREM for the eil101 instance.

	MREM-A2	MREM-P2	MREM-A3	MREM-P3
Av. error(%)	10.06	9.31	2.43	2.29
Av. time (s.)	2.22	0.75	198.14	7.57

Table 3. Comparisons between MAREN and MREM.

Cities	MAREN Best tour	MAREN % valid	MREM-A2 Best tour	MREM-P2 Best tour	MREM-A3 Best tour	MREM-P3 Best tour
20	5.39	72	2.121	2.116	2.081	2.080
30	6.49	64	4.633	4.621	4.516	4.517
50	11.84	83	6.020	5.975	5.738	5.734

Table 1 shows a comparison with others neural approaches for eil51 instance in terms of average time and error. So the Hopfield network with simulated annealing (Hopf.+SA), the Hopfield with local minima escape strategies model (LME) [18] and a model that uses augmented Lagrange multipliers to escape from local minima (LME-A) [13] have been tested jointly with different dynamics of MREM: asynchronous with 2-opt moves (MR-A2) and 3-opt moves (MR-A3), N-parallel and 2-opt moves (MR-P2) and 3-opt moves (MR-P3). It must be pointed out that the standard Hopfield model obtains an error of 273.7% (416% for eil101) [18], LME model obtains an average error of 43.2% and Hopf.+SA takes more of ten days for the eil101 instance. These results are highly improved by all reported methods. The best results are obtained by MREM methods. It can be observed that parallel methods obtain slightly shorter tours in much less time. Table 2 compares different dynamics for MREM.

We have also compared MREM with the MAREN network [6]. They reported results for different numbers of cities randomly placed on the interior of a square

Table 4. Comparisons between KNIES and MREM.

Instance	Optimal tour	KNIES		MREM-P2		MREM-P3		
		Best Q.	Best Q.	Av. Q.	Av. time	Best Q.	Av. Q.	Av. time
eil51	426	2.86	2.58	6.71	0.22	0.00	2.29	1.53
eil101	629	4.66	3.81	9.31	0.75	1.27	3.69	7.57
st70	675	1.51	0.88	7.27	0.42	0.00	2.13	2.83
att532	27686	6.74	6.57	9.63	25.95	2.05	3.78	600.84
bier127	118282	2.76	2.80	7.91	1.38	1.10	3.43	14.37
eil76	538	4.98	5.01	8.77	0.44	0.37	3.33	3.67
kroA200	28568	5.71	5.92	12.40	3.90	3.10	6.63	36.35
lin105	14379	1.29	2.26	7.96	0.24	0.00	2.18	1.71
pcb442	50778	10.44	7.13	11.63	13.15	2.47	3.97	310.14
pr107	44303	0.42	1.23	6.17	1.07	0.00	0.76	8.90
pr124	59030	0.08	1.16	5.35	1.51	0.00	1.61	8.99
pr136	96772	4.53	4.73	9.27	1.39	1.07	3.46	15.89
pr152	73682	0.97	0.54	5.18	2.20	0.64	1.22	16.84
rat195	2323	11.92	7.66	12.49	2.96	3.35	5.23	39.00
rd100	7910	2.09	2.82	9.21	0.95	0.00	3.48	7.31

of edge length 1. Table 3 shows the best tour of 100 trials and percentage of valid tours obtained by MAREN. Note that MREM always obtains valid tours.

Of all the neural network methods available for solving the TSP, it is believed that the algorithms based on the Kohonen's self organizing map are both the most accurate and the fastest. In 1999, Aras et al. [2] have reported a new self-organizing neural network to solve the TSP called KNIES. In that paper they compares the efficiency of the Pure Kohonen Network [11], the Guilty Net [3] and the approach of Angénol et al. [1]. Their results, for several instances from TSPLIB, proves that KNIES obtains the most accurate results, so they claim that KNIES is the most accurate neural network strategy reported for the TSP, at that date.

Table 4 compares the experimental results obtained by MREM with those reported by Aras [2]. Note that the reported results given by Aras are the best values recorded after running KNIES for a large number of parameter settings, where four parameters have to be adjusted. There are not comparisons in terms of time with KNIES since the authors do not give description of computation times. Results for MREM have been obtained for 100 runs with random initial state. The table shows that MREM with 3-opt moves obtains better solution than MREM with 2-opt moves but it takes more time.

4 Conclusions

A new type-Hopfield multivalued recurrent network has been introduced. This network is not only a generalization of the binary Hopfield network to multi-valued neurons, but it also introduces a more general types of neuron outputs different from real numbers. The only condition is the existence of a similarity function between the outputs of the neurons, so a wide range of outputs are allowed, vectors or non numerical sets among them.

The energy function is established in terms of the similarity function between the neuron outputs. This function allows an easy formulation for the TSP, where our model presents the advantage of that it has not fine tuning of parameters, as others networks do when they are solving C.O. problems. For others C.O. problems, the similarity function offers a wide range of possibilities and it usually includes restrictions in the energy function avoiding penalization terms.

The neural model has been applied to the TSP obtaining N-parallel neural implementations for classical 2-opt and 3-opt algorithms. The MREM neural models with these dynamics obtains good results that improves the actually reported for others neural models.

References

1. Angénol B., Vaubois, G. and Texier, J. *Self-organizing feature maps and the T.S.P.* Neural Networks, V. 1, pp. 289-93, 1988.
2. Aras, N. Oomen, B.J. and Altinel, I.K., *The Kohonen network incorporating explicit statistics and its application to the T.S.P.* Neural Networks, V. 12, pp. 1273-84, 1999.
3. Burke, L. I. and Damany, P., *The guily net for the T.S.P.* Computer and Operational Research, V. 19, pp. 255-65, 1992.
4. Croes, G. A. *A method for solving T.S.P.* Oper. Research, V. 6, pp 791-812, 1958.
5. Durbin, R. and Willshaw, D. *An analogue approach to the T.S.P. using an elastic net methods*, Nature, V. 326, 689-91, 1987.
6. M. H. Erdem & Y. Ozturk, *A New family of Multivalued Networks*, Neural Networks 9, 6, pp 979-89, 1996.
7. Zhi-Hong Guan, Guanrong Chen and Yi Qin, *On equilibria, Stability, and Instability of Hopfield Neural Networks*, IEEE Trans. N. N., V. 11, No 2, pp 534-41, 2000.
8. J.J. Hopfield, *Neural networks and physical systems with emergent collective computational abilities*, Proc. National Academy of Sciences USA, 79, 2254-58, 1982.
9. J. J. Hopfield, *Neurons with graded response have collective computational properties like those of two-state neurons*, Proc. Nat. Ac. of Sciences USA, 81, 3088-92, 1984.
10. J. J. Hopfield, & D.W. Tank, *Neural computations of decisions in optimization problems*, Biological Cybernetics, 52, 141-52, 1985.
11. Kohonen, T. *Self-organizing maps*. Berlin: Springer, 1994.
12. Kohring, G.A., *On the Q-state Neuron Problem in Attractor Neural Networks*. Neural Networks, V. 6, 573-81, 1993.
13. Li, S.Z., *Improving convergence and solution quality of Hopfield-type neural networks with augmented Lagrange multipliers*, IEEE Trans. N.N, V. 7, 1507-16, 1996.
14. E. Mérida Casermeiro, *Red Neuronal recurrente multivaluada para el R. patrones y la opt. comb.*, Ph.D. dissertation. University of Málaga, Spain, (in spanish), 2000.
15. Mérida-Casermeiro, E., Galán-Marín, G., Muñoz-Pérez. J., *An Efficient Multivalued Hopfield Network for the T.S.P.* Neural Processing Letters 14, 203-16, 2001.
16. Enrique Mérida, José Muñoz and Rafaela Benítez, *A Recurrent Multivalued Neural Network for the N-Queens Problem*. LNCS, Vol 2084, 522-529, 2001.
17. Enrique Mérida-Casermeiro, José. Muñoz-Pérez and M.A. García-Bernal, *An Associative Multivalued Recurrent Network*. LNAI, Vol. 2527, 509-518, 2002.
18. Peng, M., Gupta, N. K. and Armitage, A. F., *An investigation into the improvement of local minima of the Hopfield Network*. Neural Networks 9, 1241-53, 1996.
19. Reinelt, G., *TSPLIB- a T.S.P. library*, ORSA J. on Computing 3, 376-84, 1991.
20. Wilson, V. & Pawley, G.S. *On the stability of the TSP algorithm of Hopfield and Tank*, Biological Cybernetics 58, pp 63-70, 1988.

Parallel ACS for weighted MAX-SAT

Habiba Drias, and Sarah Ibri

USTHB, Computer Science Department,
Laboratory of Research in Artificial Intelligence
BP 32 El-Alia Bab-Ezzouar, 16 111 Algiers, Algeria
Drias@wissal.dz

Abstract. The ant colony system or ACS is an important approach of the Ant Colony Optimization meta-heuristic. In this work we aim to study the efficiency of ACS in solving the weighted max-sat problem. This will require an adaptation of all the rules of this approach to the elements which characterize our problem. All the ACO algorithms contain a low dependence level, this feature makes them beneficent to parallelize. Thereby, we will also study various ways to parallelize the ACS algorithm proposed in the first step of this work and discuss their impacts and differences.

1 Introduction

ACO is a population based meta-heuristic inspired by the behavior of real ants. It exploits the positive feedback as well as the greedy search. The basic elements of this meta-heuristic include some transition and update rules, applied by the artificial ants, which cooperate to find a good solution to the problem in consideration. The ant algorithms are composed of a colony of artificial ants, where each ant builds a solution starting from an initial state, and moving through a sequence of neighbor states. An ant chooses its path according to the transition rule, and updates the pheromone using the update rule. The computational complexity of the sequential ACO algorithms impedes their use for large problem instances. In fact, the generation and evaluation of solutions by artificial ants are the most costly in execution time. These parts offer a low dependence degree, which make the algorithm very suitable for a concurrent execution. In this paper we introduce a sequential ACS algorithm to solve the weighted maximum satisfiability problem or MAX-W-SAT and then, we propose two different ways to parallelize it. The basic idea of both methods is to divide the colony in several sub-colonies, to affect each sub-colony to a process and to launch processes in parallel. The synchronization mode and the communication way used by the processes constitute the main differences between the synchronous and the asynchronous methods proposed in this work. The last step of this study is to tune the empirical parameters of the algorithms and to compare their performance with those existing in the literature.

1.1 Weighted MAX-SAT problem

Formally the MAX-W-SAT problem can be described by following components:

- A set of n Boolean variables $X = \{x_1, x_2, \dots, x_n\}$.
- A set of m clauses $C = \{c_1, c_2, \dots, c_m\}$, each clause being a disjunction of literals. A literal is a variable or its negation.

- A weight w_i associated with each clause c_i .
- A solution s of an instance MAX-W-SAT is a sequence of n literals $\{x_{ij} \mid i=1..n \text{ and } j=0,1 / (x_{il}=x_i \text{ and } x_{i0}=\bar{x}_i)\}$ where x_i is the negation of x_i
- The cost of a solution s is equal to $\sum_{Ci \in SAT(s)} w_i$, $SAT(s)$ is the set of satisfied clauses by the solution s .

2 ACS with random walk for MAX-W-SAT

The communication used among ants is named *stygmeric communication*. It is characterized by a modification of physical states and a local access to these states by the ants. Like real ants, an ACO algorithm is composed of a colony of concurrent agents, *the artificial ants*, and uses an artificial pheromone or *pheromone information* as a mean of communication. The artificial ants cooperate to search a good solution to the problem. Each ant builds its own solution starting from a random initial state and moving through a sequence of states. It chooses the next state where to move with a probability calculated by the transition rule. The latter is composed of the pheromone information and possibly a heuristic value related to the application domain. The ant can add a pheromone when constructing the solution, we talk about *online step-by-step update* in this situation, or after the construction, and *online delayed update* is evoked in that case. The amount of the pheromone to add is proportional to the quality of the constructed solutions.

In the ACO algorithms, an evaporation mechanism similar to the evaporation of real pheromone is simulated, its goal is to direct the search towards new regions. The amounts of the evaporated and added pheromone are defined by the update rule. When an ant builds a solution and updates the pheromone information, it dies. Another technique called *offline update* consists in adding pheromone to the components of the best solution found by the ants at the end of each iteration of the algorithm. An ACO algorithm can also be enriched by some extra-competencies like local optimization and backtracking procedures.

The seminal work on ACO was the ant system algorithm *AS*, it was designed to solve the traveling Salesman Problem or TSP. This algorithm gave good results only for small instances. For this reason another improving versions have been proposed like AS with elitist strategy or *ASelite*, the rank based version or *ASrank* and the Max-Min AS or *MMAS*. All these algorithms know for most situations, premature convergence. Another more interesting algorithm called Ant Colony system or ACS, has been introduced to improve the Ant System.

At our knowledge, the ACS approach is for the first time studied for solving the satisfiability problem. A colony of artificial ants is used to solve a MAX-W-SAT instance. Each artificial ant starts from a complete initial solution generated randomly and applies to it a number of successive flips in order to build a final solution. The objective of the algorithm is to find a solution s which minimizes the sum of unsatisfied clauses weights, which is equivalent to maximize the weights sum of satisfied clauses.

Pheromone initialization. The pheromone is deposited on each literal of the instance. It represents the information on the previous use of this literal. In fact the higher the amount of pheromone is on a literal, the greater is the probability of choosing it again. The pheromone information is expressed by a function called *phero* such that: $\text{phero}: \text{Xx}\{0,1\} \rightarrow [0,1]$. It is implemented using an $n \times 2$ matrix and initialized with a small value $\tau=0.1$. The main framework of the ACS algorithm is described below. *Nbants* and *MaxIter* are respectively, the total number of ants and the maximum number of iterations determined by the physical machine limits.

ACS-SAT algorithm

```

begin
    Pheromone initialization ();
    For i=1 to MaxIter do
        begin for k=1 to nbants do
            begin generate a random initial solution ( $S_0$ );
                build a solution ( $S_k$ ) using the walkbuild procedure;
                apply online delayed update of the pheromone();
            end;
            determine the best solution of the iteration();
            apply offline-update of pheromone();
        end;
    end.

```

Building solutions. The procedure *walkbuild* outlined below is a stochastic local search algorithm or SLS for short, which have been proved to be very powerful in practice[7]. Each ant k starts from an initial solution s_0 generated randomly and constructs by an iterative process its own solution. The ant chooses at each iteration, a literal x_i to flip, with a probability P_{xi}^k computed by the following proportional rule.

$$P_{xi}^k(t) = \frac{\text{phero}[i, j]^\alpha \text{heur}[i, j]^\beta}{\sum_{x_j \in N^k} \text{phero}[l, k]^\alpha \text{heur}[l, k]^\beta} \quad x_l = 1 - k \text{ and } x_i = 1 - j \text{ at time } t \quad (1)$$

α and β are the parameters which control respectively the pheromone and the heuristic. N_k is the set of non taboo variables.

$$\text{heur}[i, j] = \frac{\sum_{c \in SAT(xij)} \text{weight}(c)}{\text{total weight of clauses.}} \quad (2)$$

$SAT(x_{ij})$ is the set of satisfied clauses when $x_i=j$.

Random Walk Strategy. The ant algorithm we have designed belongs to the WalkSAT/TABU algorithms [7], which have been successfully adapted to problems from various domains. The role of the random walk strategy is to bring diversification in the search and explore different parts of the state space. Let q be a random variable uniformly distributed over $[0,1]$, and $q_0 \in [0,1]$ a tunable parameter. If $q \leq q_0$ then:

$$P_{xi}^k(t) = \begin{cases} 1 & \text{if } \text{phero}(i,j) = \text{argmax}\{\text{phero}[i,j]^{\alpha} \text{heur}[i,j]^{\beta}\} \\ 0 & \text{else} \end{cases} \quad (3)$$

else the probability is computed using formula (1).

Argmax is the state j having the maximum value $\text{phero}[i,j]^{\alpha} \text{heur}[i,j]^{\beta}$. Using this decision rule, we can control the search exploration degree. In fact, increasing q_0 leads to concentrate the search on the literals that belong to the best solutions because the literals having the maximum value *argmax* are often chosen. However a small value of q_0 allows the choice of other literals, favoring thus the search exploration. After flipping a variable x , it is inserted in a taboo list. It keeps the taboo state for an iteration number equal to the taboo list length.

The performance of the ACO algorithms can be still augmented by applying an improvement technique to enhance the solutions built by the ants. In our algorithm, before each pheromone update a solution improvement technique is performed.

```
procedure walkBuild()
begin for i=1 to nbflip do
    begin generate a random variable q;
        if (q<=q0) then
            determine the variable x with the maximal probability
        else choose a variable x using rule (2);
            flip(x);
            put x in the taboo list;
    end;
end.
```

Pheromone update. The pheromone evaporation denoted by the evaporation rate ρ , is simulated by rule (4). Then another amount is added to the literals of the solution s^* obtained by the improvement technique. In the online delayed update, rule (5) is applied to add pheromone whereas in the online delayed update, rule (6) is taken into account. *bestsol* is the best solution found from the beginning of the algorithm.

$$\forall i=1..n \text{ and } \forall j=0..1, \text{phero}[i,j] = (1-\rho) * \text{phero}[i,j] \quad (4)$$

$$\forall (x_i=j) \in s^*, \text{phero}[i,j] = \text{phero}[i,j] + \rho * (1 - (\text{cost}(s^*) / \text{total weight})) \quad (5)$$

$$\forall (x_i=j) \in s^*, \text{phero}[i,j] = \text{phero}[i,j] + \rho * (\text{cost}(\text{bestsol}) / \text{cost}(s^*)) \quad (6)$$

3 Parallel ACS-SAT

In this section, we introduce two methods to parallelize the sequential algorithm described above. In both methods, the ant colony is divided into sub-colonies and affected to processes that run in parallel and apply the sequential algorithm. In order to balance the load between p processes, we affect the ants m_i to the process j , so that $j=i \bmod p$. In this way, the processes will have the same load, and each sub-colony will behave like a complete colony. The two proposed strategies, namely, synchronous and asynchronous differ from each other by the synchronization and the communication approaches.

Synchronous method. This method consists in using synchronous parallel computation, which requires synchronization points between processes, while particularizing the role of one as master and the others as slaves.

The master process creates the slaves processes, dispatches the ant colony to them and launches processes to work in parallel. When all the slaves finish their respective tasks, the master searches for the best found solution. It applies the offline update rule, and then launches slaves again. In this method processes do not communicate explicitly but share a common pheromone table, which is kept in a critical memory area where only one update is allowed at a time. The access to the pheromone table for reading or writing must fulfill the conditions below in order to keep the table consistent:

1. WriteFlag=0 and ReadFlag>0
2. WriteFlag=1 and ReadFlag=0.

Handling such problem requires some synchronization and mutual exclusion mechanisms. The use of monitors can guarantee the mutual exclusion whereas synchronization can be assured by signals, which are implemented in Java, our implementation language, by the *wait()* and *notify()* methods. Each reading (writing) must be preceded by a call to the *StartRead* (*StartWrite*) function and followed by the *EndRead* (*EndWrite*) function.

Synchronous algorithm

```
//Master's code
begin
    Initialize pheromone;
    Create slaves processes;
    Dispatch colony to slaves;
    For i=1 to MaxIter do begin
        Launch slaves;
        Wait for slaves ending;
        Find the best solution;
        Offline update of pheromone;
    end;
    stop all the slaves ;
end

StartRead()
Begin
    while (writing=true) do
        Wait()//passive wait;
        ReadFlag :=ReadFlag+1;
        if (ReadFlag=1) then
            reading:=true;
    End ;

EndRead()
Begin
    ReadFlag :=ReadFlag-1;
    if(ReadFlag=0) then
        reading:=false;
        notifyAll() ;
    End ;

//Slave's code (slave i)
begin  While not end do
begin
    Wait for master's signal;
    for each ant k of the process
    i do
    begin
        initialize a solution;
        build solution sk;
        online pheromone update;
    end
end
end

StartWrite()
Begin
while ((reading =true) or
(writing=true)) do
    Wait() ;
    Writing :=true ;
End ;

EndWrite()
Begin
    Writing :=false;
    notifyAll();
End;
```

Asynchronous method. The asynchronous method allows a straight communication between processes. This will avoid the synchronization points of the first approach and would theoretically reduce the execution time. In this method, each process carries out some local iterations to its sub-colony. When it finishes, it sends the best solution found in these iterations to the other processes and receives the best solutions found by them. It chooses the best one of all the processes, applies the

offline pheromone update and then begins again other local iterations. To eliminate wasted time owing to simultaneous access to the pheromone table, we use one table for each process or sub-colony.

Table1. Empirical parameters values

parameter	role	Value
q_0	Intensification/diversification rate	0.9
ρ	Evaporation rate	0.7
NbFlip	Number of flips applied in solutions construction	2xN/3
MaxStep	Iterations number of improvement technique	30
MaxIter	Iterations number of the algorithm	160
α, β	Weight of pheromone/heuristic in the decision rule	1,0

Asynchronous algorithm

```
//code common to all processes
Begin cpt :=0 ; (*counts the number of local iterations *)
  For i=1 to MaxIter do
    Begin for each ant of process i do
      Begin Generate an initial solution ;
        build a solution  $s_k$  ;
        Apply online pheromone update;
      end; cpt :=cpt + 1 ;
      if cpt = LocalIter then
        begin exchange solutions with other processes ;
          choose the best solution ;
          cpt :=0 ;
        end; Apply offline pheromone update;
    End;
End ;
```

LocalIter is the empirical parameter that determines the communication frequency between processes.

4 Experimental results

Numerous tests have been performed to evaluate the performance of the algorithms described above on instances of *Johnson* class. These problems are characterized by a set of 100 variables and a number of clauses between 800 and 950. All the clauses have a positive weight less than 1000. The algorithms have been implemented in Java and tested on a personal computer (Pentium 533MHZ, 128MO of RAM) working with Linux operating system. First, we began by tuning empirical parameters values. Table 1 contains the results of this step.

In the asynchronous parallel algorithm, the communication between sub-colonies is controlled by the empirical parameter *localIter*. It defines the number of local iterations done before each information exchange. The tests have shown that increasing *localIter* (which leads to reduce the exchanges frequency), can improve the quality of solutions. But beyond 30, the latter may degrade (fig.1).

In table2, we compare the performance of the algorithms proposed in this paper. Solution1 is the maximum value and solution2 the average obtained by 20 executions. The results show that:

- The synchronous parallel algorithm gives good maximal solutions. However its average solutions are surpassed by those of sequential and asynchronous algorithms.
- The asynchronous algorithm is at least as efficient as the sequential version. Therefore, we can maintain that when having a parallel machine, the asynchronous algorithm can be very efficient. Finally in table3, comparison between the best solutions of our algorithms (sequential, synchronous and asynchronous) and those obtained by Grasp and Scatter Search shows the efficiency of the ACS algorithm for nearly all the tested instances.

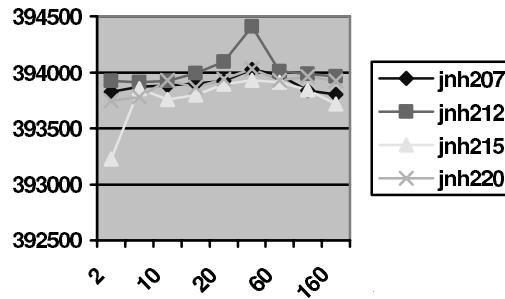


Fig.1. The impact of the communication frequency on the solution quality

Table2. Comparison of implemented algorithms

	sequential		synchronous		asynchronous	
	Solution1	Solution2	Solution1	Solution2	Solution1	Solution2
Jnh1	420925	420829	420925	420706	420925	420832
Jnh12	420925	420419	420925	420348	420925	420671
Jnh14	420733	420469	420824	420346	420824	420460
Jnh16	420911	420698	420914	420569	420911	420751
Jnh19	420497	420002	420455	419846	420497	420254
Jnh201	394238	394232	394238	394177	394238	394205
Jnh203	394119	393397	393766	393164	394105	393631
Jnh208	394159	393598	394013	393501	394159	393675
Jnh211	393863	393302	393979	393225	393836	393470
Jnh214	394163	393413	394013	393472	394163	393495
Jnh217	394238	394154	394238	394185	394238	394215
Jnh219	394053	393498	394059	393527	394053	393643
Jnh301	444842	444668	444842	444501	444842	444678
Jnh303	444299	443634	444351	443823	444318	443960
Jnh305	443714	443017	443471	442788	443857	443291
Jnh307	444083	443520	444083	443515	444083	443690
Jnh309	444578	444225	444578	444249	444578	444227

5 Conclusion

In this study we have used the ACS approach to solve the MAX-W-SAT problem. The main aim of this work was the parallelization of the sequential version of ACS. In this viewpoint we have implemented two strategies, the first one is synchronous and the second one asynchronous. The empirical tests have revealed the impact of

the communication frequency on the search quality. They show that a moderate frequency of communication is, in the case of our asynchronous parallel algorithm, more beneficial for the search. The comparison of the solutions found by our algorithms and those obtained by other meta-heuristics shows superior performance of the ACS sequential and parallel algorithms to solve MAX-W-SAT. Besides, we think that on a real parallel machine and using larger instances, tests could provide more conclusions concerning the speed up of our parallel algorithms.

Table 3. Comparison with other meta-heuristics

	ACS-SAT	GRASP	SS-SAT
Jnh301	444842	444612	444842
Jnh302	444459	443906	443895
Jnh303	444351	444063	444223
Jnh304	444533	444310	444533
Jnh305	443857	444112	443594
Jnh306	444838	444603	444515
Jnh307	444083	443836	443662
Jnh308	444568	444215	444250
Jnh309	444578	444273	444483
Jnh310	444274	444010	444313

References

1. Aguilar, J.: A general Ant Colony Model to solve combinatorial optimization problems, (1999)
2. Bulheimer, J B., Kotsis, G., Straub C.: A new rank based version of the ant system, A computational study walking paper 1997,to appear in the annals of operations research
3. Coloroni, A., Dorigo, M., Manniezo. V.: Distributed optimization by ant colonies, In proce of the first European conference on artificial life, pages 134-142 Elsevier, 1992
4. Cordon, O., Deviana, I., Herrera, F., Moreno, L.: A new ACO model integrating evolutionary computation concepts, ANTS2000, Brussels, 2000, 22-29
5. Delisle, P., Krajecki, M., Gravel, M., Gagné C., Parallel implementation of an ant colony optimization meta-heuristic with OpenMP, EWOMP'2001, Barcelone, (2001)
6. Doerner, K., Hartl, R.F., Reimann, M.: Cooperative ant colonies for optimizing resource allocation in transportation, evolutionary workshop, LNCS 2037, 70-79, (2001)
7. Dorigo, M., G.Dicaro. M.: Ant colony optimisation: A new meta-heuristic, artificial life vol5, Nº3 137-172, (1999)
8. Drias, H., Khabzaoui, M.: Scatter Search with Random Walk Strategy for SAT and MAX-W-SAT problems, IEA-AIE'2001, LNAI 2070, Springer , Budapest, (2001) 35-44
9. Holger H., Hoos, H.: On the runtime behaviour of stochastic local search algorithms for SAT, ΛΛΛΙ, 661-666, (1999)
10. Manniezo, V., Carbonaro. A.: Ant colony optimization, Kluwer, 21-44, (2001)
11. Middendorf, M., Reishle, F., Schmeck, H.: Information exchange in multi-colony ant algorithms, LNCS 1800, 645-652, (2000)
12. Parlados, P.M., Pistoulis L., Resende, M.G.C., A parallel grasp for Max-Sat problems. Center for applied optimization, Springer, PARA 96: 575-585 (1997)
13. Randall, M., Tonkes A., Intensification and diversification strategies in ant colony search. Tech rep, School of information technology, Bond University, (2000)
14. Resende, M., Pistoulis L., Parlados, PM., Approximate solution of weighted MAX-SAT problems using Grasp, Disc Math and Theo Comp Sci, vol 35, 393-405, (1997)

Learning to Generate Combinatorial Action Sequences Utilizing the Initial Sensitivity of Deterministic Dynamical Systems

Ryunosuke Nishimoto^{1,2} and Jun Tani^{1,2}

¹ The University of Tokyo, 3-8-1 komaba, Meguro-ku, tokyo 153-8902, Japan

² Brain Science Institute, RIKEN 2-1 Hirosawa, Wako-shi, Saitama, 351-0198 Japan
`{ryu,tani}@brain.riken.go.jp`

Abstract. This study shows how sensory-action sequences of imitating finite state machines (FSMs) can be learned by utilizing the deterministic dynamics of recurrent neural networks (RNNs). Our experiments indicated that each possible combinatorial sequence can be recalled by specifying its respective initial state value and also that fractal structures appear in this initial state mapping after the learning converges. We also observed that the RNN dynamics evolves, going back and forth between regions of window and chaos in the learning process, and that the evolved dynamical structure turns out to be structurally stable after adding a negligible amount of noise.

1 Introduction

It is not well understood how various action sequences are learned and generated in brains. Previously, Tanji and Shima [1] conducted electro-physiological experiments on Macaque monkeys in which pre-SMA activities were recorded while the monkeys generated action sequences of combining 'Push', 'Pull', and 'Turn' behaviors. The results showed that there exist "motor preparatory neurons" activities in pre-SMA that correspond to specific behavior sequences during the motor preparatory period. For instance, the neurons which activate only before the combinatorial action "Push-Pull-Turn" won't activate in the case of "Push-Turn-Pull". Therefore, it is reasonable to assume that these motor preparatory neurons contain certain information related to the action sequences.

After considering the neural mechanisms for the above combinatorial actions, we developed a new nerural network model that utilizes the initial sensitivity characteristics of dynamical systems. Our interpretation of the findings of Tanji and Shima[1] is that pre-SMA might set the initial conditions of a forward model network (assumed to be in cerebellum [2]) which generates future motor sequences. The forward model can generate different sensory-motor sequences by means of its forward dynamics depending on the initial state given to it. In our proposed scheme, the sensory-action sequences for future behavior plans are internally represented in terms of the corresponding initial internal states. Therefore, specific behavior sequences can be regenerated, once their corresponding initial states are memorized in association with goals.

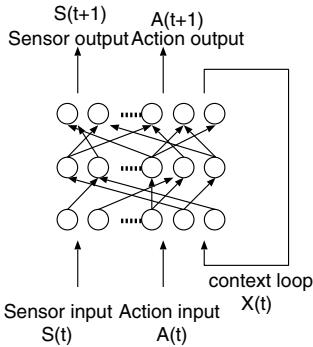


Fig. 1. The RNN employed in this paper.

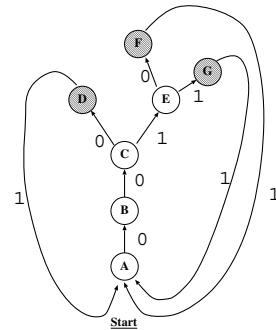


Fig. 2. The finite state machine environment.

A question now arises: “Could this model account for the mechanism of combinatorial action selections in sequences that are required in the general action planning paradigm?” For example, assume that a robot is navigating through a maze environment that is represented by a graph structure consisting of multiple branching points. In this situation, there could exist infinite number of paths that the robot can go through by means of arbitrary combinations of branching. How can the forward model generate this sort of different combinations of actions as dependent on the initial state values? In this paper, we will show that the initial sensitivity characteristics of deterministic chaos can account for the combinatorial mechanism in generating action selections. We will also show that this scheme of action generation can be sufficiently robust under certain conditions.

2 Neural network model

By utilizing the recurrent-type neural network (RNN) model [3–7] as shown in Fig1, sensory-action sequences are learned in terms of the forward model. More specifically, the RNN learns to predict the incoming sensory state S_{t+1} and next branching action A_{t+1} by perceiving the current sensory inputs S_t and the branching action A_t through the agent’s iterative search. The internal state X_t represented by context units in the input layer is mapped to X_{t+1} in the output layer. As will be discussed in the experimental results section, the mechanism of generating combinatorial branching can be achieved by adequately self-organizing the internal structure utilizing the context units.

In the learning of the sample sequences, the connective weights of the RNN can be obtained through the iterative calculation utilizing the back-propagation through time (BPTT) algorithm [8]. The value of the i th unit of context units $x_{0,i}$ in the initial step for each training sequence is iteratively computed by the steepest descent method utilizing δ information in the initial step. This iterative search of the initial state is conducted for each sequence, simultaneously with

the one for the connective weights. In the actual simulation, only two context unit values are updated in the initial step. The values of the remaining context units are fixed at 0.5, in the initial step. It is expected that the learned RNN can regenerate each sequence by means of its forward dynamics computation , starting from each initial state, as determined from when the learning converges.

To generate the action sequences, the environment and the learned RNN are coupled with the sensory-action loop. The forward computation [9] starts in the RNN with the setting of the initial sensory and action values and two initialcontext unit values in the input layer. The action outputs as well as all the context unit outputs are fed into their corresponding units in the input layer for the next step of the forward computation. Simultaneously, the current state in the environment is updated based on the action generated by the RNN. The resultant sensory values given in the environment are fed into the sensory input units of the RNN. This iteration of the forward computation is continued as long as the sensory prediction in the output layer accords with its actual sensation in the environment. Note that this open-loop computation of the RNN coupled with the FSM is completely deterministic.

3 Experiment setting

In our numerical experiments, a maze-like environment is represented simply by a finite state machine (FSM) in which an agent explores, as shown in Fig.2. Each node in the target FSM represents a corresponding landmark in which sensory inputs of (A,B,C,D,E,F,G) are perceived with given binary branching actions of (0,1) at every node. It should be noted that this FSM is closed; –i.e., starting from the start node “A”, the agent goes through one of the goal nodes of “D”, “F” and “G” and returns to the starting node within a finite number of steps. Note also that number of the agent’s possible paths increases exponentially when the agent cycles around this environment more times.

In the learning experiment, a set of the sensory-action training sequences were prepared by utilizing the FSM environment. More specifically, the agent was guided in such a way as to loop around the FSM environment for three cycles, starting from the start node “A”. (Cycle 1 starts at node “A”, and the agent must pass through one of the goal nodes “D”, “F”, or “G” before it can return to “A”). Because of multiplicative combinations of branching at each node, there exist $3^3 = 27$ different sequences for the agent traveling through three cycles. In the learning experiment, 21 sample sequences out of the 27 were used for the purpose of testing the generalization capabilities in the learning process.

The questions that could be studied with the experiment are as follows:(1) How is the initial sensitivity utilized in generating complex sequences? (2) To what extent can the learning be generalized by using only partial training data? (3) What type of dynamics evolves for embedded complex sequences? (4) How robust is the dynamics in terms of generating sequences?

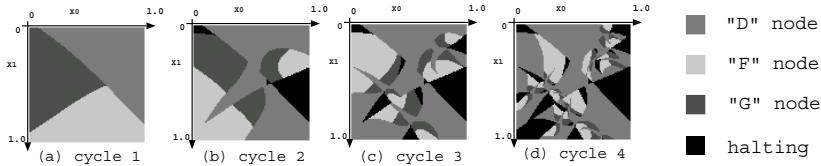


Fig. 3. The initial sensitivity map at each cycle.

Table 1. The number of generated sequences and the percentage of halting sequences

Cycle	generated patterns /possible combinations	halting percentage
Cycle 1	3/3 (100.0%)	0.00%
Cycle 2	9/9 (100.0%)	9.10%
Cycle 3	27/27 (100.0%)	14.48%
Cycle 4	78/81 (96.3%)	22.71%

4 Results and Analysis

The learning process converged with a mean square error of 4.16×10^{-6} after 50000 BPTT steps. The details of the convergence process will be explained later. Firstly, we observed how the sensory-action sequences could be generated depending on the two variables of the initial internal state by plotting their phase diagram. For this purpose, multiple steps of the forward activation are computed while varying the setting of the initial values of the two context units used for adaptation in the learning phase.

Figure 3(a) shows a mapping between the initial state values and the nodes (“D”, “F” or “G”) reached before returning the start node during one cycle. The colored regions represent the regions of the initial state that lead to the goal nodes “D”, “F” or “G”. It can be seen that the three regions are about the same size. For multiple cycles, it is assumed that the initial state space has to be divided into a number of sub-regions in order to generate all possible action sequences. For comparison, figure 3(b)(c)(d) shows the same mapping at each cycle. The color indicates the goal nodes “D”, “F” or “G” reached by the end of the cycle. In the map of the second cycle, each “D”, “F” or “G” region of the first cycle has been further sub-divided into “D”, “F”, “G” and halting sub-regions.

This means that all possible combinations of action sequences within two cycles are mapped into these separate sub-regions in the initial state space. The black regions represent a halting region for which the initial state sensory prediction is incorrect and the forward computation has halted. Similar recursive division structures in the initial state space were observed in cycle 3 and cycle 4. This suggests that the combinatorial action generation properties of the FSM are imitated by the RNN recursive functions by means of self-organizing fractal-like structures in the mapping from the initial state to the goals reached.

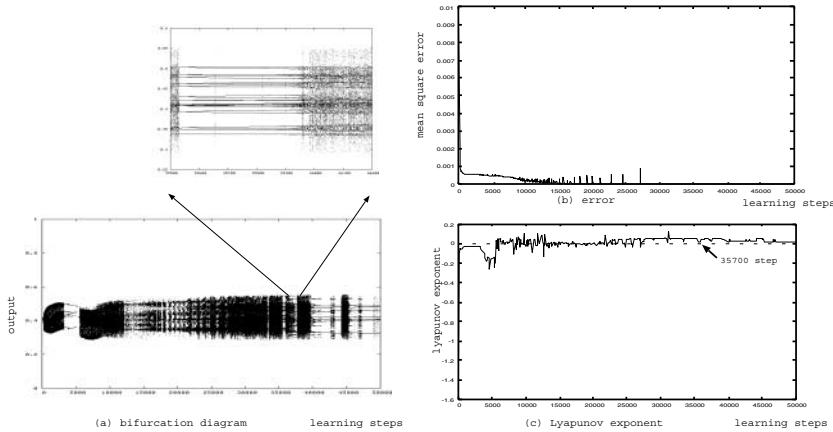


Fig. 4. (a) bifurcation diagram, (b) the mean square learning error and (c) the Lyapunov exponent of the RNN dynamics during the learning process.

We determined how many valid action sequences were generated, as well as how many the halting sequences were generated using the learned RNN by varying the two initial context unit values for each cycle, up to four cycles. The table 1 shows the ratio of number of generated sequences versus number of all possible sequences for each cycle and the percentage of halting sequences generated. All possible sequences were generated in each of the first three cycles, and nearly all sequences were generated in the fourth cycle. This result suggests that the learning process achieved a certain amount of generalization even though a partial training set was utilized. (Remember that the training was conducted using 21 sample sequences out of 27 possible sequences for the case of three cycles.)

Next, we examined how the structure developed during the learning process. Figure 4 shows (a) the bifurcation diagram, (b) the learning mean square error and (c) the Lyapunov exponent during the learning process. The bifurcation diagram shows that a typical pattern of nonlinear dynamic systems occurred where regions of chaos (densely plotted parts) and windows (indicating limit cycling dynamics) intermingle with each other. The bifurcation pattern enlarged after about 36000 learning steps. The mean square error reached a minimum after 15000 learning steps (Figure 4 (b)). The Lyapunov exponent was either zero or slightly positive after the learning error converged (Figure 4 (c)). In the figure, the Lyapunov exponent is slightly positive from the 28000th to the 40000th learning step, where the window structures can be barely seen; whereas, it is almost zero where the window structure is apparent. To examine in detail the dynamic structures of the RNN developed during the learning process, the phase plot and the initial state sensitivity mapping were drawn for specific time points in the learning process. Figure 5 shows plots at 35700, 36000 and 49900 learning steps, respectively. An average of one half of all context outputs and that for the

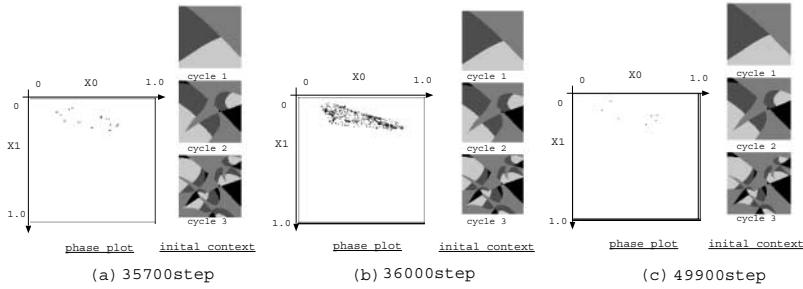


Fig. 5. The phase plot diagram and the initial state sensitivity map are shown for the cases of 35700, 36000 and 49900 learning steps.

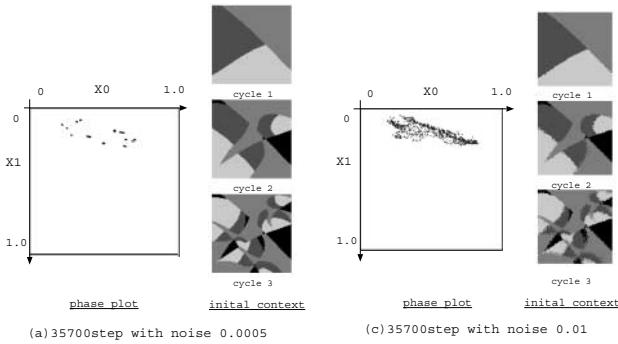


Fig. 6. (a) and (b) show the phase plots and the initial sensitivity map for 35700 learning steps with added 0.0005 and 0.01 magnitude noise, respectively.

remained context outputs are computed for the last 1000 steps and plotted in the 2-dimensional space. In these phase plots, a strange attractor suggestive of chaos appears only in the case of 36000 learning steps. All other plots indicate generation of limit cycling dynamics, since only finite number of points are observed in those plots. The initial state mappings have similar structures. The apparent contradiction regarding these observations is that the dynamical structure of the RNN seems to continue to change dramatically between chaos and limit cycling even after the 35700th step where the learning error is minimized and the initial sensitivity map becomes constant. The mathematical theory of the symbolic dynamics, as well as that of Markov partitions [10], indicates that deterministic dynamical systems can imitate stochastic ones in branching of FSMs only if they can generate chaotic dynamics. Thus, how can the RNN exhibiting a limit cycle with fixed periodicity imitate the FSM characteristics of the combinatorial branching? One possible reason explaining this contradiction might be that the transient dynamics before converging to the limit cycle is relatively long and relatively complex. This is apparent from the result that diverse action

sequences of relatively long steps are generated by utilizing the initial sensitivity characteristics even in the case of limit cycling dynamics, as shown in the initial state sensitivity map for 35700 learning steps.

To examine the dynamical structure including the transient region nearby the invariant set of limit cycles, the phase plots were re-drawn by adding a relatively small amount of noise to the forward dynamics of the RNN. It was expected that the dynamics would be slightly perturbed by the added noise, by which the dynamical structure nearby the invariant set might be revealed. Gaussian noise of magnitude 0.0005 or 0.01 was added to the context unit loops for every forward dynamics iteration of the 35700 learning step cases. (See Figures 6) The attractor when the added noise magnitude is 0.01 is quite similar to the one of deterministic chaos observed in the 36000 learning step of Figure 5 (b). Looking at the initial sensitivity map in Figures 6 (a) and (b), one can see that the characteristics of the initial sensitivity are mostly preserved. (Note that the sizes of the halting regions are also equal.) These observations confirm that the essential dynamical structure including the transient region near the invariant set becomes structurally stable after the error is minimized, although the attractor itself goes back and forth between limit cycling and chaos.

This means that the acquired dynamic function of the RNN for imitating the FSM is sufficiently robust and that the behavioral characteristics of the trained RNN in terms of generating long action sequences can become almost the same as the one of the target FSM simply by adding a negligible amount of noise.

5 Discussion and Summary

The current study investigated how a specific action sequence can be recalled from among other learned sequences by setting its respective initial state condition of a forward model by using an RNN. In the case of learning to imitate target FSMs, it was shown that fractal like structures are self-organized in the initial state sensitivity map by which most of possible combinations in branching sequences can be generated. It was also found that action sequences of the target FSMs can be imitated by organizing the complex transient dynamics in the network. Although it was observed that the invariant set of the RNN dynamics dramatically changes between chaos and limit cycling even after the learning has almost converged, our analysis showed that the acquired dynamical structure becomes structurally stable after the addition of a negligible amount of noise and that the original initial sensitivity properties of deterministic dynamics are almost preserved in such noise-added conditions.

One may ask what are the essential advantages of utilizing the initial sensitivity characteristics in generating action sequences. One obvious advantage is that sensory-action sequences of arbitrary length can be encoded by their respective initial state values. Additionally, the scheme can be extended to perform goal-directed planning. In the case of our FSM environment, a goal node can be specified in terms of its sensory state and possible action sequences for reaching this node can be searched by generating simulated action sequences by

modulating the initial state value for the forward dynamics. If the generated sequence is found to be worthwhile memorizing for future re-use, the system only has to memorize its corresponding initial state value in an additional memory system. Although there are many prior studies investigating the RNN capabilities for learning FSMs or grammatical structures, no studies have focused on utilizing the initial sensitivity of the deterministic RNN dynamics for generating structured sequences.

There are many unanswered questions related to the current study. One essential question is how to extend the current scheme of a discrete time system to that of a continuous time system. This question will be important when the scheme is applied to real sensory-motor systems operated in the continuous time domain. When applied to real sensory-motor systems, the issues of on-line planning will also become important. We also have to study the possibility of incremental learning instead of off-line learning utilized in the current study. We expect that the general framework for utilizing the initial sensitivity of the RNN deterministic dynamics should, in the future, be studied with accompanying experiments on goal-directed behaviors in real sensory-motor systems.

References

1. J. Tanji and K. Shima. Role for supplementary motor area cells in planning several movements ahead. *Nature*, 371:413–416, 1994.
2. D. Wolpert and M. Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11:1317–1329, 1998.
3. M.I. Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. In *Proc. of Eighth Annual Conference of Cognitive Science Society*, pages 531–546. Hillsdale, NJ: Erlbaum, 1986.
4. J.L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.
5. J.B. Pollack. The induction of dynamical recognizers. *Machine Learning*, 7:227–252, 1991.
6. J.F. Kolen. *Exploring the computational capabilities of recurrent neural networks*. PhD thesis, The Ohio State University, 1994.
7. C.L. Giles, G.Z. Sun, H.H. Chen, Y.C. Lee, and D. Chen. Higher order recurrent networks and grammatical inference. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 380–387. San Mateo, CA: Morgan Kaufmann, 1990.
8. D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In D.E. Rumelhart and J.L. Mclelland, editors, *Parallel Distributed Processing*. Cambridge, MA: MIT Press, 1986.
9. M.I. Jordan and D.E. Rumelhart. Forward models: supervised learning with a distal teacher. *Cognitive Science*, 16:307–354, 1992.
10. B. I. Hao. *Elementary Symbolic Dynamic and Chaos in Dissipative System*. Singapore: World Scientific, 1989.

BICONN: A Binary Competitive Neural Network

J. Muñoz-Pérez, M. A. García-Bernal, I. Ladrón de Guevara-López
and J.A. Gomez-Ruiz

Dpto. Lenguajes y Ciencias de la Computación
E. T. S. Ingeniería Informática. Universidad de Málaga
Campus de Teatinos s/n
29071-Malaga, Spain.
e-mail: munozp@lcc.uma.es

Abstract. In this paper a competitive neural network with binary synaptic weights is proposed. The aim of this network is to cluster or categorize binary input data. The neural network uses a learning mechanism based on activity levels that generates new binary synaptic weights that evolve toward *medianoids* of the clusters or categorizes that are being formed by the process units of the network, since the *medianoid* is the better representation of a cluster for binary data when the Hamming distance is used. The proposed model has been applied to codebook generation in vector quantization (VQ) for binary fingerprint image compression. The binary neural network find a set of representative vectors (codebook) for a given training set minimizing the average distortion.

1 Introduction

The unsupervised competitive learning is a mechanism that allows the detection of regularities in the patterns inputs. It was introduced by Grossberg ([1]) and von der Malsburg ([2]) and developed by Amari *et al* ([3] and [4]), Bienstock *et al* ([5]) and Rumelhart *et al* ([6]). The aim of unsupervised competitive learning is to cluster or categorize the input data. Similar inputs should be classified as being in the same category, and so should fire the same output unit.

A competitive neuronal network consists of a single layer of M process units (neurons) fully connected to a same input $\mathbf{x} \in \mathbb{R}^N$ and producing an outputs $y_i \in \{0,1\}$, $i=1,2,\dots,M$. We say that process unit i is active if $y_i=1$. For each input (stimulus) there is only one active unit. This active unit is called the *winner* and is determined as the unit with largest *activation potential*. The activation potential of the process unit i is the inner products $\mathbf{w}_i^T \mathbf{x}$, where \mathbf{w}_i is the synaptic weight vector of process unit i and $\|\mathbf{w}_i\|=1$. Thus the winner is that process unit with the weight vector closest to the input vector \mathbf{x} (in Euclidean distance sense). That is, the best match of the input vector \mathbf{x} with the synaptic weight vectors. The synaptic weight vector of the winning process unit, r , is updated according to the standard competitive learning rule,

$$\Delta \mathbf{w}_r = \eta(\mathbf{x} - \mathbf{w}_r) \quad (1)$$

which moves \mathbf{w}_r directly toward \mathbf{x} . In this way, the synaptic weight vectors will become the *centroids* of the M clusters or categories that are formed by the networks. However, since the learning parameter $\eta \in (0, 1)$ then the new weight vector \mathbf{w}_r could be no binary. We look for a new learning rule where the weight vectors became *medianoids* of cluster of input data. Note that the *medianoid* of a cluster of binary inputs is its best binary representation.

A well known binary competitive network is the Hamming network that is a maximum likelihood classifier that can be used to determine which of a group of prototype vectors is more similar to the input vector (stimulus). The prototype vectors determine the synaptic weights. The measure of similarity between the input vector and stored prototypes vectors is given by N minus the Hamming distance between these vectors. Another learning model that also allows the formation of clusters is the method ART1 (Adaptative Resonance Theory) that was developed in [7] (a simplified version has been shown in [8]). This algorithm adjusts the winning vector \mathbf{w}_r by deleting any bits in it that are not also in \mathbf{x} . However, while \mathbf{w}_r preserves its binary nature, the new prototype \mathbf{w}_r can only have fewer and fewer 1s as training progresses.

In this paper a new model with binary inputs, output and synaptic weights is proposed based on a learning rule that preserves the binary nature of the synaptic weights and they evolve toward the *medianoids* of clusters of input data. Moreover, the binary synaptic weight vectors are not normalized as in the simple competitive learning and so we have 2^N possible binary vectors instead of N . This algorithm uses only binary operations and so it is very computationally efficient. The rest of this paper is organized as follows. In the section 2 we develop the model Biconn (Binary Competitive Neural Network). We present an application to compression of fingerprint images by codebook generation in section 3. Conclusions are presented in the section 4.

2 A Binary Competitive Neural Networks

Consider the Hamming space \mathbf{H}^N ,

$$\mathbf{H}^N = \{\mathbf{x} = (x_1, x_2, \dots, x_N)^T \in \mathcal{R}^N : x_i \in \{0, 1\}, i=1, 2, \dots, N\}$$

The Hamming distance between the binary vectors $\mathbf{x} = (x_1, x_2, \dots, x_N)^T$, $\mathbf{y} = (y_1, y_2, \dots, y_N)^T \in \mathcal{R}^N$, is given by expression $d_H(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T (\mathbf{1} - \mathbf{y}) + (\mathbf{1} - \mathbf{x})^T \mathbf{y}$. Let $C = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p\}$ be a set of p vectors of \mathbf{H}^N . We define the *medianoid* of C as the vector $\mathbf{m} = (m_1, m_2, \dots, m_N)$ of \mathbf{H}^N given by expression

$$m_i = \begin{cases} 1 & \text{if } u_i \geq v_i \\ 0 & \text{if } u_i < v_i \end{cases}$$

where u_i is the number of i -th components of vectors of C with value one and v_i is the number of i -th components of vectors of C with value zero. It is easy to prove that the *medianoid* de C is the best representation of C by a vector of \mathbf{H}^N when the Hamming

distance is used. That is, a *medianoid* of C is the vector \mathbf{w} that minimizes the distortion function given by the expression

$$\sum_{j=1}^p d_H(\mathbf{x}_j, \mathbf{w}). \quad (2)$$

Next we propose a binary competitive neural network to cluster or categorize input data where synaptic weight vectors evolve toward the *medianoids* of clusters.

The binary neural network consist of a single layer of M output units, $\{O_1, O_2, \dots, O_M\}$, each receiving the same input $\mathbf{x} = (x_1, x_2, \dots, x_N)'$ and producing an outputs $\{y_1, y_2, \dots, y_M\}$. They are fully connected to a set of inputs (x_1, x_2, \dots, x_N) via connections w_{ij} , $i=1, 2, \dots, M$, $j=1, 2, \dots, N$, that are called synaptic weights. We consider that inputs, outputs and synaptic weights are binary. Only one of the output units, called the winner, can be activated at a time. The active unit is determined as the unit with the largest net input, where the net input of unit i is given by the expression

$$h_i = \sum_{j=1}^N w_{ij} x_j - \frac{1}{2} \sum_{j=1}^N w_{ij} = \sum_{j=1}^N \left(x_j - \frac{1}{2} \right) w_{ij} \quad (3)$$

and its output is

$$y_i(k) = \begin{cases} 1 & \text{if } h_i = \max_r \{h_r\} \\ 0 & \text{otherwise} \end{cases}, \quad i=1, 2, \dots, M$$

Thus, the winner is that unit with weight vector closest (in Hamming distance sense) to the input vector. It is established in the following proposition:

Proposition 1

$$h_i > h_r \Leftrightarrow d_H(\mathbf{x}, \mathbf{w}_i) < d_H(\mathbf{x}, \mathbf{w}_r) \quad (4)$$

where $\mathbf{w}_i = (w_{i1}, w_{i2}, \dots, w_{iN})'$ is the synaptic vector of the unit i .

Proof.

We have

$$\begin{aligned} d_H(\mathbf{x}, \mathbf{w}_i) &= (d_E(\mathbf{x}, \mathbf{w}_i)^2) = (\mathbf{x} - \mathbf{w}_i)^T (\mathbf{x} - \mathbf{w}_i) \\ &= \mathbf{x}^T \mathbf{x} - 2 \mathbf{x}^T \mathbf{w}_i + \mathbf{w}_i^T \mathbf{w}_i \\ &= \mathbf{x}^T \mathbf{x} - 2 h_i \end{aligned}$$

since $\mathbf{w}_i^T \mathbf{w}_i = w_{ii}$. Thus, the result is followed. \square

Next we present a learning process to determine the synaptic weights. First, we consider a set $C = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p\}$ of p inputs. We want to find M synaptic vectors to minimize the distortion function, that is,

$$E(\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_M) = \sum_{i=1}^M \sum_{j=1}^p d_H(\mathbf{x}_j, \mathbf{w}_i) \delta_i(\mathbf{x}_j, \mathbf{w}_1, \dots, \mathbf{w}_M) \quad (5)$$

where

$$\delta_i(\mathbf{x}_j, \mathbf{w}_1, \dots, \mathbf{w}_M) = \begin{cases} 1 & \text{if } d_H(\mathbf{x}_j, \mathbf{w}_i) = \min_{1 \leq r \leq M} \{d_H(\mathbf{x}_j, \mathbf{w}_r)\} \\ 0 & \text{otherwise} \end{cases}$$

In this way, synaptic vectors are the best representation of inputs data by M binary vectors. That is, the synaptic vectors must be medianoids of clusters of the input data.

The rule of simple competitive learning for continuous inputs leads to an update of the neural network so that its synaptic vectors are moved toward input vectors, that is, the new synaptic vector is a linear combination between the old synaptic vector and the input vector. However, the lineal combination of two binary vectors could be a no binary vector and so the new synaptic vector could be a no binary vector. It is necessary a new learning process such that the new synaptic vector becomes a binary vector at the same time that it also comes near to input vector. Thus, we propose a new learning rule based on *activity levels* to attain this goal. In each stage, we updates the components of the winner synaptic vector according to a learning rate $\eta_{ij}(k)$ that depends on the *activity level*. The *activity level* of the j th component of the synaptic weight \mathbf{w}_i after presentation of the k th training pattern, $\mathbf{x}(k) = (x_1(k), x_2(k), \dots, x_N(k))^T$, is given by

$$a_{ij}(k) = \sum_{s=1}^k \left(x_j(s) - \frac{1}{2} \right) y_i(s) = \sum_{s \in C_i(k)} \left(x_j(s) - \frac{1}{2} \right) \quad (5)$$

where $C_i(k) = \{s \in Z_+ : y_i(s) = 1, s \leq k\}$. It give us a balance between zeros and ones of the j th components of presentation patters with $y_i(s)=1, s \leq k$. That is to say, if $a_{ij}(k) > 0$ then the median of the set $\{x_j(s), s \in C_i(k)\}$ is one while if $a_{ij}(k) < 0$ then the median is zero. Note that

$$a_{ij}(k) = a_{ij}(k-1) + (x_j(k) - \frac{1}{2}) y_i(k). \quad (6)$$

The new learning rule is given by expression

$$\Delta w_{ij}(k+1) = w_{ij}(k+1) - w_{ij}(k) \quad (7)$$

$$= \eta_{ij}(k) (x_j(k) - w_{ij}(k)) \quad (8)$$

where

$$\eta_{ij}(k) = \begin{cases} 0 & \text{if } \operatorname{sgn}(a_{ij}(k)) = \operatorname{sgn}(a_{ij}(k-1)) \\ 1 & \text{otherwise} \end{cases}$$

That is, the weight w_{ij} is changed only when the unit i is winner and the sign of the *activation level* is modified by the new training pattern. Note that the expression (8) is similar to simple competitive rule but the learning parameter is now different. In this way, the synaptic weight vectors will become *medianoids* of inputs patters.

It is easy to prove that this learning rule guarantees that the new synaptic vector will be more similar to input vector than the old synaptic weight. It is established in the next proposition.

Proposition 2

$$d_H(\mathbf{w}_i(k+1), \mathbf{x}(k)) \leq d_H(\mathbf{w}_i(k), \mathbf{x}(k)), \quad i=1,2,\dots,M. \quad (9)$$

Proof:

It is obvious since

$$w_{ij}(k) = \begin{cases} x_j(k) & \text{if } \eta_{ij}(k) = 1 \\ w_{ij}(k) & \text{if } \eta_{ij}(k) = 0 \end{cases}$$

Thus

$$\begin{aligned} d_H(\mathbf{w}_i(k+1), \mathbf{x}(k)) &= \sum_{j: \eta_{ij}=0} |w_{ij}(k) - x_j(k)| \\ &\leq \sum_{j=1}^N |w_{ij}(k) - x_j(k)| \\ &= d_H(\mathbf{w}_i(k), \mathbf{x}(k)). \end{aligned}$$

Next we study the binary learning parameter η_{ij} . Let $\{\mathbf{x}(r), r=1,2,\dots,p\}$ be drawn independently according to a finite mixture distribution. Then $\eta_{ij}(k)$ is a random variable that depends on $\{x_j(s), s \in C_i(k)\}$. If $P(X_j(r)=1)=p, r \in C_i(k)$, then

$$\begin{aligned} P(\eta_{ij}(k)=1) &= P(a_{ij}(k) \neq a_{ij}(k-1)) \\ &= P\left(\sum_{r \in C_i} X_j(r) \geq \frac{|C_i|}{2}, \sum_{r \in C_i} X_j(r) + X_j(k) < \frac{|C_i|+1}{2}\right) \\ &\quad + P\left(\sum_{r \in C_i} X_j(r) < \frac{|C_i|}{2}, \sum_{r \in C_i} X_j(r) + X_j(k) \geq \frac{|C_i|+1}{2}\right) \\ &= \begin{cases} \binom{k}{k/2} p^{k/2} (1-p)^{1+k/2} & \text{if } k \text{ is even} \\ \binom{k}{(k-1)/2} p^{(k-1)/2} (1-p)^{(k+1)/2} & \text{if } k \text{ is odd} \end{cases} \quad (10) \end{aligned}$$

This probability tends to zero as k tend to infinite. So the learning parameter $\eta_{ij}(k)$ entails a convergence to zero. Like competitive neural networks, this network evolves until a local minimum is reached.

In this way we have the following algorithm:

Step 1: Initialization.

Generate M binary synaptic vectors $\mathbf{w}_1, \dots, \mathbf{w}_M$.

Step 2: k th iteration. Synaptic potentials.

Given a input vector, $(x_1(k), x_2(k), \dots, x_N(k))$ determine the synaptic potentials h_i , $i=1, 2, \dots, M$, by expression

$$h_i = \sum_{j=1}^N (x_j(k) - 0.5) w_{ij}$$

Step 3: Winner unit

The unit r is activated (winner), $y_r=1$, if $h_r \geq h_j$, $j=1, 2, \dots, M$.

Step 4: Update activity levels

$$a_{ij}(k) = a_{ij}(k-1) + (x_j(k) - \frac{1}{2}) y_i(k)$$

Step 4: Update synaptic weight

$$\Delta w_{ij}(k+1) = \eta_{ij}(k) (x_j(k) - w_{ij}(k))$$

and

$$\eta_{ij}(k) = \begin{cases} 0 & \text{if } \operatorname{sgn}(a_{ij}(k)) = \operatorname{sgn}(a_{ij}(k-1)) \\ 1 & \text{otherwise} \end{cases}$$

Paso 6: Stop rule.

If the learning parameters are equal to zero during an epoch of training, stop. In other case, go to step 2.

On the other hand, in an update in batch mode we only have to assign to each synaptic vector, \mathbf{w}_i , the *medianoid* vector of the set of inputs that activate unit i .

3. Experimental results: Codebook generation in vector quantization

The aim of this network is to cluster or categorize the input data. It can be used for data encoding and compression through vector quantization, where each input vector is replaced by the code of the winner output unit. In this application, the digital images to be encoded are decomposed into small blocks, say 3 by 3 pixels, called *vectors*. The resulting vectors are represented by the “nearest” of a reduced set of prototype vectors, called *codewords*. The set of codewords used to represent an image, or a portion of an image, is called *codebook*. The codebook is given by the synaptic weights of the network.

To illustrate the performance of the proposed algorithm, we consider a fingerprint image of size 255×255 (see figure 1(a)). It is partitioned in 7225 blocks (windows) of size 3×3 . The problem is to find 4 prototype blocks (windows) of size 3×3 such that when each block is represented by one of the prototype blocks then we obtain the best possible representation. That is, we want to minimize the distortion function and so the prototype blocks should be medianoids.

In our experiments, we use the *peak signal to noise ratio* (PSNR) to evaluate the quality of a compressed image. For an original image $F = \{f_{ij}\}$, $i=1, 2, \dots, m$,

$j=1,2,\dots,n\}$ and its corresponding compressed image $F'=\{f'_{ij}, i=1,2,\dots,m, j=1,2,\dots,n\}$, the PSNR is defined as follows:

$$\text{PSNR} = 10 \log_{10} \frac{1}{mn} \sum_{i=1}^m \sum_{j=1}^n (f_{ij} - f'_{ij})^2$$

where 1 is the peak grey level of the image, f_{ij} and f'_{ij} are the pixel grey levels from the original and compressed images, and $m \times n$ is the number of pixels in the image. In general, the higher PSNR value of an image implies the better image quality.

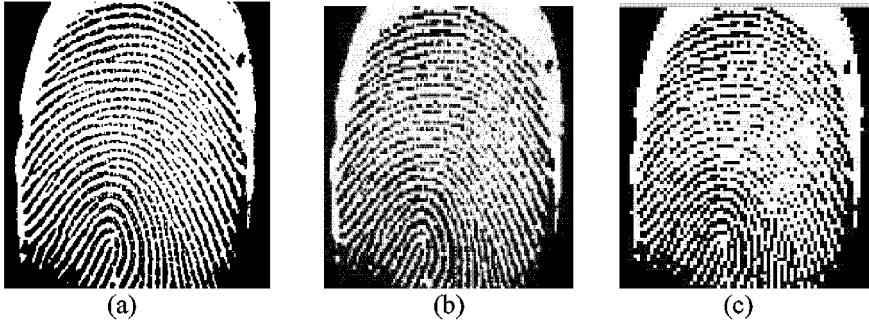


Fig. 1. (a) Fingerprint image. (b) Compressed image 9:2. (c) Compressed image 9:1.

The compressed image is constituted by prototype blocks, that is, medianoids, and when each block is replaced by its prototype block then we would obtain the most look like image to original image. In this way, we need only 2 bits to represent 4 prototype blocks while each block of the original image is represented by 9 bits, so the compression rate is about 9 to 2, that is, 2/9 bits per pixels (bpp). This problem is solved by a binary competitive network with 4 units. It finds the prototypes blocks

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \quad (10)$$

They correspond to 2851, 2886, 904 and 584 windows, respectively. The mean distortion error of a block is 0.8840 and the PSNR achieved was 10.07 dB (optimal solution). The figure 2(b) shows the compressed image. The compressed image retains features as such ridge lines, ridge bifurcation, arch, deltas, etc. Note that the new compressed image is formed by only 4 different blocks and so it can be compressed better than the original one by any other compression techniques. Like other algorithms, the algorithm also finds other solutions (local minimum), it depends by the initial set of synaptic weights, but they are optimal or near optimal with PSNR values around 10 dB. When only two prototype block are used then the neural networks has two process units and it always find the compressed image of the figure 1(c) that is only formed by the two left blocks in (10) with a PSNR equal to 8.79 dB. The compression rate is 9 to 1, that is, 1/9 bpp.

Although the experimental outcomes do not distinguish the proposed algorithm from the standard competitive learning algorithm and other algorithms such as the well-known generalized Lloyd algorithm, also referred as *k-means* algorithm (see [9]), however, it always uses binary synaptic weights and binary operations, and evolves from binary values to binary values, whereas other algorithms use real values. Hence, the proposed algorithm is more computationally efficient.

4. Conclusions

A binary competitive neural network has been proposed where synaptic weight vectors are binary vectors. The new synaptic weight vector is obtained by a learning mechanism that guarantees that it will be closer to input vector and at the same time that it will be also binary. First, we have shown that a necessary condition for an optimum solution to the problem to minimize the distortion function is that binary vectors have to be *medianoids*. In the proposed model each synaptic weight vector evolves to the *medianoid* vector of cluster that is being formed by process units of the network. Moreover, this model is more computationally efficient than the simple competitive model with continuous weights. Finally, the model has been applied to image compression and though it reaches a local minimum this is global or a good solution.

References

1. Grossberg, S. Adaptative pattern classification and universal recording: I. Parallel development and coding of neural detectors, *Biolog. Cybernetics*, **23** (1976) 121-134.
2. von der Malsburg, C. Self-organization of orientation sensitive cells in the striate cortex, *Kybernetik*, **14** (1973) 85-100.
3. Amari, S. and Takeuchi, M. Mathematical theory on formation of category detecting nerve cells, *Biological Cybernetics*, **29** (1978) 127-136.
4. Amari,S-I. Field theory of self-organizing neural nets, *IEEE Transactions on Systems, Man and Cybernetics, SMC-13* (1983) 741-748.
5. Biensstock, E., Cooper, E. & Munro, P. Theory for the development of neural selectivity: Orientation specificity and binocular interaction in visual cortex, *Journal of Neuroscience*, **2** (1982) 32-48.
6. Rumelhart, D. and Zipser, D. (1985). Feature discovery by competitive learning, *Cognitive Science*, **9** (1985) 75-112.
7. Carpenter, G. A. and Grossberg, S. A massively paralell architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing* **37**(1987) 54-115.
8. Hassoun M.H. *Fundamentals of Artificial Neural Networks*. The MIT Press, Cambridge, 1995.
9. Duda R.O., Hart P.E. and Stork D.G. *Pattern Classification*. John Wiley & Son, New York, 2001.

SASEGASA: An Evolutionary Algorithm for Retarding Premature Convergence by Self-Adaptive Selection Pressure Steering

Michael Affenzeller, Stefan Wagner

Institute of Systems Science
Systems Theory and Information Technology
Johannes Kepler University
Altenbergerstrasse 69
A-4040 Linz - Austria
ma@cast.uni-linz.ac.at

Abstract. This paper presents a new generic Evolutionary Algorithm (EA) for retarding the unwanted effects of premature convergence. This is accomplished by a combination of interacting methods. To be intent on this a new selection scheme is introduced, which is designed to maintain the genetic diversity within the population by advantageous self-adaptive steering of selection pressure. Additionally this new selection model enables a quite intuitive condition to detect premature convergence. Based upon this newly postulated basic principle the new selection mechanism is combined with the already proposed Segregative Genetic Algorithm (SEGA) [3], an advanced Genetic Algorithm (GA) that introduces parallelism mainly to improve global solution quality. As a whole, a new generic evolutionary algorithm (SASEGASA) is introduced. The performance of the algorithm is evaluated on a set of characteristic benchmark problems. Computational results show that the new method is capable of producing highest quality solutions without any problem-specific additions.

1 Introduction

Evolutionary Algorithms (EAs) may be described as a class of bionic techniques that imitate the evolution of a species. The most important representatives of EAs are Genetic Algorithms (GAs) and Evolution Strategies (ES).

The fundamental principles of GAs were first presented by Holland [6]. Since that time GAs have been successfully applied to a wide range of problems including multimodal function optimization, machine learning, and the evolution of complex structures such as neural networks. An overview of GAs and their implementation in various fields is given by Goldberg [5] or Michalewicz [9].

Evolution Strategies, the second major representative of EAs, were introduced by Rechenberg [10] and Schwefel [13]. Applied to problems of combinatorial optimization, ES tend to find local optima quite efficiently. Though, in the

case of multimodal test functions, global optima can be detected by ES only if one of the starting values is located in the absorbing region of a global optimum.

The advantage of applying GAs to hard problems of combinatorial optimization lies in the ability to search the solution space in a broader way than heuristic methods based upon neighborhood search. Nevertheless, also GAs are frequently faced with a problem which, at least in its impact, is quite similar to the problem of stagnating in a local but not global optimum. This drawback, called premature convergence in the terminology of GAs, occurs when the population of a GA reaches such a suboptimal state that the genetic operators can no longer produce offspring that outperform their parents (e.g. [4]).

Inspired by Rechenberg's 1/5 success rule for Evolution Strategies, we have developed an advanced selection model for Genetic Algorithms that allows self-adaptive control of selection pressure in a quite intuitive way. Based upon this enhanced EA-model further generic extensions are being discussed.

The experimental part analyzes the new algorithms for the Traveling Salesman Problem (TSP) as a very well documented instance of a multimodal combinatorial optimization problem. In contrast to all other evolutionary heuristics known to the authors that do not use any additional problem specific information, we obtain the best known solution for all considered benchmarks.

2 The Self-Adaptive Selection Model

As there exists no manageable model for a controllable handling of selection pressure within the theory of GAs[12], we have introduced some kind of intermediate step (a 'virtual population') into the selection process which provides a handling of selection pressure very similar to that of ES [2]. As we have exemplarily pointed out in contribution [2], the most common replacement mechanisms can easily be implemented in this intermediate selection step. Furthermore, this Evolution Strategy like variable selection pressure supported us to steer the degree of population diversity. However, within this model it is necessary to adjust a parameter for the actual selection pressure and in order to steer the search process advantageously a lot of parameter tuning is essential.

Motivated by those considerations we have set up an advanced selection model by introducing a new criterion abutted on Rechenberg's 1/5 success rule. The first selection step chooses the parents for crossover in the well-known way of Genetic Algorithms by roulette wheel, linear-rank, or some kind of tournament selection strategy. After having performed crossover with the selected parents we introduce a further selection mechanism that considers the success of the applied crossover in order to assure the proceeding of genetic search mainly with successful offspring in that way that the used crossover operator was able to create a child that surpasses its parents' fitness.

In doing so, a new parameter, called success ratio ($SuccRatio \in [0, 1]$), gives the quotient of the next population members that have to be generated by successful mating in relation to the total population size. Our adaptation of Rechenberg's success rule for GAs says that a child is successful if its fitness is

better than the fitness of its parents, whereby the meaning of 'better' has to be explained in more detail: Is a child better than its parents, if it surpasses the fitness of the weaker, the better, or is it some kind of mean value of both? For this problem we have decided to introduce a Simulated Annealing (SA) like cooling strategy. Following the basic principle of SA we claim that a successful descendent has to surpass the fitness value of the worse parent at the beginning and while evolution proceeds the child has to be better than a fitness value continuously increasing in the range between the fitness of the weaker and the better parent. Like in the case of SA this strategy effects a broader search at the beginning whereas at the end of the search process this operator acts in a more and more directed way. Having filled up the claimed ratio (*SuccRatio*) of the next generation with successful individuals in the above meaning we simply fill up the rest of the next generation ($(1 - \text{SuccRatio}) * |\text{POP}|$) with individuals arbitrarily chosen from the pool of individuals that were also created by crossover but did not reach the success criteron. The actual selection pressure *ActSelPress* at the end of a single generation is defined by the quotient of individuals that had to be considered until the success ratio was reached and the number of individuals in the population $\text{ActSelPress} = \frac{|\text{virtual POP}| + \text{SuccRatio} * |\text{POP}|}{|\text{POP}|}$. Fig. 1 shows the operating sequence of the above described concepts.

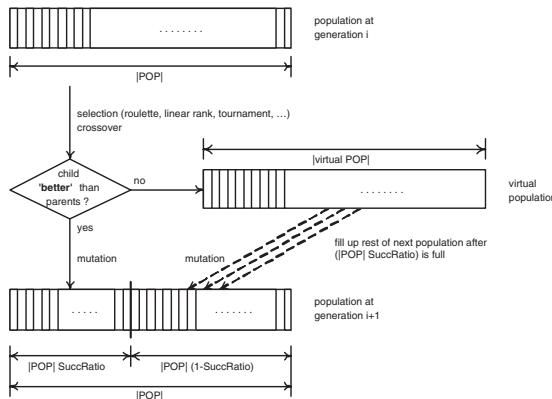


Fig. 1. Flowchart for embedding the new selection principle into a GA

With an upper limit of selection pressure (*MaxSelPress*), defining the maximum number of children considered for the next generation that may be produced in order to fulfill the success ratio, this new model also acts as a precise detector of premature convergence: If it is not possible anymore to find enough (*SuccRatio* * $|\text{POP}|$) children outperforming their own parents, even if (*MaxSelPress* * $|\text{POP}|$) candidates have been generated premature convergence has occurred.

As a basic principle of this selection model a higher success ratio causes higher selection pressure. Nevertheless, higher settings of success ratio and therefore of

selection pressure do not essentially cause premature convergence. This is because per definition the new selection step (after crossover) does not accept clones that emanate from two identical parents. In traditional GAs such clones represent a major reason for premature convergence of the whole population around a suboptimal value, whereas the new selection step specifically counteracts against this phenomenon.

Experiments performed on the variable selection pressure model already indicate the supremacy of this approach. Also the corresponding canonical Genetic Algorithm is fully included in our new superstructure if the success ratio is simply set to 0. Furthermore, we are going to discuss new aspects and models built up upon the described self adaptive selection model In the following section.

3 Generic GA-Concepts Based Upon the Self-Adaptive Selection Model

When applying Genetic Algorithms to complex problems, one of the most frequent difficulties is premature convergence. Concisely speaking, premature convergence occurs when the population of a Genetic Algorithm reaches such a suboptimal state that the genetic operators can no longer produce offspring that outperform their parents (e.g. [4]).

A critical problem in studying premature convergence is the identification of its occurrence and the measure of its extent. Srinivas and Patnaik [15], for example, use the difference between the average and maximum fitness as a standard to measure premature convergence and adaptively vary the crossover and mutation probabilities according to this measurement. On the other hand, as in the present paper, the term 'population diversity' has been used in many papers to study premature convergence (e.g. [14]) where the decrease of population diversity is considered as the primary reason for premature convergence.

The following generic extensions, that are built up upon the self-adaptive variable selection pressure model, aim to retard premature convergence in a general way.

3.1 SASEGASA: The Core Algorithm

In principle, the new SASEGASA (Self Adaptive SEgregative Genetic Algorithm with Simulated Annealing aspects) introduces two enhancements to the basic concept of Genetic Algorithms. Firstly, we bring in a variable selection pressure, as described in section 2, in order to control the diversity of the evolving population. The second concept introduces a separation of the population to increase the broadness of the search process and joins the subpopulation after their evolution in order to end up with a population including all genetic information sufficient for locating a global optimum.

The aim of dividing the whole population into a certain number of subpopulations (segregation) that grow together in case of stagnating fitness within those subpopulations (reunification) is to combat premature convergence which is the

source of GA-difficulties. This segregation and reunification approach is an efficient method to overcome premature convergence [1] called the SEGA algorithm (SEgregative GA).

The principle idea of SEGA is to divide the whole population into a certain number of subpopulations at the beginning of the evolutionary process. These subpopulations evolve independently from each other until the fitness increase stagnates in all subpopulations because of too similar individuals within the subpopulations, i.e. local premature convergence. Then a reunification from n to $(n - 1)$ subpopulations is performed by joining an appropriate number of adjacent subpopulation members.

Metaphorically speaking this means, that the a certain number of villages (subpopulations) at the beginning of the evolutionary process are slowly growing together to bigger cities, ending up with one big town containing the whole population at the end of evolution. By this approach of width-search essential building blocks can evolve independently in different regions of the search space at the beginning and during the evolutionary process. In the case of a standard GA those building blocks are likely to disappear early and, therefore, their genetic information can not be provided at a later phase of evolution, when the search for a global optimum is of paramount importance.

Within the classical SEGA algorithm there is no criterion to detect premature convergence and there is also no self-adaptive selection pressure steering mechanism. Even if the results of SEGA are quite good with regard to global convergence it requires an experienced user to adjust the selection pressure steering parameters and as there is no criterion to detect premature convergence the dates of reunification have to be implemented statically.

Equipped with our new self adaptive selection technique we have both: A self-adaptive selection pressure (depending on the given success ratio) as well as an automated detection of local premature convergence, if the current selection pressure becomes higher then the given maximal selection pressure parameter (*MaxSelPress*). Therefore, a date of reunification has to be set, if local premature convergence has occurred within all subpopulations in order to increase genetic diversity again.

Again, like in the context of the new selection model which is included in SASEGASA as well, it should be pointed out that a corresponding Genetic Algorithm is unrestrictedly included in SASEGASA, when the number of subpopulations (villages) is set to 1 and the success ratio is set to 0 at the beginning of the evolutionary process. Moreover, the introduced techniques also do not use any problem specific information.

4 Experimental Results

Empirical studies with different problem classes and instances are the most effective way to analyze the potential of heuristic optimization searches like Evolutionary Algorithms.

In our experiments, all computations are performed on a Pentium 4 PC with 1 GB RAM. The programs are written in the C# programming language. For the tests we have selected the Traveling Salesman Problem (TSP) as a well documented instance of a typical multimodal combinatorial optimization problem. We have tested the new concepts on a selection of symmetric as well as asymmetric TSP benchmark problem instances taken from the TSPLIB [11] using updated results for the best or at least the best known solutions. In all experiments, the results are represented as the relative difference to the best known solution defined as $\text{relativeDifference} = (\frac{\text{Result}}{\text{Optimal}} - 1) * 100\%$.

Especially we aim to point out the main effect of the present contribution - namely that an increasing number of subpopulations at the beginning of the evolutionary process allows to achieve scalable improvements in terms of global convergence. As Tab. 1 shows, the global solution can be scaled up to highest quality by just increasing the number of evolving subpopulations. This definitely represents a new achievement in the area of Evolutionary Algorithms with distributed subpopulations. For all the experiments in Tab. 1 the starting size of one subpopulation is fixed at a value of 100, mutation rate is 5%, and the upper limit of selection pressure is set to a value of 10 for the TSPs respectively 15 for the ATSPs.

The results in Tab. 1 present the best solution quality of five runs of each test-instance as well as the average best result-value of the five runs in terms of average difference to the best known solution.

Indeed, as Tab. 1 shows, the optimal solution could be found for all benchmark test cases if the initial number of subpopulations is set high enough.

Even if the achieved results are clearly superior to most of the results reported for applications of Evolutionary Algorithms to the TSP [8], it has to be pointed out again, that all introduced and applied additions to a standard evolutionary algorithm are generic and absolutely no problem specific local pre- or post-optimization techniques have been applied in our experiments. Additional experiments performed on non-standardized scheduling problems (job-shop and multiprocessor) show comparable potential and underscore the generic potential of the new techniques in various fields of applications.

5 Conclusion

In this paper an enhanced Genetic Algorithm and two upgrades have been presented and exemplarily tested on some TSP benchmarks. The proposed EA-based techniques couple aspects from Evolution Strategies (selection pressure and success rule in our new selection procedure), Simulated Annealing (growing selective pressure) as well as a special segregation and reunification strategy with crossover and mutation in the general model of a Genetic Algorithm. Therefore, established crossover and mutation operators for certain problems may be used analogously to the corresponding Genetic Algorithm. The investigations in this paper have mainly focused on the avoidance of premature convergence and on

Table 1. Experimental results of the SASEGASA-algorithm for TSPLIB benchmark problems with variable number of subpopulations for tuning global solution quality.

Problem	Parameters and Operators				Results (in %)	
	<i>noOfSubPopulations</i>	SuccessRatio	Crossover	Iterations	Best	Average
berlin52	1	0,8	OX,ERX,COSA	139	6.7	8.6
berlin52	5	0,8	OX,ERX,COSA	218	0.0	0.0
berlin52	10	0,8	OX,ERX,COSA	301	0.0	0.0
ch130	1	0,8	OX,ERX,COSA	295	52.9	59.4
ch130	5	0,8	OX,ERX,COSA	584	12.2	13.24
ch130	10	0,8	OX,ERX,COSA	783	4.8	6.11
ch130	20	0,8	OX,ERX,COSA	1024	1.6	2.5
ch130	40	0,8	OX,ERX,COSA	1426	0.63	0.89
ch130	80	0,8	OX,ERX,COSA	2067	0.45	0.74
ch130	160	0,8	OX,ERX,COSA	3518	0.0	0.15
kroA200	1	0,8	OX,ERX,COSA	584	77.6	87.3
kroA200	5	0,8	OX,ERX,COSA	1035	22.5	24.9
kroA200	10	0,8	OX,ERX,COSA	1310	12.4	13.1
kroA200	20	0,8	OX,ERX,COSA	1604	4.7	7.4
kroA200	40	0,8	OX,ERX,COSA	2243	0.9	2.6
kroA200	80	0,8	OX,ERX,COSA	2842	0.6	1.3
kroA200	160	0,8	OX,ERX,COSA	4736	0.0	0.3
rbg323	1	0,8	OX,ERX,COSA	1463	20.59	26.24
rbg323	5	0,8	OX,ERX,COSA	2690	4.37	8.02
rbg323	10	0,8	OX,ERX,COSA	5991	2.11	2.84
rbg323	20	0,8	OX,ERX,COSA	13456	0.30	0.53
rbg323	40	0,8	OX,ERX,COSA	40762	0.15	0.20
rbg323	80	0,8	OX,ERX,COSA	100477	0.00	0.06
rbg323	160	0,8	OX,ERX,COSA	212418	0.00	0.00

the introduction of methods which make the algorithm more open for scalability in terms of solution quality versus computation time.

Concerning the speed of SASEGASA, it has to be pointed out that the superior performance concerning global convergence requires a higher computation time, mainly because of the greater total population size $|POP|$ and because of the increase of generated individuals in our new self adaptive selection mechanism under higher selection pressure. Nevertheless, in contrast to other implementations in the field of evolutionary computation, it is absolutely remarkable, that it has become possible to almost linearly improve the global solution quality by introducing greater population sizes and an accordingly greater number of subpopulations, whereas the computational costs are 'only' increasing linearly due to the greater number of individuals having to be taken into account. This allows to transfer already developed GA-concepts to increasingly power-

ful computer systems in order to achieve better results. Using parallel computer architectures seems especially suited to increase the performance of SASEGASA.

Anyway, with special parameter settings the corresponding Genetic Algorithm is fully included within the introduced concepts achieving a performance only marginally worse than the performance of the equivalent Genetic Algorithm. In other words, the introduced models can be interpreted as a superstructure of the GA model or as a technique upwards compatible to Genetic Algorithms. Therefore, an implementation of the new algorithm(s) for a certain problem should be quite easy to do, presumed that the corresponding Genetic Algorithm (coding, operators) is known.

References

1. Affenzeller, M.: A New Approach to Evolutionary Computation: Segregative Genetic Algorithms (SEGA). Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence, Lecture Notes of Computer Science 2084 (2001) 594–601
2. Affenzeller, M.: Transferring the Concept of Selective Pressure from Evolutionary Strategies to Genetic Algorithms. Proceedings of the 14th International Conference on Systems Science 2 (2001) 346–353
3. Affenzeller, M.: Segregative Genetic Algorithms (SEGA): A Hybrid Superstructure Upwards Compatible to Genetic Algorithms for Retarding Premature Convergence. International Journal of Computers, Systems and Signals (IJCSS), Vol. 2, Nr. 1 (2001) 18–32
4. Fogel, D.B.: An Introduction to Simulated Evolutionary Optimization. IEEE Trans. on Neural Networks 5(1) (1994) 3–14
5. Goldberg, D. E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley Longman (1989)
6. Holland, J. H.: Adaptation in Natural and Artificial Systems. 1st MIT Press ed. (1992)
7. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by Simulated Annealing. Science 220 (1983) 671–680
8. Larranaga, P., Kuijpers, C.M.H., Murga, R.H., Inza, I., Dizdarevic, S.: Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. Artificial Intelligence Review 13 (1999) 129–170
9. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (1996)
10. Rechenberg, I.: Evolutionsstrategie. Friedrich Frommann Verlag (1973)
11. Reinelt, G.: TSPLIB - A Traveling Salesman Problem Library. ORSA Journal on Computing 3 (1991) 376–384
12. Schneburg, E., Heinzmann, F., Feddersen, S.: Genetische Algorithmen und Evolutionsstrategien. Addison-Wesley (1994)
13. Schwefel, H.-P.: Numerische Optimierung von Computer-Modellen mittels Evolutionsstrategie. Birkhäuser Verlag, Basel (1994)
14. Smith, R.E., Forrest, S., Perelson, A.S.: Population Diversity in an Immune System Model: Implications for Genetic Search. Foundations of Genetic Algorithms 2 (1993) 153–166
15. Srinivas, M., Patnaik, L.: Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms . IEEE Transactions on Systems, Man, and Cybernetics 24(4) (1994) 656–667

Cooperative Ant Colonies for Solving the Maximum Weighted Satisfiability Problem

Habiba Drias, Amine Taibi, and Sofiane Zekour

USTHB, Computer Science Department,
Laboratory of Research in Artificial Intelligence
BP 32 El-Alia Bab-Ezzouar, 16 111 Algiers, Algeria
Drias@wissal.dz

Abstract. This paper introduces the ant colonies approach for the maximum weighted satisfiability problem, namely MAX-W-SAT. We describe an ant colonies algorithm for MAX-W-SAT called AC-SAT and provide an overview of the results of the empirical tests performed on the hard Johnson benchmark. A comparative study of the algorithm with well known procedures for MAX-W-SAT is done and shows that AC-SAT outperforms the other evolutionary meta-heuristics especially the scatter search, which has been developed recently.

1 Introduction

Many NP-complete problems are better solved now than in the past. One undeniable reason of the improvement in solutions quality and running time performance is the development of an important number of new powerful meta-heuristics. If the problems sizes have been very limited in getting a reasonable execution time in the nearest past, larger ones are more manageable nowadays. They have increased in a tremendous way mostly because of the competition of new approaches and the recent physical machines performances.

The algorithm A* has marked the beginning of the introduction of heuristics in solving complex problems. Since then a plethora of meta-heuristics approaches often inspired from natural phenomena have been developed. Simulated annealing, genetic algorithms, taboo search and scatter search are examples of these methods.

The ant colonies or AC for short is another meta-heuristic mimicking the real behavior of ants colonies when searching for food. The approach has gained a large reputation since it has solved with success many combinatorial optimization problems like the travelling salesman problem or TSP[4], the quadratic assignment problem or QAP[8], the vehicle routing problem or VRP[1] and the job shop scheduling problem or JSSP[3].

The maximum weighted satisfiability problem or MAX-W-SAT is an NP-Hard optimization problem. It is central in both computational complexity theory and artificial intelligence discipline. In addition, it has an important number of applications, especially in knowledge bases of learning systems and VLSI technology. SAT and many of its variants have been widely and constantly studied these last two decades. Competitive general purpose algorithms based on meta-heuristics like genetic algorithms, taboo search and scatter search are known for this problem. In this

paper, the AC approach is proposed for this problem. An algorithm has been designed and tested on the real-life Johnson problems. Numerical results are compared with those of other well known approaches and in particular with scatter search.

2 The MAX-W-SAT Problem

The satisfiability problem or SAT in abbreviation is a problem of logic. Its importance comes from the fact that logic is a theoretical tool used in problem solving and many other domains. SAT is defined to answer the question whether there exists an interpretation of variables that satisfies a logical formula written in the setting of propositional or first order calculus. Since every well formed formula can be converted in a conjunction of disjunctions of literals, an instance of SAT is often presented in this form. More precisely, an instance of SAT is a set of clauses that are disjunctions of literals, a literal is defined to be a Boolean variable with or without a negation. When a valuation of variables satisfies all the clauses of the instance, we say that the instance is satisfiable. Otherwise, we say that the data is contradictory. In this situation, another question arises: find an assignment that satisfies the maximum number of clauses. This problem is known as the maximum satisfiability problem or MAX-SAT. When weights are associated to clauses, the goal is to find the assignment that maximizes the sum of weights of the clauses that are simultaneously satisfied. This problem is named maximum weighted satisfiability problem or MAX-W-SAT. The following example is an instance of three clauses c_1 , c_2 and c_3 having each, three literals built over five Boolean variables x_1 , x_2 , x_3 , x_4 and x_5 .

$$\begin{aligned} c_1 &= x_1 + \neg x_3 + x_4 \\ c_2 &= x_3 + \neg x_4 + \neg x_5 \\ c_3 &= \neg x_2 + x_3 + \neg x_5 \end{aligned}$$

The plus sign denotes the Boolean disjunction operator while the minus expresses the unary negation.

3 The Ant Colonies Optimization

The ant colonies optimization meta-heuristic or ACO has revealed its efficiency for many combinatorial optimization problems. It is based on a simple mechanism of communication between artificial agents imitating the behavior of real ants. The ants are capable of cooperating in searching in a collective and asynchronous manner an optimal path linking the food source to their nest. They start exploring the food source in a random way. When an ant finds food, it deposits on its way back to the nest, a hormonal substance called pheromone, inciting and guiding the other ants to get straightly to the food source.

The ant colony metaphor is based on artificial ants that build the solutions in a progressive manner. They take their itinerary through a search space according to probabilistic decision rules. When a solution is found, it is partially evaluated and

the corresponding pheromone information is updated. A concise translation of the ant colonies behavior can be described by the following algorithm framework:

```

Begin
    initialize the pheromone;
    repeat for each ant until stop condition
        begin
            generate at random a solution;
            build the solution using the pheromone;
            update the pheromone;
        end
    end

```

4 Solving MAX-W-SAT with ACO

The adaptation of the meta-heuristic for our problem requires the design of the following components: the artificial world where the ants live, the fitness function that evaluates solutions, the probabilistic rules that direct the ants moves and the strategy of updating the pheromone.

The artificial world or search graph. According to the nature of MAX-W-SAT problem, the search space is a hyper-cubic graph, $G = (V, E)$ such that:

- $V = \{0,1\}^n$ is a finite set of vertices, each vertex corresponds to a truth assignment of variables, the total number of these vertices is equal to 2^n .
- $E = \{(x,y) \text{ such that } x \text{ and } y \in V \text{ and have only one different bit}\}$, the total number of the edges is equal to $n2^{n-1}$.

In this graph, the vertices corresponding to optimal solutions, represent the food sources for the real ants whereas the vertices corresponding to initial solutions, represent the nests. The artificial ants are supposed to cross a part of the graph in order to find the shortest path existing between initial and optimal solutions. An optimal solution is a vertex denoting an assignment that maximizes the fitness function. The artificial agents cooperate in order to emerge the optimal solution.

The fitness function. The fitness function measures the solution quality. In SAT, this function counts the number of clauses satisfied by the solution whereas, in MAX-W-SAT, it computes the weights sum of the clauses satisfied simultaneously.

The pheromone structure. The artificial ants store the pheromone on the literals. The pheromone information is structured as a table of two dimensions, one for the variables and the other for the possible values that can be taken by these variables. Each time the pheromone is updated, it is stored in this table.

The improvement method. Our ants colonies approach is hybridized with an improvement method in order to increase the algorithm performance. The improvement technique is applied to the initial solutions and the generated ones to enhance their quality. Our improvement method consists in choosing the best solution in the neighborhood of the current solution, a neighbor is obtained by

just flipping a bit. This operation is repeated until no improvement in the evaluation function is observed.

The overall procedure. The ACO procedure or AC-SAT is summarized as follows:

```

Procedure AC-SAT;
Begin
    Initialize the pheromone table using a heuristic;
    for (Iter = 1 to max-Iteration) do
        Begin For each ant k= 1 to nb-ants do
            Begin
                Generate at random a solution s;
                s' = build-sol(s); (*Build the solution using
                the pheromone*)
                s'' = Improve the solution s';
                if f(s'') > f(best) then best = s'';
            End;
            Update the pheromone;
        End;
    End;
End;
```

where Max-itcr and nb-ants are empirical parameters representing respectively the maximum number of iterations and the total number of ants. The process starts generating an artificial ants population and initializing the pheromone. Each ant has to construct a solution using the pheromone information initialized by the process or communicated by all the other ants afterwards. Once a solution is built, it is improved by means of a local search and the pheromone is updated. This phenomena simulates the process of knowledge acquisition and the forgetting of the past. The whole process is iterated until the optimal solution is found or after a certain number of iterations dictated by the physical machine limits. Let consider the following instance to better understand the main instructions of the algorithm:

$$\begin{array}{lll}
 c_1 = x_1 + x_3 & w_1=3 & c_4 = x_2 + x_4 & w_4=1 \\
 c_2 = x_1 + x_4 & w_2=4 & c_5 = -x_1 & w_5=2 \\
 c_3 = x_2 + x_4 & w_3=2 & c_6 = -x_2 & w_6=3
 \end{array}$$

The pheromone table called *phero* is a matrix of two dimensions. The lines are indexed by the problem variables and the column specifies the positive or negative form of the literal. An entry in the table reports the pheromone quantity brought by the literal. The table is initialized as follows:

$$\text{Phero}[x_i, k] = \frac{\sum \text{weight}(c_j)}{\sum \text{weight}(c_l)}$$

where c_l is any clause and c_j a clause containing x_i if $k = 1$ and $-x_i$ if $k = 0$. The initialization of the pheromone table for this example is shown in Table 1.

Table 1. Initialization of the pheromone table for the example

	0	1
x_1	2/15	7/15
x_2	3/15	3/15
x_3	0	3/15
x_4	0	7/15

The probabilistic decision rules. The solution is built by an ant according to the pheromone information. In fact, a certain number of bits of the current solution will change according to the following formula:

$$P(x_i=k) = \frac{\text{Phero}[x_i, k]}{\text{Phero}[x_i, 0] + \text{Phero}[x_i, 1]}$$

That computes the probability that a literal x_i takes the value k . A random number is generated and compared to this probability to deduce the value of the bit. The procedure that constructs a solution is as follows:

```
Procedure build-sol (s: assignment): assignment;
begin
    let s = ( $x_1, x_2, \dots, x_n$ );
    for j:= 1 to max-changes do
        begin draw at random a variable  $x_i$ ;
            (* apply probabilistic decision rule *)
            compute  $P(x_i=0)$ ;
            generate a random number r such that  $0 \leq r \leq 1$ ;
            if  $r \leq P(x_i=0)$  then  $x_i = 0$  else  $x_i = 1$ ;
        end;
    return s;
end;
```

The total number of bit changes is denoted by max-changes and is set by empirical tests.

The pheromone updating. The delayed pheromone update strategy is adopted, that is it is undertaken once the ants finish their search cycle. The evanescence process is first simulated by decreasing the pheromone values according to the following formula:

$$\text{Phero}[x_i, j] = (1-\alpha) \text{phero}[x_i, j] \quad \text{for } i=1..n \text{ and } j \in \{0, 1\}$$

Where $0 < \alpha < 1$ is the fourth empirical parameter representing the pheromone rate. Then the pheromone is reinforced by the quality of the solution found such that each ant contribution is inversely proportional to the solution evaluation. This pheromone is computed as follows:

$$\text{Phero}[x_i, j] = (1-\alpha) \text{phero}[x_i, j] + \alpha \frac{1}{1+f(s)} \times \frac{f(\text{worst})-f(s)}{f(\text{worst})-f(\text{best})}$$

where worst and best are respectively the worst and the best solution. α allows the control of the pheromone quantity added. When it is close to 1, the ants become unstable because of the lack of the pheromone. However when it is close to 0, the solutions stagnate and do not progress toward the optimal ones.

5 Numerical Results

The procedure AC-SAT has been implemented in C on a Pentium personal computer and numerical tests have been performed on the benchmark instances available on the web site <http://www.research.att.com/~mgcr/data/index.html>. These hard problems translating real-life problems have been converted from the Johnson class of the second DIMACS Challenge implementations[11]. The weights of the clauses have been drawn from 1 to 1000 and assigned at random to clauses, the number of clauses being ranged from 800 to 950. The Johnson class namely ‘jnh’, has been used in many works for testing algorithms performance. It includes three subclasses characterized by the variables number, which is equal to 100 for all instances. The clauses number is

- 800 for the subclass 1: Jnh201 ..Jnh220
- 850 for the subclass 2 : Jnh01..jnh19
- 900 for the subclass 3 : Jnh301..Jnh310

Each subclass contains instances that are satisfied or not. On each instance, 10 executions have been undertaken. The solutions quality as well as the running time have been considered as performance criteria.

5.1 Parameters setting

Preliminary tests have been carried out in order to fix the key parameters of the AC-SAT algorithm. Fig. 1 shows an example of the results of the tests done on some Johnson problems for setting the parameter max-iter. Identical tests have been performed on all the other instances and for all the other parameters. Table 2 summarizes the parameters values obtained after these extensive experiments.

Table 2. Empirical parameters values for AC-SAT

Pop-size	6
α	0.1
maxchanges	[40,60]
Max-iter	30

5.2 Performance comparison

Comparison of AC-SAT with SS-SAT, GRASP and the optimal solution is done. GRASP is a parallel greedy algorithm proposed for solving MAX-W-SAT[11]. It has been considered in this comparison because of its high efficiency. SS-SAT is another evolutionary algorithm based on scatter search approach and recently developed [6].

Table.3 contains for each Johnson problem, the optimal and the best solutions found respectively by AC-SAT, SS-SAT and GRASP. The bold numbers are the best values found. Their number is clearly more important for AC-SAT. This result translates the fact that AC-SAT outperforms SS-SAT and also GRASP for most instances of problems and its solutions are very close to the optimum. The execution time of AC-SAT is also very competitive and more interesting than that of SS-SAT. A schematic view of these results is shown through the curve of fig.2. The same observation can be done on the performance of AC-SAT.

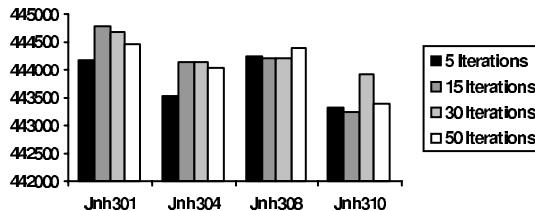


Fig. 1. The results of tests for setting the parameter max-iter

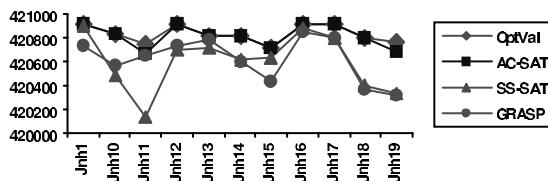


Fig. 2. A schematic comparison between ac-sat, ss-sat, grasp and the optimal solution for the subclass jnh01-Jnh19

6 Conclusion

In this paper, an ant colony algorithm called AC-SAT has been designed and implemented for the maximum satisfiability problem. From the practical viewpoint and on the basis of the empirical results, AC-SAT outperforms the scatter search algorithm recently developed and the best version of GRASP. The robustness of the AC method relies on the design of the solution construction procedure based on probabilistic decision rules, the pheromone updating strategy and its hybridization with an improvement technique. For the time being, we are studying and experimenting the parallelization of the AC approach for MAX-W-SAT in order to achieve increase in performance in terms of solutions quality and running time. The results will be communicated in the near future.

References

1. Bullnheimer, B., Hartl, R.F., Strauss, C.: Applying the ant system to the vehicle routing problem, in 2nd Metaheuristics Int Conf, MIC'97, Sophia-Antipolis, France, (July 1997)
2. Dorigo, M., Caro, G.D., Gambardella, L.M.: Ant Algorithms for discrete optimization, Tech Rep iridia/98-10, Université libre de Bruxelles, 1998
3. Colorni, A., Dorigo, M., Maniezzo, V., Trubian, M.: Ant system for Job-Shop Scheduling, Jorbel, 34(1), 39-53, (1994)

4. Dorigo, M., Gambardella, L.M.: Ant Algorithms for the Traveling Salesman Problem, Biosystems, 43, 73-81, (1997)
5. Dorigo, M., Caro, G.D.: Ant Colony Optimization : A New meta heuristic, IEEE, (1999).
6. Drias, H. , 'Scatter search with random walk strategy for solving hard MAX-W-SAT problems', in proc of IEA-AIE2001, LNAI 2070, Springer , Budapest, 35-44, (2001).
7. Drias, H. , 'Genetic Algorithm versus Scatter search and solving hard MAX-W-SAT problems', in proc of IWANN'2001, LNCS 2084, (2001) Granada, vol 1 586-593
8. Gambardella, L.M., Taillard, E., Dorigo, M.: Ant Algorithms for the QAP, Technical Report 97-4, IDSIA, Lugano, Switzerland, (1997)
9. Glover, F., Laguna M., Marti, R.: scatter search, internet document, (2000).
10. Hansen P., Jaumard, B.: Algorithms for the Max-SAT , comp 44, 279-303, (1990).
11. Resende, M.G., Pitsoulis L.S., Pardalos, P.M.: Approximate Solution of Weighted MAX-SAT Problems using GRASP, in Dimacs series on DM and TCS vol 35, 393-405, (1997).

Table 3. Empirical results comparing ac-sat, ss-sat, grasp and the optimal solutions

data	Opt sol	Ac-sat	ac-sat time (s)	ss-sat	ss-sat time (s)	Best of grasp
Jnh01	420925	420925	107.15	420892	203.99	420737
Jnh10	420840	420840	179.25	420479	901.47	420565
Jnh11	420753	420674	200.15	420141	439.82	420642
Jnh12	420925	420925	124.65	420701	250.65	420737
Jnh13	420816	420816	192.94	420716	930.12	420783
Jnh14	420824	420824	237.23	420616	298.22	420592
Jnh15	420719	420719	240.26	420632	624.63	420429
Jnh16	420919	420914	297.18	420889	716.51	420851
Jnh17	420925	420925	222.71	420794	401.95	420807
Jnh18	420795	420795	210.71	420404	129.64	420372
Jnh19	420759	420680	204.75	420330	403.10	420323
Jnh201	394238	394238	162.56	394238	161.72	394238
Jnh202	394170	394170	184.26	393752	269.99	393983
Jnh203	394199	394135	191.78	393876	294.10	393889
Jnh205	294238	394238	283.75	394060	203.37	394224
Jnh207	394238	394237	122.36	394107	313.58	394101
Jnh208	394159	394159	186.71	393560	137.19	393987
Jnh209	394238	394238	186.20	394238	395.38	394031
Jnh210	394238	394238	219.12	394067	394.46	394238
Jnh211	393979	393979	187.56	393742	408.04	393739
Jnh212	394238	394227	188.78	394082	758.00	394043
Jnh214	394163	394163	193.67	394152	1159.15	393737
Jnh215	394150	394150	184.37	393942	202.51	393858
Jnh216	394226	394226	189.13	393933	391.10	394042
Jnh217	394238	394238	144.6	394238	799.40	394232
Jnh218	394238	394238	217.12	394238	755.92	394099
Jnh219	394156	394070	130.45	393942	236.38	393792
Jnh220	394238	394238	166.23	393985	430.44	394053
Jnh301	444854	444807	204.4	444842	1267.63	444670
Jnh302	444459	444459	260.2	443895	698.77	444248
Jnh303	444503	444457	222.32	444223	437.91	444244
Jnh304	444533	444533	310.11	444533	1175.98	444310
Jnh305	444112	443970	330.28	443594	463.33	444112
Jnh306	444838	444838	321.46	444515	604.53	444658
Jnh307	444314	444314	231.45	443662	128.35	444159
Jnh308	444724	444621	116.15	444250	270.98	444222
Jnh309	444578	444578	211.02	444483	662.58	444349
Jnh310	444391	444353	247.29	444313	132.63	444282

Designing a Phenotypic Distance Index for Radial Basis Function Neural Networks

Jesús González, Ignacio Rojas, Héctor Pomares and Julio Ortega

Department of Computer Architecture and Computer Technology
University of Granada

Abstract. MultiObjective Evolutionary Algorithms (MOEAs) may cause a premature convergence if the selective pressure is too large, so, MOEAs usually incorporate a niche-formation procedure to distribute the population over the optimal solutions and let the population evolve until the Pareto-optimal region is completely explored. This niche-formation scheme is based on a distance index that measures the similarity between two solutions in order to decide if both may share the same niche or not. The similarity criterion is usually based on a Euclidean norm (given that the two solutions are represented with a vector), nevertheless, as this paper will explain, this kind of metric is not adequate for RBFNNs, thus being necessary a more suitable distance index. The experimental results obtained show that a MOEA including the proposed distance index is able to explore sufficiently the Pareto-optimal region and provide the user a wide variety of Pareto-optimal solutions.

1 Introduction

The automatic optimization of a RBFNNs from training data [8, 9, 18, 19] is a problem in which two clearly competing objectives must be satisfied. The model's prediction error must be minimized in order to achieve a well fitted model, while the number of Radial Basis Functions (RBFs) should be as low as possible to obtain a reliable interpolator. The problem here is how to satisfy both objectives simultaneously. Improving one of them will probably worsen the other. This kind of problem is known as a Multi-Objective Problem (MOP) [16, 11, 2, 10], and their solutions are usually sub-optimal for each objective in particular, but "acceptable" taking all the objectives into account, where "acceptable" is totally subjective and problem-dependent.

The algorithms proposed in the literature to construct RBFNNs from examples usually try to find a unique model with a compromise between its complexity and its prediction error. This is not an adequate approach. In MOPs there is usually more than one alternative optimal solution (each making different compromises between multiple objectives) that can be considered equivalent. Thus, it is very difficult to adapt conventional optimization techniques to solve MOPs because they were not designed to deal with more than one solution simultaneously [6]. Nevertheless, Evolutionary Algorithms (EAs) maintain a population of potential solutions for the problem, thus making it easier to adapt them to

solve MOPs [6]. In particular, the fitness of the solutions must be adapted to comprise all the objectives to be satisfied and new mutation operators must be designed to alter the structure of RBFNNs. With these changes, an EA becomes a MOEA able to search the Pareto-optimum region and find a wide variety of solutions with different compromises between the competing objectives.

2 Evolving competing objectives

The great difference between single objective and multiple objective problems is that the set of solutions is not completely ordered for MOPs. In the case of RBFNNs, as the complexity of a net gets higher, it can achieve a lower approximation error, but it loses generalization properties, so if we want to satisfy both objectives (low approximation error and high generalization properties), there will exist a set of Pareto-optimal solutions with different compromises between the competing objectives. Thus, good multiobjective algorithm should find as many Pareto-optimum solutions as possible, to provide the final user with the possibility of choosing the right solution following his own criteria.

There have been proposed several approaches in the literature to adapt EA to MOPs, such as the MOGA presented in [6] or the NSGA described in [21]. In [8] we used the approach proposed in [6], which defines the concept of *rank* of a solution as:

$$\text{rank}(\iota_j) = 1 + \text{dom}_j \quad (1)$$

where dom_j represents the number of solutions dominating ι_j in the current population ¹. Note that rank improves when becomes smaller, that is, as the rank of ι_j gets a lower value, ι_j represents a better solution for the MOP, thus, all the Pareto-optimum solutions will be assigned a rank value of one.

This simple modification in the fitness evaluation of the RBFNNs allows a generic EA to solve an MOP transparently, that is, without changing any other of its components, although the design of expert evolutionary operators can improve the search results, as reported in [9]. The shortcoming of this scheme is that at the end of the ranking procedure there may exist many solutions having the same rank, so, the selection procedure will maintain or delete blocks of solutions for the next generation, what is likely to produce a large selection pressure which may cause a premature convergence of the algorithm [7].

To avoid this premature convergence problem, MOGA and our MOEA [8] use a niche-formation method to distribute the population over the Pareto-optimal region. The difference between MOGA and our MOEA is that the former performs a fitness sharing step based on the objective function values and our MOEA uses the parameter values for this step, thus maintaining the diversity in the parameter set rather than in the objective function values. This change allows to distinguish two different solution with the same fitness value, but needs

¹ A solution ι_j dominates another solution ι_k if ι_j is better or equal than ι_k for all the objectives in the MOP (approximation error and generalization properties)

a phenotypic distance index to estimate the similarity between two solutions of the problem.

Two solutions are considered belonging to the same niche if their phenotypic distance is lower than the threshold σ_{sh} [4]. The idea behind this sharing scheme is to divide the population into several niches formed by similar solutions, and modify the fitness of the members of a niche inversely proportional to the size of the niche. To carry out this task, the following equation is applied [21]:

$$Sh(d(\iota_j, \iota_k)) = \begin{cases} 1 - \left(\frac{d(\iota_j, \iota_k)}{\sigma_{sh}} \right)^2, & \text{if } d(\iota_j, \iota_k) < \sigma_{sh} \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

where $d(\iota_j, \iota_k)$ denotes the phenotypic distance index between ι_j and ι_k described below.

Finally, the fitness of every solution is modified according to:

$$f'_j = f_j \cdot \sum_{k=1}^n Sh(d(\iota_j, \iota_k)) \quad (3)$$

being n the size of the population.

3 A Phenotypic Distance Index for RBFNNs

Most of the approaches in the literature use a Euclidean norm in the parameter space to estimate the phenotypic distance between two solutions, given that these two solutions are coded as arrays of parameters [5, 4, 21], but this metric is not suitable for RBFNNs. Using this metric, two identical RBFNNs with the same RBFs (centers and radii), but coded in different positions in the array of RBFs, would obtain a distance different than 0. This simple example demonstrate the necessity of a more adequate phenotypic distance for this problem.

Another issue to consider is that in the MOP that we are solving, two different solutions may have different complexity (number of RBFs), thus, the distance index must be designed having this fact in mind. Taking these considerations into account, this paper proposes the following phenotypic distance index:

$$d(\iota_j, \iota_k) = \frac{\sum_{i_1=1}^{m_j} \min \{ \|c_{j,i_1} - c_{k,i_2}\| : 1 \leq i_2 \leq m_k \}}{m_j + m_k} + \frac{\sum_{i_2=1}^{m_k} \min \{ \|c_{k,i_2} - c_{j,i_1}\| : 1 \leq i_1 \leq m_j \}}{m_j + m_k} \quad (4)$$

where m_j and m_k are respectively the number of RBFs in ι_j and ι_k , and c_{j,i_1} y c_{k,i_2} their centers. Basically, this index calculates the Euclidean norm between

each center in one RBFNN and its closest center the other RBFNN, and then calculates the mean value of this measures.

The proposed phenotypic index is not affected by the coding order of the RBFs in the solutions, different number of RBFs in the nets and it is also robust against RBFNNs with several identical RBFs. Thus, it is able to distribute the solutions over the Pareto-optimal region and provide the user a wide variety of possibilities, as shown in the following section.

4 Experimental Results

As described above, the MOEA [8] with the new phenotypic distance index is able to obtain in only one execution several optimum solutions for different configurations (a Pareto-optimum frontier of solutions) for a given training set of examples. In this case, the examples belong to the Mackey-Glass time-delay differential equation [13]:

$$\frac{ds(t)}{dt} = \alpha \cdot \frac{s(t - \tau)}{1 + s^{10}(t - \tau)} - \beta s(t) \quad (5)$$

Following previous studies [22], the parameters were fixed to $\alpha = 0.2$, $\beta = 0.1$, thus obtaining a chaotic time series with no clearly defined period; it does not converge or diverge, and is very sensitive to initial conditions.

As in [12], the time series values at integer points were obtained applying the fourth-order Runge-Kutta method to find the numerical solution for the above equation. The values $s(0) = 1.2$, $\tau = 17$, and $s(t) = 0$ for $t < 0$ were assumed. This data set can be found in the file `mgdata.dat` belonging to the FUZZY LOGIC TOOLBOX OF MATLAB 5.

Following the conventional settings to perform a long term prediction of these time series, we predict the value $s(t + 85)$ from the current value $s(t)$ and the past values $s(t - 6)$, $s(t - 12)$, and $s(t - 18)$; thus, the training vectors for the model have the following format:

$$[s(t - 18), s(t - 12), s(t - 6), s(t), s(t + 85)] \quad (6)$$

As argued by Moody and Darken in [15], this prediction problem is a significant challenge in which classical methods do little better than chance, and thus the use of RBFNNs is justified.

The first 500 input vectors were used to train the model and the next 500 vectors were used to test the RBFNNs obtained. The algorithm was run several times with a population of 25 solutions for 1000 generations, and the Levenberg-Marquardt minimization algorithm was applied to the best solutions found to fine-tune their parameters. Table 1 compares the obtained result with other approaches presented in the literature in terms of their *Root Mean Squared Error* (RMSE), defined as:

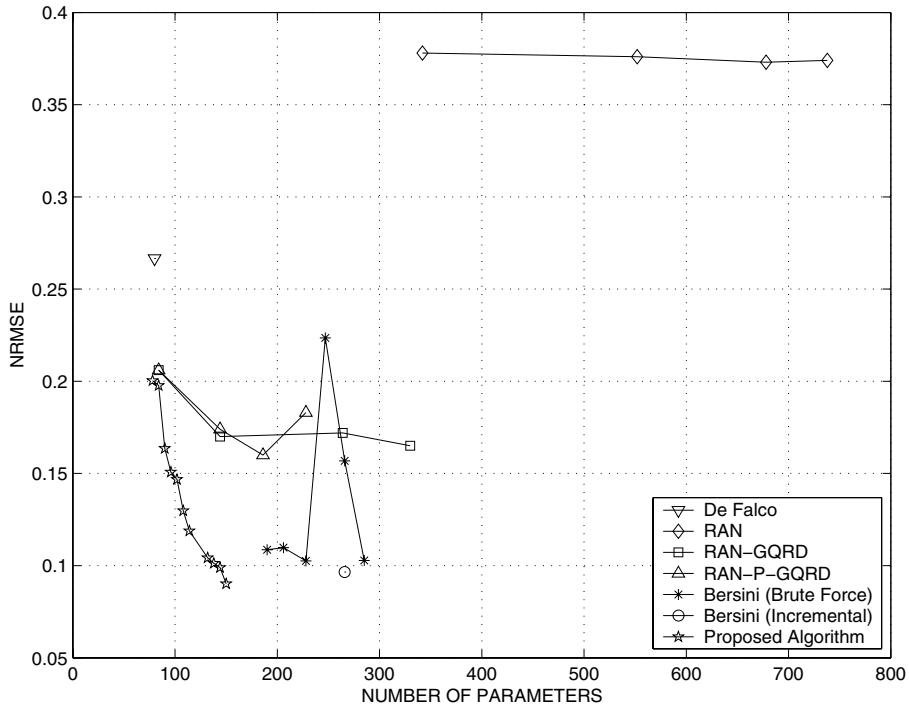


Fig. 1. Comparison of the proposed algorithm with others applied in the literature to predict the $s(t + 85)$ value of the Mackey-Glass time series

$$\text{RMSE} = \sqrt{\frac{\sum_{k=1}^n (y^k - \mathcal{F}(\mathbf{x}^k; \Phi, \Omega))^2}{n}} \quad (7)$$

Some of the other approaches are also based on RBFNNs, such as the model RAN [17], which iteratively constructs an RBFNN analyzing the novelty of the input data, or the modifications of RAN proposed in [20], which include the Givens QR decomposition (RAN-GQRD) to obtain the weights of the net and a pruning criterion (RAN-P-GQRD) to reduce the complexity of the net. The results are compared with other paradigms too. One of them [1] presents two different algorithms to train fuzzy systems, one using brute force and another incremental, and it is shown that the brute force approach presents an unstable behavior as the number of rules is increased and it not reaches the approximation errors obtained by the incremental algorithm. The other one [3] applies BGAs (*Breeder Genetic Algorithms*) to train MLPs. It can be appreciated that the algorithm with the new phenotypic distance index is able to distribute the

Algorithm		m	n_p	Test NRMSE
MLP + BGA (De Falco <i>et al.</i> 1998)		16	80	0.2666
RAN (Platt 1991)	$\epsilon = 0.1$	57	342	0.378
	$\epsilon = 0.05$	92	552	0.376
	$\epsilon = 0.02$	113	678	0.373
	$\epsilon = 0.01$	123	738	0.374
RAN-GQRD (Rosipal <i>et al.</i> 1998)	$\epsilon = 0.1$	14	84	0.206
	$\epsilon = 0.05$	24	144	0.170
	$\epsilon = 0.02$	44	264	0.172
	$\epsilon = 0.01$	55	330	0.165
RAN-P-GQRD (Rosipal <i>et al.</i> 1998)	$\epsilon = 0.1$	14	84	0.206
	$\epsilon = 0.05$	24	144	0.174
	$\epsilon = 0.02$	31	186	0.160
	$\epsilon = 0.01$	38	228	0.183
Fuzzy Systems (Bersini <i>et al.</i> 1997)	Brute Force	10	190	0.1086
		11	206	0.1098
		12	228	0.1026
		13	247	0.2235
		14	266	0.1568
		15	285	0.1028
Proposed Algorithm	Incremental	14	266	0.0965
		13	78	0.2003 ± 0.0178
		14	84	0.1977 ± 0.0164
		15	90	0.1635 ± 0.0401
		16	96	0.1507 ± 0.0193
		17	102	0.1467 ± 0.0178
		18	108	0.1297 ± 0.0175
		19	114	0.1188 ± 0.0131
		20	120	0.1268 ± 0.0174
		21	126	0.1187 ± 0.0104
		22	132	0.1042 ± 0.0135
		23	138	0.1012 ± 0.0132
		24	144	0.0989 ± 0.0063
		25	150	0.0901 ± 0.0066

Table 1. Comparison of the proposed algorithm with others applied in the literature to predict the $s(t+85)$ value of the Mackey-Glass time series; m represents the number of RBFs or rules (depending on the model), and n_p is the number of free parameters

population over the Pareto-optimal solutions that dominate all the solutions in Table 1. Figure 1 summarizes graphically the results.

5 Conclusions

This paper presents a new phenotypic distance index to improve the fitness sharing procedure in a MOEA. The objective of this new index is to better distribute the solutions in the population over the Pareto-optimal region and thus avoid the premature convergence of the algorithm.

Section 4 has shown that with this new index, a wide variety of RBFNNs have been obtained, each one Pareto-optimal, but with a different compromise between the two conflicting objectives: approximation error and complexity.

Acknowledgment

This work has been partially supported by the Spanish *Ministerio de Ciencia y Tecnología*, under project DPI2001-3219.

References

1. H. Bersini, A. Duchateau, and N. Bradshaw. Using Incremental Learning Algorithms in the Search for Minimal and Effective Fuzzy Models. In *Proceedings of the 6th IEEE International Conference on Fuzzy Systems*, pages 1417–1422, Barcelona, Spain, July 1997. IEEE Computer Society Press.
2. V. Chankong and Y. Y. Haimes. *Multiobjective Decision Making Theory and Methodology*. North-Holland, New York, 1983.
3. I. De Falco, A. Della Cioppa, A. Iazzetta, P. Natale, and E. Tarantino. Optimizing Neural Networks for Time Series Prediction. In R. Roy, T. Furuhashi, and P. K. Chawdhry, editors, *Proceedings of the 3rd On-line World Conference on Soft Computing (WSC3). Advances in Soft Computing - Engineering Design and Manufacturing*, Internet, June 1998. Springer Verlag.
4. K. Deb. Evolutionary Algorithms for Multi-Criterion Optimization in Engineering Design. In K. Miettinen, P. Neittaanmäki, M. M. Mäkelä, and J. Périaux, editors, *Proceedings of Evolutionary Algorithms in Engineering and Computer Design, EU-ROGEN'99*. John Wiley, Apr. 1999.
5. K. Deb and D. E. Goldberg. An Investigation of Niche and Species Formation in Genetic Function Optimization. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 42–50, San Mateo, CA, 1989. Morgan Kaufmann.
6. C. M. Fonseca and P. J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423. Morgan Kaufmann, 1993.
7. D. E. Goldberg and K. Deb. A COMPARISON OF SELECTION SCHEMES USED IN GENETIC ALGORITHMS. pages 69–93. Morgan Kaufmann, San Mateo, CA, 1991.

8. J. González, I. Rojas, H. Pomares, and J. Ortega. RBF Neural Networks, Multiobjective Optimization and Time Series Forecasting. In Mira and Prieto [14], pages 498–505.
9. J. González, I. Rojas, H. Pomares, and M. Salmerón. Expert Mutation Operators for the Evolution of Radial Basis Function Neural Networks. In Mira and Prieto [14], pages 538–545.
10. A. E. Hans. Multicriteria optimization for highly accurate systems. In W. Stadler, editor, *Multicriteria Optimization in Engineering and Sciences, Mathematical Concepts and Methods in Science and Engineering*, volume 19, pages 309–352, New York, 1988. Plenum Press.
11. C. L. Hwang and A. S. M. Masud. *Multiple Objective Decision Making – Methods and Applications*, volume 164 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, Berlin, 1979.
12. J. S. R. Jang. ANFIS: Adaptive Network-based Fuzzy Inference System. *IEEE Trans. Syst., Man, Cybern.*, 23:665–685, May 1993.
13. M. C. Mackey and L. Glass. Oscillation and Chaos in Physiological Control Systems. *Science*, 197(4300):287–289, 1977.
14. J. Mira and A. Prieto, editors. *Connectionist Models of Neurons, Learning Processes and Artificial Intelligence*, volume 2084 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.
15. J. E. Moody and C. J. Darken. Fast Learning in Networks of Locally-Tuned Processing Units. *Neural Computation*, 1:281–294, 1989.
16. V. Pareto. *Cours D'Economie Politique*, volume I and II. F. Rouge, Lausanne, 1896.
17. J. Platt. A Resource Allocation Network for Function Interpolation. *Neural Computation*, 3:213–225, 1991.
18. I. Rojas, J. González, A. Cañas, A. F. Díaz, F. J. Rojas, and M. Rodriguez. Short-Term Prediction of Chaotic Time Series by Using RBF Network with Regression Weights. *Int. Journal of Neural Systems*, 10(5):353–364, 2000.
19. I. Rojas, H. Pomares, J. González, J. L. Bernier, E. Ros, F. J. Pelayo, and A. Prieto. Analysis of the Functional Block Involved in the Design of Radial Basis Function Networks. *Neural Processing Letters*, 12(1):1–17, Aug. 2000.
20. R. Rosipal, M. Koska, and I. Farkaš. Prediction of Chaotic Time-Series with a Resource-Allocating RBF Network. *Neural Processing Letters*, 7:185–197, 1998.
21. N. Srinivas and K. Dev. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, 1995.
22. B. A. Whitehead and T. D. Choate. Cooperative-Competitive Genetic Evolution of Radial Basis Function Centers and Widths for Time Series Prediction. *IEEE Trans. Neural Networks*, 7(4):869–880, July 1996.

The Antiquadrupolar Phase of the Biquadratic Neural Network

David R. C. Dominguez

EPS, Universidad Autonoma de Madrid,
Cantoblanco, 28049 Madrid, Spain
david.dominguez@ii.uam.es

Abstract. We study the macroscopic dynamics of a three-state mean-field neural network, by using information theory principles. The results are expressed in terms of the relevant order parameters of the system. The network is composed by bilinear and biquadratic synaptic interactions, which yields an improvement of the storage and information properties of the network, measured by the mutual information and the basins of attraction. The introduction of a self-adaptive mechanism in the synapses is analyzed. The coexistence of the usual retrieval phase with a quadrupolar phase, as well as with a new informative *anti-quadrupolar* phase, is observed.

1 Introduction

Patterns in nature are not just binary. Most research on pattern recognition works on black and white statistics, forgetting the richer structure of colors or grey tones. Also, the coding of the information received by neural systems, are not limited to all/nothing, yes/not, but needs fuzzy states (i.e., maybe). This intermediate states comes from, for instance, the variability of the spike train of nervous activity. A first approach to study structured patterns is the use of sparse-coded models[1]. A deeper understanding comes from the use of ternary patterns. In both cases the tools of information theory are useful in the search for the capabilities of the neural network as associative memory device.

In a recent paper the formalism of the information theory for obtaining the effective Hamiltonian was introduced, which maximizes the Mutual Information of the system composed by a three-state neural network [2]. The last is expressed by means of three macroscopic parameters: the overlap between neurons and patterns, the neural activity and the *activity-overlap*, i.e., the overlap restricted to the active neurons. A three-state neural network is defined by a set of $\mu = 1, \dots, p$ embedded *ternary* patterns, $\{\xi_i^\mu \in [0, \pm 1]\}$ on sites $i = 1, \dots, N$, which are assumed here to be independent random variables that follow the probability distribution

$$p(\xi_i^\mu) = a\delta(|\xi_i^\mu|^2 - 1) + (1 - a)\delta(\xi_i^\mu), \quad (1)$$

where a is the activity of the patterns ($\xi_i^\mu = 0$ are the inactive ones). Accordingly, the neuron states are three-state dynamical variables, defined as

$\sigma_i \in \{0, \pm 1\}$, $i = 1, \dots, N$ and coupled to other neurons through synaptic connections, for our purpose, of a Hebbian-like form. The active states, $\sigma_i = \pm 1$, become accessible by means of an effective threshold that is built into the model Hamiltonian.

The pattern retrieval task becomes successful if the state of the neuron $\{\sigma_i\}$ matches a given pattern $\{\xi_i^\mu\}$. The measure of the quality of retrieval that we use here is the mutual information, which is a function of the conditional distribution of the neuron states given the patterns, $p(\sigma|\xi)$. The order parameters needed to describe this information are the large-N (thermodynamic) limits of the standard overlap of the μ th pattern with the neuron state,

$$m_N^\mu \equiv \frac{1}{aN} \sum_i \xi_i^\mu \sigma_i \rightarrow m = \langle \langle \sigma \rangle_\sigma | \xi | \frac{\xi}{a} \rangle_\xi, \quad (2)$$

the neural activity,

$$q_{Nt} \equiv \frac{1}{N} \sum_i |\sigma_{it}|^2 \rightarrow q = \langle \langle \sigma^2 \rangle_\sigma | \xi | \frac{\xi}{a} \rangle_\xi, \quad (3)$$

and the so called *activity-overlap*[1, 3],

$$n_{Nt}^\mu \equiv \frac{1}{aN} \sum_i |\sigma_{it}|^2 |\xi_i^\mu|^2 \rightarrow n = \langle \langle \sigma^2 \rangle_\sigma | \xi | \frac{\xi^2}{a} \rangle_\xi. \quad (4)$$

In the expressions for the thermodynamic limits, $\lim N \rightarrow \infty$, m, q, n , the index μ for the considered pattern were dropped out. The averages are over the conditional distribution, $p(\sigma|\xi)$ and over the pattern distribution, $p(\xi)$, in Eq.(1).

In the next sections we show that, when the patterns are ternary, the network should include biquadratic energy terms in order to perform correctly. Furthermore, the true expansion of the mutual information automatically introduces a mechanism of adapting the synapses according to the neural activity in each time step, which self-controls the evolution of the network and inhibits the appearance of meta-stable states.

2 Mutual Information

The Mutual Information (MI) between patterns and neurons, by regarding the patterns as the input and the neuron states as the output of the channel at each time step [4] is :

$$I[\sigma; \xi] = S[\sigma] - \langle S[\sigma|\xi] \rangle_\xi, \quad (5)$$

where

$$\begin{aligned} S[\sigma] &\equiv - \sum_\sigma p(\sigma) \ln[p(\sigma)], \\ S[\sigma|\xi] &\equiv - \sum_\sigma p(\sigma|\xi) \ln[p(\sigma|\xi)] \end{aligned} \quad (6)$$

are the entropy and the conditional entropy of the output, respectively. The quantity $\langle S[\sigma|\xi] \rangle_\xi$ is the so-called equivocation term.

The expressions for the entropies follow from the conditional probability [3],[1], and are given by:

$$S[\sigma] = -q \ln \frac{q}{2} - (1-q) \ln(1-q), \quad (7)$$

for the output entropy, and for the equivocation term, $\langle S[\sigma|\xi] \rangle_\xi = aS_a + (1-a)S_{1-a}$,

$$\begin{aligned} S_a &= -n_+ \ln n_+ - n_- \ln n_- - (1-n) \ln(1-n), \\ S_{1-a} &= -s \ln \frac{s}{2} - (1-s) \ln(1-s), \end{aligned} \quad (8)$$

where $n_\pm = (n \pm m)/2$.

We search for an energy function which is symmetric in any permutation of the patterns ξ^μ , since they are not known initially to the retrieval process [2]. This requires that the initial retrieval of any pattern ξ^μ is weak, i.e. the overlap $m^\mu \sim 0$. For general a, q , the variable σ^2 is also initially almost independent of $(\xi^\mu)^2$, so that $n^\mu \sim q$. Hence, the parameter

$$l^\mu \equiv \frac{n^\mu - q}{1 - a} = \langle \sigma^2 \eta^\mu \rangle, \quad \eta^\mu \equiv \frac{(\xi^\mu)^2 - a}{a(1 - a)}, \quad (9)$$

also vanishes initially in this limit. Note that this parameter is a recognition of a fluctuation in $(\xi^\mu)^2$ by the binary state variable σ^2 .

An expansion of the mutual information around $m^\mu = 0, l^\mu = 0$ thus gives

$$I^\mu \approx \frac{1}{2} c_1 (m^\mu)^2 + \frac{1}{2} c_2 (l^\mu)^2, \quad (10)$$

where $c_1 = a/q$, $c_2 = c_1(1-a)/(1-q)$ and the total information of the network will be given by summing over all the patterns $I_{pN} = N \sum_\mu I^\mu$. The above expression for I^μ will now be used as a measure of the mutual information during the performance of the network through growing values of m^μ and l^μ . The energy function derived from this expansion governs the network learning and retrieval dynamics, and reads $\mathcal{H} = \mathcal{H}_1 + \mathcal{H}_2$, with

$$\begin{aligned} \mathcal{H}_1 &= -\frac{1}{2} \sum_{ij} J_{ij} \sigma_i \sigma_j, \quad J_{ij} = \frac{c_1}{a^2 N} \sum_{\mu=1}^p \xi_i^\mu \xi_j^\mu, \\ \mathcal{H}_2 &= -\frac{1}{2} \sum_{i,j} K_{ij} \sigma_i^2 \sigma_j^2, \quad K_{ij} = \frac{c_2}{N} \sum_{\mu=1}^p \eta_i^\mu \eta_j^\mu, \end{aligned} \quad (11)$$

the bilinear and biquadratic terms, respectively.

These energies can be written in terms of the local fields related to the lineal and to the quadratic neuron states, respectively:

$$\begin{aligned}\mathcal{H}_1 &= -\frac{1}{2} \sum_i h_i \sigma_i, \quad h_i = \sum_j J_{ij} \sigma_j, \\ \mathcal{H}_2 &= -\frac{1}{2} \sum_i \theta_i \sigma_i^2, \quad \theta_i = \sum_j K_{ij} \sigma_j^2.\end{aligned}\quad (12)$$

3 Macro-dynamics

A noisy dynamics for the updating of the neurons at each time steps can be given by the Boltzmann factor with energy potential E ,

$$\begin{aligned}p(\sigma_{t+1}|h_t, \theta_t) &= \exp[E(\sigma_t)]/Z; \\ E(\sigma) &= h\sigma + \theta\sigma^2, \quad Z \equiv \sum_{\sigma} \exp[E(\sigma)]\end{aligned}\quad (13)$$

This is a microscopic dynamics which leads to N coupled equations. Instead, $2 \times p$ coupled equations for parameters can be written, and only the few macroscopic of them are studied.

Because we are mainly interested on the retrieval properties of our network, we take an initial configuration whose retrieval overlaps are only macroscopic of order $O(1)$ for a given pattern, let say the first one.

Supposing a network configuration in a given time step t , $\{\sigma_{i,t}\}$ with parameters m_t, l_t, q_t , the fields $h_{t=0}$ and $\theta_{t=0}$ in this time step are given by:

$$\begin{aligned}h_t &= c_{1t} \left(\frac{1}{a} \xi m_t + \omega_t \right); \quad \omega_t \equiv \sum_{\nu \geq 2}^p \frac{1}{a} \xi^{\nu} m_t^{\nu} \\ \theta_t &= c_{2t} (\eta l_t + \Omega_t); \quad \Omega_t \equiv \sum_{\nu \geq 2}^p \eta^{\nu} l_t^{\nu},\end{aligned}\quad (14)$$

where the indices $\mu = 1$ where dropped, and the rest of the patterns is regarded as some additive noise. Here we have defined the dynamical variables $c_{1t} = a/q_t$ and $c_{2t} = c_{1t}(1-a)/(1-q_t)$.

Supposing a given configuration $\{\sigma_{i,t}\}$ as a collection of independently distributed random variable, with zero-mean and variance q_t , the noises above, ω_t and Ω_t in the time step t , according to the central limit theorem are independent Gaussian distributed[1],[5], with zero mean and variance

$$\begin{aligned}Var[\omega_t] &= \frac{1}{a^2} \alpha q_t \equiv \Delta_t^2 \\ Var[\Omega_t] &= \frac{\Delta_t^2}{(1-a)^2}.\end{aligned}\quad (15)$$

In the extremely diluted case, where the first time step describes completely the dynamics, and after taking the asymptotic limit $N \rightarrow \infty$, the single-step evolution equations for the overlap m_t , the neural activity q_t and the activity overlap n_t , are the following:

$$m_{t+1} = \left\langle \frac{\xi}{a} \overline{\sigma_t} \right\rangle_{\xi} = \int D\Phi(y) \int D\Phi(z) F_{\beta} \left(c_{1t} \frac{m_t}{a} + c_{1t} y \Delta_t; c_{2t} \frac{l_t}{a} + c_{2t} \frac{z \Delta_t}{1-a} \right), \quad (16)$$

$$q_t = \left\langle \overline{\sigma_t^2} \right\rangle_{\xi} = a n_t + (1-a) s_t, \\ s_{t+1} \equiv \left\langle \frac{1-\xi^2}{1-a} \overline{\sigma_t^2} \right\rangle_{\xi} = \int D\Phi(y) \int D\Phi(z) G_{\beta} \left(c_{1t} y \Delta_t; -c_{2t} \frac{l_t}{1-a} + c_{2t} \frac{z \Delta_t}{1-a} \right) \quad (17)$$

and

$$n_{t+1} = \left\langle \frac{\xi^2}{a} \overline{\sigma_t^2} \right\rangle_{\xi} = \int D\Phi(y) \int D\Phi(z) G_{\beta} \left(c_{1t} \frac{m_t}{a} + c_{1t} y \Delta_t; c_{2t} \frac{l_t}{a} + c_{2t} \frac{z \Delta_t}{1-a} \right). \quad (18)$$

Here the averages are over the Gaussian distributions ω, Ω , according to Eq(15). The functions F_{β}, G_{β} ,

$$F_{\beta}(h, \theta) = \frac{1}{Z} 2e^{\beta\theta} \sinh(\beta h), \quad G_{\beta}(h, \theta) = \frac{2}{Z} e^{\beta\theta} \cosh(\beta h) \quad (19)$$

with $Z = 1 + 2e^{\beta\theta} \cosh(\beta h)$, are the mean $\overline{\sigma_t}$ and the square mean $\overline{\sigma_t^2}$ of the neurons over the synaptic noise with parameter $\beta = a/T$, respectively.

4 Results and Conclusions

The aspects of interest we present in this work are twofold: first, the dynamics of biquadratic network, and second the enhancement of its capability to retrieve patterns which are far from the initial states.

First we studied the Biquadratic (BQ) network with $c_1 = 1 = c_2$. The complex behavior of the biquadratic neural network, due to the biquadratic term in the energy function can be seen in the Figures 1 and 2. Although there is a symmetry for the axis m_t , corresponding to the bilinear term in the energy, there are no symmetry for the axis l_t (except if the pattern are ternary uniform). First we found the usual zero phase, Z , with $m = l = 0$ which is a repellor in every direction (the white square) and the quadrupolar, Q , phase, with $m = 0, l > 0$,

which is an attractor in the direction l . It is unstable for $\alpha = 0.12$ (Fig.1, the dark square), but it is a stable attractor for $\alpha = 0.15$ (Fig.2, the dark circle).

In the first case, the load is not too large, and the network is able to recognize the position of the active sites ± 1 *and* the signs of the patterns. So the stable attractor is the R phase, with both $m, l > 0$, which is reached after a while. In the second case, the load becomes large enough, and the network is only able to recognize the active sites, so the Q is stable and no R phase is reached.

A new, yet more strange, phase, we named *anti-quadrupolar* phase, N , with $m = 0, l < 0$, emerges in both cases. Even if one starts at *not too much* negative values of l_0 , the network reaches the $l_t > 0$ attractor, but when l_0 is smaller than a given value, the N phase can be reached. This phase, the total information about the original ternary pattern doesn't vanishes, as well as in the Q phase. In the N phase we can partially retrieve the opposite of the activities, that means, σ is uncorrelated with ξ , but σ^2 is correlated with $1 - \xi^2$ (in a complementary way). The resulting information is again finite. In opposition to the Q phase, however, it is not possible to reach an R phase through the N phase.

The Figure 3 shows large plateaus in time the system spends around the Q phase, as the network is trying to recognize the signs of ξ , until it succeeds in retrieve the pattern. It has been represented in Fig.1, by the separatrix lines connecting the Q to the R phases. The information starts small, and during $t = 70$ time steps, it remains around $i = 0$. Then, after a while as long as $\Delta_t = 200$ time steps, the informations suddenly increases to a higher value. The comparison is made with the self-control (SC) model[1], where we see that no intermediate plateau appears for models which do not include a biquadratic energy term, like the bilinear Ising models (here we used an extension of the noisy SC model, the SCT).

The appearance of the Q phase can be seen in Fig.4, for $a = 0.9$, where the evolution of a network of $N = 2,000$ neurons was simulated. In the simulation, we use the probabilistic updating according to a Gibbs distribution with energy potential given by the local fields h and θ , in the way: $E = (2 - c)h\sigma + c\theta\sigma^2$. While at $T = 0.7$ the phase is clearly the R , at $T = 0.8$, the phase is R only up to a given value of α , after which the phase is Q . For $T = 0.9$ the phase is Q from the beginning. So, the phase Q is not just a theoretical prediction, but it appears in simulation.

The work has shown a series of results comparing different schemes of adaptation of the neural activities to the presence of multi-state patterns. The main result was the appearance of a new phase of information transmission (the N), where the strong states (yes/not) are switched to the fuzzy states. It is plausible in biological learning process, in situations when the environment is very noisy. The origin of our actual model is an indication that natural evolution of neural systems are based on maximization of information. We hope this work can be extended to more states so that realistic images or other patterns can be stored and retrieved without need to compress them in binary patterns.

Acknowledgments: DRCD is supported by a Ramon y Cajal grant from the MCyT (Spain).

References

1. D.Dominguez and D.Bollé, Phys. Rev. Lett. **80**, 2961 (1998).
2. D.Dominguez and E.Korutcheva, Phys.Rev.E **62(2)**, (2000) 2620-2628
3. D.Bollé, D.Dominguez and S.Amari, Neural Networks **13**, 455 (2000); D.Bollé and D.Dominguez, Physica A **286**, 401 (2000).
4. C.E.Shannon, *A Mathematical Theory of Communication*, The Bell System Technical Journal, v.27 (1948).
5. D. Bollé, G.M. Shim, B. Vinck, and V. A. Zagrebnev, J. Stat. Phys. **74**, 565 (1994).

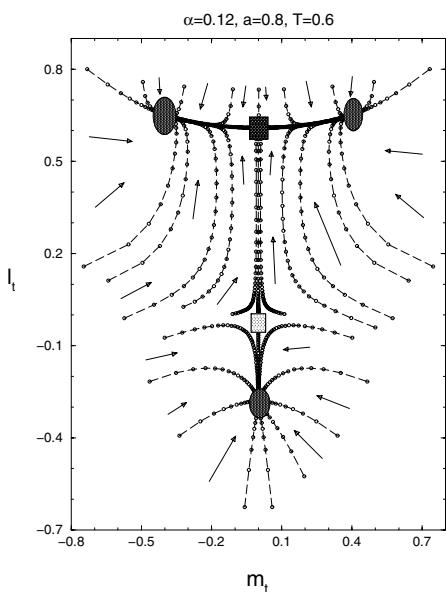


Fig. 1. Flow diagram in the order-parameter space (m, l) for a noisy network with $\alpha = 0.12$. The squares are the Q and Z saddle-points; the circles are the R and N attractors.

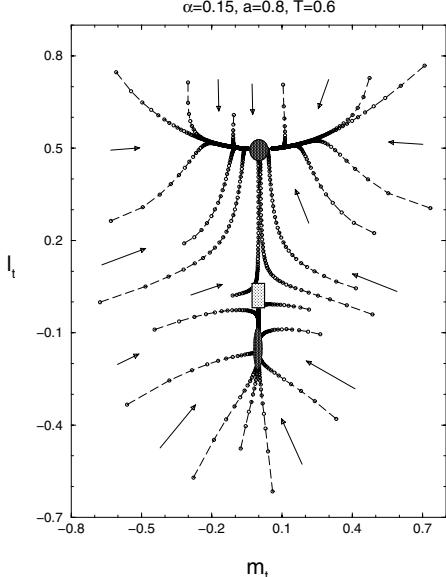


Fig. 2. Flow diagram in the order-parameter space (m, l) for a noisy network with $\alpha = 0.15$. The square is the Z repeller; the circles are the Q and N attractors.

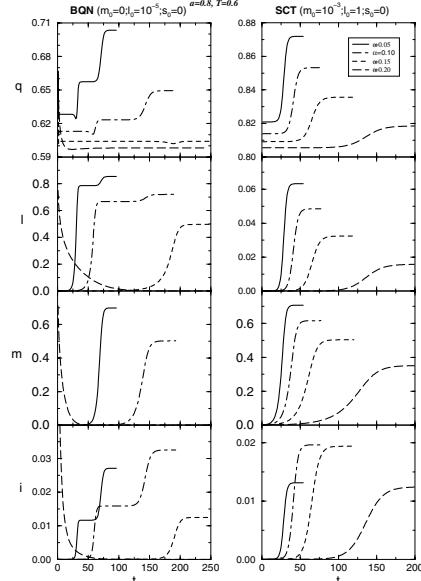


Fig. 3. Evolution behavior of the information i and the parameters m, l, q , for the biquadratic network compared with the self-control network, for several values of the load α .

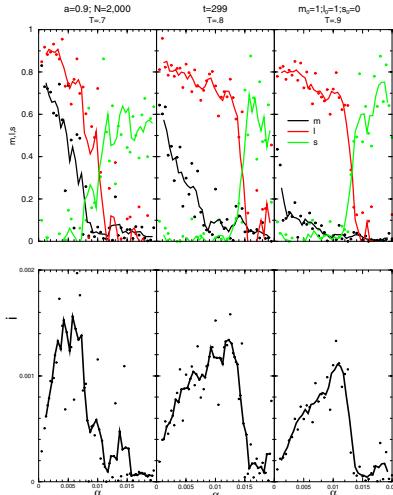


Fig. 4. Simulation for the noisy BQN with $c = 1$, near the transition from Q unstable to Q stable, for $a = 0.9$.

Co-Evolutionary Algorithm for RBF by Self-Organizing Population of neurons

A.J. Rivera¹; J. Ortega²; I. Rojas²; M.J. del Jesús¹

¹Departamento de Informática. Universidad de Jaén.
{arivera,mijesus}@ujaen.es

²Departamento de Arquitectura y Tecnología de Computadores. Universidad de Granada
{julio,ignacio}@atc.ugr.es

Abstract. This paper presents a new evolutionary procedure to design optimal networks of Radial Basis Functions (RBFs). It defines a self-organizing process into a population of RBFs based on the estimation of the fitness for each neuron in the population, and on the use of operators that, according to a set of fuzzy rules, transform the RBFs. This way, it has been possible to define cooperation, speciation, and niching features in the evolution of the population.

1. Introduction

This paper deals with the design of neural networks, more specifically networks of Radial Basis Functions [10], by using evolutionary algorithms. A Radial Basis Function Network (RBFN) implements the function $f(x)$ from R^n onto R :

$$f(x) = \sum_{i=1}^m w_i \phi_i(x) \quad (1)$$

where $x \in R^n$, and the m RBF functions ϕ_i have the form $\phi_i = \phi(|x - c_i| / d_i)$ with a scaling factor (width) $d_i \in R$ and a centre $c_i \in R^n$. Among the possible choices for ϕ in this paper we consider gaussian RBFs, $\phi(r) = \exp(-r^2)$, and $| \cdot |$ is a euclidean norm on R^n . The central problem in RBF optimization is the placement of the centers and the determination of the widths d_i and weights, to achieve the best approximation and generalization performance.

The research work done in the field of evolutionary computation to optimize neural networks can be classified according to the following major trends:

- *Competition among neural networks.* There is a population of neural networks that compete by using a fitness function corresponding to the performance of each network. Alternatives within this general procedure include: (a) evolving only the structural specification of the untrained networks, and use conventional non-evolutionary learning algorithms to train the networks [8], and (b) using evolutionary algorithms to determine both the weights and the structure of the network [7]. In this case, no additional training is required but the search space of possible neural networks is much larger. Other methods employ a tradeoff between the two previous approaches [1].

- *Evolving cooperating and competing units.* Each individual in the population is a neuron (or a group of neurons [4]) of the network instead of a population of competing networks, and the neurons reproduce or die depending on their performance [15]. Thus, the neurons of the population work together to determine a neural network that performs better with successive generations. Two problems must be solved in order to reach this goal. The first is the credit apportionment problem [14], which involves determining a performance measure for each individual in the population that reflects the performance of the whole population; this is the measure that has to be optimized, and which can be evaluated. The second problem is that of niching, which implies maintaining a population of neurons evolving to different parts of the space of solutions. In [15], the credit assigned to each RBF is based on the contribution of this RBF to the overall performance of the network, and niche creation is implicitly obtained by changing the intensity of competition between neurons according to their activation overlappings.

Optimization theory provides methods that can be used to train artificial neural networks. Indeed, as training feedforward networks can be considered as an unconstrained optimization problem, it would be possible to use algorithms such as the steepest descent, the conjugate gradient, the Newton-Raphson, and the Levenberg-Marquardt ones. Nevertheless, these methods remain stuck on the first local minimum they found. Among these algorithms, one of the most powerful is, undoubtedly, the Levenberg-Marquardt (LM) method [9]. It combines the excellent local convergence properties of a Gauss-Newton procedure, and a consistent error decrease provided by (a suitable scaled) gradient descendent technique.

2. Description of the algorithm

The procedure proposed (Figure 1) consists of two stages:

- In the first stage (steps 1 to 8 in Figure 1), the individuals of a population of Radial Basis Functions cooperate and compete, respectively, to build a neural network that performs sufficiently well for the application considered, and to obtain a set of RBFs distributed across the whole input domain of the application.
- After obtaining a good initial solution in the previous stage, it is improved by applying the Levenberg-Marquardt algorithm (step 9 in Figure 1).

In the first stage, the individuals evolve by considering several objectives. For each individual i three objectives are taken into account: p_i , a measure of its weight and width; e_i , a measure of its error, and s_i a measure of the overlapping with other neurons. Then, a set with the worst neurons (RBFs) is built, and an operator (chosen from a given set by using a fuzzy logic system [7]) is applied to delete or adjust the RBFs. The fuzzy logic system (FLS) we have defined, presents the parameters p_i , e_i and s_i , as inputs and the probability for the application of each operator as output. The procedure here described is an improvement of the one proposed in [12],[13]. This improvement allows the parameters to take and change their values automatically.

1. Initialize the RBFN
2. Train the RBFN
3. Evaluate RBFs
4. Arrange and Select worst RBFs
5. Apply operators to worst RBFs
6. Add eliminated RBFs to worst isolated points
7. Train the RBFN
8. If the *End condition* is not verified go to 3
9. Apply Levenberg-Marquardt

Fig. 1. RBF network optimization algorithm

Step 1: The algorithm builds an RBFN by initializing a population of r_{init} RBFs whose centers are allocated to randomly selected points. Their widths are set to random numbers inside the interval $[er, er \cdot pc_1]$, where er is the mean distance between RBFs.

Step 2 (and 7): After determining the population of RBFs, a given number of training iterations are applied, for example by using the LMS algorithm [16], to adapt the weights of the RBFN. The algorithm ends when the error of the RBFN, NRMSE (Normalized Root Means Square) is not reduced in two consecutive iterations.

Step 3: The fitness of each RBF is obtained. This fitness comprises three characteristics of each RBF, ϕ : its weight in the present RBFN and the number of points of training set inside $d_i(p_i)$; the error NRMSE and the standard deviation inside $d_i(e_i)$; and the distances to other RBFs (s_i). The parameter p_i takes into account the weight, w_i , and the number of points of training set inside d_i as follows:

$$p_i = rac_i / dp \quad (2)$$

Where:

$$rac_i = |w_i| \cdot (pe_i / mpe) \quad (3)$$

pe_i is the number of points of training set inside d_i and mpe is the mean of pe_i . The parameter dp normalizes the values of p_i inside $[0, 1]$, to use the fuzzy logic system. Thus:

$$dp = mrac / fac \quad (4)$$

where $mrac$ is the mean of the values of rac_i and fac depends on $mrac$ and $drac$ (standard deviation of the rac values) in the following way:

$$fac = f_{itp}(mrac / 2, pf, -mrac / 2, 3 \cdot pf, x) \quad (5)$$

$f_{itp}(x_1, y_1, x_2, y_2, x)$ returns the interpolated value of the line defined between (x_1, y_1) and (x_2, y_2) with $x = mrac - drac$. This is done in order to locate the rac values inside the

interval [0, 1], to use them by the fuzzy system. The parameter e_i in a given RBF ϕ is a measure of NRMSE and the standard deviation in d_i and is defined by:

$$e_i = rerr_i / de \quad (6)$$

where:

$$rerr_i = nrmse_i \cdot (ndesv_i / mndesv) \quad (7)$$

$nrmse_i$ is the NRMSE, $ndesv_i$ is the standard deviation and $mndesv$ is the mean of the $ndesv$ values. de is obtained as dp :

$$de = mrerr \cdot fer \quad (8)$$

$$fer = f_{lp}(mrerr / 2, pr, -mrerr / 2, 3 \cdot pr, x) \quad (9)$$

and $x = mrerr - drerr$, where $mrerr$ is the mean of the $rerr$ values, and $drerr$ is the standard deviation of $rerr$ values.

The overlapping of RBFs is quantified by a function, s_i , assigned to each RBF ϕ :

$$s_i = \sum_j^m s_{ij} \quad (10)$$

where m is the number of RBFs and s_{ij} measures the overlapping between the RBFs ϕ_i and ϕ_j :

$$s_{ij} = \begin{cases} (1 - (D(\phi_i, \phi_j) / d_i)) & \text{si } D(\phi_i, \phi_j) < d_i \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

Step 4: A set of r RBFs to be modified is determined (initially, r is set to $r_{init} / 2$, the number of neurons divided by 2). A multi-objective technique is used to built this set, by taking into account the parameters p_i , e_i , s_i of each RBF. In this way, the worst RBFs are those with lower p_i 's, higher e_i 's, and higher s_i 's.

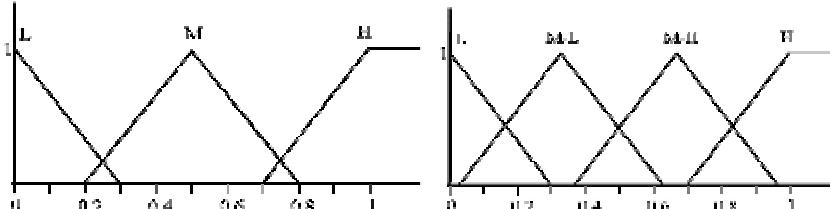
Step 5: Some operators are applied to the RBFs selected in the previous step. These operators are the following ones:

- OP_1 (*RBF elimination*): A RBF is deleted.
- OP_2 (*RBF small modification*): The centre and the width of a selected RBF are slightly modified. For each point of the training set inside the width of the RBF, err (difference between $f(x)$ and the function to approximate) is obtained, and the rules of Table 1 are applied.

Finally, the center and width are changed by adding a randomly selected amount less than a small percentage, pc_2 , of the initial width. To select the operator to apply, a fuzzy system is used. The set of fuzzy rules are applied by considering the previously described parameters (step 3) for each RBF ϕ_i (e_i , m_i , s_i), and one linguistic variable for each parameter (ve for e , vm for m , and vs for s). Figure 2 (left) shows the membership functions for the linguistic variables.

Table 1. Rules used to modify a RBF

	$err > 0$	$err < 0$
$w_i > 0$	Move the RBF towards the point and increase the width	Move the RBF away from the point and decrease the width
$w_i < 0$	Move the RBF away from the point and decrease the width	Move the RBF towards the point and increase the width

**Fig. 2.** Membership functions for vp , ve , vs (left) and for vop_1 and vop_2 (right)

The consequent variables of the rules are vop_1 and vop_2 , and have the membership functions shown in Figure 2 (right). The fuzzy rules are given in Table 2.

Table 2. Set of rules of the fuzzy inference system

Antecedents			Consequents	
vp	ve	vs	vop_1	vop_2
L			M-H	M-H
M			M-L	M-H
H			L	H
	L		L	H
	M		M-L	M-H
	H		M-H	M-H
		L	L	H
		M	M-L	M-H
		H	M-H	M-H

The mechanism used to derive a reasonable conclusion from the rules and a given condition is the Mamdani Fuzzy Model [7]. In this type of fuzzy inference system, \min and \max functions are used as fuzzy AND and OR operators. In order to extract a crisp value from a fuzzy set, the centroid area is used. Thus, from this fuzzy inference system three crisp values are obtained corresponding to the probabilities of adding a new RBF, deleting an RBF, and modifying an RBF.

Step 6. A new RBF, ϕ_n , is added for each RBF deleted by OP1. This new RBF is created in the worst approximated point, out of the d_i of any ϕ_i in the present RBFN. The centre of the new RBF is set to the coordinates of the selected point and is modified by applying a randomly perturbation. To determine the width of the RBF, er , the mean of the present d values. is calculated. The new d_n value is set to dr , the distance to the closeness RBF, modified by applying a random perturbation, taking into account that the $dr < er \cdot (1 - pc_3) \Rightarrow dr = er \cdot (1 - pc_3)$ and if $dr > er \cdot (1 + pc_3) \Rightarrow dr = er \cdot (1 + pc_3)$.

Finally, the Levenberg-Marquardt algorithm is applied (step 9 in Figure 1). In this algorithm, its parameter λ is initially set to 0.001. After each iteration, if the error grows λ is set to $\lambda \cdot 10$ and if the error decreases λ is set to $\lambda / 10$. The algorithm stops whenever the error decreases a negligible amount (less than 0.001) along three consecutive iterations.

3. Experimental results

The algorithm has been evaluated onto a benchmark of bi-dimensional functions that have been used in previous works. The experiments make use of: (a) 300 iterations of the major loop of the algorithm; (b) the NRMSE to measure the performance of our method after running it 30 times; and (c) 400 points randomly selected inside a 20x20 grid are used as training set, and the test set is built in the same way.

Table 3. Experimental parameters used in the algorithm

Parameter	Value
pc_1	1.5
pc_2	10%
pc_3	0.2
α (LMS)	0.3
pf	0.8
pr	5

The benchmark functions are:

$$f_1(x_1, x_2) = \text{sen}(x_1, x_2) \quad x_1, x_2 \in [-2, 2] \quad (12)$$

$$f_2(x_1, x_2) = \frac{1 + \sin(2x_1 + 3x_2)}{3.5 + \sin(x_1 - x_2)} \quad x_1, x_2 \in [-2, 2] \quad (13)$$

$$f_3(x_1, x_2) = 1.3356[1.5(1-x_1) + e^{2x_1-1} \sin(3\pi(x_1-0.6)^2) + e^{3(x_2-0.5)} \sin(4\pi(x_2-0.9)^2)] \quad x_1, x_2 \in [0,1] \quad (14)$$

Table 4. Results obtained with the algorithm here proposed.

RBFs	<i>f1</i>				<i>Test</i>			
	<i>Training</i>				<i>Test</i>			
	<i>Best</i>	<i>Worst</i>	<i>Mean</i>	<i>Std.dev.</i>	<i>Best</i>	<i>Worst</i>	<i>Mean</i>	<i>Std.dev.</i>
18	0.0066	0.0203	0.0128	0.0028	0.0097	0.0233	0.0156	0.0028
21	0.0050	0.0129	0.0076	0.0023	0.0063	0.0270	0.0099	0.0042
24	0.0037	0.0105	0.0065	0.0019	0.0057	0.0131	0.0086	0.0020
27	0.0029	0.0097	0.0052	0.0015	0.0041	0.0118	0.0065	0.0017
29	0.0027	0.0070	0.0046	0.0013	0.0044	0.0097	0.0067	0.0018
RBFs	<i>f2</i>				<i>Test</i>			
	<i>Training</i>				<i>Test</i>			
	<i>Best</i>	<i>Worst</i>	<i>Mean</i>	<i>Std.dev.</i>	<i>Best</i>	<i>Worst</i>	<i>Mean</i>	<i>Std.dev.</i>
20	0.0269	0.1056	0.0443	0.0138	0.0231	0.1390	0.0406	0.0193
23	0.0187	0.0641	0.0347	0.0096	0.0220	0.0515	0.0293	0.0058
25	0.0149	0.0447	0.0264	0.0064	0.0242	0.0434	0.0297	0.0044
28	0.0117	0.0328	0.0187	0.0038	0.0119	0.0184	0.0186	0.0030
30	0.0109	0.0222	0.0170	0.0026	0.0241	0.0241	0.0249	0.0039
RBFs	<i>f3</i>				<i>Test</i>			
	<i>Training</i>				<i>Test</i>			
	<i>Best</i>	<i>Worst</i>	<i>Mean</i>	<i>Std.dev.</i>	<i>Best</i>	<i>Worst</i>	<i>Mean</i>	<i>Std.dev.</i>
10	0.0520	0.1230	0.0940	0.0191	0.0601	0.1295	0.0993	0.0181
15	0.0190	0.0537	0.0281	0.0078	0.0229	0.0607	0.0330	0.0085
19	0.0094	0.0353	0.0190	0.0063	0.0131	0.0364	0.0227	0.0053
23	0.0091	0.0318	0.0144	0.0044	0.0095	0.0354	0.0158	0.0050
30	0.0052	0.0128	0.0080	0.0020	0.0074	0.0197	0.0139	0.0022

The procedure was compared to other procedures applied to the same benchmark: (a) [2] (a work where a multilayer perceptron is used); (b) Pomares [11] (a method that obtain fuzzy rules); and (c) González [5] (an algorithm that evolves populations of RBFNs). The NRMSE values provided by these algorithms are given in Table 5, where n_n is the number of neurons (RBFs in our case) or fuzzy rules obtained, and n_p is the number of free parameters to be determined by the method.

Table 5. Results for other methods

Method	<i>f1</i>		<i>f2</i>		<i>f3</i>	
	$n_n(n_p)$	NRMSE	$n_p(n_n)$	NRMSE	$n_p(n_n)$	NRMSE
Cherkassky	40(161)	0.017	40(161)	0.052	40(161)	0.008
Pomares	(164)	0.017	(64)	0.061	(19)	0.278
	(73)	0.007	(82)	0.028	(30)	0.104
	(104)	0.003	(113)	0.015	(55)	0.041
González	18(72)	0.0204	20(80)	0.0505	10(40)	0.1343
	21(84)	0.0176	23(92)	0.0362	15(60)	0.0459
	24(96)	0.0153	25(100)	0.0265	19(76)	0.0299
	27(108)	0.0103	28(112)	0.0226	23(92)	0.0155
	29(116)	0.0089	30(120)	0.0203	30(120)	0.0118

As can be seen, our procedure improves the performance of other, more complex or mature, methods.

4. Conclusions

This paper presents an evolutionary procedure to design networks of radial basis functions that propose a solution to the credit assignment problem. Here, the fitness of the population of RBFs depends on three factors: the weight and width of each RBF, the closeness to other RBFs, and the error inside each RBF width. The procedure uses a fuzzy logic system to control the application of some operators to the RBFs, and a multi-objective technique to rank the RBFs. Good experimental results are obtained, and the method performs better than other, more mature or complex, methods that have been previously proposed.

Acknowledgements. This paper has been supported by projects TIC2000-1348 and DPI2001-3219 of the Spanish Ministerio de Ciencia y Tecnología.

5. References

- [1] Angelone,P.J.; Saunders, G.M.; Pollack, J.B.: "An evolutionary algorithm that constructs recurrent neural networks". IEEE Trans. Neural Networks, Vol.5, No.1, pp.54-65. January, 1994.
- [2] V.Cherkassky, D.Gehring, F.Mulier, "Comparison of adaptive methods for function estimation from samples", IEE Trans. Neural Networks, 7 (4), pp. 969-984, 1996.
- [3] Coello, C.C.: "An Updated Survey of Evolutionary Multiobjective Optimization Techniques: State of the art and future trends". Congress on Evolutionary Computation, CEC'99, pp.3-13, 1999
- [4] N. García-Pedrajas, C. Hervás-Martínez, J. Muñoz-Pérez, "Multi-objective cooperative coevolution of artificial neural networks (multi-objective cooperative networks)", Neural Networks, vol. 15,pp. 1259-1278, 2002
- [5] González,J.; Rojas,I.; Pomares,H.; Ortega,J :"RBF Neural Networks, Multiobjective Optimization and Time Series Forecasting", Lecture Notes in Computer Science, (2084), pp.498-505. 2001
- [6] Haykin, *Neural Networks. A comprehensive foundation*, 2nd. NJ: Prentice-Hall, 1999.
- [7] Mandani, E. H.; Assilian, S.: "An experiment in linguistic synthesis with a fuzzy logic controller". Int. J. Man-Machine Studies, vol. 7, no. 1, pp. 1-13, 1975
- [8] Maniezzo, V.: "Genetic evolution of the topology and weight distribution of neural networks". IEEE Trans. on Neural Networks, Vol.5, No.1, pp.39-53. January, 1994.
- [9] Marquardt, D. W "An algorithm for least-squares estimation of non-linear inequalities," SIAM J. Appl. Math., vol. 11, pp. 431-441, 1963.
- [10] Moody, J; Darken, C.: "Fast learning networks of locally-tuned processing units," Neural Computation, vol. 3 n. 4 pp.579-588, 1991
- [11] Pomares, H.; Rojas, I.; Ortega, J.; González, J.; Prieto, A.: "A systematic approach to Self-Generating Fuzzy Rule-Table for Function Approximation". IEEE Trans. Syst., Man, and Cyber., Part B, 30(3):431-447. 2000
- [12] Rivera, A.; Ortega, J.; Prieto A.: "Design of RBF networks by cooperative/competitive evolution of units" International Conference on Artificial Neural Networks and Genetic Algorithms, ICANNGA 2001. April 2001.
- [13] Rivera, A.; Ortega, J.; del Jesús, M.; González, J; "Optimización de RBFs mediante cooperación-competición de neuronas y algoritmos de minimización de error". II Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados. MAEB03, Febrero 2003.
- [14] Smalz, R.; Conrad, M.: "Combining evolution with credit apportionment: a new learning algorithm for neural nets". Neural Networks, Vol.7, No. 2, pp.341-351, 1994.
- [15] Whitehead, B. A.; Choate, T.D."Cooperative-competitive genetic evolution of Radial Basis Function centers and widths for time series prediction". IEEE Trans. on Neural Networks, Vol 7, No. 4, pp.869-880. July, 1996.
- [16] Widrow, B.; Lehr, M.A.: "30 Years of adaptative neural networks: perceptron, madaline and backpropagation". Proceedings of the IEEE, Vol. 78 n. 9, September 1990.

Studying the Capacity of Grammatical Encoding to Generate FNN Architectures

Germán Gutiérrez, Beatriz García, José M. Molina, Araceli Sanchis

SCALAB. Departamento de Informática. Universidad Carlos III de Madrid.

Avda. Universidad 30, 28911-Leganés (Madrid)-SPAIN.

{ggutierrez, masm}@inf.uc3m.es, molina@ia.uc3m.es, 100027154@alumnos.uc3m.es

Abstract: Many methods to codify Artificial Neural Networks have been developed to avoid the defects of direct encoding schema, improving the search into the solution's space. A method to estimate how the search space is covered and how are the movements along search process applying genetic operators is needed in order to evaluate the different encoding strategies for Feedforward Neural Networks. A first step of this method is considered with two encoding strategies, a direct encoding method and an indirect encoding scheme based on graph grammars: generative capacity, how many different architectures the method is able to generate.

1. Introduction

In the last years, many works have been centered toward automatic resolution of the design of neural networks architecture [1,2,3,4,5,6]. Two representation approaches exist to find the optimum net architecture using Genetic Algorithms (GA). One based on the complete representation of all the possible nodes and/or connections, Direct Encoding Methods (DEM), where every node and/or connection of the neural network is indicated into the chromosome of the GA [7,8]. And other based on an indirect and compact representation of the architecture instead of codifying the complete network, called Indirect Encoding Methods (IEM) [1,9,10].

IEM are applied in order to reduce the length of the genotype, the search space, and to make the problem more scalable. No exhaustive analysis of those features exists in the literature. The main problem is the definition of an objective measure to evaluate different codifications and search strategies. The measure should be able to evaluate the generative capacity and the efficiency of the search strategy, analyzing the different neural network architectures generated with several methods.

Typically, a good problem representation requires the representation of any possible solution, and a good search strategy requires that similar genotypes produce similar architectures. In this way, all the search space is covered and the fitness function is able to guide the search.

The goal is to evaluate and compared different encoding algorithms. In this way, we need to estimate the neural networks generated, to analyze the generative capacity of the method (how many different architectures the method is able to generate) over

the whole search space and the search strategy (how the Genetic Algorithm generates neural networks).

Hamming distance might be used to show how the neural network space is covered by the encoding scheme. However, indirect approaches could not be applied because each chromosome and its corresponding binary matrix have a different meaning depending on the expansion method. We proposed an objective measure to evaluate in the same way different methods and the evaluation is made over the generated neural networks architecture space. Thus different encoding methods can be compared.

Our interest is focused in the evolution of architectures of feedforward neural networks (FNN) with one hidden layer, just the number of hidden nodes and connections between different layers. The task of weight training is left to be carried out by the back propagation algorithm. So, only the topology of FNN will be codified into the chromosome.

In this work, the generative capacity of two different encoding methods are studied and compared: a DEM that represents every possible connection of a FNN and an IEM based on Graph Grammars with a expansion process to obtain the binary matrix that represent the FNN architecture. Using the procedure defined in this paper, we could analyze the generative capacity (representation) and the search space in the domain of genotypes of direct codification scheme and compare with the indirect one. An objective assessment and the results concerning to the generative capacity (ability to cover the FNN architecture space) of both encoding methods proposed are shown. A random initial population of chromosomes will be created for both encoding methods, the expansion process will be applied to each chromosome for the IEM and conclusions obtained from the histograms of neural networks achieve will be exposed. In [11], a preliminary study of the representation capacity of another IEM based on Cellular Automata [12] is presented using that objective measure. The results showed that the IEM based on cellular approach is able to cover the search space.

Section 2 and 3 is related with the direct and indirect encoding schemes, respectively, used in this work. Section 4 describes the objective assessment. The generative capacity of the encoding schemes is shown in section 5. Finally, some conclusions are presented in section 6

2. Direct Encoding Method

In direct encoding schema the most usual way to codify the architecture is to place the chromosome (a string) as the concatenation of rows (or columns) of the binary connections matrix, where each cell represents the existence or absence of a set connection between two nodes. Since we are interested only in FFN architectures with one hidden layer, rejecting any other type of neural network, as recurrent neural networks, the direct encoding scheme used in this work only represents useful connections, connections between nodes of one layer and immediately next layer. Thus, the length of the chromosome is reduced.

The direct encoding scheme used in this work is based on a binary matrix of dimension $\text{Dim}_x \times \text{Dim}_y$, where Dim_x is equal to the number of input neurons plus the number of output neurons and Dim_y corresponds with the maximum number of

hidden neurons to be considered (see Fig 1). To relate that matrix with an architecture of a FNN with one hidden layer, the meaning for the grid position (i,j) is defined as follows. Let's n the number of input neurons; if $i \leq n$ then (i,j) represents a connection between the i -th input neuron and the j -th hidden neuron; if $i > n$, (i,j) represents a connection between the j -th hidden neuron and the $(i-n)$ -th output neuron. That relation is shown in Fig 1. The chromosome is just the concatenation of the matrix rows and $\text{Dim}_x \times \text{Dim}_y$ gives the length of the chromosome.

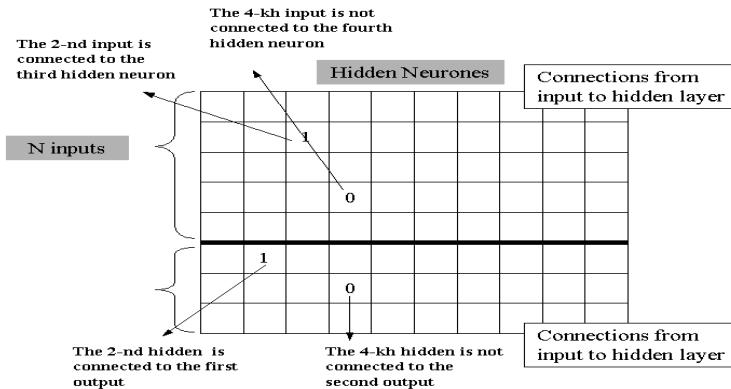


Fig. 1. Matrix connection for direct encoding method.

3. Method based on Graph Grammar/Indirect Encoding Method

In 1990, Kitano [9] presents a new method for designing neural networks by means of Genetic Algorithms, where neural networks are represented through graph grammars, codified in the chromosomes of the individuals. The selected indirect encoding method is based on an improvement of Graph Grammars used by Kitano.[13]

Whole system is composed of three modules, Genetic Algorithm Module, Grammar Module and Neural Network Module. In [14] a detailed description of Neural Network and Genetic Algorithm Modules can be found. The Grammar Module takes charge of generating FNN architecture from a Graph Grammar indicated into the chromosome. The generated FNN is trained and relevant information about the goodness of FNN is used as the fitness value for the genetic algorithm module. And finally the genetic algorithm module takes charge of generates Graph Grammars of next generation.

Into each chromosome a bi-dimensional Graph Grammar is described and a bi-dimensional word, a binary matrix, is obtained, in a deterministic way, from the bi-dimensional Graph Grammar [13]. The binary matrix obtained represents a connection matrix, where each element a_{xy} represents the existence ($a_{xy}=1$) or the absence ($a_{xy}=0$) of connection from node labelled x to node labelled y .

Since the goal in this paper is to compare the generative capacity of both methods, the binary matrix, the word, obtained from the Graph Grammar codified into a

chromosome has to be translated into the matrix connection described in section 2. Every backward connection or every direct link from a input node to a output node is not considered on the translation process: Only connections corresponding to a link from input nodes to a hidden node or a link from hidden node to output node are hold.

4. Generative Capacity

The goal is to evaluate different encoding algorithms studying the generative capacity (how many different architectures the method is able to generate) over the whole FNN architecture space and the search strategy (how the Genetic Algorithm generates neural networks).

In order to analyze the behavior of the different encoding methods, two kind of information have to be studied:

The first one is related with the ability of the methods to generate an initial population of chromosome covering the complete FNN space. As the initial population has a finite size, the individuals are a subset of the search space. Clearly, an algorithm that generates random individuals following an uniform distribution over the search space is better than algorithms that introduce a bias in the initial population.

The second one is related with the way the algorithm searches in the search space. The best situation to apply genetic algorithm is that small changes in the genotype produce small changes in the fitness value. This situation avoids the epistasis problem. This problem could not be addressed directly (as in other genetic approximations) with the indirect encoding methods because of the particular process introduced to translate the genotype in to the neural network.

We propose to approximate the points one and two, analyzing the number of hidden neurons (nh) and the number of connections (nc). We propose to calculate the histogram of nh and nc for the initial population (point one), and to evaluate random walks (representing in a 3-D graph the fitness landscape over nh and nc) for the search (point two).

In this work, results concerning to the first point are shown. The study involves the following steps:

1. Generate a random initial population of chromosomes with a uniform distribution.
2. Apply the translation process for each encoding method and for each chromosome to obtain the correspondent binary matrix or neural network architecture.
3. Make a histogram of the number of hidden neurons (nh) in the binary matrix.
4. Make a histogram of connections (nc) in the binary matrix.
5. 3-D histogram considering the bidimensional space $nh \times nc$.
6. Represent the percentage of connections for each FNN obtained with the same number of hidden nodes

5. Experimental Results

In order to analyze the generative capacity of both encoding methods, a initial population of 10^4 chromosomes is randomly generated for each encoding method. In both schemes every value for an element of the chromosome has the same probability than the other values. With direct encoding method, in any element of the binary matrix (the chromosome) there is a “0” or a “1” with the same probability (1/2). For IEM based on graph grammars the chromosome codifies a bidimensional grammar. Into the chromosome, the production rules (the right side of the production rules), the expansion method and the level of recursion are codified in binary. Each allele has the same probability (1/2) to be a “0” or a “1”.

In these results a generic domain is used, with 2 inputs and 2outputs ($n_I = n_O = 2$). Then the maximum number of hidden neurons of FNN considered for DEM is 16 ($(n_I + n_O)^2$), [11]. Therefore the length of the chromosome for direct encoding scheme is 64, equal to the maximum number of connection considered.

In case of IEM, the length of the chromosome is fixed and not depends on the problem. The maximum number of hidden nodes and connections depends on the size of the binary matrix immediately obtained from the derivation process for each chromosome. The size of that matrix depends on which rules are codified in the chromosome and especially in the value of two parameters that stop the derivation process [13].

For both encoding schemes if in the binary matrix the elements of a column are 0, it means that the hidden node that this column represents has not any connection from input layer or to output layer. In this case, it could be eliminated from the FNN.

For grammatical approach only the results within range 0 to 16 of hidden nodes and 0 to 64 of connections are exposed to compare with results of DEM, that are limited to these values for the size of FNN obtained. But bigger FNNs are obtained for the IEM.

Fig 2 (a) (b) show FNN with the same number of hidden nodes obtained from the initial population of chromosome. FNN architectures with the same number of connections are shown if Fig 3 (a) (b) for both direct and indirect encoding scheme.

In Fig 4 (a) and Figure 4 (b) the 3-D histograms for both encoding schema is shown. In them, the number of FNN with a same number of hidden nodes and a same number of connections is displayed.

In Fig 5 (a) (b) the percentage of connectivity for FNN with the same number of hidden nodes for both encoding schemes are shown. The maximum number of connections for a given number of hidden nodes (h) is ($h \times (n_I + n_O)$), the minimum number of connection is ($n_I + n_O$)

In Fig 2 (a) and Fig 3 (a) is observed that for direct encoding scheme the probability density function, corresponding to this discrete distribution, could be obtained exactly from the probability that a column has no element equal to “1” and an element of matrix connection (the chromosome in DEM) is equal to “1”. It is clear that must be a binomial distribution. But for indirect encoding (see Fig 2(b) and 3(b)) it is not feasible, at least too complex, to obtain exactly the discrete probability density function that corresponding to the histogram of nc and nh .

Besides, in Fig 5 (a) (b) can be observed that the percentages of connectivity of FNN obtained with both encoding schemes. EM are limited to some values (35 - 75 %). By opposite, for IEM (see Fig 5 (b)) FNN with percentages of connectivity ranging from minimum, $\left(\frac{100}{n_i+n_o} \right)$, to maximum are obtained.

All these advantages for IEM, considering a initial set of chromosomes randomly generated with a uniform distribution, could make the search more feasible for the evolutionary algorithm.

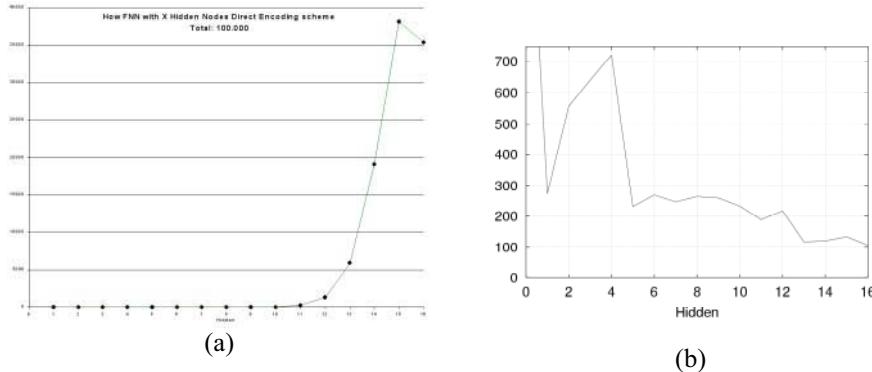


Fig. 2.. FFN have the same number of hidden nodes. DEM (a), IEM (b).

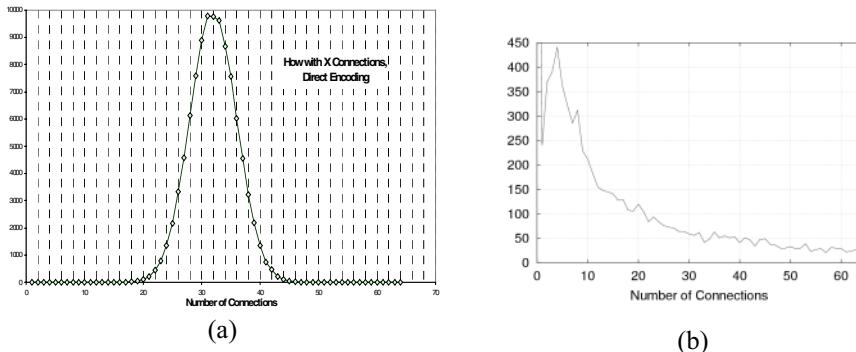


Fig. 3. How many FNN have the same number of connections. DEM (a), IEM (b).

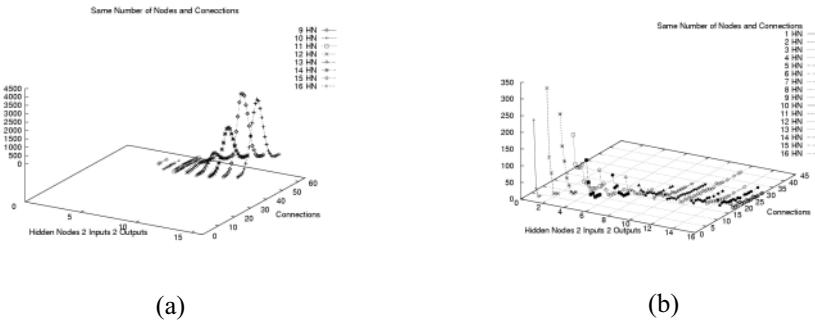


Fig. 4. FNN have the same number of hidden nodes and connections. DEM (a), IEM (b).

As it is observed in all figures, it is clear that indirect encoding scheme cover the FNN space for a initial random population of chromosomes better than the DEM. With DEM no FNN with less than 10 hidden nodes and only FNN with a number of connections ranging between 18 and 47 are obtained. Therefore only a limited subset of space of FNN is obtained from a uniform distribution of chromosome space.

On the contrary, with IEM and considering only the subset of FNN space with less or equal than 16 hidden nodes and 64 connections, the space of FNN is covered. Even though FNN with few hidden nodes and connections are majority obtained.

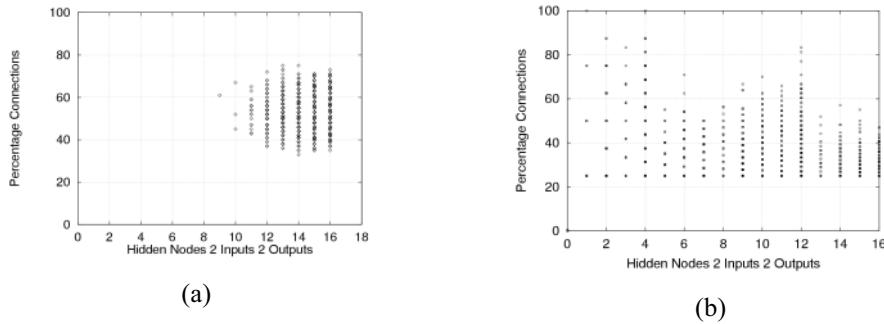


Fig. 5. Percentage of connectivity. DEM (a), IEM (b).

6. Conclusions

Indirect Encoding methods are applied in order to reduce the length of chromosomes. As a Evolutionary Algorithm begins the search with a random initial population, it is important that it covers as completely as possible the search space. The analysis presented in this work shows that direct encoding methods do not carry out this condition, on the contrary than the indirect encoding scheme based on grammars aforesaid.

In future works, estimations of what density distribution functions correspond with the histograms shown will be study. Random walks over the space of chromosomes will be evaluated in order to get to know how the genetic algorithm look for in the search space for both encoding methods, direct and indirect.

References

- [1] S. Harp, Samad T. and Guha A. Towards the Genetic Synthesis of Neural Networks. Proceedings of the Third International Conference on Genetic Algorithms and their applications, pp 360-369, San Mateo, CA, USA, 1989.
- [2] G.F. Miller, P.M. Todd and S.U. Hegde. Designing neural networks using genetic algorithms. In Proc. of the third international conference on genetic algorithms and their applications, pp 379-384, San Mateo, CA, USA, 1989.
- [3] S. Harp, Samad T. and Guha A. Designing Application-Specific Neural Networks using the Genetic Algorithm, Advances in Neural Information Processing Systems, vol2, 447-454, 1990.
- [4] F. Gruau. Genetic Synthesis of Boolean Neural Networks with a Cell Rewriting Developmental Process. Proc. of COGANN-92 International Workshop on Combinations of Genetic Algorithms and Neural Networks, pp. 55-74, IEEE Computer Society Press, 1990.
- [5] F. Gruau. "Neural Network Synthesis Using Cellular Encoding and the Genetic Algorithm". Ph.D. Thesis, Ecole Normale Supérieure de Lyon, (1994).
- [6] F. Gruau. Automatic Definition of Modular Neural Networks. Adaptive Behavior, vol. 2, 3, 151-183, 1995.
- [7] T. Ash. Dynamic Node Creation in Backpropagation Networks ICS Report 8901, The Institute for Cognitive Science, University of California, San Diego (Saiensu-sh, 1988), 1988.
- [8] D.B. Fogel, Fogel L.J. and Porto V.W. Evolving Neural Network, Biological Cybernetics, 63, 487-493, 1990.
- [9] H. Kitano. Designing Neural Networks using Genetic Algorithms with Graph Generation System, Complex Systems, 4, 461-476, 1990.
- [10] J.W.L. Merril and R.F. Port. Fractally configured Neural Networks. Neural Networks, 4, 53-60, 1991.
- [11] G. Gutierrez, J.M. Molina, I. Galván, A. Sanchis. An Objetive Measure to Compare some Automatic Generation Methodsof NN Architectures. IEEE International Conference on Systems, Man and Cybernetics, 2002. Tunisia.
- [12] J. G. Gutierrez, I. M. Galván, A. Sanchis, J. M. Molina. Generative Capacities of Cellular Automata Codification for Evolution of NN Codification. International Conference on Artificial Neural Networks 2002 (ICANN 2002). LNCS 2415 Jose R. Dorronsoro (Ed.) Springer.
- [13] M. A. Guinea, G. Gutierrez, I. Galván, A. Sanchis, J. M. Molina. Generative Capacities of Grammars Codification for Evolution of NN Architectures. International Conference on Evolutionary Computation, WCCI, 2002. USA.
- [14] G. Gutiérrez, P. Isasi, J.M. Molina, A. Sanchís and I. M. Galván. Evolutionary Cellular Configurations for Designing Feed-Forward Neural Networks Architectures. Connectionist Models of neurons, Learning Processes and Artificial Intelligence. 6th International Work-Conference on Artificial Neural Networks, IWANN 2001. Proceedings, Part I. LNCS 2084. J. Mira, A Prieto (Eds.) Springer.

Optimal Phasor Measurement Unit Placement using Genetic Algorithms

F. J. Marín¹, F. García-Lagos², G. Joya², F. Sandoval²

¹ Dpto. Electrónica. E. T. S. I. Informática.

Universidad de Málaga. Campus Teatinos s/n. 29071 Málaga, Spain

fjmarin@uma.es

² Dpto. Tecnología Electrónica. E. T. S. I. Telecomunicación.

Universidad de Málaga. Campus Teatinos s/n. 29071 Málaga, Spain

{lagos, joya, sandoval}@dte.uma.es

Abstract. A genetic algorithm-based procedure for solving the optimal phasor measurement units (PMUs) placement problem is presented. A PMU measures voltage and current phasors at the bus where is placed. These measurements must realize the electrical power system observable, in order to perform the state estimation. Our proposal have two essential advantages: (1) it determines the minimal number of PMUs and their geographic distribution making the network observable; (2) it shows the relationship between the number of current phasors that must be measured on each PMUs and the necessary number of PMUs for a given network. The placement algorithm has been tested on 4 standards IEEE-bus, ranging in size from 14 to 118 buses.

1 Introduction

In the context of an Energy Management System (EMS), State Estimation is the operation oriented to calculate the most probable values of the system state variables: magnitude and phase angle of voltage in all buses. This operation may be considered as the core of the EMS, because from it, a model of the system can be realized, so allowing every other operation. Classical State Estimation uses several measurements, such as active and reactive power flows and voltage magnitude, which are obtained from different sources and, probably, in different instants. This approach must solve a complex non linear equation system because the voltage phase angles (second state variables) have been traditionally non available due to the impossibility of their synchronized measurement. This synchronization is necessary since to angle phase values must be measured at the same instant time. However, recent advances of Global Position Systems (GPS) have led to new insights into the measurement process (small size, low cost, high accuracy, and, mainly, synchronization capability). Thus, the development in time synchronising (less than 1 microsecond, this is 0.018 degrees for the 50 Hz Spanish power system) allows to measure phasor values (magnitude and angle phase) of the electrical variables (voltage and current). This new approach reduces the original complex system of

equations to a linear system, converting the classical state estimation to a faster and numerically stable problem consisting of multiplying a constant matrix by the phasor measurement vector.

From this perspective, it is non strange that Phasor Measurement Units (PMUs) are more and more substituting or complementing the classical Remote Terminal Units (RTUs) in the Supervisory Control And Data Acquisition (SCADA) systems. PMUs measure voltage and current phasors and can repeatedly calculate watts, frequency and phase angle during a power line cycle.

If either phasors or no phasors are available for a particular power system, a minimum number of appropriately distributed measurements are needed in order to carry out its state estimation. When these measurements are available, we say the system is observable. The formulation of the observability analysis can be performed by means of measurement assignments to branches with topological algorithms (topological or graphical observability) [1], [2], [3], or well as numerical approach of the measurement and network loop equations (numerical or floating-point observability) [4], [5]. In this paper we are interested in the topological observability.

In this work, we face the problem of finding an optimal distribution of the minimal number of PMUs in a powers system in order to guarantee the observability for a linear state estimation. In [5] a minimal set of PMUs is found using a dual search algorithm: a bisecting search and a simulated annealing algorithm. In this approach, PMUs must have the capability of measuring the voltage phasor in a bus and the current phasor of all the lines concurring to this bus, with indifference of this number of lines. This is a non realistic condition because the number of possible measurements is clearly limited in real PMUs and it can not be always adjusted to the number of lines of every bus. In this sense, our work presents two essential results: on one hand, we obtain the minimal set of PMUs that guarantees the system observability as a function of the number of current phasors that a PMU can measure; and, on the other hand, we found the minimal number of measurement that a PMU must measure in order to guarantee the observability with the lowest number of PMUs. This number of measurements of a PMU is, in all the studied cases, minor than the maximum number of lines by bus required in [5]. Thus, we show that it is not necessary to dispose of PMUs with the maximal number of current phasors to obtain the minor number of PMUs.

Our method is based on a Genetic Algorithm [6] which presents as the main characteristic the selected encoding for each individual of the population. This encoding is carried out by means of a Chromosome representing the connectivity of the power system. The placement method has been tested on 4 standards IEEE-bus (14-bus, 30 bus, 57-bus, and 118-bus) networks.

2 Observability analysis

If the set of measurements is sufficient in number and well-distributed geographically, the state estimator will give an estimate of the system state (voltages in module and phase of each bus). In this case, we say the network is observable. If

we have a voltage phasor measurement in each bus, then the network is observable. Besides, if the current phasor are measured, then the measurement set is redundant, and state estimation process can filtered erroneous measured. Unfortunately, in general it's no possible to measure each voltage and current phasors of the system. Thus, the system must have the sufficient number of PMUs making the system observable.

The non-linear equations relating the measurements and the state vector are:

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) + \mathbf{e} \quad (1)$$

where \mathbf{z} is the $(m \times 1)$ measurement vector, $\mathbf{h}(\cdot)$ is the $(m \times 1)$ vector of non-linear functions, \mathbf{x} is the $(2N \times 1)$ true state vector, \mathbf{e} is the $(m \times 1)$ additive measurement error vector, m is the number of measurements, and N is the number of buses.

In our case, we are interested in power systems provided with PMUs (linear model). That is why we must rewrite equation (1) for the linear case. Given an N -bus network, and m measurements of voltage and current phasors, the linear equations relating the measurements and the state vector are:

$$\mathbf{z} = \mathbf{H}(\mathbf{x}) + \mathbf{e} \quad (2)$$

where the vector \mathbf{z} is linearly related to the n -dimensional state vector \mathbf{x} containing N -bus voltage phasors, resulting in $n=2N-1$ state variables, $\mathbf{H}(\cdot)$ is the $(m \times N)$ matrix, and \mathbf{e} is the $(m \times 1)$ additive measurement error vector.

The observability of the power system can be assessed as: if matrix \mathbf{H} is of full rank and well-conditioned, then the network is considered to be observable [3]. This is called numerical observability. There is another different observability concept, the topological observability, which is defined as the existence of at least one tree that connects all the buses through branches with a metered or a calculated current phasor (pseudo-phasor). The essential advantage of this approach is the fastness due to the lack of floating point operations.

As we have mentioned above, the number of buses observable is inferred by the rank of the matrix \mathbf{H} (the gain matrix). So, the number of buses whose state variables can not be calculated (noted as Φ) is determined by [5]:

$$\Phi = N - (\text{Rank}(\mathbf{H}) + 1) / 2 \quad (3)$$

When the system is completely observable, there are not unobservable buses, and the result of equation (3) is zero ($\Phi=0$).

3 Genetic Algorithms

Genetic Algorithms (GAs) are evolutive algorithms of search inspired on natural selection mechanisms. They work over a population of *individuals* or *chromosomes*, each representing a potential solution of the problem. A chromosome is composed by genes. A *fitness* value is assigned to each individual indicating the goodness of the

solution that it represents. At each generation, a new set of possible solutions is created by selecting the current individuals according their fitness (*selection*), and applying them recombination (*crossover*) and *mutation operators* borrowed from the natural genetic.

The GA starts with a randomly generated initial population of individuals, each of them representing, by a particular codification, a possible solution of our problem. More appropriated the fitness value of an individual is more probability for being selected as a parent for the next generation it has. The crossover operator exchanges substrings between two individuals; once two individuals are selected, a position (the crossover position) is randomly obtained; from this position the exchanges between the substrings take place to yield two new offspring which replace to the parents. The mutation operator randomly changes the value of one gene of the individual. This operator first randomly selects a gene of the individual, and then changes its value. After these operators are applied, a new generation is obtained and the fitness value of its individuals must be calculated. The process is repeated until a predetermined stop criterion is fulfilled. After several generations, the result is a high number of fitted individuals in the populations.

4 PMU Placement. Representation and Methodology

Our objective is find the minimal number of PMUs placed in an N-bus power system (and their distribution) so that the system is topologically observable. This is a NP-complete problem, with a solution space of 2^N possible combinations. It can be mathematically expressed as follows:

$$\min \sum_{i=1}^N PMU_i \quad / \quad \Phi = 0 \quad (4)$$

where $PMU_i = 1$ if there is a PMU placed in the bus i and $PMU_i = 0$ in other case; and $\Phi = 0$ means that the system is observable (3). This combinatorial optimization problem is difficult to resolve due to the discrete nature of the search space with multiple local minima.

From the measurements supplied by the availables PMUs, the remain variables can be calculate by applying the Kirchhoff's and Ohm's laws. The Kirchhoff's Current Law is used to calculate one unknown incident current phasor in a node if all remaining incident current phasors of this node are known. The Kirchhoff's Voltage Law is applied as follows: if in a bus the voltage phasor and one incident current phasor are both known, then the voltage phasor at other end of the branch can be calculated. Finally, the Ohm's law can be used to calculate the current phasor of a branch if voltage phasors are known at both ends.

The chromosome of an individual is formed by binary genes, where each bus of the network has a gene that indicates the existence of a PMU in that bus (bus gene), and one gene for each line of this bus (line genes), used to indicate if the corresponding current measurement is available in this branch. Also, each bus has

an additional gene for each incident branch. Both speed and proper running are closely dependent on an appropriate codification. In our case, individuals have as components as buses in the networks. Figures 1 and 2 show the structure of the chromosome of an individual for the IEEE 14-bus network, where we have shown the optimal solution for a PMU with 4 current phasors and 1 voltage phasor (PMUs on buses 2, 6 and 9).

In order to calculate the fitness of each individual, an observability analysis is carried out. The goal is to make the whole network observable with a minimum set of PMU. With phasor measurement, the observability implies that each bus of the network must have one phasor voltage measurement or a phasor voltage pseudo-measurement. The pseudo-measurement can be created using Kirchhoff's voltage and current laws. Our method uses the following rules: a pseudo-measurement can be assigned to an unobservable bus if one of its neighbours has a phasor voltage measurement and the line between both has a current phasor measurement or pseudo-measurement. When the buses at the end of a branch are observables, a current phasor pseudo-measurement can be assigned to this branch. Finally, if a bus has all their branches observables except one, a pseudo-measurement can be assigned to this branch. The observability analysis recursively applies these rules to find the number of buses that are not observables (N_H) and the number of PMUs in the network (N_{PMU}), using an optimized graph search procedure. The fitness function is calculated according to equation (5):

$$f = a N_{PMU} + b N_H + c N_{PMU} N_H \quad (5)$$

where a, b and c are constants. Our best experimental results are obtained using a=1, b= 2, and c=1.

The placement problem is resolved through the following steps:

Step 1. Given the power network, build its corresponding *chromosome*.

Step 2. Create the initial population.

Step 3. For each individual, calculate its fitness function by equation (5).

Step 4. Apply selection operator.

Step 5. Apply crossover operator.

Step 6. Apply mutation operator.

Step 7. Apply elitist strategy.

Step 8. Go to step 3 until G generations are completed.

Once computed the fitness value (step 3) for each individual, the algorithm proceeds selecting individuals for reproduction (step 4) on the basis of their relative fitness (in our case, smaller is the fitness value better is the solution represented by an individual). The default selection strategy works by a *roulette wheel* mechanism to probabilistically select individuals based on the relative fitness value associated to each individual. In step 5, individuals are selected and mutually crossed to produce new individuals that take part of both parent's genetic material. Mutation (step 6) is applied to produce a new genetic structure. In our case, mutation is applied over two randomly selected gene positions (row of buses and columns of branches) on the individual. At this point, a new generation can start with the fitness evaluation for this new population. In our case, it is desirable to deterministically maintain one o

more of the best fitted individuals through the use of an *elitist strategy*, in step 7. The algorithm works during G iterations (step 8).

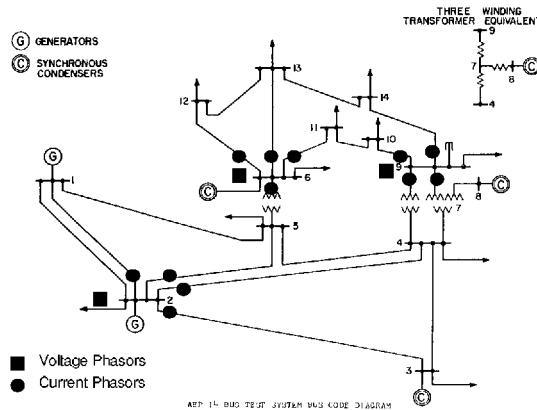


Fig. 1. IEEE 14-bus power system with 3 PMUs.

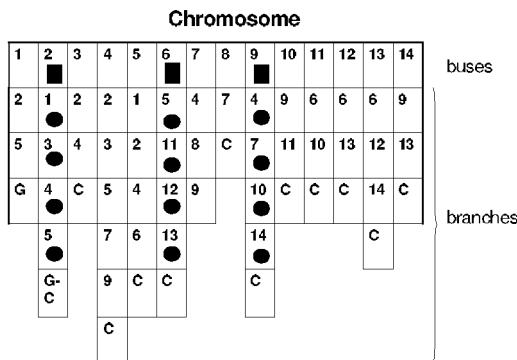


Fig. 2. Codification of the IEEE 14-bus network for PMUs on buses 2, 6, 9.

5 Simulation Results

The placement algorithm has been tested on 4 standards IEEE-bus (14-bus, 30-bus, 57-bus, and 118-bus) networks. Results obtained with the GA are given in Table 1, where λ is the minimal number of PMUs needed to make the system topological observable (equation (3)); $\delta = (\lambda / \text{buses}) * 100$ is the percentage between λ and the number of buses of the network; b is the largest number of incident branches of each network; ph is the necessary number of current phasors that must be measured in each PMU to give an optimal solution; and R is the ratio between ph and b .

Table 1. Optimal Number of PMUs

Power System	λ	$\delta (\%)$	b	ph	R=ph/b
IEEE 14-bus	3	21.4	6	4	0.66
IEEE 30-bus	7	23.3	8	3	0.38
IEEE 57-bus	12	21.1	7	3	0.43
IEEE 118-bus	29	24.6	10	5	0.50

Figure 3 shows the number of needed PMUs as a function of the number of available current phasor measurements by PMU (it is supposed that, in addition, one voltage phasor measurement must be available for each PMU) for different IEEE networks. For the IEEE-14 network the best PMUs placement consists of 3 PMUs placed on buses 2, 6 and 9. These PMUs must guarantee 4 current phasor measurements. For the IEEE 30-bus, 7 PMUs are needed on the buses 1, 5, 10, 12, 15, 20, and 27, with the readiness of only 3 current phasor measurements in each PMU. We have implemented an exhaustive algorithm for observability on IEEE 14-bus, and 30-bus networks, obtaining the same results than the GA [7]. For the IEEE 57-bus 12 PMUs are needed with only 3 current phasor measurements. Finally, for the IEEE 118-bus, 29 PMUs are needed with only 5 current phasors measurements, which means the 50% of the maximal number of incident branches in this network.

Results show that, for each power system, there are a minimal number of current phasor measurements by PMUs that guarantee the optimal solution. Thus, using PMUs with an unlimited number of current phasors measurements, as in [5], is an extreme and unnecessary case of possible solutions.

6 Conclusions

In this work we present a Genetic Algorithm based method for solving the Phasor Measurement Units placement problem in a power system. This placement must guarantee the topological observability of the system. On one hand, the proposed method determines the optimal number of PMUs and their geographic distribution as a function of the number of available phasor measurements in a PMU. On the other hand, the method determines the minimal number of phasor measured by a PMU in order to guarantee the minimal number of PMUs. This characteristics give a marked degree of realism to the method, which is not presents in the other reported solutions, which need to use PMUs with as phasor measurers as the maximal number of concurrent lines in all buses of the system. A very distinctive aspect of the method is the way as the individuals are codified in the GA. This codification permits a rapid and clear quantification of the fitness value of each individual.

Currently, our application will be tested to include restrictions such as obligation or prohibition to place a PMU on a determined bus of the network, possibility to use PMUs of different manufacturers, where the number of current phasors available is different, etc.

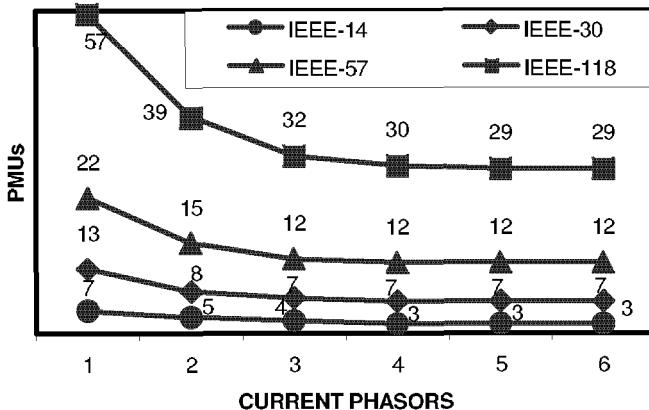


Fig. 3. Relationship between PMUs and number of current phasors that must be measured for topological observability in IEEE 14, 30, 57 and 118- bus.

Acknowledgments

This work has been partially supported by Eliop S.A., under contract No. 8.06/58.1508 and MCYT, project No. TIC 2001-1758. Thanks are due to Red Eléctrica de España (REE), Subdirección Regional Centro, for comments and suggestions.

References

- Monticelli, A. , Wu, F.F.: Network Observability: Theory. *IEEE Trans. on Power Apparatus and Systems*, vol. 104, no. 5, (1985) 1042-1048
- Expósito, A., Abur, A.: Generalized Observability Analysis and Measurement Classification. *IEEE Trans. On Power Systems*, vol. 13, no. 3, (1998) 1090-1095
- Gou, B., Abur, A.: A Direct Numerical Method for Observability Analysis. *IEEE Trans. on Power Systems*, vol. 15, no. 2, (2000) 625-630
- Clements, K.A., Krumpholz, G.R., Davis, P.W.: Power System State Estimation with Measurement Deficiency: An Algorithm that Determines the Maximal Observable Subnetwork. *IEEE Trans. On Power Apparatus and Systems*, vol. 101, no. 7, (1982) 3044-3052
- Baldwin, T.L., Mili, L., Boisen, M.B., Adapa R.: Power System Observability with Minimal Phasor Measurement Placement. *IEEE Trans. on Power Systems*, vol. 8, no. 2, (1993) 707-715
- Goldberg, D.E.: *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Reading, MA (1989)
- García-Lagos, F., Joya G., Marín, F.J., Sandoval, F.: Observability Analysis using phasors. Technical Report. Dpto. Tecnología Electrónica. Universidad de Málaga, (2003)

On the Evolutionary Inference of Temporal Boolean Networks

Carlos Cotta

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

Abstract. The problem of inferring genetic networks under the Temporal Boolean Network model is considered here. This is a very hard problem for which an heuristic approach is proposed. This approach is based on the use of evolutionary algorithms (EAs) to refine the results of a specialized algorithm (ID3). Experimental results provide support for the usefulness of this approach, showing a consistent enhancement of the ID3 solutions.

1 Introduction

Genetic Networks are tools for modeling gene interactions that can be used to explain a certain observed biological behavior. In a genetic network, the functionality of a gene is described by the global effect it has on the cell, as a result of its interaction with other genes. Such interactions must be inferred from experimental data in order to construct an adequate genetic network for modeling the biological process under scrutiny. The way this inference is done depends on the particular genetic-network model used. For example, one can consider Bayesian Networks [9], Boolean Networks [18], Petri Nets [12], and Weight Matrices [19] among others. In this work we will focus on Temporal Boolean Networks (TBNS) [17], a generalization of the Boolean Network model that takes into account the time-series nature of the data, and tries to incorporate into the model the possible existence of delayed regulatory interactions among genes. The basic notions about this model of genetic networks will be presented in Section 2.

Several exact algorithms have been proposed for the inference of TBNS from experimental data. It turns out that the inference problem is very hard, and some of these algorithms can become impractical up from a certain problem size. For this reason, the use of heuristic approaches is in order. In this sense, we will study the utilization of evolutionary algorithms (EAs) [4] for this purpose. The details of the application of EAs to this problem will be provided in Section 3.

It is important to notice that we consider the utilization of EAs not substituting but complementing other existing algorithms. In effect, EAs can successfully use information provided by the latter in order to build improved solutions. Section 4 will provide empirical evidence of this fact. This work will close with some discussion and prospects for future research in Section 5.

2 Background

In this section we will introduce the essentials of the TBN model considered in this work. Firstly, some basic definitions and notation will be presented in Subsection 2.1. Subsequently, we will provide some highlights on existing algorithms for the inference of TBNs in Subsection 2.2.

2.1 Temporal Boolean Networks

As mentioned in the previous section, TBNs are a generalization of Boolean Networks. It would be then appropriate to start defining the latter.

A Boolean Network is a tuple $BN(G, F)$, where $G(V, E)$ is a directed graph, and $F = \{f_v \mid v \in V\}$ is a set of Boolean functions, such that f_v is attached to vertex v . Let $K_G(v)$ be the in-degree of vertex v in graph G . Then, the signature of the Boolean function attached to v is $f_v : \mathbb{B}^{K_G(v)} \longrightarrow \mathbb{B}$, where $\mathbb{B} = \{0, 1\}$. Each vertex v represents a different gene, and can be labeled with a binary value $-0(\text{OFF})$ or $1(\text{ON})-$ that indicates its expression level. The existence of an edge (v', v) in G indicates that the value of gene v' exerts some influence on the value of gene v , i.e., v' is a regulatory factor of v . The precise way in which this regulation works, and how it is combined with other regulatory factors that bind to v is captured by means of the corresponding Boolean function f_v .

Having defined these concepts, the dynamics of the genetic network is modeled as follows: first of all, time is divided in discrete time steps. In each step, the values of all genes are synchronously updated using the attached Boolean functions. More precisely, the value of gene v at time $t+1$ is computed by having function f_v being fed with the values at time t of all genes $v', v'', \dots, v^{(K_G(v))}$ for which incoming edges to v exist. The output of f_v for this particular input is the expression level of v at time $t+1$.

Let $\psi : V \longrightarrow \mathbb{B}$ be a labeling of vertices in V , and let Ψ be the set of all such labelings. Now, we define an example as a pair $(I, O) \in \Psi^2$. A Boolean network BN is said to be *consistent* with this example if it holds for each $v \in V$ that $f_v(I(v'), \dots, I(v^{(K_G(v))})) = O(v)$. In the context of time-series data, we will have a sequence $\Lambda = \langle \lambda_1, \dots, \lambda_m \rangle \in \Psi^m$, and we will be interested in checking the consistency of the network with examples $(\lambda_i, \lambda_{i+1})$, $1 \leq i < m$. Plainly, this represents the capability of the network for reproducing the observed data. In case the network were not consistent with the whole time series, it would make sense to measure the degree of agreement with it. This is done using the *accuracy* measure. First, let the error $\epsilon_{BN}^A(v)$ for gene v be defined as the fraction of states of v that were incorrectly predicted by the network across the time series Λ . Now the accuracy of the network for a time series Λ is

$$\text{accuracy}_{BN}(\Lambda) = 1 - \frac{1}{|V|} \sum_{v \in V} \epsilon_{BN}^A(v). \quad (1)$$

It can be easily seen that the accuracy of a network is 1.0 if, and only if, it is fully consistent with the time series Λ . In case more than one time series were

available, i.e., A_1, \dots, A_q , a combined accuracy measure can be computed by averaging the accuracy values for all time series A_i , $1 \leq i \leq q$.

Temporal Boolean networks are a extension of BNs that tries to overcome some of the limitations of the basic model, i.e., the synchronous updating of gene values. Unlike the case of plain Boolean networks, in TBNs the state of a gene at a certain time step is not only relevant for the next time step; on the contrary, its regulatory influence can span across several time steps.

In order to formally define a TBN we only need to specify a labeling of edges in the graph. Thus, a TBN is a tuple $TBN(G, F)$, where F has the same interpretation as above, and G is a graph $G(V, E, \varphi)$ with φ being defined as a function $\varphi : E \rightarrow \mathbb{N}$. Were a certain edge (v', v) be labeled with l , the state of v' at time t would be relevant for computing the new state of v at time $t + (l+1)$. It is then easy to see that a plain Boolean network is a particular case of TBN in which all edges are labeled with 0. We will denote by T the maximum size of the time window in which the state of a gene can have regulatory effects, i.e., $T = 1 + \max_{e \in E} \varphi(e)$ (hence $T = 1$ for a plain Boolean network).

2.2 Inference of TBNs

A number of algorithms have been proposed for the inference of TBNs from data. Actually most of them correspond to the simpler Boolean network model, but they can be readily extended to deal with TBNs.

One of the first algorithms that were proposed is REVEAL (REVerse Engineering ALgorithm) [11]. This algorithm combines Information-Theory tools and exhaustive search in order to find a network consistent with the data. More precisely, the algorithm tries to identify an adequate set of inputs for each gene g by considering all possible k -tuples of genes for increasing values of k (1 up to n , the total number of genes). A k -tuple Γ is considered a valid input if the *mutual information* between gene g and genes in Γ is equal to the entropy of g , i.e., Γ is enough to explain all state variations for g . If the underlying model behind the data is known to have in-degree bounded by K , then the complexity of this algorithm can be shown to be $O(mn \binom{n}{K}) = O(mn^{K+1})$, where m is the size of the data set.

Another algorithm was proposed by Akutsu *et al.* [1]. This algorithm was termed BOOL-1 in a later work [2], and consists of examining all possible K -tuples of inputs, testing all Boolean functions of each K -tuple until a consistent set of inputs is found. As it is the case for REVEAL, this algorithm has $O(mn^{K+1})$ worst-case complexity.

Since the number of network topologies is $O(n^{K+1})$, these algorithms can be essentially viewed as brute-force approaches. As a matter of fact, it has been shown by Cotta and Moscato [6] that this bound cannot be improved unless a very unlikely condition regarding the structure of parameterized complexity classes held [8].

This hardness has motivated the utilization of heuristic approaches. A popular one is ID3 [13], a well known algorithm in Machine Learning. This algorithm is based on the incremental construction of the input set for each variable using a

greedy search guided by the information gain criterion. The approach presented in next section is based on the synergistic utilization of evolutionary algorithms and existing heuristics such as ID3.

3 An EA-based Approach for Inferring TBNs

In this section we will show how to deploy EAs on the inference problem we are considering. Firstly, the representation and evaluation function utilized are described in Subsection 3.1. Then, the procedures used for initialization and reproduction are discussed in Subsection 3.2.

3.1 Representation and Evaluation

Choosing a representation in which the relevant properties of solutions for the problem considered are explicitly shown [14] is crucial for the success of the EA. In the problem of inferring TBNs from data, the relevant properties of solutions are the edges among genes, and their labels. The chosen representation is then based on these units. More precisely, we have considered solutions as a list of triplets (v', v, l) , $v, v' \in V$, $l \in \{0, \dots, T - 1\}$, each one corresponding to an edge present in the TBN. It turns out that such lists can be efficiently stored and manipulated in terms of an $(n \times n)$ -matrix M of natural numbers in the range $[0, T]$; having $M_{ij} > 0$ implies that an edge exists from v_i to v_j , and its label is $M_{ij} - 1$ (conversely, the edge list can be viewed as a sparse encoding of this matrix). This matrix is not allowed to have more than K non-zero entries per column, i.e., K is the maximum in-degree of any node. Repairing by pruning the input set of a node is performed whenever its size is greater than allowed, e.g., after applying mutation.

EA individuals thus specify the wiring of the TBN. When submitted to evaluation, the Boolean functions attached to each vertex must be firstly learned. This is done by scanning the data set Λ and assigning the most common output to each input combination found in Λ . This is similar to the maximum likelihood estimation of parameters done in, e.g., Bayesian networks, and can be done in linear time in the number of patterns in Λ . Once these functions have been determined, the TBN is effectively evaluated using accuracy –recall Equation (1)– as the fitness function to be maximized.

3.2 Initialization and Operators

In order to have the EA started, it is necessary to create the initial population of solutions. This is typically addressed by randomly generating the desired number of solutions. When the alphabet used for representing solutions has low cardinality (as it is here the case), this random initialization provides a more or less uniform sample of the solution space. The EA can subsequently start exploring the wide area covered by the initial population, in search of the most promising regions.

This random initialization can be complemented with the inclusion of heuristic solutions in the initial population. The EA can thus benefit from the existence of other algorithms, using the solutions they provide both for refinement and as information source. This is termed *seeding*, and it is known to be very beneficial in terms of convergence speed, and quality of the solutions achieved [16]. In order to avoid injected solutions taking over the whole population in a few iterations, a non-fitness-proportionate selection mechanism (focusing on qualitative fitness comparisons instead of on quantitative comparisons) must be used. This has been the approach we have considered: injecting solutions provided by the ID3 algorithm in the initial population, and using binary tournament [5].

Once the population has been created, the EA conducts its search using two reproductive operators, recombination and mutation. One of the advantages of the representation chosen is the fact that it is *orthogonal* [15], that is, any $(n \times n)$ -matrix M of natural numbers in the range $[0, T]$ represents a valid TBN (there are no constraints, e.g., regarding the presence of cycles as it is the case in bayesian networks). This means that classical operators can be used. In this case, we have considered the utilization of uniform crossover (UX) for recombination. This operator ensures a high interchange of gene-values during recombination. As to mutation, it is performed using a simple mechanism: a random edge (v', v, l) is selected and removed from the solution if it is present, or added to it otherwise (any (v', v, l') that existed would then be removed). The objective here is obtaining an unbiased source of fresh information that keep diversity in the population.

4 Experimentation

The experiments have been conducted in order to test the ability of the proposed approach in recovering specific TBNs using a sample of their output. To do so, we have randomly generated networks of different sizes, in-degrees, and temporal delays. More precisely we have considered networks of $n = 16$ and $n = 32$ genes, in-degrees of $K = 5$ and $K = 7$ edges, and temporal delays from 0 up to 3 time steps (i.e., $1 \leq T \leq 4$). For each parameter combination we have generated 5 different networks, following the procedure described in [17]. Once all networks have been generated, their output is sampled by randomly setting their states for T time steps, and then iterating their behavior for 100 time steps. This is repeated 5 times, so a total of 5 time series of 100 patterns each is obtained for each network.

The evolutionary algorithm used is a (50,1)-EA, i.e., an EA with a population size of 50 individuals, generating a single descendant in each step, and inserting this new individual in the population by substituting the worst one. Recombination is done with probability $p_R = 0.9$, and mutation of genes with probability $p_M = 1/n^2$. Binary tournament is used for selecting individuals for reproduction as mentioned in Subsection 3.2. Finally, the EA is run for a total number of evaluations $maxevals = 10,000$ for $n = 16$, and $maxevals = 20,000$ for $n = 32$.

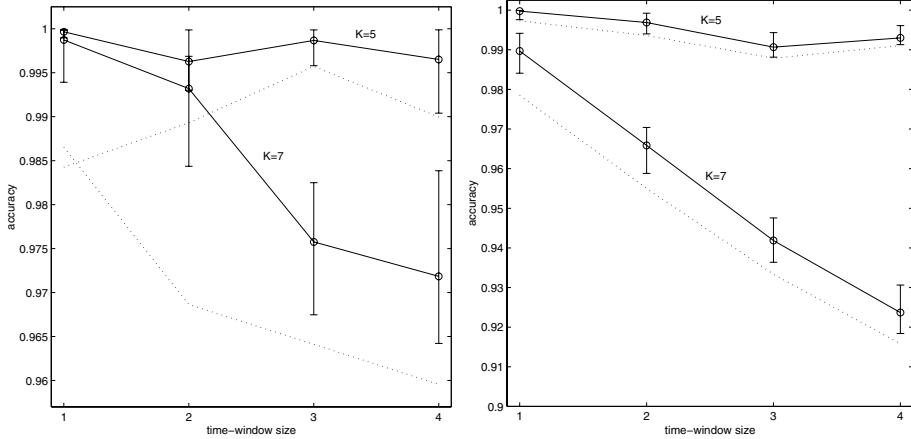


Fig. 1. Accuracy of the TBNs generated by the EA (solid line). The results of ID3 are included as a reference (dotted line). (Left) $n = 16$ (right) $n = 32$

Table 1. Percentage of success of the EA and the ID3 algorithm in finding the target TBN for different network sizes, in-degree bounds, and time-window sizes.

algorithm	size	$K = 5$				$K = 7$			
		$T = 1$	$T = 2$	$T = 3$	$T = 4$	$T = 1$	$T = 2$	$T = 3$	$T = 4$
EA	$n = 16$	95%	69%	82%	71%	81%	43%	0%	1%
	$n = 32$	97%	60%	26%	51%	9%	0%	0%	0%
ID3	$n = 16$	20%	20%	40%	40%	20%	0%	0%	0%
	$n = 32$	60%	40%	20%	40%	0%	0%	0%	0%

Using these parameters, the EA is run 20 times on each of these 5 time-series data sets. Besides considering the accuracy of the solutions obtained, their similarity to the target network is also measured. This can be done using the *sensitivity* and *specificity* metrics [10]. These are respectively defined as the fraction of matched edges (edges in both the generated solution and in the original TBN) with respect to the total number of edges in the generated solution or in the original TBN. Notice that both metrics will yield the same result when the original network and the solution provided by the EA indicate the same number of dependencies for each gene.

First of all, the accuracy results of the EA are shown in Fig. 1. Notice that in all cases the EA manages to improve the accuracy of the ID3 solution. This improvement is generally larger for low values of T . In this case, the EA is able to find the target network most of the times. This can be corroborated by taking a look at Fig. 2, where sensitivity values are shown, and Table 1, where the number of successful runs (i.e., runs in which the target network is found) is shown. As it can be seen, the performance line is located very close to 1.0 in this case (specificity values are equivalent in all cases to those for sensitivity, due to

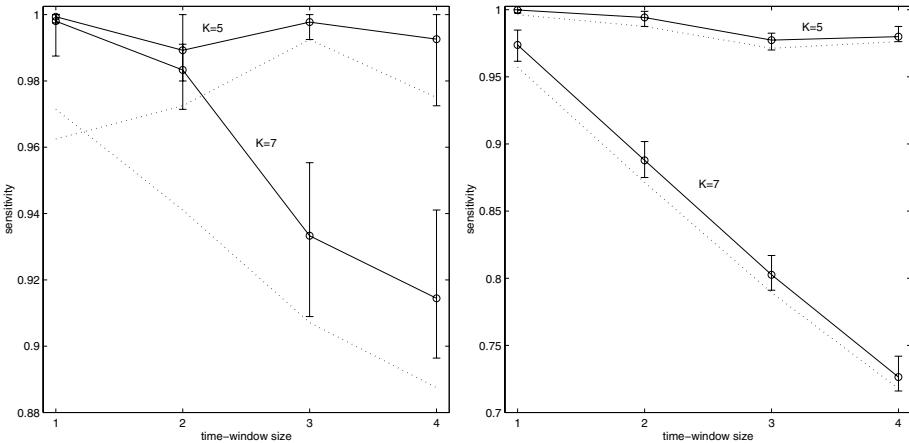


Fig. 2. Sensitivity of the TBNs generated by the EA (solid line). The results of ID3 are included as a reference (dotted line). Specificity values are equivalent in this case. (Left) $n = 16$ (right) $n = 32$

the fact that the networks generated by the EA have exactly K inputs per gene, as target networks do).

The performance curve of the EA drops at a slightly higher rate in the case $K = 7$. However, this is also the case for ID3, and reflects the higher difficulty of this set of instances. Actually, this in-degree value can be considered as rather high since genes are believed to be influenced on average by no more than eight to ten other genes [3]. Obviously, the search space is also larger for increasing K , so longer evolution times may be required in this case.

5 Conclusions

An evolutionary approach for the inference of genetic networks has been presented in this work. This approach can exploit pre-existing heuristics by incorporating the solutions they provide to the initial population. This seeding boosts convergence towards probably optimal solutions.

The empirical results obtained from its evaluation are encouraging. It has been shown that the results of a specialized algorithm (ID3) can be consistently improved. In this sense, we would like to emphasize the generalizability of the approach: ID3 has been considered as the heuristic for seeding the initial population, but it can be easily changed for any other heuristic; hence, if an improved algorithm is devised, it can be readily used to seed the initial population.

Future work will be directed to test the applicability of this approach in more complex scenarios, e.g., assuming the existence of white noise in the data. Work is in progress here. Another interesting line for future developments is the utilization of ideas taken from a related field as it is the inference of bayesian networks [7].

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

References

1. T. Akutsu, S. Miyano, and S. Kuhara. Identification of genetic networks from a small number of gene expression patterns under the boolean network model. *Pacific Symposium on Biocomputing*, 4:17–28, 1999.
2. T. Akutsu, S. Miyano, and S. Kuhara. Inferring qualitative relations in genetic networks and metabolic pathways. *Bioinformatics*, 16(8):727–734, 2000.
3. A. Arnone and B. Davidson. The hardwiring of development: Organization and function of genomic regulatory systems. *Development*, 124:1851–1864, 1997.
4. T. Bäck, D.B. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation*. Oxford University Press, New York NY, 1997.
5. T. Bickle and L. Thiele. A mathematical analysis of tournament selection. In L.J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 9–16, San Francisco, CA, 1995. Morgan Kaufmann.
6. C. Cotta and P. Moscato. The k -FEATURE SET problem is $W[2]$ -complete. *Journal of Computer and Systems Science*, 2003. In press.
7. C. Cotta and J. Muruzábal. Towards a more efficient evolutionary induction of bayesian networks. In *Parallel Problem Solving from Nature VII*, volume 2439 of *Lecture Notes in Computer Science*, pages 730–739. Springer-Verlag, Berlin Heidelberg, 2002.
8. R. Downey and M. Fellows. *Parameterized Complexity*. Springer-Verlag, 1998.
9. N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using bayesian networks to analyze expression data. *Journal of Computational Biology*, 7:601–620, 2000.
10. T.E. Ideker, V. Thorsson, and R.M. Karp. Discovery of regulatory interactions through perturbation: Inference and experimental design. *Pacific Symposium on Biocomputing*, 5:305–316, 2000.
11. S. Liang, S. Fuhrman, and R. Somogyi. REVEAL, a general reverse engineering algorithm for inference of genetic network architectures. *Pacific Symposium on Biocomputing*, 3:18–29, 1997.
12. H. Matsuno, A. Doi, M. Nagasaki, and S. Miyano. Hybrid Petri net representation of gene regulatory network. *Pacific Symposium on Biocomputing*, 5:338–349, 2000.
13. J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.
14. N.J. Radcliffe. Non-linear genetic representations. In R. Männer and B. Manderick, editors, *Parallel problem Solving from Nature II*, pages 259–268, Amsterdam, 1992. Elsevier Science Publishers.
15. N.J. Radcliffe. The algebra of genetic algorithms. *Annals of Mathematics and Artificial Intelligence*, 10:339–384, 1994.
16. C. Ramsey and J.J. Grefenstette. Case-based initialization of genetic algorithms. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 84–91, San Mateo CA, 1993. Morgan Kauffman.
17. A. Silvescu and V. Honavar. Temporal boolean network models of genetic networks and their inference from gene expression time series. *Complex Systems*, 13(1):54–70, 2001.
18. R. Somogyi and C. Sniegoski. Modeling the complexity of genetic networks: Understanding multigenetic and pleiotropic regulation. *Complexity*, 1(6):45–63, 1996.
19. D.C. Weaver, C.T. Workman, and G.D. Stormo. Modeling regulatory networks with weight matrices. *Pacific Symposium on Biocomputing*, 4:112–123, 1999.

Specifying evolutionary algorithms in XML

Juan Julián Merelo Guervós, Pedro Ángel Castillo Valdivieso, Gustavo Romero López, Maribel García Arenas

GeNeura Team, Depto. Arquitectura y Tecnología de Computadores Universidad de Granada (Spain) todos@geneura.ugr.es, <http://geneura.ugr.es>

Abstract. This paper describes EvoSpec, an XML-based language for describing evolutionary computation algorithms, used by the Perl evolutionary computation library `Algorithm::Evolutionary`. It presents a survey on the state of the art of specification languages for evolutionary algorithms, emphasizing those that are based on XML, gives some guidelines for implementing XML-based specification languages, and finally, describes EvoSpec and how it can be used to represent EC experiments.

1 Introduction

Specification languages have a long history in soft computing, and there are several reasons for this: they represent problem descriptions closer to the problem domain, they also make easier to interchange information between different applications that implement the same algorithms, and, at the same time, language design gives some insight on the underlying structure of the algorithms. In the case of evolutionary algorithms, abstracting them, revealing its internal structure, is akin to a great unified theory of evolutionary algorithms, reducing them to a population, a main loop that renovates them via variation operators, and data structures used to represent solutions. This general structure represents all evolutionary algorithms, from EDAs (estimation of distribution algorithms) [1] to genetic programming [2] or grammatical evolution [3].

Once this general structure has been described, it is easy to create variations on it, which can be represented using an specification algorithm. Different types of main loops could be represented using different data structures, there would be different ways of representing operators, and so on and so forth. Finally, each language should have its parser or interpreter, that takes the problem representation and generates source code compilable by a high-level language compiler, or that runs the problem itself. The problem with a new language that is created from scratch, easy as it might be to learn, are manifold: it must be accompanied by its parser (although those are not difficult to write), and it usually presents a steep learning curve, so most soft computing specification languages, such as NeuDL (Neural Description Language, [?]) or EASEA [4] are not really widely used outside the community that created it. We are not claiming that we have the silver bullet for solving this problem, but using a language that is widely understood, whose interpretation is quite straightforward from any computer

language, will be a step in the right direction. That is why we are choosing XML (eXtensible Markup Language) as a (meta)language for implementing the system we are presenting in this paper [5–7].

XML is a language proposed and steered by the World Wide Web consortium as a framework for data interchange; but it is more than that, it is a new way of understanding programming: document-oriented programming as opposed to input/output or event-driven programming. XML documents can also be understood as programs, whose output is the tree that univocally represents the document (the DOM, Document Object Model, tree). Besides, there are tools that can handle XML very easily: XML databases such as Tamino, XML transformations such as SAX filters and XSLT logicsheets [5], which can extract easily information from them, and browsers such as the latest versions of Internet Explorer and Mozilla/Netscape which can display them directly.

Having a common (or at least easily processable) language for all evolutionary applications enables another, scientific, objective: reproducibility. It is almost impossible to reproduce an experiment from what is shown in most papers published today. Even if all requisites and parameters are clearly explained, you still have to program the algorithm, using whatever system you choose. But if scientists committed themselves to publish an specification and source code to interpret it along with the paper, reproducibility would be enhanced, it would be much easier to compare results, and scientist could build on what other scientist have done. Besides, the integration of XML and EC make possible to use XML documents as another native data structure, at the same level as bit-strings or LISP S-functions. In this paper we will present a possible solution to this conundrum: the *EvoSpec* programming language, derived from XML, with which any (serial) evolutionary computation experiment can be represented. We will also introduce the `Algorithm::Evolutionary` evolutionary computation Perl module, a library built around it, which can parse and run algorithms specified using this language (as well as run by itself).

The rest of the paper is organized as follows: next section (section 2) is devoted to reviewing the state of the art in specification of evolutionary algorithms using XML. *EvoSpec*, the language proposed here, will be presented in section 3, followed on some tips on how to use it 4. Finally, the paper will be closed with some conclusions and a discussion of results presented.

2 State of the art

XML and evolutionary algorithms realms have remained traditionally apart from each other, but there have been a few researchers that have approached the topic. Among them, we would like to highlight the EAML language (Evolutionary Algorithm Markup Language, an XML dialect) described by Veenhuis and others [8]. This language consists of set of XML tags with a defined and fixed semantics, specifies an evolutionary algorithm, from which C++ code can be generated. EAML attempts to be an exhaustive description of an evolutionary algorithm, but it does not really achieve its target, since, for instance, variation

operators are tags, instead of being attributes values of a generic *operator* tag; this means that adding a new operator (say, a n-ary *orgy* operator) would require a redesign of the language's syntax (defined through a DTD, or Data Type Dictionary). There is no provision for user-defined operators or data structures. Tag attributes can have any value or a constrained one (depending on how you define them in the DTD), but validated XML requires tags to be defined in its DTD first. In any case, for a restricted form of a evolutionary algorithm it is a valid approach, and it is a step in the right direction; *EvoSpec*, which is used by *Algorithm::Evolutionary*, tries to overcome these errors. This language has been used by the XML-Men project [9], that cites as its main disadvantage the fact that the fitness function must be written in Java. The Open BEAGLE [10] also support XML as a language for serialization, but this feature is not documented, and, in fact, the only way to obtain an XML file is to run one of the examples; a DTD or data type dictionary is not provided either; however, the language design is much more flexible, structured around a *register* with *entries*, and a *Vivarium*, or runtime section. However, the meaning of the XML file is left to the program to interpret (for instance, there is no fitness function definition); that is, it acts more as a configuration file for running a program, than a problem description language. The serialization part (enabling the program to write and read its internal state) is much more important than the problem description part: the algorithm is just described by a set of keys and its values.

There are also other languages for evolutionary algorithm description, such as EASEA [4]; however, this is a language designed to generate C++ and Java (in either of 3 different EC libraries) programs from its description, and is not intended as an universal evolutionary algorithm description language that can be parsed from any other language, or generate programs in any language (it could be used as such, however).

3 Description of *EvoSpec*, a language for describing evolutionary computation experiments

The design principles involved in creating *EvoSpec* is that it should be easily extensible, general, and able to describe any evolutionary computation experiment. The main goal for achieving this is to stick to very few XML elements, and allow many specific features to be passed as parameters, which are not checked by the XML parser. That, in turn, has the main disadvantage that it is up to the application that uses the XML file to find out which features are present and which are not, but the main advantage is that it fulfills the three criteria outlined above.

Another criteria is that the XML format would be used not only for configuration files, but also as serialization format for each and every object in the environment. This also calls for a recursive structure of the XML file, but means a delegation of responsibilities that is very convenient for XML parsing.

```

<?xml version="1.0"?>
  <!DOCTYPE ea SYSTEM "EvoSpec.dtd">
  <ea version='0.4'>
    <initial>
      <op name='Creator' >
        <param name='number' value='20' />
        <param name='class' value='Vector' /> <!-- and other params.. -->
      </op>
      <op name='Easy' type='unary'>
        <param name='selrate' value='0.4' />
        <param name='maxgen' value='100' />
        <code type='eval' language='perl'> <src><![CDATA[code goes here]]>
          </src></code>
      <op name='GaussianMutation' type='unary' rate='1'>
        <param name='avg' value='0' /> <!-- other params .. -->
      </op> <!-- other ops: Crossover... -->
    </op>
  </initial>
</ea>

```

Fig. 1. XML document that describes a simple genetic algorithm for the "tide" function

As is seen in figure 1, there are only a few elements or tags involved in describing an evolutionary algorithm: initial, pop, op, param, code, src, indi, atom and ea.

ea is the root tag, that is, the tag that encompasses the rest of the XML file; all valid XML files need a (single) root element.

The initial element is equivalent to a configuration section, and contains itself several sections. In principle, the number of sections is not bounded, but it is usual to have only two, one related to the individuals, and another related to the population. Imagine that section as a set of transformations to be performed in turn. If you put a pop descriptor at the beginning of the initial element, and then an operator, the operator will be applied to the population; if it is an operator that creates the population, and then several others, the population will be created and then each operator will be applied in turn. The initial element is like a pipeline that applies operators to a population, which is implied or not, in document order (*document order* in the context of XML parsing indicates the order in which tags appear in the document, parsing from the first to the last line)

So, basically what can be found or included in this section are operators and population elements. A population element might look like this:

```

<pop size='20'>
  <param name='type' value='BitString' />
  <param name='length' value='64' />
</pop> 

```

. The pop element has an attribute (**size**) that indicates the number of individuals it will contain, and then, using parameters, describes the kind of individuals that are going to be created. This section includes parameters that describe the individual type and any other thing needed to create it; the parameters names used are **type**, for the class in which the individuals are going to be instantiated, and other parameters such as **length**, that will be passed to the class constructor. Every class will use its own parameters; in both cases, checking that the class or data structure that it describes indeed exists within the particular EC implementation is made in runtime.

After, or instead of, the declaration of the initial population, there will be declarations for one or more operators that will act on the population. If there is no population element, the first operator should create the population, just like this one:

```
<op name='Creator' >
  <param name='number' value='20' />
  <param name='class' value='BitString' />
  <param name='options'>
    <param name='length' value='64' />
  </param>
</op>
```

The Creator used here is a 0-ary population operator: takes a population, which could have no individuals, and adds individuals as specified by the XML fragment; in this case, 20 individuals, of class BitString, with the **length** as an option that will be passed to the individual constructor. This form of the operator, actually, is completely equivalent to the population declaration shown above, and has exactly the same effect.

Once the population has been created, a second operator, which will most likely be the genetic algorithm itself, must be applied to it. In this case, we apply the **Easy** operator: an easy-to-build and -use genetic algorithm, with default mutation and crossover operator. A couple of parameters, given using the **param** tag, are passed to the operator: the selection rate (**selrate**) and the maximum number of generations (**maxgen**). Each operator has its own parameters, which are normally (or should be) specified in its documentation. Wrong parameters will not hurt, but if required parameters are not set, the runtime system will complain (as it should), in any case, it should be noted that checking that the parameters, its values and its quantity is correct is left to the environment that reads them.

There is another part that is required: the fitness function, which falls within the **code** tag. The language and the type of function should be specified (although currently, in the **Algorithm::Evolutionary** implementation, only the Perl language is supported, as well as only the **eval** or **fitness** function). Any source code could be passed this way. Since this entry includes free Perl code, it must be enclosed in a **CData** tag, so that it is not interpreted by the XML parser.

Although they could be in any place within the **op** tag, following the **code** tag we can find the operators that are going to be included within the **Easy**

genetic algorithm . In this case, the operators are Mutation and Crossover, and they follow the same convention as its parent. Type and rate must be specified, as well as any parameters. These parameters are interpreted in the context of the operator, so they could be applied, in principle, to the population or to individuals within it; its exact functionality is, once again, left to the runtime system.

Another design requisite was that the *EvoSpec* language should be valid also for serialization, that is, the XML document could represent the state of the algorithm at any point during execution; normally, whatever is generated during runtime will go after the initial element, and will usually consist of a population section like this one.

```
<pop> <indi type='BitString' fitness='16' > <atom>0</atom>
[... and so on]
<atom>1</atom> <atom>0</atom> <atom>0</atom> </indi> </pop>
```

The population is composed of individuals *indi*, which could be divided into atoms. Atoms do not always make sense, only in the case in which the data structure that is going to be evolved can be easily divided into a series of identical (sub-)data structures. In this case, a BitString is a string of 0/1s, which constitute its atoms.

4 Using *EvoSpec*

EvoSpec is now part of the EC framework `Algorithm::Evolutionary`, which is written in Perl and released as a GPL (GNU's General Public License), and available from its SourceForge site <http://opeal.sf.net> and from the CPAN (Comprehensive Perl Archive Network, at <http://www.cpan.org>). This Perl module is built from the ground up to be able to understand and generate XML files everywhere, and has been presented elsewhere ([11–15]).

The distribution includes also some general tools for these XML files: a validator (written in Perl), that checks the XML file against the DTD and issues an error if it contains some grammatical error, an XSL document (which is a program that transforms XML documents into other documents) that converts the output of an experiment into XML, and several examples that cover all features of the *EvoSpec* language. In principle, it should not be too difficult for anybody using other framework that can understand XML to use *EvoSpec*. There are two possible ways of doing it: write a parser that reads it, and instantiates objects, or generates code, according to the specifications made in the file, or write an XSL file that transforms an *EvoSpec* document to any other format; any XML file, written using any XML grammar, can be transformed in any other XML file by several possible methods, one of which is XSLT logicsheets (XML stylesheet language for transformations). Given two grammars, and knowing what are the correspondences between them, it should not be too difficult to transform from one document to another.

The main problem this transformation presents is the amount of information given as tags or as attributes values or as element content; converting tags to tags is easy, if the equivalence is known, same goes for converting tags to text, but converting text to tags is more difficult, since parsing text is not a task that XML tools do easily, and, besides, plain text is not, per se, structured.

The EvoSpec documents can be interpreted using this simple Perl script:

```
use Algorithm::Evolutionary::Experiment;
open (X, shift );
my $xml = join("", <X>);
close X;
$xp = Algorithm::Evolutionary::Experiment->fromXML( $xml );
my $popRef = $xp->go();
```

which shows its expressivity as a problem description language.

5 Conclusion, discussion, and future work

This paper introduces EvoSpec, an XML-based language for describing Evolutionary Computation Problems. Documents written using this language can be parsed to generate source code in another language, which can in turn be used to run an evolutionary algorithm, or used directly to generate all objects needed for running an evolutionary algorithm. The module that uses it, `Evolutionary::Algorithm`, does precisely this. The same language can be used to serialize evolutionary algorithms; this serialization can in turn be stored permanently in a file or database, or converted to other (readable) format such as PDF or HTML.

The fact that this language completely describes evolutionary algorithms can be used to create permanent stores of evolutionary experiments; it would be advisable, for the reproducibility of evolutionary algorithms, to create repositories of experiments, so that anybody who wants to check whether the results presented in a paper are correct, could do it; besides, repositories of XML documents along with any code needed to interpret them would make incremental development much easier, since it would be easier to build on what is already done than write an application from scratch.

Finally, future work will involve the development of filters to convert EvoSpec documents to other formats, be them XML or not, and continue development of the `Evolutionary::Algorithm` module in Perl. Also, interfaces for other EC libraries such as JEO [16]. EvoSpec will also be extended to include the specification of parallel algorithms.

Acknowledgments

This work has been supported in part by INTAS 97-30950.

References

1. H. Mühlenbein and G. Paaf. From recombination of genes to the estimation of distributions: Binary parameters, 1996.
2. M. Grauer and D. B. Pressmar, editors. *Applied Parallel and Distributed Optimization*, volume 367 of *Lecture Notes in Mathematical Systems and Economics*. Springer, Berlin, 1991.
3. Anthony Brabazon, Michael O'Neill, Robin Matthews, and Conor Ryan. Grammatical evolution and corporate failure prediction. In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1011–1018, New York, 9–13 July 2002. Morgan Kaufmann Publishers.
4. P. Collet, E. Lutton, M. Schoenauer, and J. Louchet. Take it EASEA. In Marc Schoenauer, Kalyanmoy Deb, Guenter Rudolph, Xin Yao, and Hans-Paul Schwefel Evelynne Lutton, Juan Julian Merelo, editors, *PPSN VI*, number 1917 in LNCS, pages 891–901. Springer Verlag, 2001.
5. Elliott Rusty Harold. *XML Bible*. IDG Books worldwide, 1991.
6. O'Reilly Networks. XML.com: XML from the inside out. Web site at <http://www.xml.com>.
7. Erik T. Ray. *Learning XML: creating self-describing data*. O'Reilly, January 2001.
8. Christian Veenhuis, Katrin Franke, and Mario Köppen. A semantic model for evolutionary computation. In *Proceedings IIZUKA*, 2000.
9. Gregory Oschwald et al. Xmlmen. Available at <http://csci.mrs.umn.edu/twiki/view/CSci4553/XmlMen>, May 2002. Course Wiki for CSci 4553, taught at the University of Minnesota.
10. Christian Gagné Marc Parizeau. Open BEAGLE, a versatile EC framework. Available from <http://www.gel.ulaval.ca/beagle/>. Web pages for the Open BEAGLE EC framework.
11. Juan-Julián Merelo-Guervós. OPEAL, una librería de algoritmos evolutivos en Perl. pages 54–59. Universidad de Extremadura, Febrero 2002.
12. Juan J. Merelo Guervós. Evolutionary computation in Perl. In Münich Perl Mongers, editor, *YAPC::Europe::2002*, pages 2–22, 2002.
13. Juan J. Merelo Guervós. Algoritmos evolutivos en Perl. Ponencia presentada en el V Congreso Hispalinux, disponible en <http://congreso.hispalinux.es/ponencias/merelo/ae-hispalinux2002.html>, Noviembre 2002.
14. Juan-Julián Merelo-Guervós; Francisco-Javier García-Castellano; P.A. Castillo; M.G. Arenas. How Evolutionary Computation and Perl saved my conference. In Sánchez [17], pages 93–99.
15. Juan-David Fernández-Garrido; Juan-Miguel Rodríguez-Fernández; Juan Julián Merelo-Guervós. Algoritmos genéticos aplicados a la liga fantástica yahoo. In Sánchez [17].
16. M.G. Arenas; P.A. Castillo; G. Romero; Juan Julián Merelo-Guervós. Distribución de información en algoritmos evolutivos P2P. In Sánchez [17], pages 85–92.
17. Luciano Sánchez, editor. *MAEB03, Segundo Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados*, Febrero 2003.

Analysis of the Univariate Marginal Distribution Algorithm modeled by Markov chains

C. González, J. D. Rodríguez, J. A. Lozano, and P. Larrañaga

Department of Computer Science and Artificial Intelligence,
University of the Basque Country, Spain
`{ccpgomoc,scsrofej,ccplalj,ccplamup}@sc.ehu.es`

Abstract. This work presents an analysis of the convergence behaviour of the Univariate Marginal Distribution Algorithm (UMDA) when it is used to maximize a number of pseudo-boolean functions.

The analysis is based on modeling the algorithm using a reducible Markov chain, whose absorbing states correspond to the individuals of the search space. The absorption probability to the optimum and the expected time of convergence to the set of absorbing states are calculated for each function. This information is used to provide some insights into how the absorption probability to the optimum and the expected absorption times evolve when the size of population increases. The results show the different behaviours of the algorithm in the analyzed functions.

1 Introduction

Estimation of Distribution Algorithms (EDAs) constitute a new and promising paradigm for EAs [5, 9]. Introduced by Mühlenbein and Paab [9], EDAs are based on Genetic Algorithms (GAs) and constitute an example of stochastic heuristics based on populations of individuals, each of which encodes a possible solution of the optimization problem. These populations evolve in successive generations as the search progresses, organized in the same way as most Evolutionary Computation heuristics. In contrast to GAs, which consider the crossover and mutation operators as essential tools to generate new populations, EDAs replace those operators by estimating and sampling the joint probability distribution of the selected individuals.

Unfortunately, the bottleneck of this new heuristic lies in estimating the joint probability distribution associated with the database containing the selected individuals. To avoid this problem, several authors have proposed different algorithms where simplified assumptions concerning the conditional dependencies between the variables of the joint probability distribution are made. A review of different approaches in the combinatorial and numerical fields can be found in [4, 5].

The purpose of this paper is to further investigate the convergence behaviour of the simplest EDA –UMDA–, which is applied to the maximization of a number of pseudo-boolean functions.

The analysis is based on modeling the algorithm using a Markov chain whose absorbing states correspond to the individuals of the search space. Hence some natural questions immediately arise. What is the absorption probability to any absorbing state (particularly to the optimal point)? How long must we wait until the set of absorbing states are visited? Answering these questions will enable us to learn the effects that changes in the individual and population size have on the absorption probability to the optimum and the expected absorption times. In order to do so we calculate those quantities for UMDA on the maximization of an example of linear, pseudo-modular, unimax and almost positive functions, for different values of population size N and individual length l . Due to the high computational cost of these calculations we have to use low values of N and l .

The remainder of this paper is organized as follows: Section 2 introduces the UMDA algorithm. In Section 3 the Markov chain that models the algorithm is described, and some useful theoretical results are revised. The studied functions are introduced in Section 4. Section 5 explains the experiments carried out and analyzes the results. Finally, we draw conclusions in Section 6.

2 The UMDA algorithm

The Univariate Marginal Distribution Algorithm was proposed by Mühlenbein [7] in 1998. A pseudocode for this algorithm can be seen in Figure 1. UMDA

UMDA

$D_0 \leftarrow$ Generate M individuals (the initial population) randomly
Repeat for $t = 1, 2, \dots$ until the stopping criterion is met
 $D_{t-1}^{Se} \leftarrow$ Select the $N \leq M$ individuals from D_{t-1} according to
 the selection method
 $p_t(\mathbf{x}) = p(\mathbf{x}|D_{t-1}^{Se}) = \prod_{i=1}^l \frac{\sum_{j=1}^N \delta_j(X_i=x_i|D_{t-1}^{Se})}{N} \leftarrow$ Estimate the joint
 probability distribution
 $D_t \leftarrow$ Sample M individuals (the new population) from $p_t(\mathbf{x})$

Fig. 1. Pseudocode for a general UMDA algorithm.

uses the simplest model to estimate the joint probability distribution of the selected individuals in each generation, $p_t(\mathbf{x})$. This joint probability distribution is factorized as a product of independent univariate marginal distributions, $p_t(\mathbf{x}) = p(\mathbf{x}|D_{t-1}^{Se}) = \prod_{i=1}^l p_t(x_i)$.

Each univariate marginal distribution is estimated from marginal frequencies, $p_t(x_i) = \frac{\sum_{j=1}^N \delta_j(X_i=x_i|D_{t-1}^{Se})}{N}$, where:

$$\delta_j(X_i = x_i|D_{t-1}^{Se}) = \begin{cases} 1 & \text{if in the } j\text{-th case of } D_{t-1}^{Se}, X_i = x_i \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

In practice UMDA is used with elitism and the estimation of the parameters is carried out by means of Laplace correction [1]: $p_t(x_i) = p(x_i|D_{t-1}^{Se}) = \frac{\sum_{j=1}^l \delta_j(X_i=x_i|D_{t-1}^{Se})+1}{N+2}$. This ensures convergence to the optimum [2].

Previous works on UMDA [6, 8] have been carried out by assigning a dynamical system to the algorithm, and showing that the algorithm can converge to any local optima of the search space. However this result is subject to the use of an infinite population, far from the finite population case, where the algorithm can converge to any point of the search space.

We analyze UMDA with finite population in order to provide new insights into the circumstances under which UMDA will (will not) perform well.

3 Using Markov chains to model UMDA

Let us introduce some notation. The search space is represented by $\Omega = \{0, 1\}^l$, where $l \in \mathbb{N}$. The cardinality of the search space is $|\Omega| = 2^l = n$. We consider the optimization problem $\max_{\mathbf{x} \in \Omega} f(\mathbf{x})$, where $f : \Omega \rightarrow \mathbb{R}$ is the objective function.

We use a specific version of UMDA to solve the above problem. The algorithm works as follows: at each step t we have a population of size $M = 2N$, D_{t-1} , from which we select the N best individuals (truncation selection), obtaining D_{t-1}^{Se} . Later, using these selected individuals the joint probability distribution $p_t(\mathbf{x})$ is estimated as in Figure 1. Finally we obtain the new population D_t sampling $2N$ individuals from $p_t(\mathbf{x})$.

Given that the probability distribution at step t only depends on the probability distribution at step $t - 1$, the UMDA algorithm above can be modeled using a Markov chain, where the states of the chain are the different probability distributions the algorithm can take. If we take into account that each probability distribution can be represented as a probability vector $\mathbf{q} = (q_1, \dots, q_l)$ (where q_i is the probability of obtaining a 1 in the i th gene), the set of states can be expressed as follows:

$$E = \left\{ (q_1, \dots, q_l) \mid q_i \in \left\{0, \frac{1}{N}, \dots, \frac{N-1}{N}, 1\right\}, i \in \{1, \dots, l\}\right\}. \quad (2)$$

The cardinality of the state space is $c = |E| = (N + 1)^l$. It is important to note that the cardinality of the space states c increases exponentially as the individual size does. This is the reason why, in order to have a reasonable computational cost, the experiments will be carried out for small values of l .

We also want to stress that the Markov chain is not irreducible. More precisely, the absorbing states of the Markov chain correspond to the individuals of the search space while the transient states are the rest, i.e. the states with some component not equal to 0 or 1.

To aid in comprehension, note that each absorbing state is associated with a uniform population of selected individuals (which is formed by N copies of

the same individual). Taking into account that the absorbing states are the individuals of the search space, the algorithm could converge to any of them.

Calculation of the absorption probabilities and the expected absorption times

Let's suppose that the states of the Markov chain are ordered in such a way that the absorbing states are in the last places. Therefore the transition probability matrix P associated with the Markov chain can be written as follows:

$$P = \begin{pmatrix} Q & R \\ \emptyset & I \end{pmatrix} \quad (3)$$

where \emptyset is the null matrix, and I is the identity matrix.

The formulas for the absorption probability and expected absorption times can be obtained from matrices: $W = (I - Q)^{-1}$ and $U = WR$. These results can be seen in [10].

The *expected absorption time starting from the i th state v_i* is given by the expression $v_i = \sum_j w_{ij}$.

The *absorption probabilities to an absorbing state j starting from the i th state, u_{ij}* are given by the elements of the matrix $U = (u_{ij})$.

The computational cost of the calculation of the above quantities depends on the cost of inverting W . Since the dimension of W is $(N + 1)^l - 2^l$, it is directly related to the individual size l .

Calculation of the transition probability matrix P

The calculation of the transition probability matrix P is basic to obtain the remaining quantities. Each entry of $P = (p_{ij})$ is the probability of going from state \mathbf{q}_i to state \mathbf{q}_j in a step of the algorithm, and can be obtained as follows:

$$p_{ij} = P(\mathbf{q}_j | \mathbf{q}_i) = \sum_D P(\text{obtain population } D | \text{ the vector } \mathbf{q}_i \text{ is sampled}) , \quad (4)$$

where D varies in the populations that can be obtained from \mathbf{q}_i , and from the selected individuals D^{Se} of D , the probability vector \mathbf{q}_j is obtained. Unfortunately, in order to calculate (4) it is necessary to solve a system of equations with a large number of degrees of freedom (which depends on l), which has a high computational cost. This is the reason why we use a different method to estimate the elements of matrix P . In Section 5.1 we explain in detail how the estimation was carried out.

4 The functions used

The particular **linear function** analyzed in this work is $f(\mathbf{x}) = c_0 + \sum_{i=1}^l c_i x_i$, $x_i \in \{0, 1\}$, $c_i \in \mathbb{R}$ such that $c_i > \sum_{j=0}^{i-1} c_j$. It is clear that $(1, \dots, 1)$ is the only global maximum for this function.

The **pseudo-modular function** we have used is $f(\mathbf{x}) = \sum_{i=1}^l \prod_{j=1}^i x_j$, $x_j \in \{0, 1\}$, whose optimal solution is $(1, \dots, 1)$.

In the experiments we have carried out we used a well known **unimax function**, the long path function [3] (we can not include it here for reasons of space). We want to stress that this function only has sense for odd values of l .

The **almost positive** function analyzed in this paper is $f(\mathbf{x}) = l - \sum_{i=1}^l x_i + (l+1) \prod_{i=1}^l x_i$, $x_i \in \{0, 1\}$. The optimal solution is $(1, \dots, 1)$, while the individual $(0, \dots, 0)$ is a local optimum point.

5 Experimental results

Our aim is to find the absorption probability to the optimum and the expected absorption times to some absorbing states when the UMDA algorithm is used to maximize the pseudo-boolean functions introduced in the previous section. Once we have those quantities we analyze how they evolve when the size of population N varies. We have made our analysis when $l = 2$ and $2 \leq N \leq 8$, and when $l = 3$ and $3 \leq N \leq 8$.

Estimation of the transition probability matrix P

The computational complexity of the exact calculation of $P = (p_{ij})$ forces us to estimate these values instead of carrying out an exact calculation. To obtain the values of the j th row of P corresponding to probability vector \mathbf{q}_j , we carry out k times the following two steps for each j :

- \mathbf{q}_j is taken as the initial vector of probabilities, each time carrying out the basic steps of UMDA: (i) The initial population is obtained sampling \mathbf{q}_j , (ii) The N best individuals are selected, giving us the selected population, (iii) The new vector of probabilities \mathbf{q}_i is obtained from the selected population.
- After the previous step the obtained probability vector is picked up.

If we denote by k_i the number of times that we reach the state \mathbf{q}_i , then each value p_{ij} , $i \in \{1, \dots, c\}$ of row j is estimated as $p_{ij} = \frac{k_i}{k}$.

It is clear that when the number of experiments carried out k increases the estimation improves. In our experiments we chose k in order to obtain a reasonable computational cost. We made a number of $k = 5,000,000$ experiments which fixes the 5th decimal of p_{ij} .

The absorption probability to the optimum and the expected absorption time to some absorbing state

In practice, the first probability vector used to be $(1/2, \dots, 1/2)$, so we have calculated the absorption probability (resp. time) to the optimum from this state when N is even. In case of odd N we have used the mean of the probability vectors that are the nearest to $(1/2, \dots, 1/2)$ (see Table 1).

Summarizing the results

The results can be seen in Figure 3. The absorption probability to the optimum (and the expected absorption time to some absorbing state) is given in a graph, where the y axis shows the absorption probability (resp. the expected absorption time) and the x axis shows the size of the population N . The same graph shows the results for $l = 2$ and $l = 3$.

$1/N$	2	3
2	$(\frac{1}{2}, \frac{1}{2})$	
3	$(\frac{1}{3}, \frac{2}{3})$ $(\frac{2}{3}, \frac{1}{3})$ $(\frac{1}{3}, \frac{1}{3}, \frac{2}{3})$ $(\frac{1}{3}, \frac{2}{3}, \frac{1}{3})$ $(\frac{2}{3}, \frac{1}{3}, \frac{1}{3})$ $(\frac{1}{3}, \frac{2}{3}, \frac{2}{3})$ $(\frac{2}{3}, \frac{1}{3}, \frac{2}{3})$ $(\frac{2}{3}, \frac{2}{3}, \frac{1}{3})$	
4	$(\frac{2}{4}, \frac{2}{4})$	$(\frac{2}{4}, \frac{2}{4}, \frac{2}{4})$
5	$(\frac{2}{5}, \frac{3}{5})$ $(\frac{3}{5}, \frac{2}{5})$ $(\frac{2}{5}, \frac{2}{5}, \frac{3}{5})$ $(\frac{2}{5}, \frac{3}{5}, \frac{2}{5})$ $(\frac{3}{5}, \frac{2}{5}, \frac{2}{5})$ $(\frac{2}{5}, \frac{3}{5}, \frac{3}{5})$ $(\frac{3}{5}, \frac{2}{5}, \frac{3}{5})$ $(\frac{3}{5}, \frac{3}{5}, \frac{2}{5})$	
6	$(\frac{3}{6}, \frac{3}{6})$	$(\frac{3}{6}, \frac{3}{6}, \frac{3}{6})$
7	$(\frac{3}{7}, \frac{4}{7})$ $(\frac{4}{7}, \frac{3}{7})$ $(\frac{3}{7}, \frac{3}{7}, \frac{4}{7})$ $(\frac{3}{7}, \frac{4}{7}, \frac{3}{7})$ $(\frac{4}{7}, \frac{3}{7}, \frac{3}{7})$ $(\frac{3}{7}, \frac{4}{7}, \frac{4}{7})$ $(\frac{4}{7}, \frac{3}{7}, \frac{4}{7})$ $(\frac{4}{7}, \frac{4}{7}, \frac{3}{7})$	
8	$(\frac{4}{8}, \frac{4}{8})$	$(\frac{4}{8}, \frac{4}{8}, \frac{4}{8})$

Table 1. Starting probability vectors chosen.

Several comments can be made in view of the graphs. We can distinguish the behaviour of the first three functions (the easiest to optimize) from the almost positive.

As was expected, for the first three functions, the absorption probability increases with N . This probability is near to one in the linear and pseudo-modular functions, which means that in almost all executions the algorithm will converge to the optimum. On the other hand, in the unimax function this probability is lower than 0.6. However, it seems that the growth of this probability with N is higher in this third function. Similarly this probability is smaller when l is bigger. The small number of generations to reach convergence is noteworthy.

We obtained surprising results regarding the expected absorption times. In the linear and pseudo-modular functions this time does not increase with population size. We think it is related to the absorption probability. Because the absorption probability is higher the algorithm converges faster. On the other hand in the unimax function this time increases linearly with N . In this last case the function is harder to optimize than the others so the algorithm needs more time to converge. The same as before, absorption time increases with l .

The case of the almost positive function is the most interesting. While with $l = 2$ the absorption probability increases a little with N , in dimension 3 this probability decreases. Apparently going to 0 with N . It seems the algorithm can be absorbed by the local optimum point $(0, \dots, 0)$. So this function is hardly optimized with UMDA.

6 Conclusions and future work

In this work we have used Markov chains to model and analyze some interesting questions about UMDA algorithm behaviour on pseudo-boolean functions. We hope that further theoretical studies on the behaviour of UMDA can be based on this work.

For each analyzed function, we have calculated the absorption probabilities to the optimal point and the expected absorption times. This calculation enables us to see the effects that changes in population size have on these two quantities. The analysis shows the behaviour of the algorithm when the complexity of the function increases: the absorption probability decreases while the expected

absorption time increases. Even in the almost positive function the absorption probability goes near to zero when N increases.

There is much further work to be done to increase our understanding of the EDAs algorithm's behaviour on different classes of problems. A first task would be to increase both individual length and population size. Another interesting direction would be to explore the relationship between the probability of reaching the optimum or any other point of the search space. It would also be helpful to arrive at an analogous model and analysis for other EDAs.

7 Acknowledgments

This work was supported by the University of the Basque Country under grant no. 9/UPV/EHU 00140.226-12084/2000. Also C. González is supported by UPV-EHU.

References

1. B. Cestnik. Estimating Probabilities: A Crucial Task in Machine Learning. In *Proceedings of the European Conference in Artificial Intelligence*, pages 147–149, 1990.
2. C. González, J. A. Lozano, and P. Larrañaga. Mathematical Modelling of Discrete Estimation of Distribution Algorithms. In P. Larrañaga and J. A. Lozano, editors, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, pages 147–163. Kluwer Academic Publishers, 2002.
3. J. Horn, D. E. Goldberg, and K. Deb. Long Path Problems. In Y. Davidor, H. P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature, PPSN*, volume 3, pages 149–158. Berlin and Heidelberg: Springer, 1994.
4. P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña. Combinatorial Optimization by Learning and Simulation of Bayesian Networks. In C. Boutilier and M. Goldszmidt, editors, *Proceedings of Uncertainty in Artificial Intelligence, UAI-2000*, pages 343–352. Morgan Kaufmann, 2000.
5. P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2002.
6. T. Mahnig and H. Mühlenbein. Mathematical Analysis of Optimization Methods Using Search Distributions. In A. S. Wu, editor, *Proceedings of the 2000 Genetic and Evolutionary Computation Conference, Workshop Program*, pages 205–208, 2000.
7. H. Mühlenbein. The Equation for Response to Selection and its Use for Prediction. *Evolutionary Computation*, 5:303–346, 1998.
8. H. Mühlenbein and T. Mahnig. Evolutionary Computation and Wright's Equation. *Theoretical Computer Science (in press)*, 2001.
9. H. Mühlenbein and G. Paaß. From Recombination of Genes to the Estimation of Distributions I. Binary Parameters. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature, PPSN-IV*, pages 178–187, 1996.
10. H. M. Taylor and S. Karlin. *An Introduction to Stochastic Modeling*. Academic Press, 1993.

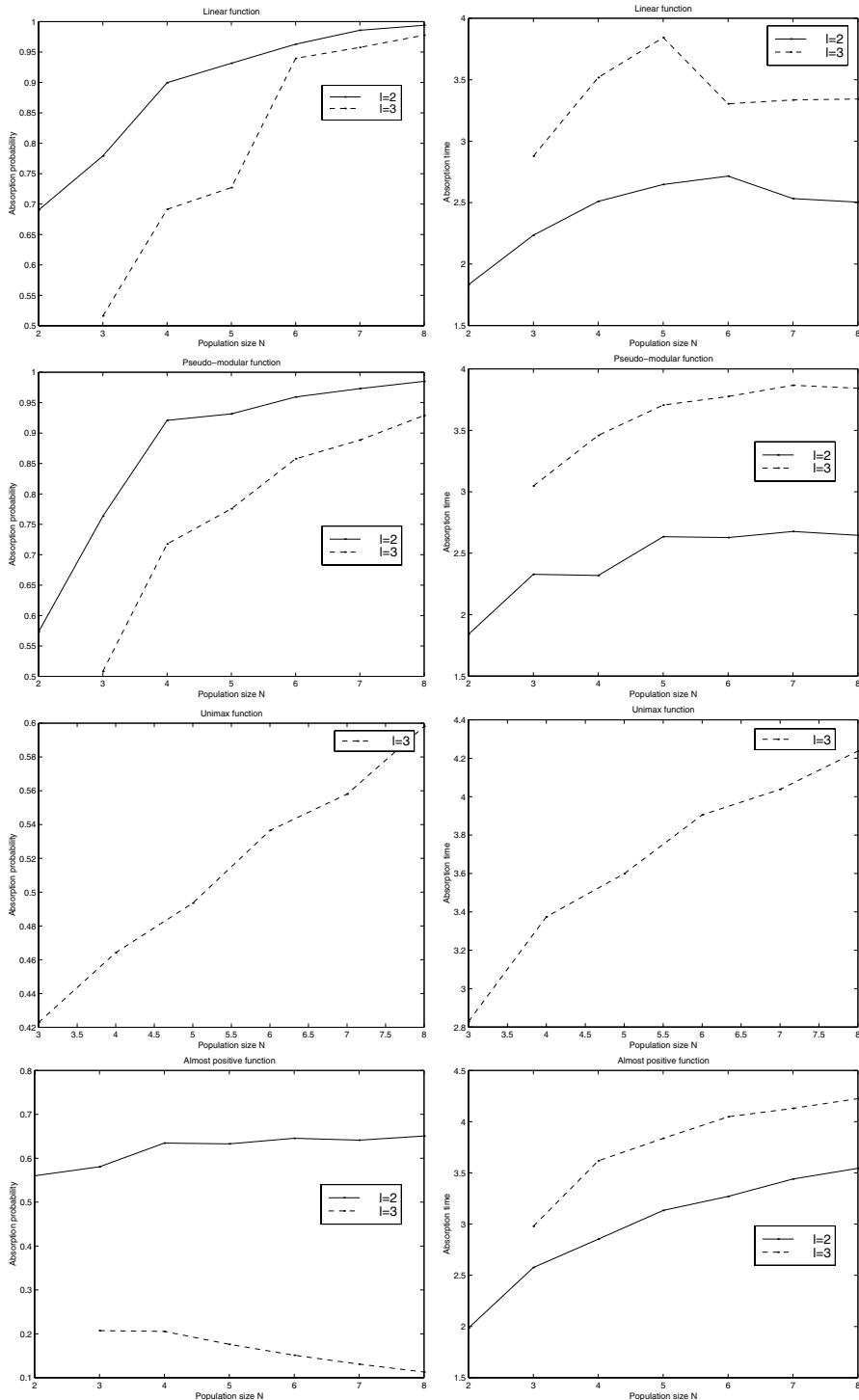


Fig. 2. Absorption probability and absorption time for the linear, pseudo-modular, unimax and almost positive functions.

Node level crossover applied to neural network evolution *

E. Sanz-Tapia, N. García-Pedrajas, D. Ortiz-Boyer, and C. Hervás-Martínez

Universidad de Córdoba, Dpto. de Informática y Análisis Numérico,
{ersanz, npedrajas, dortiz, chervas}@uco.es

Abstract. In this work we present an application of the *Confidence Interval Based Crossover using L₂ Norm* (CIXL2) and BLX- α crossovers to the evolution of neural networks. CIXL2 is a new crossover operator, based on obtaining the statistical features of the best individuals of the population. These features are used as virtual parents for the crossover operator.

Due to the *permutation problem* of neural network coding that negatively affects the crossover operator, we have adopted a novel approach. Crossover is made at node level. Instead of performing crossover over two whole networks, we perform crossover over two nodes. We present here the adaptation of two crossover methods: CIXL2 and BLX- α . All the nodes of the best networks are clustered, by means of a k -means algorithm, and a redefinition of the operators is carried out using the obtained clusters. Each node is mated within the cluster where it belongs.

Keywords

Neural network evolution, genetic algorithms, node level crossover.

1 Introduction

In the area of neural networks design one of the main problems is finding suitable architectures for solving specific problems. The election of such architecture is very important, as a network smaller than needed would be unable to learn and a network larger than needed would end in over-training.

Evolutionary computation is a set of global optimization techniques that have been widely used in late years for training and automatically designing neural networks. There have been many applications to parametric learning and to both parametric and structural learning[9], since G. F. Miller *et al.*[5] proposed that evolutionary computation was a very good candidate to be used to search the space of topologies because the fitness function associated with that space is complex, noisy, non-differentiable, multi-modal and deceptive.

* This work was supported in part by the Project TIC2002-04036-C05-02 of the Spanish CICYT and FEDER funds.

2 Node level crossover

Genetic algorithms are based on a representation independent of the problem, usually the representation is a string of binary, integer or real numbers. This representation (the genotype) codifies a network (the phenotype). This is a dual representation scheme. The interpretation function maps between the elements in the recombination space (on which the search is performed) and the subset of structures that can be evaluated as potential task solutions. The ability to create better solutions in a genetic algorithm relies mainly on the operation of *crossover*. This operator forms offspring by recombining representational components from two or more members of the population.

The benefits of crossover come from the ability of forming connected substrings of the representation that correspond to above-average solutions. These substrings are called *building blocks*. Crossover is not effective in environments where the fitness of an individual of the population is not correlated with the expected ability of its representational components. Such environments are called *deceptive*[3]. Deception is a very important feature in most representations of neural networks, so crossover is usually avoided in evolutionary neural networks[1].

One of the most important forms of deception arises from the many-to-one mapping from genotypes in the representation space to phenotypes in the evaluation space. Two networks which order their hidden nodes differently will have different genotypes in most representation schemes, but these two networks are functionally equivalent.

This is one of the most common problems of the application of genetic algorithms to neural networks, known as the *permutation problem* or the *competing conventions problem*. To solve this problem D. Thierens[8] proposed a non-redundant genetic coding of neural networks that avoids the permutation problem. This coding avoids different representations for *exactly* the same network. Many other works just disregard the problem.

Our approach in this paper consists of performing crossover at node level. As the coding of a node is ordered, if two nodes are topologically close, its functionality is also similar. In order to mate only similar nodes, a cluster analysis is carried out, and each node is mated within its cluster. Next section explains more deeply the evolutionary process. A similar kind of crossover has been used before in coevolution of nodes[6].

3 Redefinition of crossover operators

The evolution of the population of networks is different from the standard methods of evolving neural networks. There is no crossover performed over whole networks. Networks are only subject to mutation, in its two forms: parametric and structural mutation.

Crossover is performed at node level. In this way we avoid the problems of mixing the nodes of two networks that have been discussed above. In order to

be effective crossover needs a high correlation between the functionality and the coding of the individuals. When a network is coded as a string of numbers, two functionally similar networks can have very different representations, and vice-versa.

If we consider the representation of a node alone, the scenario is different. Consider the discrepancy between the outputs of nodes i and j over input x

$$|y_i - y_j| = |f(\sum_k w_{ik} x_k) - f(\sum_k w_{jk} x_k)|, \quad (1)$$

if we use a hyperbolic tangent transfer function, we can consider f an approximately linear function in an interval around the origin, so

$$\begin{aligned} &= |f(\sum_k w_{ik} x_k) - f(\sum_k w_{jk} x_k)| \\ &= |f(\sum_k x_k (w_{ik} - w_{jk}))|. \end{aligned} \quad (2)$$

So, the discrepancy is a linear function of the difference among the weight vector, \mathbf{w}_i and \mathbf{w}_j ; so, two nodes will have similar functional behavior if and only if their weight vectors are similar. This reasoning yields us to the conclusion that a node level crossover is more probable to obtain good offspring than a network level crossover.

3.1 Confidence interval based crossover

Confidence interval based crossover is a new crossover based on statistical features of the best individuals of the population[4]. This crossover has been successfully applied to function optimization with[7] and without constraints. The crossover is based on obtaining the information of the confidence intervals of the genes using the L_2 norm of the best individuals, so it is called *Confidence Interval Based Crossover using L_2 Norm* (CIXL2).

Let β be the set of N individuals that forms the population and $\beta^* \subset \beta$ the subset of the n fittest individuals, and q the number of genes on each chromosome. Let us assume that the genes, β_i , of the chromosomes of the individuals in β^* are independent random variables, then we can consider β_i a random variable with a normal distribution, with a localization parameter of the form μ_{β_i} , we have the model $\beta_i = \mu_{\beta_i} + e_i$, for each $i = 1, \dots, q$, being e_i a random variable.

If we assume that the n fittest individuals form actually a simple random sample $(\beta_{i_1}, \beta_{i_2}, \dots, \beta_{i_n})$ of the normal distribution of the fittest individuals of the population β_i^b , the model can be written:

$$\beta_{ij}^b = \mu_b + e_{ij}, \quad \text{for } j = 1, 2, \dots, n. \quad (3)$$

We can define the dispersion function, D_2 , induced by the L_2 norm as: $D_2(\mu_b) = \sqrt{\sum_{j=1}^n (\beta_{ij}^b - \mu_b)^2}$. The estimator using this dispersion function of the localization parameter is the mean of the distribution β_i^b , that is, $\hat{\mu}_b = \bar{\beta}_i^b$.

When the distribution of the genes is normal, the sample mean estimator follows a normal distribution $N(\mu_b, \sigma_b^2/n)$. Under these assumptions we have a bilateral confidence interval for the localization of the genes, for a confidence coefficient $1 - \alpha$, of the form:

$$I_{1-\alpha}(\mu_b) = \left[\bar{\beta}_i^b - t_{n-1,\alpha/2} \frac{S_b}{\sqrt{n}}; \bar{\beta}_i^b + t_{n-1,\alpha/2} \frac{S_b}{\sqrt{n}} \right], \quad (4)$$

where $S_b = \left[\sum_{j=1}^n (\beta_{ij}^b - \hat{\beta}_i^b)^2 / (n-1) \right]^{1/2}$ is the standard deviation, and t_{n-1} is a Student t of $n-1$ degrees of freedom.

Crossover operator method. From the confidence interval of equation 4 we build three individuals that are considered the parents of the proposed crossover. These three parents are formed by: all the lower limit values of the confidence intervals of the genes, individual CILL; all the upper limit values of the confidence intervals of the genes, individual CIUL; and all the means of the confidence intervals of the genes, individual CIM. These individuals divide the domain of the gene values into three subintervals: $I_i^L \equiv [a_i, CILL_i]$, $I_i^M \equiv [CILL_i, CIUL_i]$, and $I_i^R \equiv (CIUL_i, b_i]$.

The interval based crossover operator using L_2 norm, CIXL2, creates an offspring β^s , from an individual of the population, $\beta^f = (\beta_1^f, \beta_2^f, \dots, \beta_p^f)$, and the three individuals CILL, CIUL, and CIM obtained from the confidence interval. We consider these four individuals and their fitness (being $f(\beta)$ the fitness value of individual β) and distinguish three cases depending on the position of β^f in one of the three subintervals defined above. These three cases are:

- Case 1:** $\beta_i^f \in I_i^L$. If $f(\beta^f) > f(CILL)$ then $\beta_i^s = r(\beta_i^f - CILL_i) + \beta_i^f$ else $\beta_i^s = r(CILL_i - \beta_i^f) + CILL_i$.
- Case 2:** $\beta_i^f \in I_i^M$. If $f(\beta^f) > f(CIM)$ then $\beta_i^s = r(\beta_i^f - CIM_i) + \beta_i^f$ else $\beta_i^s = r(CIM_i - \beta_i^f) + CIM_i$.
- Case 3:** $\beta_i^f \in I_i^R$. If $f(\beta^f) > f(CIUL)$ then $\beta_i^s = r(\beta_i^f - CIUL_i) + \beta_i^f$ else $\beta_i^s = r(CIUL_i - \beta_i^f) + CIUL_i$.

where r is a uniform random number belonging to $[0, 1]$.

The node level crossover is made following the algorithm of 1.1. In order to evaluate the virtual parents, three virtual networks are created with the upper and lower bounds, and means of the confidence intervals of the clusters of the best networks nodes.

3.2 BLX- α applied to node crossover

The definition of this crossover is the following[2]. The crossover generates just one offspring $\beta^s = \{\beta_1^s, \beta_2^s, \dots, \beta_i^s, \dots, \beta_n^s\}$ from two parents β^{p_1}, β^{p_2} , where β_i^s is randomly selected from the interval $[\beta_{min} - I\alpha, \beta_{max} + I\alpha]$, being $\beta_{min} =$

Algorithm 1.1 Node level crossover using CIXL2

```

1: Select  $n$  best networks
2: Perform  $k$ -means over nodes of selected networks
3: Obtain confidence intervals of each cluster
4: Create virtual individuals and evaluate them
5: repeat
6:   Select a parent by roulette selection
7:   for Each node do
8:     Classify the node
9:     Perform CIXL2 within its cluster
10:  end for
11: until Refill new population

```

$\min(\beta_i^{p_1}, \beta_i^{p_2})$, $\beta_{max} = \max(\beta_i^{p_1}, \beta_i^{p_2})$, and $I = \beta_{max} - \beta_{min}$. The exploration/exploitation compromise of this crossover depends on the value of the α parameter. The optimum value obtained in the experimentation is $\alpha = 0.5$.

We have also redefined BLX- α crossover in order to use it at node level, as we cannot randomly choose two networks of the population and carry out a crossover of their nodes. The redefinition of the crossover is shown on algorithm 1.2.

Algorithm 1.2 Node level crossover using BLX- α

```

1: Select a parent by roulette selection
2: for Each node do
3:   Classify the node
4:   Perform standard BLX- $\alpha$  with the centroid of its cluster
5: end for

```

So, each node is mated with the center of the cluster where it belongs. As BLX- α depends on a random value, many different children will be obtained although all of them share one of the parents.

4 Network coding and evolution

An individual of the population of networks is just a string of real number following the standard codification. The evolution of the population of networks is the following, once the fitness of each individual has been obtained:

- The best 10% of the population is selected for reproduction. This elitist selection prevents the elimination of the best solutions along the evolutionary process.
- To fill the 60% of the new population we select individuals of the population by means of a roulette algorithm and perform a node level crossover.

- To fill the 20% of the new population we select individuals of the population by means of a roulette algorithm and perform a parametric mutation that consists of a basic back-propagation algorithm (10%) or a random step in the space of network weights (10%).
- To fill the 10% of the new population we select individuals of the population by means of a roulette algorithm and perform a structural mutation that consists of one of the two following operations:
 - Removing one of the nodes of the network.
 - Adding a new node to the network.

In order to keep the nodes comparable for the k -means algorithm we cannot remove/add connections to a node as another mutation operator.

5 Experimental setup

Each set of available data was divided into three sets: 50% of the patterns were used for learning, 25% of them for validation and the remaining 25% for testing the generalization of the networks.

The population had 1000 networks with a maximum of 16 hidden nodes. For CIXL2 crossover we chose $1 - \alpha = 0.7$ and $n = 10$, for BLX- α crossover we had $\alpha = 0.5$.

In order to assess the validity of application of node level crossover to neural network evolution, we compare CIXL2 and BLX- α crossovers with C4.5 algorithm, an MLP trained with the quickprop algorithm and a modular neural network using the adaptive mixture of local experts model.

In order to assess the true difference in performance among CIXL2 and BLX- α and the other methods, we conducted t test whose p -values are shown on each table.

For the comparison of the different training methods we chose three problems of the UCI machine learning repository: Pima Indian, Heart Disease, and Credit Card data sets. All of them are widely used in neural network bibliography.

6 Results

In all the tables we show, for each data set, the averaged error of classification over 30 repetitions of the algorithm, the standard deviation, the best and worst individuals, and the averaged number of nodes and connections of the best networks of each experiment. The measure of the error is $E = \frac{1}{P} \sum_{i=1}^P e_i$, where P is the number of patterns and e_i is 0 if pattern i is correctly classified, and 1 otherwise.

Table 1 shows the results obtained in classifying the Pima data set. We can see how the CIXL2 and BLX- α crossovers are the best performing methods. They clearly outperforms the other algorithms. The difference in performance is statistically significant at a confidence level of 5% as it is shown on the table.

Table 1. Error rates for Pima data set

Model	Training				Generalization				<i>t</i> -test	
	Mean	StD	Best	Worst	Mean	StD	Best	Worst	CIXL2	BLX- α
CIXL2	0.2137	0.0034	0.2049	0.2205	0.2116	0.0154	0.1719	0.2396	–	0.2432
BLX- α	0.2144	0.0036	0.2083	0.2205	0.2069	0.0154	0.1719	0.2396	0.2432	–
quickprop	0.2313	0.0069	0.2222	0.2413	0.2219	0.0189	0.1979	0.2604	0.0221	0.0000
C4.5	0.1718	–	–	–	0.2552	–	–	–	–	–
Mix. exp.	0.2528	0.0135	0.2413	0.2847	0.2485	0.0211	0.2135	0.2865	0.0000	0.0000

Table 2 shows the results obtained in classifying the Heart data set. We can see how the CIXL2 crossover is the best performing method together with BLX- α . These two algorithms and quickprop have almost the same performance, all of them outperform the other algorithms, with a difference that is statistically significant as it is shown on the table.

Table 2. Error rates for Heart data set

Model	Training				Generalization				<i>t</i> -test	
	Mean	StD	Best	Worst	Mean	StD	Best	Worst	CIXL2	BLX- α
CIXL2	0.1111	0.0055	0.0990	0.1188	0.1377	0.0233	0.0882	0.1912	–	0.4891
BLX- α	0.1104	0.0060	0.1040	0.1238	0.1338	0.0202	0.1029	0.1765	0.4891	–
quickprop	0.0975	0.0226	0.0495	0.1188	0.1471	0.0196	0.1176	0.1765	0.0938	0.0111
C4.5	0.0644	–	–	–	0.2206	–	–	–	–	–
Mix. exp.	0.0500	0.0135	0.0347	0.0743	0.1853	0.0210	0.1618	0.2206	0.0000	0.0000

Table 3 shows the results obtained in classifying the Card data set. We can see how the performance of the two crossover methods is very similar, outperforming the other models. Nevertheless, it is interesting to note that C4.5 performs very well for this problem, it is a reasonable result as many of the features of the patterns are not numerical and C4.5 is a hierarchical classification method.

Table 3. Error rates for Card data set

Model	Training				Generalization				<i>t</i> -test	
	Mean	StD	Best	Worst	Mean	StD	Best	Worst	CIXL2	BLX- α
CIXL2	0.0754	0.0049	0.0665	0.0867	0.1263	0.0088	0.1105	0.1453	–	0.4568
BLX- α	0.0729	0.0055	0.0636	0.0838	0.1244	0.0092	0.1105	0.1395	0.4568	–
quickprop	0.0861	0.0112	0.0618	0.1004	0.1407	0.0094	0.1221	0.1512	0.0000	0.0000
C4.5	0.0830	–	–	–	0.1337	–	–	–	–	–
Mix. exp.	0.1120	0.0097	0.0907	0.1236	0.1401	0.0043	0.1337	0.1453	0.0000	0.0000

7 Conclusions

We have shown that node level crossover is an effective way of evolving neural networks. We have also successfully introduced the *Confidence Interval Based Crossover* in the field of evolutionary neural networks. We have achieved a performance comparable, at least, to other classical methods of neural network training.

We have also proved that BLX- α crossover has an interesting performance when it is applied at node level. This result is very important as BLX- α is one of the best operators described in the bibliography of genetic algorithms.

As work for the future we are working on the optimization of the definition of CIXL2 and BLX- α at node level. We are also dealing with a node level redefinition of other well-known crossovers.

Acknowledgements

The authors would like to acknowledge R. Moya-Sánchez for her helping in the final version of this paper.

References

1. P. J. Angeline, G. M. Saunders, and Jordan B. Pollack. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, 5(1):54 – 65, January 1994.
2. L. J. Eshelman and J. D. Schaffer. Real-coded genetic algorithms and interval-schemata. In L. D. Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 187–202. Morgan Kaufmann, San Mateo, 1993.
3. D. E. Goldberg. Genetic algorithms and Walsh functions: Part 1, a gentle introduction. *Complex Systems*, 3:129 – 152, 1989.
4. C. Hervás-Martínez, D. Ortiz-Boyer, and N. García-Pedrajas. Theoretical analysis of the confidence interval based crossover for real-coded genetic algorithms. In J. Merelo, P. Adamidis, H-G. Beyer, J. Fernández, and H-P. Schwefel, editors, *Parallel Problem Solving from Nature – PPSN VII*, number 2439 in Lecture Notes in Computer Science, pages 153–161, Granada, september 2002. Springer-Verlag.
5. G. F. Miller, P. M. Todd, and S. U. Hedge. Designing neural networks. *Neural Networks*, 4:53–60, 1991.
6. D. E. Moriarty and R. Miikkulainen. Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation*, 4(5), 1998.
7. D. Ortiz-Boyer, C. Hervás-Martínez, and N. García-Pedrajas. Crossover operator effect in function optimization with constraints. In J. Merelo, P. Adamidis, H-G. Beyer, J. Fernández, and H-P. Schwefel, editors, *Parallel Problem Solving from Nature – PPSN VII*, number 2439 in Lecture Notes in Computer Science, pages 184–193, Granada, september 2002. Springer-Verlag.
8. D. Thierens. Non-redundant genetic coding of neural networks. In *Proc. of the 1996 IEEE International Conference on Evolutionary Computation*, pages 571–575, Piscataway, NJ, 1996. IEEE Press.
9. X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 9(87):1423–1447, 1999.

Genetic Search of Block-Based Structures of Dynamical Process Models

A. López¹, L. Sánchez²

Depts. of Ingeniería Eléctrica (1) and Informática (2). C. Universitario de Viesques.
Ed. Departamentales, 33204, Gijón, Asturias.
e-mails: antonio@isa.uniovi.es, luciano@ccia.uniovi.es

Abstract. Genetic identification of models of dynamical systems is becoming a well established research field. Nowadays it is hard to obtain more precise numerical results than *state of the art* methods, but, in our opinion, there is still room to improve the understandability of genetically induced models. In this paper it is proposed a method that focuses in the comprehensibility of the final model, while keeping most of the numerical precision of former studies.

The main innovation in this work is centered in the concept of “understandable” system. We do not use state space designed, rule based models, but z-transform based models, comprising linear, discrete dynamical models of first or second order and memoriless nonlinear elements (saturation, dead zone or other nonlinear gains.) This way, we provide control engineers with their preferred representation in moderate to complex models, and facilitate the task of designing control systems for these processes.

Keywords: Block-based models, Process control, GA-P algorithms, System Identification.

1 Introduction

Genetic identification of models of dynamical systems is becoming a mature research field. It is widely admitted than both GA and GP can find very precise representations of dynamical models. But the use of the structures found with GP algorithms is not very extended in practice. This is mostly due to the complexity of the final output: the output of canonical GP, even in simple nonlinear models [7] is hard to interpret, and not very different in complexity from a neural network.

The comprehensibility of a model of a dynamical system has also been studied under different perspectives. In our opinion, the most advanced methods have been developed in the fuzzy community (see, for instance [2]). Genetic Fuzzy systems are able to produce a set of linguistically understandable fuzzy rules that define the state-space behavior of a dynamical system. Fuzzy models have important advantages over black-box models like the formerly mentioned neural networks, or least-square fitted polynomials. But there are many circumstances in which control engineers are not comfortable with a state space, rule based

description of a dynamical model. According to our own experience, many control systems are best designed in the classical framework (z transform). The preferred output for a small to moderately complex dynamical model is a block-based structure where blocks are either

- A linear, discrete dynamical model of first or second order.
- A memoriless nonlinear element (saturation, dead zone or other nonlinear gain).

In fact, there are preliminary works in GP in which is used an structure with certain similarities to the one being proposed herein [8, 9]. But the work in this field, up to our knowledge, was not continued. In this paper we propose to use more modern hybrid genetic techniques than those used in these preliminary works and to improve both numerical properties and the quality of the block based representation. Our objective is to produce models with a high linguistic quality, and to promote their practical use.

1.1 Structure of the Paper

The outline of this paper is as follows: in Sect. 2, different genetic-based models of dynamical systems are discussed. In Sect. 3 the algorithm proposed here is described. Then (Sect. 4) an experimental validation of our proposal is done, modeling both a synthetic and a real process and comparing the results with those obtained with previous works. The paper finishes (Sect. 5) with the concluding remarks and future work.

2 Genetic-based comprehensible models

Most of the evolutionary methods for system identification from sampled data focus in nonlinear state space-based models. For this kind of models, the objective of the learning process is the production of a set of difference equations defining the dynamics of the process. Unfortunately, for practical purposes, a set of equations that relates all state variables between them is hard to manage in all but small sized problems. Modular representations are usually preferred, because they allow to determine groups of variables affected by specific parameters.

Genetic Programming has been applied to learn such modular models. One of the first examples was given in [6], where a structured Genetic Algorithm, in a tree based representation, is used. The set of functions that was proposed contained only two-input quadratic functions, which are not the building blocks that control engineers expect to find in structured models. Some implementations nearer to usual practice can be found in [3, 7] and other, less common approaches to model the dynamics of a system, are described in [4]. Most of these schemes introduce dynamic considerations by means of extended terminal sets, that include either input and input-output delayed variables.

One of the most complete methods is described in SMOG [8, 9]. The problem is addressed there as a search of a diagram block based representation of a

1. Initialize random population of models.
2. Tune parameters of models (Hooke-Jeeves algorithm).
3. Calculate fitness.
4. Selection of models and application of genetic operators.
5. Go to 2).

Fig. 1. SMOG evolution. Canonical GP is used for structural search and Hooke-Jeeves method is used for parameter tuning

model of the process in a tree codification. Under the considered approach (see Fig. 1), hierarchical evolutionary algorithms are applied: canonical GP is used for the evolution of model structures and combined with deterministic numerical optimization methods (Hooke and Jeeves algorithm) for parameter tuning. An iterative search of structure and parameters is done: each model considered is parametrically tuned by means of Hooke-Jeeves algorithm as a previous step to fitness evaluation. Genetic operators defined for evolution affects only the structure of the models.

We will show in this paper that, according to our experimentation, better results can be obtained if a new representation in which the search is not done hierarquically, but by means of an hybrid method between GA and GP (the GA-P algorithm, [5]). GA-P is able to search *in parallel* in both structure and parameter spaces. It is remarked that we use the word “hierarquical” to describe an algorithm in which GP operators are used first, to find an structure, and then a local search, classical or genetic, is launched to optimize the parameters of that structure. In other words, the GP serves to find structures, but the numbers found in the terminal nodes only serve as a starting point for the parameter search.

3 Proposed Algorithm

Our algorithm will evolve a set of diagram block representations of the process. A diagram block is, in turn, a series, parallel or feedback association of subsystems. Series association is intrinsic, and parallel association will be allowed by means of arithmetic operators, such as + and -, and feedback representation will be allowed by means of an special operator [1] described next.

Regarding the catalog of subsystems, we used only memoryless version of the common non-linear features of physical systems, such as dead zones or saturations. All the dynamic behavior is delegated to linear elements: we include in the function set a reduced group of linear models (first and second order damped linear subsystems, unitary delay and static gain) such that it is possible to get higher order systems by means of series association.

3.1 GA-P Algorithms

GA-P [5] is an hybrid between genetic algorithms and genetic programming, that was first used in symbolic regression problems. Individuals in GA-P have

two parts: a tree based representation and a chain of numerical parameters. Different from canonical GP, the terminal nodes of the tree never store numbers but linguistic identifiers that are pointers to the chain of numbers. We do not use the initial implementation of GA-P but rather a niche-based strategy described in [10].

3.2 Representation

The parameter part of the GA-P is a standard, real coded vector of values. The structure has to be adapted to the problem at hand. Otherwise, the tree based representation of GA-P would make it impossible to model a wide set of systems, such as those involving nested or not unitary feedbacks. The reason is that a block diagram is not a tree when it includes feedback, but a directed graph. The proposed representation circumvents this problem by means of a special feedback node. Both input and the feedback branches originate in it. The terminal nodes of the feedback branch (marked as “**”) are recessive. This way, standard structural modification operators can be applied at any point in the individual to evolve structures.

It also contains a third link to another node from which the feedback signal will be taken, converting the representation in a graph. This pointed node must be contained in the path between the feedback node and the output node of the system. Otherwise, feedback node loses its significance. This consideration must be present in the creation and modification of individuals as a consequence of structural genetic operators. When an individual does not accomplish this condition after a structural modification, invalid feedback nodes are reinitialized.

Algebraic loops are neglected by means of the implicit inclusion of a unit delay in the feedback branch. To prevent series associations of delays, dynamic blocks used respond instantly. But, known the fact that physical systems never respond instantly to an excitation, a unit delay is also implicitly linked to the output of the model.

3.3 Genetic Operators

Two sets of operators are applied in the evolutionary process: structural and parameter genetic operators. Again, in the parameter part a fairly standard arithmetic crossover, followed by a local search adjust. The structural operators (GP crossover) must take care of the special tree-based encoding of a graph that is made here, thus feedback links are inhibited during the process.

4 Numerical Results

To validate our approach, as a first test, an empirical control system of a first order process with a proportional saturated controller and a sensor without dynamics (see Fig. 3(a)) was modelled by means of the defined GA-P strategy.

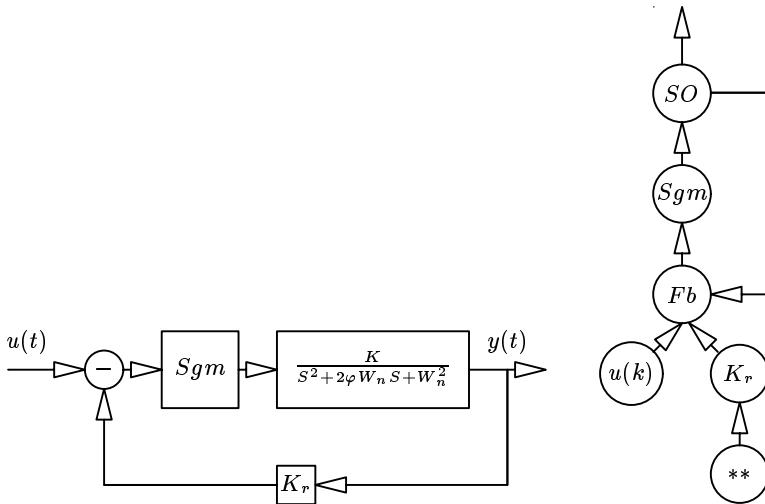


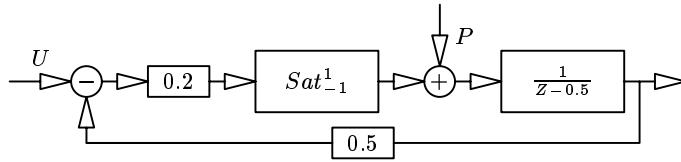
Fig. 2. Block diagram representation of a feedback system (left) and its genetic representation (right.) “SO” stands for “Second Order” and “ Sgm ” for “sigmoid function”. Also, “ $**$ ” stands for a recessive terminal

It was also compared with a hierarchical process. Both approaches were stopped after certain number of evaluations of the objective function.

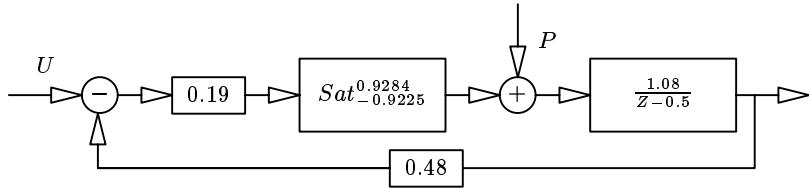
Experiments were repeated 10 times each. Table 1 contains validation errors for each experiment. Observe that GA-P improves slightly the results, but the differences are not significant. The gain with GA-P is in the identified structure (see Figs. 3(b) and 3(c), where the best models obtained by both approaches are shown). In this case, GA-P found exactly the structure of the target model, explaining very well the data relationships. In contrast, the hierarchical method was trapped in a local minimum of the structure. It is only capable of fitting the sampled data.

As a final test, a real system was modelled by means of the proposed scheme. A DC motor was selected, in order to have information enough to contrast the GA-P solution with a known model for the process (usually a first or second order dumped linear subsystem with a non-linear dead zone component).

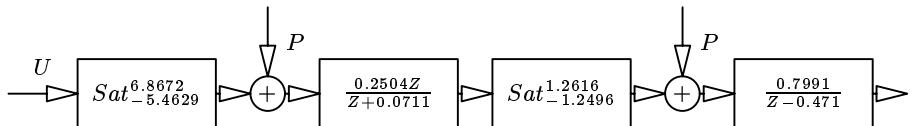
Experimental conditions were the same as in the preceding section. Table 1 contains the numerical validation errors for each experiment. From it, it can be concluded that the best result was found at experiment 10, shown in Fig. 4. Solution is close to a known model for the system: the search scheme is capable of capturing the most significant relationships in the data. This figure also includes a comparison between the motor and the model responses using a squared input signal. Observe that the behavior is correctly reproduced and the noise is smoothed as expected.



(a) Target Model



(b) GA-P Search Best Model



(c) Hierarchical Search Best Model

Fig. 3. Modelling a synthetic example. Upper part: target model. Central and lower parts: structures of the learned models. “ Sat_a^b ” stands for “saturation” block with limits in a, b

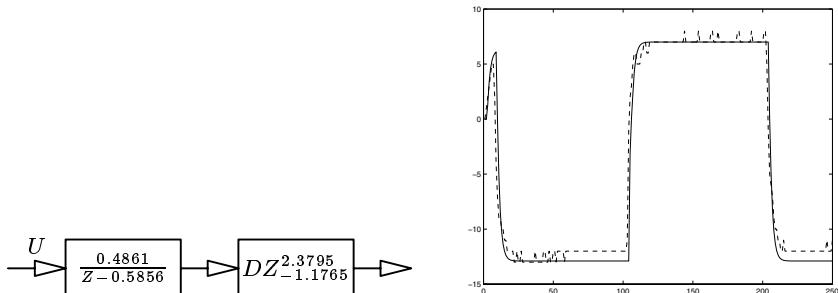


Fig. 4. Modelling of a direct current motor. Left: best model found (“ DZ_a^b ” stands for “dead zone” block with limits in a, b). Right: Comparison of model (continuous line) and system (plotted line) responses

Table 1. Numerical results: GA-P (left) and hierarchical (center) modelling errors for the synthetic problem. Right table: modelling errors for the direct current motor

Experiment	Error	Experiment	Error	Experiment	Error
1	0.00017	1	0.00206	1	0.9196
2	0.0004	2	0.00301	2	0.7755
3	0.00005	3	0.00129	3	0.7354
4	0.00004	4	0.00184	4	0.8433
5	0.00019	5	0.00287	5	0.9223
6	0.00005	6	0.00112	6	0.9259
7	0.00005	7	0.00111	7	1.1809
8	0.00006	8	0.00107	8	1.0134
9	0.00029	9	0.00147	9	1.0976
10	0.00007	10	0.00263	10	0.6933
Average	0.00014	Average	0.00185		

5 Concluding Remarks

We have shown that the use of hybrid methods can improve the understandability of the model of a dynamical system. A parallel evolutive search of parameters and structure was proposed, and it was shown that it did not waste time optimizing parameters for invalid structures neither discards structures too early, thus the quality of the output was improved.

Perhaps the main interest in this work is in the concept of “understandable” system. We did not use state space designed, rule based models, neither we used ordinary GP arithmetical expressions, but z-transform based models, comprising linear, discrete dynamical models of first or second order and memoriless nonlinear elements (saturation, dead zone or other nonlinear gains.) This way, we provide control engineers with their preferred representation in moderate to complex models.

Acknowledgements

This work was funded by Spanish Ministry of Science and Technology and by FEDER funds, under the grant TIC-04036-C05-05.

References

1. H. Lopez A.M. Lopez and L. Sanchez. Graph based GP applied to dynamical systems modeling. *IWANN 2001. Connectionist Models of Neurons, Learning Processes and Artificial Intelligence*, pages 725–732, 2001.
2. Cordón, O., Herrera, F. (2000) “A proposal for improving the accuracy of linguistic modeling”. *IEEE Transactions on Fuzzy Systems*, 8, 3, pp. 335-344.

3. G.J. Gray, D.J. Murray-Smith, Y. Li, and K.C. Sharman. Nonlinear model structure identification using genetic programming. In *Late Breaking Papers at the Genetic Programming 1996 Conference*, pages 32–37, Stanford University, CA, USA, 1996.
4. H. Hiden, M. Willis, B. McKay, and G. Montague. newblock Non-linear and direction dependent dynamic modelling using genetic programming. In *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 168–173, Stanford University, CA, USA, 1997.
5. L.M. Howard and D.J. D'Angelo. The GA-P: A genetic algorithm and genetic programming hybrid. *IEEE Expert*, 10(3):11–15, June 1995.
6. H. Iba, T. Karita, H. Garis, and T. Sato. System identification using structured genetic algorithms. In *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93*, pages 279–286, University of Illinois at Urbana-Champaign, 1993.
7. M.A. Keane, J.R. Koza, and J.P. Rice. Finding an impulse response function using genetic programming. In *Proceedings of the 1993 American Control Conference*, volume III, pages 2345–2350, Evanston, IL, USA, 1993.
8. P. Marenbach. Using prior knowledge and obtaining process insight in data based modelling of bioprocesses. *System Analysis Modelling Simulation*, 31:39–59, 1998.
9. P. Marenbach, K.D.. Betterhausen, and S. Freyer. Signal path oriented approach for generation of dynamic process models. In *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 327–332, Stanford University, CA, USA, 1996.
10. L.A. Sanchez and J.A. Corrales. Niching scheme for steady state GA-P and its application to fuzzy rule based classifiers induction. *Mathware and Soft Computing*, 7(2-3):337–350, 2000.

Visualization of Neural Net Evolution

G. Romero, M.G. Arenas, P.A. Castillo, J.J. Merelo

Dpto. de Arquitectura y Tecnología de Computadores

E.T.S.I. Informática. Universidad de Granada

Periodista Daniel Saucedo Aranda s/n. 18071 Granada (Spain)

e-mail: gustavo@geneura.ugr.es URL: <http://geneura.ugr.es>

Abstract. This paper gives an overview of how visualization techniques can help us to improve an evolutionary algorithm that trains artificial neural networks. Kohonen's self-organizing maps (SOM) are used for multidimensional scaling and projection of high dimensional search spaces. The SOM visualization technique used here makes visualization of the evolution process easy and intuitive.

1 Introduction

Evolutionary Algorithms (EA) produce a vast amount of data concerning run performance and progress. Apart from simple convergence toward the solution, the extraction of useful information to get further insight into the state and course of the algorithm is a non-trivial task. Understanding run behavior is difficult due to the fact that EAs searches stochastically large problem spaces.

EA users often examine how the quality of the solutions found changes over time by using a fitness versus generation number graph. Although such a graph illustrates the improvements in the quality of the solutions considered during the run, it does not illustrate the structure of the solutions being considered, the regions of the search space being explored, or how to fine-tune the algorithm parameters so that the search space is explored better. Visualization is proposed as a useful method for solving this problem.

The rest of the paper is organized as follows: Section 2 contains a classification and description of visualization techniques. Section 3 describes the state of the art in multidimensional visualization and scaling methods. Sections 4 and 5 are dedicated to the study of the visual effects of genetic operators on an individual and selective pressure over a whole population. Finally, section 6 offers conclusions and suggestions for future work.

2 Visualization techniques

Visualization techniques have been used in EAs for their ability to represent run data in an intuitive form. As the old saying goes, “a picture is worth a thousand words”. These techniques provide a baseline for a better understanding of the evolutionary process.

Visualization techniques can be divided into several categories according to the following criteria:

- **What is seen?** We may graphically represent either the genotype (the representation of a problem solution) or the phenotype (the qualitative interpretation of the representation). In some problems, both of them, genotype and phenotype, can be visualized at once.
 - Visualizing **phenotypes** is the easiest and oldest visualization technique used in the evolutionary computation field. Normally, an individual's phenotype, is reduced to a single number called fitness. Researchers plot fitness against time or the fitness of other individuals.
 - The **genotype** is the representation of a problem solution. For an easy problem like minimizing the Rastrigin function for one dimension it can be a real number or a binary string representing a real value. In the maximization of the Onemax function for two dimensions it can be as simple as a pair of real numbers or a bit string representing the values of x and y . For the TSP it can be an integer vector representing a path through the cities. If possible, genotype visualization allows us to see the structure of the solutions.

This is a less common way of visualizing what is happening with our evolutionary algorithm, and is not usually as easy as showing phenotype progress. For problems involving three or fewer dimensions, we can plot fitness as a function of genotype. The problem appears when more dimensions are involved, where only a multidimensional scaling method accomplishes this task.
- **How many generations?** Two kinds of data can be obtained. Firstly, in order to get a picture of the EA progress, we can represent data produced over many generations. Secondly, a graphic representation of the state of the algorithm is obtained with the data yielded in every new generation.
 - Looking at one generation's data will yield a picture of the **state** of the population at that moment. Many conclusions can be drawn from a look at the state of an evolutionary algorithm: the existing degree of diversity, the areas of the search space being explored. Any of the following pictures and sequences (figures 5 and 6) will answer some questions about what solutions look like and how good they are.
 - Sometimes data from many generations can be seen at the same time, showing the **course** of the evolutionary process. One of the major drawbacks of evolutionary computation is the lack of user interaction during the process. By using some kind of visualization we can follow the process much better and stop it if it is getting lost in the search space or, even better, tune it to improve its work.

3 Multidimensional visualization

For the visualization of multidimensional data, a method to transform multidimensional data to a lower dimension (preferably 2 or 3) is needed. This transformation should provide a lower-dimensional picture where the dissimilarities between the data points of the multidimensional domain correspond to the dissimilarities of the lower-dimensional domain. These transformation methods are referred to as **multidimensional scaling** ([1–3]).

To measure the dissimilarity, the distances between pairs of data points is used. These distances tend to be genuine distances in the high-dimensional domain, such as: euclidean distance; Manhattan distance; or curvilinear distance.

There exist many projection methods with different strengths and weaknesses. In [4] König gives a survey of previous and new unsupervised projection techniques with an emphasis to their computational complexity and structure preservation. Some of the more well known are Principal Component Analysis (PCA) [5], Sammon's projection [6], Curvilinear Component Analysis (CCA) [7], Curvilinear Distance Analysis (CDA) [8], k-means based ones [9], Self-Organizing Maps (SOM) [10] and Locally Linear Embedding (LLE) [11].

Linear methods, like PCA, are computationally light but fail to handle nonlinear structures. Nonlinear methods, such as Sammon's projection, CCA and CDA, can handle nonlinear structures, but they are computationally intensive. They also emphasize local distances over global ones and are therefore excellent for showing the local shape of a data set.

The self-organizing map is a data visualization technique developed by Teuvo Kohonen which reduces the dimensions of data through the use of self-organizing neural networks. The SOM algorithm is based on unsupervised, competitive learning. It provides a topology preserving mapping from the high dimensional space to map units. Map units, or neurons, usually form a two-dimensional lattice and thus creating a mapping from a high dimensional space onto a plane. The topology preserving property means that the mapping preserves the relative distance between the points. Points that are close to each other in the input space are mapped to nearby map units in the SOM. The SOM can thus serve as a cluster analyzing tool of high-dimensional data. Also, the SOM has the capability to generalize, which means that the network can recognize or characterize inputs it has never encountered before. A new input is assimilated with the map unit it is mapped to. With other methods, once the data points are transformed from high-dimensional space to the lower-dimensional one, no new points can be projected without repeating the whole process. SOM has the advantage of being able to transform very easily new points between spaces very easily once it has been trained.

4 Visual effect of a genetic operator on an individual

We are going to study the effects of a genetic operator on an individual. An evolutionary algorithm to train multilayer perceptrons was used. Several runs of

this algorithm with identical parameters and initial populations were recorded. The only difference between them is that only one genetic operator was used at a time. Population size was reduced to 1 to study how the operator affects this single initial individual (the same for all the experiments). Two unary operators were studied: a random additive mutation and another mutation based on Fahlman's Quickprop algorithm [12]. The dataset used in this experiment is ecoli [13] and cross validation was employed (50% for training, 25% for validation and 25% for test).

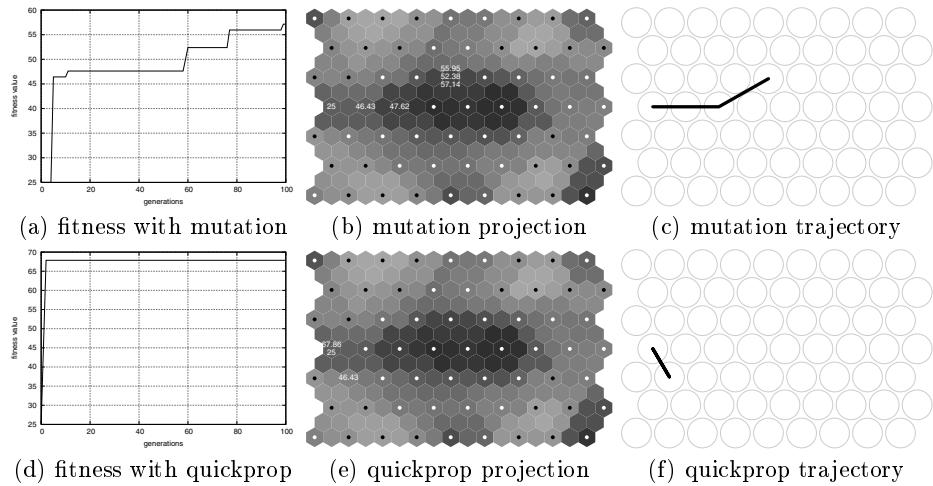


Fig. 1. Trajectories of a multilayer perceptron evolved with an evolutionary algorithm. (a, b, c) Left column represents the evolution of the net using random additive mutation only. (d, e, f) Right column represents the evolution of the net using mutation based on quickprop only. Numbers in figures are fitness values representing accuracy percentage over the ecoli test set.

A SOM was trained with 10000 unique individuals collected over 100 runs. This way the map represents the search space explored by the algorithm during the neural net training. The SOM and the projections of these two runs can be seen in figure 1.

As the algorithm uses elitism, the fitness values in figures 1a and 1d never decrease. The algorithm behavior is predictable: mutation causes random jumps between different areas represented by the SOM cells (figure 1b and 1c). Also, the fitness values increase in a random fashion (figure 1a) and never reach a near optimum value. With quickprop the effect is the opposite: perceptron accuracy improvement is continuous until a local optima is found (figure 1d) and the changes in its internal representation are subtle. Thus its projection over the SOM hardly moves from one cell to another (figure 1e and 1f). With mutation, accuracy increases gradually and better nets are found up until the last

generations. With quickprop, improvement is very fast and only happens in the firsts few generations since it gets trapped in a local maximum. Using mutation the route traveled by an individual during its transformation used to be longer (figure 1c).

Solutions found by each genetic operator are different as can be seen in figures 1b and 1e, or in figures 1c and 1f. Contrary to the advantage of genotype visualization with SOM, this information cannot be extracted from fitness graphs from figures 1a and 1d.

5 Visual effects of selective pressure

To study how selective pressure affects visually the evolution of a population a new set of 20 experiments were performed per parameter combination. This time the only changing parameter is the selective pressure. Tournament selection will be used, and its parameters, the number of individuals are preselected from the whole population, t , and how many of these are going to breed, w , will be changed.

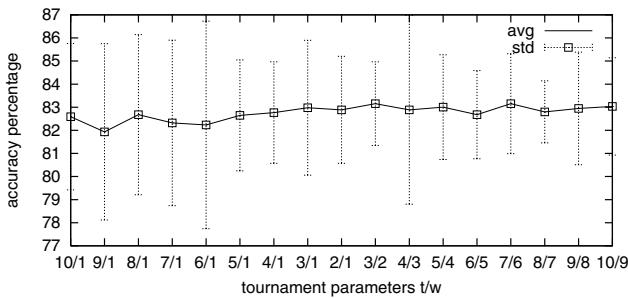


Fig. 2. Average and standard deviations of a set of experiments with several combinations of tournament parameters t and w (see section 5). X axis represents t/w . Y axis represents fitness values as recognition accuracy over ecoli test set.

In figure 2 we can see average and standard deviation of maximum fitness values obtained for every set of experiments with the same combination of parameters t/w . Selective pressure does not seem to have a significant affect on the evolutionary algorithm. In any case, best results are obtained with moderated values around 4/3.

To visualize the effects of selective pressure two experiments will be projected, one with the highest value 10/1 and another with the lowest one, 10/9. Figure 3 shows the evolution of fitness values for both experiments. The SOM used for projection is the one shown in figure 4.

From the projections of figures 5 and 6 several conclusions can be extracted. The evolutionary process progress differently in the two runs. In the first, the

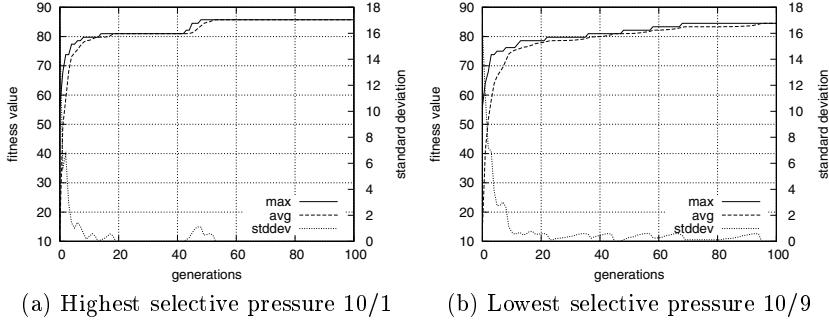


Fig. 3. Fitness evolution for two experiments with different selective pressure. (a) Highest selective pressure 10/1. (b) Lowest selective pressure 10/9.

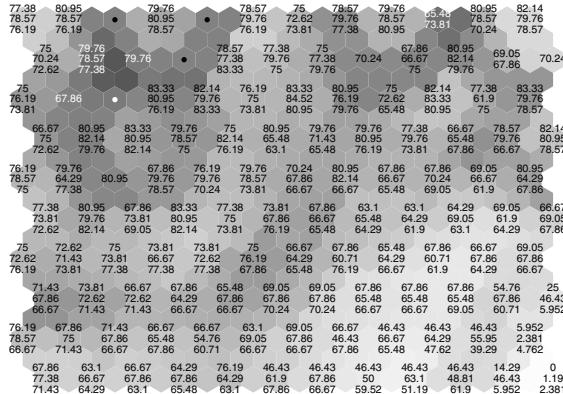


Fig. 4. SOM used for population projection of two experiments with different values of selective pressure in figures 5 and 6.

high selective pressure leads the population to a premature convergence in a few generations. In 5 generations only two very similar areas of the search space remain in exploration. In second run the convergence process takes more generations to happen and some degree of diversity is maintained throughout the whole evolutionary process.

Another difference is the area of search space that has been explored. The run with high pressure, 10/1, only explores 26.62% of the search space represented by the SOM in figure 4. On the other side, the run with low selective pressure, 10/9, is able to explore 42.21% of the same search space.

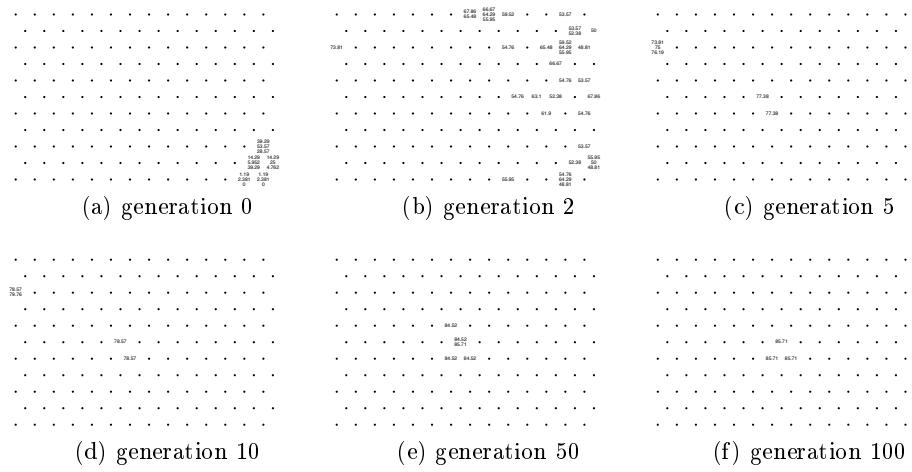


Fig. 5. Projection of the evolution of multilayer perceptrons with the highest selective pressure 10/1. Its fitness evolution can be observed in figure 3a.

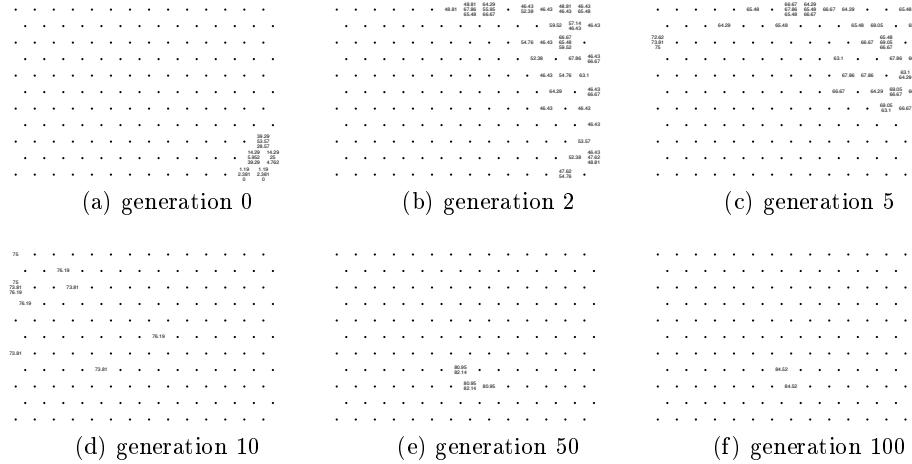


Fig. 6. Projection of the evolution of multilayer perceptrons with the lowest selective pressure 10/9. Its fitness evolution can be observed in figure 3b.

6 Conclusions and future work

SOM can be used to represent search space for evolutionary computation problems. In this work we used it to see how a genetic operator affects to an individual in section 4 and after that we used it to see what is happening inside several

runs of the same algorithm just changing one parameter, the selective pressure, in section 5.

In the first case, we satisfy our curiosity about what regions of the search space are crossed by an individual during its evolution. In the second case, we discover that although the selective pressure does not affect in a significant way the quality of the best solution obtained, it affects the population helping to maintain or to eliminate its diversity degree. All this information cannot be extracted from the simple fitness versus time graph, but it can be easily recovered from the projection of the population over the representation of its search space that the SOM is.

It would be interesting to apply this visualization technique to other genetic operators, as well as to selection and replacement algorithms to discover how they work.

References

1. T.F. Cox and M.A.A. Cox. *Multidimensional Scaling*. London: Chapman & Hall, 1994.
2. B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge, GB: Cambridge University Press, 1996.
3. L. Tsogo and M. Masson. Multidimensional scaling methods for many-objects sets: a review. *Multivariate Behavioral Research*, 35(3):307–320, 2000.
4. Andreas König. A survey of methods for multivariate data projection, visualisation and interactive analysis. In *Proceedings of the 5th International Conference on Soft Computing and Information/Intelligent Systems (IIZUKA'98)*, pages 55–59, October 1998.
5. Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.
6. J.W. Sammon Jr. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, 18:401–409, 1969.
7. Pierre Demartines and Jeanny Hrault. Curvilinear component analysis: a self organizing neural network for non linear mapping of data sets. *IEEE Transactions on Neural Networks*, 8:148–154, 1997.
8. John Aldo Lee, Amaury Lendasse, and Michael Verleysen. Curvilinear Distance Analysis versus Isomap. In *ESANN 2002, 10th European Symposium on Artificial Neural Networks*, pages 185–192, Bruges (Belgium), April 2002.
9. Arthur Flexer. On the use of self-organizing maps for clustering and visualization. In *Principles of Data Mining and Knowledge Discovery*, pages 80–88, 1999.
10. Teuvo Kohonen. *The Self-Organizing Maps*, volume 30 of *Information Sciences*. Springer-Verlag, second extended edition, 1997.
11. Sam T. Roweis and Lawrence K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290:2323–2336, December 2000.
12. S.E. Fahlman. Faster-Learning Variations on Back-Propagation: An Empirical Study. In *Proceedings of the 1988 Connectionist Models Summer School*. Morgan Kaufmann, 1988.
13. Paul Horton and Kenta Nakai. Better Prediction of Protein Cellular Localization Sites with the k Nearest Neighbors Classifier. In *In Proceeding of the Fifth International Conference on Intelligent Systems for Molecular Biology*, pages 147–152, Menlo Park. USA. AAAI Press.

Separable Recurrent Neural Networks Treated with Stochastic Velocities

A. Castellanos-Moreno

February 27, 2003

Departamento de Física, Universidad de Sonora
A. Postal 1626, Hermosillo 83000, Son., México.

Abstract

Stochastic velocities are taken from the kinematics of Stochastic Mechanics to study finite-size effects in separable recurrent neural networks. Since time-evolution of the physical state of these finite neural networks can be studied through the separation of their deterministic behaviour in the overlap space, plus a Gaussian noise around their paths, closed expressions for the stochastic velocities are found. Then their application to study different models of neural networks is a matter of algebra. This mathematical tool give us new insight in the physics of the problems, as it is demonstrated by revisiting one example, solved in previous literature: a neural network presenting a fixed point without detailed balance.

PACS: 87.30, 05.20

1 Introduction

The goal of this paper is to study separable recurrent neural networks (SRNN), with sequential dynamics and far away from the saturation regime, by using stochastic velocities. A mathematical tool developed within the framework of Stochastic Mechanics (SM) [1].

SRNN has been studied in the limit of an infinite number, N , of neurons (a review can be found in [2]). It was found that their macroscopic description can be considered as a nonlinear dynamical system. Besides, finite SRNN with sequential dynamics, and far away from saturation, has been studied by using stochastic processes as described by the Fokker-Planck equation [3].

SM is a theory developed to obtain Quantum Mechanics from stochastic processes occurring in the configuration space [4][5]. Besides several quantum mechanical systems has been studied[6]. At least two types of dynamics has been distinguished in the literature: a) one that sometimes is named as Einstein processes, to be described by the linear Fokker-Planck equation, and b) other

type to be named de Broglie processes, identified with the quantum behaviour [7][8]. Paying attention to processes (a), SM can be used to get a better understanding of another physical systems (please see [9] for details). One of them is SRNN.

This paper is organized as follows: the general formalism and one example is presented in the 2nd. section; stochastic kinematics is discussed in the 3th. section by presenting the formalism and by getting new insight about the example mentioned in the 2nd. section. A short conclusion is included at the end.

2 General formalism and some results

General formalism and one example are presented now. Deeper details can be seen in [3][12]. SRNN is a system composed by a large, but finite, number N of interconnected neurons $\sigma \in \{-1, 1\}^N$, whose state, at a given time, is described by the vector $\sigma(t) = (\sigma_1(t), \sigma_2(t), \dots, \sigma_N(t))$. Each variable $\sigma_i(t)$ evolves in time by describing sequential stochastic alignment of the spins under the action of an external field given by $h_i(\sigma) = \sum_j J_{ij}\sigma_j + \theta_i$, with $J_{ij} = \frac{1}{N} \sum_{\mu\nu=1}^p \xi_i^\mu A_{\mu\nu} \xi_j^\nu (1 - \delta_{ij})$ the synaptic strength of the connection going from neuron j to neuron i ; θ_i is a response threshold, and the autointeractions J_{ii} are excluded. The network can store a finite number $p \ll N$ of quenched binary patterns $\xi_i = (\xi_i^1, \dots, \xi_i^p) \in \{-1, 1\}^p$, with $\{\xi_i^\mu\}$ chosen at random and $\mu = 1, \dots, p$. The presence of the matrix $\{A_{\mu\nu}\}$ permit us to introduce nonsymmetric interactions. The microscopical dynamics of the system is defined by a master equation for the probability density $p_t(\sigma)$, given by

$$\frac{d}{dt}p_t(\sigma) = \sum_i \left\{ w_i(\hat{F}_i\sigma)p_t(\hat{F}_i\sigma) - w_i(\sigma)p_t(\sigma) \right\}. \quad (1)$$

The transition $\sigma_i(t) \rightarrow -\sigma_i(t)$, occurs with a probability

$$w_i(\sigma) = \frac{1}{2}[1 - \sigma_i \tanh(\beta h_i(\sigma))],$$

$T = \beta^{-1}$ is the temperature, and \hat{F}_i is an operator such that $\hat{F}_i f(\sigma_1, \dots, \sigma_N) = f(\sigma_1, \dots, -\sigma_i, \dots, \sigma_N)$.

A macroscopical description of the system is obtained by defining the order parameters $m_\mu = \frac{1}{N} \sum_i \xi_i^\mu \sigma_i$, $\mu = 1, \dots, p$, to compare the microscopic state at time t , of the neural network, σ , with each one of the patterns stored, ξ^μ . These are separated in two parts: $m_\mu(t) = m_\mu^* + N^{-\frac{1}{2}}q_\mu(t)$, such that the first one obeys the well known mean-field law:

$$\frac{d}{dt}m^*(t) = \langle \xi \tanh \beta [\xi \cdot \mathbf{A}m^*(t) + \theta] \rangle_{\xi, \theta} - m^*(t), \quad (2)$$

and the second term is an stochastic process defined as

$$\mathbf{q}(\sigma) = \sqrt{N} [\mathbf{m}(\sigma) - \mathbf{m}^*(t)] \quad \mathcal{P}_t(\mathbf{q}) = \int d\mathbf{m} P_t(\mathbf{m}) \delta \left[\mathbf{q} - \sqrt{N} [\mathbf{m} - \mathbf{m}^*(t)] \right]. \quad (3)$$

The probability density of this fluctuating part $\mathbf{q}(\sigma)$, satisfies the next Fokker-Planck equation:

$$\frac{\partial}{\partial t} \mathcal{P}_t(\mathbf{q}) = \sum_{\mu=1}^p \frac{\partial}{\partial q_\mu} \{ \mathcal{P}_t(\mathbf{q}) F_\mu[\mathbf{q}; t] \} + \sum_{\mu=1}^p \sum_{\nu=1}^p \frac{\partial^2}{\partial q_\mu \partial q_\nu} \{ \mathcal{P}_t(\mathbf{q}) D_{\mu\nu}[\mathbf{m}^*; t] \} \quad (4)$$

in which the flow term $\mathbf{F} = \{F_\mu[\mathbf{q}; t]\}$ can be written as the sum of two terms: a vector \mathbf{K} depending on the specific microscopic realization of the pattern components; and the matrix \mathbf{L} (acting on \mathbf{q}) depending only on \mathbf{m}^* . Since this FPE describes a time dependent Ornstein-Uhlenbeck process, its formal solution is Gaussian:

$$\mathcal{P}_t(\mathbf{q}) = \frac{\exp \left\{ -\frac{1}{2} [\mathbf{q} - \langle \mathbf{q} \rangle_t] \cdot \boldsymbol{\Xi}^{-1}(t) [\mathbf{q} - \langle \mathbf{q} \rangle_t] \right\}}{(2\pi)^{\frac{p}{2}} \sqrt{\det \boldsymbol{\Xi}(t)}}, \quad (5)$$

so that the process can be fully characterized by its two first statistical moments, which are found to be given by

$$\frac{d}{dt} \langle \mathbf{q} \rangle_t + \mathbf{F}[\mathbf{q}; t] = 0, \quad \frac{d}{dt} \boldsymbol{\Xi}(t) = -\mathbf{L}\boldsymbol{\Xi}(t) - \boldsymbol{\Xi}(t)\mathbf{L}^T(t) + 2\mathbf{D}(t) \quad (6)$$

where $\boldsymbol{\Xi}$ is the time-dependent centered correlation matrix for the fluctuations $\Xi_{\mu\nu} = \langle q_\mu q_\nu \rangle - \langle q_\mu \rangle \langle q_\nu \rangle$. Several models has been studied in the literature cited above.

2.1 Example: a fixed point without detailed balance

This model is a neural network operating at temperature $T = \beta^{-1}$ with two stored patterns, $p = 2$ and defined by the matrix $A = \{\{1, \varepsilon\}, \{0, 1\}\}$. It has a stable fixed point in $(m_\beta, 0)$, where m_β is the solution of the equation: $m_\beta = \tanh(\beta m_\beta)$, (please see [3] for details). With $K_0 = \frac{1}{\sqrt{N}} \sum_i \xi_i^\mu \xi_i^\nu$, the coefficients of the Fokker-Planck equation (4) are:

$$\mathbf{L}(\mathbf{m}_\infty^*) = \begin{pmatrix} L_{11}^\infty & L_{12}^\infty \\ L_{21}^\infty & L_{22}^\infty \end{pmatrix} = \begin{pmatrix} 1 - \beta + \beta m_\beta & -\beta \varepsilon (1 - m_\beta) \\ 0 & 1 - \beta + \beta m_\beta \end{pmatrix}. \quad (7)$$

$$\mathbf{K}^{(\infty)} = \begin{pmatrix} 0 \\ -K_0 m_\beta \end{pmatrix} \quad \mathbf{D}^{(\infty)} = (1 - m_\beta) \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \quad (8)$$

Computer simulations support the results listed below: a) fluctuations around the fixed point are

$$\langle q_1 \rangle_\infty (\beta, \varepsilon) = \frac{\beta \varepsilon \left(1 - m_\beta^2\right) K_0 m_\beta}{(1 - \beta + \beta m_\beta)^2} \quad \langle q_2 \rangle_\infty (\beta, \varepsilon) = \frac{\left(1 - \beta + \beta m_\beta^2\right) K_0 m_\beta}{(1 - \beta + \beta m_\beta)^2} \quad (9)$$

With the next short writing $y = L_{12}^\infty / L_{11}^\infty$, the centered correlation matrix has the next components: $\Xi_{11} = -\frac{1}{\beta \varepsilon} (y + y^3)$, $\Xi_{12} = \frac{1}{2\beta \varepsilon} y^2$, and $\Xi_{22} = -\frac{1}{\beta \varepsilon} y$.

3 Stochastic kinematics applied to finite neural networks

3.1 General formalism

Stochastic velocities are presented in this section. For details please see [9]. An stochastic process, $\mathbf{q}(t)$, must have associated a velocity, $\mathbf{v}(t)$, to measure the changes of the position of the state-point when time elapses, but since $\mathbf{q}(t)$ is a stochastic process, the total time-derivative operator to relate it with the velocity can not be used, so that the expression: $\mathbf{v}(t) = \frac{d}{dt} \mathbf{q}(t)$, is not true any more. To solve this problem, velocity has been written as the sum of a systematic (current) velocity, \mathbf{v}_c , plus a stochastic component, \mathbf{u} : $\mathbf{v} = \mathbf{v}_c + \mathbf{u}$ [7]. The behavior of these velocities can be classified by introducing an operator \widehat{T} to produce a temporal inversion as $\widehat{T}\mathbf{G}(\mathbf{q}, t) = \mathbf{G}(\mathbf{q}, -t)$, where $\mathbf{G}(\mathbf{q}, t)$ is a dynamical variable of the physical system. Time-inverted velocities are $\tilde{\mathbf{v}}_c = \widehat{T}\mathbf{v}_c$ and $\tilde{\mathbf{u}} = \widehat{T}\mathbf{u}$. There are physical arguments [7] to expect the next transformation rules under time-inversion: $\tilde{\mathbf{v}}_c = -\mathbf{v}_c$ and $\tilde{\mathbf{u}} = \mathbf{u}$. So that $\tilde{\mathbf{v}} = -\mathbf{v}_c + \mathbf{u}$. Therefore $\mathbf{v}_c = \frac{1}{2}(\mathbf{v} - \tilde{\mathbf{v}})$, and $\mathbf{u} = \frac{1}{2}(\mathbf{v} + \tilde{\mathbf{v}})$.

New operators has been obtained in stochastic mechanics to relate $\mathbf{q}(t)$ with $\mathbf{v}(t)$, these are [7]:

$$\widehat{\mathcal{D}} = \frac{\partial}{\partial t} + \mathbf{v} \cdot \nabla_q + \nabla_q \cdot \mathbf{D} \nabla_q, \quad \widetilde{\mathcal{D}} = -\frac{\partial}{\partial t} + \tilde{\mathbf{v}} \cdot \nabla_q + \nabla_q \cdot \mathbf{D} \nabla_q \quad (10)$$

with v_μ , $D_{\mu\nu}$ the limits when $\Delta t \rightarrow 0$ of the first and second order moments of a coarse-graining average on a joint-probability distribution $\mathcal{P}(\mathbf{q} - \mathbf{q}', t)$ useful to smooth the roughness of the stochastic paths in a local neighborhood (cell), \mathcal{B} , of the point \mathbf{q} :

$$E\{F(\mathbf{q}, t)\} = \lim_{\Delta t \rightarrow 0} \langle \frac{\Delta F(\mathbf{q}, t)}{\Delta t} \rangle_{\mathcal{B}} = \int_{\mathcal{B}} dq'_1 \cdots dq'_p \mathcal{P}(\mathbf{q} - \mathbf{q}', t) \left(\frac{\Delta F(\mathbf{q}, t)}{\Delta t} \right), \quad (11)$$

with $\mathcal{P}(\mathbf{q} - \mathbf{q}', t)$ normalized inside the cell \mathcal{B} and

$$F(\mathbf{q}, t) = \Delta q_\mu(t), \quad \text{or} \quad 2\Delta q_\mu(t)\Delta q_\nu(t).$$

This average give us locally averaged quantities in the same sense that moving average is used to smooth time series in mathematics. The limit $\Delta t \rightarrow 0$ can not be taken arbitrarily, instead of that, two times must be considered: the smaller time T_0 needed to observe a systematic motion, and the correlation time t_c of the stochastic process $\mathbf{q}(t)$. The interval Δt must be such that $T_0 \gg \Delta t \gg t_c$. Relations between $\mathbf{q}(t)$ and the current velocities are

$$\mathbf{v}(t) = \widehat{\mathcal{D}}\mathbf{q}(t), \quad \tilde{\mathbf{v}}(t) = \widetilde{\mathcal{D}}\mathbf{q}(t) \quad (12)$$

to be taken as the sought-for generalization of the total time-derivative used in Newtonian mechanics. A mathematical approach can be found in [13].

Defining the operators:

$$\widehat{\mathcal{D}}_c = \widehat{\mathcal{D}}^{(-)} = \frac{1}{2} (\widehat{\mathcal{D}} - \widetilde{\mathcal{D}}) \quad \widehat{\mathcal{D}}_s = \widehat{\mathcal{D}}^{(+)} = \frac{1}{2} (\widehat{\mathcal{D}} + \widetilde{\mathcal{D}}), \quad (13)$$

one gets another relations between current and stochastic velocities with the process $\mathbf{q}(t)$:

$$\mathbf{v}_c = \widehat{\mathcal{D}}_c \mathbf{q}(t), \quad \mathbf{u} = \widehat{\mathcal{D}}_s \mathbf{q}(t) \quad (14)$$

It is easy to note that $\widehat{\mathcal{D}}_c = \widehat{\mathcal{D}}$, and $\widehat{\mathcal{D}}_s = \mathbf{u} \cdot \nabla_q + \nabla_q \cdot \mathbf{D} \nabla_q$.

Another useful combination of operators is the exit operator $\widehat{\mathcal{D}}_e = \widehat{\mathcal{D}}_c + \widehat{\mathcal{D}}_s$, identical to $\widehat{\mathcal{D}}$. Thus, \mathbf{v} is denoted \mathbf{v}_e and renamed as the exit velocity. Besides one can define the access operator $\widehat{\mathcal{D}}_a = \widehat{\mathcal{D}}_c - \widehat{\mathcal{D}}_s = -\widetilde{\mathcal{D}}$, to get the access velocity $\mathbf{v}_a = \mathbf{v}_c - \mathbf{u}$. The relation of these velocities to the process $\mathbf{q}(t)$ is

$$\mathbf{v}_e = \widehat{\mathcal{D}}_e \mathbf{q}(t), \quad \mathbf{v}_a = \widehat{\mathcal{D}}_a \mathbf{q}(t) \quad (15)$$

If one has an ensemble of neural networks, one has a cloud of state-points in the space $\{\mathbf{q}\}$, then \mathbf{v}_a is the local average velocity of the state-points reaching the point \mathbf{q} , whereas \mathbf{v}_e is the local average velocity of the state-points leaving the point \mathbf{q} .

Once that one arrives at this level of the formalism, the usual step to get a dynamics in SM is to build an expression for the stochastic acceleration and a relation with a stochastic force. But in SRNN one does not have a force in the Newtonian sense, then, the idea has been to follow the approach presented in [10], to relate the coefficients, A_μ and $D_{\mu\nu}$, of the forward Fokker-Planck equation (4) to the stochastic velocities (please see [9] for details). One gets

$$v_\mu^e(\mathbf{q}, t) = A_\mu(\mathbf{q}, t), \quad u_\mu = \sum_{\nu=1}^p \frac{1}{\mathcal{P}(\mathbf{q}, t)} \frac{\partial (D_{\mu\nu} \mathcal{P}(\mathbf{q}, t))}{\partial q_\nu}. \quad (16)$$

With $v_\mu^c = v_\mu^e - u_\mu$ and $v_\mu^a = v_\mu^e - 2u_\mu$, one can determine the current and the access velocities. The diffusion velocity is written in terms of the distribution probability, $\mathcal{P}(\mathbf{q}, t)$, so that the solution to the Fokker-Planck equation is needed to know u_μ . In general, this is the weak point of this approach, but SRNN are described in terms of gaussian processes, so that it is easy, and useful, to find the stochastic velocities for time-dependent Ornstein-Uhlenbeck processes as

$$\mathbf{u} = -\mathbf{D}\mathbf{\Xi}^{-1}(\mathbf{q} - \langle \mathbf{q} \rangle), \quad \mathbf{v}_e = -\mathbf{L}\mathbf{q} - \mathbf{K} \quad (17)$$

$$\mathbf{v}_c = -\mathbf{L}\mathbf{q} - \mathbf{K} + \mathbf{D}\mathbf{\Xi}^{-1}(\mathbf{q} - \langle \mathbf{q} \rangle), \quad \mathbf{v}_a = -\mathbf{L}\mathbf{q} - \mathbf{K} + 2\mathbf{D}\mathbf{\Xi}^{-1}(\mathbf{q} - \langle \mathbf{q} \rangle) \quad (18)$$

3.2 Applications to SRNN: Fixed point without detailed balance revisited

For this case, exit and diffusion velocities are

$$\mathbf{v}_e = \begin{pmatrix} -f_\beta q_1 - \beta\varepsilon G_\beta q_2 \\ -f_\beta q_2 + K_0 m_\beta \end{pmatrix} \quad \mathbf{u} = \begin{pmatrix} U_1 q_1 + U_2 q_2 + U_3 \\ U_2 q_1 + U_1 q_2 + W_3 \end{pmatrix}, \quad (19)$$

$$\begin{aligned} U_1 &= \frac{1}{d_n} g_\beta (4 + 8\beta G_\beta + 4\beta^2 G_\beta^2), \quad U_2 = \frac{1}{d_n} 2\varepsilon \beta G_\beta^2 g_\beta, \\ U_3 &= \frac{2K_0 m_\beta \beta G_\beta g_\beta}{d_n}, \quad W_3 = \frac{2K_0 m_\beta}{d_n} [-2 - 4\beta G_\beta + (3\varepsilon^2 - 2)\beta^2 G_\beta^2], \end{aligned}$$

where $d_n = -4 - 8\beta G_\beta + \beta^2 G_\beta^2 (5\varepsilon^2 - 4)$, $G_\beta = m_\beta^2 - 1$, $f_\beta = 1 - \beta G_\beta$, and $g_\beta = 1 + \beta G_\beta$.

The current velocity is

$$\mathbf{v}_c = \varepsilon \begin{pmatrix} A_1 q_1 + A_2 q_2 - U_3 \\ B_1 q_1 + B_2 q_2 + B_3 \end{pmatrix}, \quad (20)$$

with

$$\begin{aligned} A_1 &= -\frac{5\varepsilon}{d_n} (\beta G_\beta)^2 g_\beta, \quad A_2 = \frac{1}{d_n} \{ 2\beta G_\beta (1 + 2\beta G_\beta) + \beta^3 G_\beta^3 (2 - 5\varepsilon^2) \}, \\ B_1 &= -\frac{2\beta G_\beta}{d_n} (1 + 2\beta G_\beta + \beta^2 G_\beta^2), \quad B_2 = -\frac{\varepsilon (\beta G_\beta)^2}{d_n} g_\beta, \\ B_3 &= -\frac{\varepsilon K_0 m_\beta}{d_n} (\beta G_\beta)^2. \end{aligned}$$

Detailed balance is recovered when $\varepsilon = 0$. In this case, one gets $\lim_{\varepsilon \rightarrow 0} \mathbf{v}_c = 0$ and $\lim_{\varepsilon \rightarrow 0} \mathbf{u} \neq 0$. Thus, an ensemble of neural networks give us a cloud of state-points in the neighborhood of the fixed point $\mathbf{q}_\infty = (\langle q_1 \rangle_\infty(\beta, \varepsilon), \langle q_2 \rangle_\infty(\beta, \varepsilon))$, presenting agitation, but without a systematic motion to go far away from \mathbf{q}_∞ . Since \mathbf{v}_c vanishes, component by component, one can take it as the coarsen version of the detailed balance at the microscopic level.

For nondetailed balance ($\varepsilon \neq 0$) one has $\mathbf{v}_c \neq 0$. This is, just apparently, a contradiction because in one hand, there is a fixed point, on the other hand, the current velocity is not zero. It is easily solved by introducing an angular momentum, $\mathbf{M}(\mathbf{q})$. In three dimensions

$$\begin{aligned} \mathbf{M}(\mathbf{q}) &= \mathbf{q} \times \mathbf{v}_c = \sum_{\mu, \nu=1}^3 \epsilon_{\gamma \mu \nu} \hat{r}_\gamma q_\mu v_\nu^c \\ &= \hat{k} \varepsilon [B_1 q_1^2 - A_2 q_2^2 + (B_2 - A_1) q_1 q_2 + B_3 q_1 + U_3 q_2]. \end{aligned} \quad (21)$$

The last expresion is found by written $q_3 = 0$. Note that $\mathbf{M}(\mathbf{q}) \neq 0$ if $\varepsilon \neq 0$, so that the current velocity is due to a rotation of the vector, $\mathbf{m}(t) = \mathbf{m}_\infty^* + \frac{1}{\sqrt{N}} \mathbf{q}(t)$, around the fixed point. This new picture can be compared with the current probability's rotation found in [3]

$$\nabla \times \mathbf{J}(\mathbf{q}) = -\varepsilon \beta \left[1 - (m_\beta)^2 \right] \mathcal{P}(\mathbf{q}), \quad (22)$$

where $\mathbf{J}(\mathbf{q}) = \mathcal{P}(\mathbf{q}) [\mathbf{D}\Xi^{-1} - \mathbf{L}] (\mathbf{q} - \langle \mathbf{q} \rangle_\infty)$.

Now, the absence of detailed balance can be visualized with the presence of a vortex around \mathbf{m}_∞^* .

4 Conclusions

Stochastic velocities give us a useful tool to study finite-size effects in SRNN. One model has been revisited to show how these concepts can help us to get a better picture of the phenomenon under consideration. It was demonstrated that a neural network with a fixed point without detailed balance is such that the state in the overlap space has a rotation with an angular momentum associated to it. When one has a nonzero angular momentum, the neural network does not have detailed balance.

References

- [1] Nelson E., “Quantum Fluctuations”, Princeton University Press (New Jersey 1985) and references there in.
- [2] Coolen A.C.C., and D. Sherrington, ‘Dynamics of Attractor Neural Networks’, in ‘Mathematical Approaches to Neural Networks’ (North-Holland 1993; ed. J.G. Taylor), 293-306

- [3] Castellanos A., Coolen A.C.C., and Viana L. “Finite-size effects in separable recurrent neural networks”, J. Phys. A. **31**, 6615-6634, (1998). L. Viana, A. Castellanos, A.C.C. Coolen, “Finite Size Effects in Neural Networks”, in “Foundations and Tools for Neural Modeling Lecture Notes in Computing Science” **1606**, J. M. Mira, and J. V. Sánchez-Andrés, (eds.), Springer (Berlin 1999). A. Castellanos, and L. Viana, “Stationary Processes and Equilibrium States in Non-symmetric Neural Networks”, RMF **48**, 310-316, (2002).
- [4] Nelson E., “Derivation of the Schrödinger equation from Newtonian mechanics”, Phys. Rev. **150**, 1079-1085 (1966).
- [5] Nelson E., “*Dynamical Theories of Brownian Motion*”, Princeton U., (Princeton 1966).
- [6] Namsrai K., “Nonlocal Quantum Field Theory and Stochastic Quantum Mechanics”, Reidel (Tokyo 1986) and references there in.
- [7] de la Peña L., “New formulation of the stochastic theory and Quantum Mechanics”, J. Math. Phys. **10**, 1620-1630 (1968).
- [8] Santos E., “Brownian motion an the stochastic theory of Quantum Mechanics”, in: “Irreversibility and the many-body problem”, J. Biel and J. Rae, eds. Plenum (New York 1973).
- [9] Castellanos A., “Random Systems Described with Stochastic Velocities”, Physica A 316, 189-202 (2002).
- [10] de la Peña L. and Cetto A. M., “The Quantum Dice (an introduction to stochastic electrodynamics)”, Kluwer Academic Publishers, (Dordrecht 1996) and references there in.
- [11] Laughton, S.N., and A.C.C. Coolen, ‘Macroscopic Lyapunov Functions for Separable Stochastic Neural Networks with Detailed Balance’, J. Stat. Phys. **80** (1995), 375-387.
- [12] Castellanos, A., Ph. D. Thesis, CICESE-UNAM, México (1998).
- [13] Blanchard Ph., Ph. Combe, and W. Zheng, “Mathematical and Physical Aspects of Stochastic Mechanics, Lectures Notes in Physics” **281**, Springer-Verlag, (Berlin 1987). Nelson E., in Ecole d’Ete de Probabilités de Saint-Flour XV-XVII, Lectures Notes in Mathematics **1362**, P.L. Hennequin ed., Springer-Verlag, (New York 1988). Pavon M., “Derivation of the wave function collapse in the context of Nelson’s stochastic mechanics”, J. Math. Phys. 40, 5565-5577, (1999).

Studying the Convergence of the CFA Algorithm

Jesús González, Ignacio Rojas, Héctor Pomares and Julio Ortega

Department of Computer Architecture and Computer Technology
University of Granada

Abstract. This paper studies the convergence properties of the previously proposed CFA (Clustering for Function Approximation) algorithm and compares its behavior with other input-output clustering techniques also designed for approximation problems. The results obtained show that CFA is able to obtain an initial configuration from which an approximator can improve its performance.

1 Introduction

Training algorithms for Radial Basis Function Neural Networks (RBFNNs) usually use a clustering step to initialize the RBF centers. This approach has proved successful in classification problems. Nevertheless, in recent papers some authors have attempted to solve function approximation problems by means of clustering techniques without making significant changes to them [3, 7]. Therefore, new clustering algorithms specially fitted to the problem of function approximation, which take into account the differences between both problems, would improve the performance of the approximator. For example, as the design objective in a function approximation problem is to construct the relationships between the input/output examples, it is necessary to incorporate the output variable into the clustering algorithm. Nevertheless, this process can be performed in different ways, as will be explained below.

2 State of the Art

Although clustering algorithms have been widely used to make the initial models of function approximators, they were initially designed for classification problems. The output space in both problems is quite different, being infinite and continuous for the former but finite and discrete for classifiers. Another important difference is the final objective of the model, which is the interpolation of a target function f for new input vectors in function approximation and the assignment of unknown examples to a set of a priori defined set of labels in classification.

The first important change in clustering techniques to obtain a good initialization for function approximation problems is incorporate the target function response in the clustering process. This kind of clustering techniques have been

denominated input-output clustering algorithms [4, 6], because they take into account also the output of the model, not as the traditional clustering algorithms or input clustering techniques. Several approaches have been proposed to incorporate the target function response into the clustering process in order to reach a better initialization of the prototypes. In this section we will summarize two well known input-output clustering algorithms, ACE [6] and CFC [4], and we will also review the functioning of our previously proposed algorithm CFA [2].

2.1 ACE (Alternating Cluster Estimation) Algorithm

Traditionally, membership functions and prototype locations are restricted to particular shapes and positions determined by the updating of equations derived from the objective function for a clustering model. Nevertheless, the final user might be interested in the use of a certain type of membership function which could be better fitted to the problem in question. In [6], an Alternating Cluster Estimation (ACE) is proposed; this uses an alternating iteration architecture, but membership and prototype functions are selected directly by the user.

The way ACE incorporates the available outputs into the training set is by constructing a new training set $Z = \{z^i : i = 1, \dots, n\} \subset \mathbb{R}^{n_i+n_o}$, where each data vector z^i is a concatenation of an input vector $x^i \in \mathbb{R}^{n_i}$ with its corresponding output vector $y^i \in \mathbb{R}^{n_o}$. Once this new training set has been generated, ACE can be used to implement well known clustering algorithms such as HCM (Hard C-Means) and FCM (Fuzzy C-Means) [1], or to design new clustering techniques better fitted to a particular problem. For example, in [6] an instance of ACE is proposed called *Dancing Cones* (DC), based on the use of hyper-conic membership functions to assign the samples to the clusters. Once the final configuration is obtained, the projections of these hyper-cones are isosceles triangles that can be used to form a fuzzy model with IF-THEN rules.

2.2 CFC (Conditional Fuzzy Clustering) Algorithm

The main objective of this algorithm [4] is to develop clusters while preserving the homogeneity of the clustered patterns. This means that samples belonging to the same cluster must meet a similarity criterion in the input and also in the output space.

The CFC algorithm is based mainly on the FCM algorithm, but introduces the concept of *context-sensitivity*, which requires the output variable of a cluster to satisfy a particular condition. This condition or context (termed \mathcal{A}) can be treated as a fuzzy set, with \mathcal{A} being defined via the corresponding membership function. The clustering problem can be reformulated as one of making groups of input data, taking into account that their associated output must be \mathcal{A} , with the restriction \mathcal{A} being expressed as a fuzzy set.

An important consequence of the use of this method is that the computational complexity is reduced by splitting the original problem into a series of context-driven clustering problems, which are usually represented as trapezoidal fuzzy

sets designed by an expert human operator. This feature makes it difficult to compare this method with other approaches, and means it cannot be automated.

2.3 CFA (Clustering for Function Approximation) Algorithm

The goal this clustering algorithm [2] is to increase the density of prototypes in the input areas where the target function presents a more variable response, rather than just in the zones where there are more input examples. This approach is more adequate for function approximation problems because it helps to minimize the approximation error by means of reducing the output variance not explained by the model. As all clustering algorithms, CFA reveals the structure of training data in the input space, but it also preserves the homogeneity in the output responses of data belonging to the same cluster. To carry out this task, this technique tries to minimize the following objective function

$$d = \frac{\sum_{j=1}^c \sum_{\mathbf{x}^i \in P_j} \|\mathbf{x}^i - \mathbf{p}^j\|^2 \omega_{ij}}{\sum_{j=1}^c \sum_{\mathbf{x}^i \in P_j} \omega_{ij}} \quad (1)$$

where ω_{ij} weights the influence of each training example \mathbf{x}^i in the final position of the j -th prototype. This index informs about the existing controversy between the expected output for the prototype \mathbf{p}^j and the output of the training example \mathbf{x}^i . The greater the distance between the expected output of \mathbf{x}^i and the estimated output of the cluster P_j it belongs to, the greater the influence of \mathbf{x}^i in the final result. Mathematically, ω_{ij} is defined as:

$$\omega_{ij} = \frac{|f(\mathbf{x}^i) - o^j|}{\max_{l=1}^n \{f(\mathbf{x}^l)\} - \min_{l=1}^n \{f(\mathbf{x}^l)\}} + \mu_{\min} \quad (2)$$

with $\mu_{\min} > 0$. The first term of the sum calculates a normalized distance (in the range [0,1]) between $f(\mathbf{x}^i)$ and o^j (the estimation of the output response of the cluster P_j), and the second term is a minimum contribution threshold (when there is no controversy ω_{ij} is equal to μ_{\min}). As μ_{\min} decreases, CFA forces the prototypes to concentrate on input zones where the output variability is greater. This action is justified because it preserves the homogeneity in the output space of the points \mathbf{x}^i belonging to each cluster P_j by means of minimizing the distance of $f(\mathbf{x}^i)$ with respect to o^j . On the contrary, if μ_{\min} increases, CFA pays more attention to the input space, acquiring the behavior of HCM for a sufficiently large value of μ_{\min} . This parameter controls the convergence of the proposed algorithm and thus, is studied in detail in the following section. For a detailed description of the clustering algorithm, the reader is referred to [2].

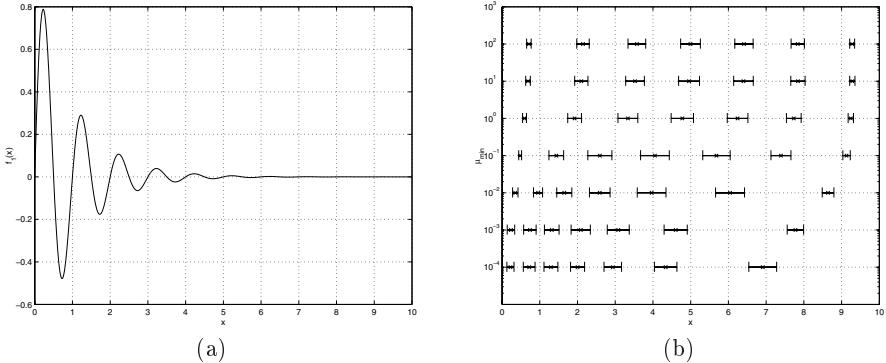


Fig. 1. (a) Function f_1 . (b) Effect of μ_{\min} in the allocation and deviations of prototypes.

3 Study of the Convergence Properties of CFA

The parameter μ_{\min} (2) controls how CFA deals with the output variability in the training data. CFA can converge to a great variety of prototype allocations depending on the value of this parameter. A large value for μ_{\min} implies that CFA pays more attention to input space than to output variability, and as μ_{\min} is lowered, the variability of the target function has a greater influence on the algorithm behavior. To gain an insight into the role of μ_{\min} in the algorithm functioning, an experiment was designed in which μ_{\min} takes the values $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^0, 10^1, 10^2\}$, and for each one of these values, we ran CFA 25 times to approximate the target function

$$f_1(x) = \frac{\sin(2\pi x)}{e^x}, \quad x \in [0, 10] \quad (3)$$

with 7 prototypes, and a training set of 1000 samples of f_1 generated by evaluating inputs taken uniformly from the interval $[0, 10]$.

Figure 1 shows a graph of the mean positions and deviations of the prototypes for each different value of μ_{\min} . This graph reveals that CFA begins to move prototypes to the input zones where f_1 is more variable when μ_{\min} is less than one. This effect is due to the design of equation (2), which assigns a minimum weight of μ_{\min} for examples in input regions where the target function is constant and a maximum weight of $1 + \mu_{\min}$ for examples in more variable zones.

As an example, the CFA algorithm was applied to initialize the RBF centers of an RBFNN. The functioning of this model can be summarized by the following equation:

$$F(\mathbf{x}) = \sum_{j=1}^c \omega_j \phi_j(\mathbf{x}) \quad (4)$$

μ_{\min}	NRMSE (dev)	Time (dev)
10^2	0.964 (0.004)	0.50 (0.20)
10^1	0.963 (0.004)	0.53 (0.25)
10^0	0.956 (0.004)	0.59 (0.20)
10^{-1}	0.942 (0.009)	1.31 (0.43)
10^{-2}	0.846 (0.092)	8.8 (3.8)
10^{-3}	0.666 (0.138)	56 (39)
10^{-4}	0.647 (0.128)	359 (255)

Table 1. Mean and standard deviation of the approximation error and the time (in seconds) required to approximate f_1 for each value of μ_{\min} .

where ϕ_j is an RBF and ω_j is its associated weight. Basis functions are implemented using gaussian functions. After establishing all the parameters concerning the RBFs (centers and radii), the RBFNN becomes a linear model and its weights can be calculated optimally using pseudo-inverse methods [5].

Table 1 shows a representative example of the mean and standard deviation of the approximation error of the models obtained by CFA, and of the computation time needed to reach convergence. The smaller the value of μ_{\min} , the bigger the contribution of the output fluctuations of f_1 to the final prototype allocation. Nevertheless, as μ_{\min} decreases, the time required by the algorithm increases dramatically while the approximation error of the model only suffers small changes. From Table 1 it can be concluded that values of μ_{\min} below 0.0001 do not significantly improve the approximation error, taking into account the computation time needed by CFA to reach convergence.

4 Results

This section compares the CFA algorithm with the other two input-output clustering techniques described in this paper: ACE and CFC.

4.1 CFA vs. ACE

This example compares the performance of the CFA algorithm with the *Dancing Cones* (DC) instance of ACE proposed in [6]. We consider the approximation of the following function, also used in [6]:

$$f_2(x) = 0.2 + 0.8(x + 0.7 \sin(2\pi x)), \quad x \in [0, 1] \quad (5)$$

from 21 equidistant input/output training examples belonging to the interval $[0, 1]$. As in [6], we performed two experiments, the first one with the original training set, and the second one with a modified training set to which a random 5% white noise was added. For each of these two experiments, the training data were approximated with models with 4–10 prototypes (rules for DC or RBFs for CFA), and for each combination of training set and number of prototypes,

c	NRMSE (dev)			
	Original training data		Perturbed training data	
	ACE (DC)	CFA	ACE (DC)	CFA
4	1.125 (0.045)	0.380 (0.035)	1.002 (0.023)	0.422 (0.062)
5	1.179 (0.020)	0.327 (0.078)	1.159 (0.026)	0.345 (0.052)
6	0.625 (0.018)	0.309 (0.089)	1.270 (0.120)	0.296 (0.087)
7	0.633 (0.090)	0.181 (0.051)	0.839 (0.357)	0.207 (0.025)
8	0.523 (0.072)	0.167 (0.048)	0.646 (0.219)	0.189 (0.043)
9	0.714 (0.295)	0.161 (0.039)	0.633 (0.321)	0.166 (0.036)
10	0.526 (0.096)	0.098 (0.024)	0.468 (0.071)	0.164 (0.027)

Table 2. Mean and standard deviation of the approximation error just after the initialization procedure to initialize models using from 4 to 10 prototypes to approximate f_2 with the original training data.

the clustering algorithms were run several times. All the models obtained were tested with a set of 100 test points in the interval [0,1]. Table 2 shows the mean and standard deviation of the test NRMSEs obtained.

For the fuzzy models obtained with DC, the product was used as the T -norm, the sum as the T -conorm, and the center of gravity (COG) as the defuzzification mechanism.

Although CFA only performs changes in the prototypes, differing from DC, which also models the consequents of the fuzzy rules, the results obtained show that the proposed algorithm is better fitted for function approximation problems. CFA converges to better models and obtains similar solutions in different executions of the algorithm. This fact is directly derived from the standard deviations of the NRMSE obtained by the two algorithms (see Table 2).

4.2 CFA vs. CFC

This last example compares the performance of the CFA algorithm with the CFC algorithm to obtain the initial RBF centers for an RBFNN. As in [4], a normalized RBFNN

$$F(\mathbf{x}) = \frac{\sum_{j=1}^c \omega_j \phi_j(\mathbf{x})}{\sum_{j=1}^c \phi_j(\mathbf{x})} \quad (6)$$

was used to approximate the two-input data produced by the target function

$$f_3(x_1, x_2) = \frac{(x_1 - 2)(2x_1 + 1)}{1 + x_1^2} \cdot \frac{(x_2 - 2)(2x_2 + 1)}{1 + x_2^2}, \\ x_1, x_2 \in [-5, 5] \quad (7)$$

Alg.	c	NRMSE (dev)	
		Limited data	Complete data
CFA	4	0.798 (0.017)	0.926 (0.008)
	5	0.797 (0.024)	0.898 (0.019)
	6	0.779 (0.007)	0.812 (0.044)
	7	0.752 (0.021)	0.758 (0.053)
	8	0.672 (0.032)	0.713 (0.016)
	9	0.593 (0.032)	0.708 (0.041)
	10	0.575 (0.016)	0.705 (0.035)
	6	0.756 (0.017)	0.838 (0.073)
	9	0.709 (0.054)	0.812 (0.100)
	12	0.614 (0.021)	0.823 (0.056)

Table 3. Mean and standard deviation of the approximation error just after the initialization procedure to approximate f_3 .

The above function was approximated using two different training sets, a limited training set of 26 examples and a complete training set of 441 examples obtained from a grid of 21×21 points equidistributed in the input interval defined for f_3 .

For CFA, both training sets were learned with RBFNNs with 4–10 RBFs, and for CFC, we used the same output context proposed in [4] and described by the following three trapezoidal linguistic labels:

$$\mathcal{A}_1(y) = \mathcal{T}(y; -8, -7, -3, -0.6) \quad (8)$$

$$\mathcal{A}_2(y) = \mathcal{T}(y; -1, -0.4, 0.4, 1) \quad (9)$$

$$\mathcal{A}_3(y) = \mathcal{T}(y; 0.6, 3, 7, 8). \quad (10)$$

Each context was learned with 2, 3 and 4 prototypes, thus generating RBFNNs of 6, 9 and 12 RBFs respectively. As in the above examples, for each combination of network structure and training set, each clustering technique, CFA with $\mu_{\min} = 0.001$, and CFC with $r = 2$, was run several times to obtain several initial configurations.

Table 3 shows average NRMSEs and standard deviations of the approximation error obtained. These results reveal that CFA also supervises the clustering and groups data according to their output similarity, but with one important difference with respect to CFC: the contexts in the output space are obtained directly from the training data, and so there is no need for an expert designer to define them a priori. This automatization means the algorithm obtains better results as it adjusts the output contexts dynamically.

5 Conclusions

When we have a set of input/output data from the observation of a system to be identified, determining the structure of the function approximator becomes an important issue.

The learning methods currently used to train non-linear models usually perform a local search of the closest minimum to the starting configuration, and so it is desirable for the initial configuration to be as close as possible to the global minimum. Incorporating some information about the target function into the clustering algorithm used to obtain the initial model has been proved to obtain better results than if it is constructed in an unsupervised way.

The proposed algorithm improves the initialization performed by other Input-Output clustering techniques. This is directly derived from its objective function, which increases the density of prototypes in those input areas where the target function presents a more variable response, thus minimizing the variance of the output response of the training examples belonging to the same cluster.

It is emphasized that the proposed clustering method does not need any prior assumption about the structure of the data, and that it is able to initialize function approximators from noise data.

Acknowledgment

This work has been partially supported by the Spanish *Ministerio de Ciencia y Tecnología*, under project DPI2001-3219.

References

1. J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum, New York, 1981.
2. J. González, I. Rojas, and H. Pomares. Neural Networks, Clustering Techniques, and Function Approximation Problems. In *ICANN 2002, LNCS 2415*, pages 553–568, Madrid, Spain, 2002.
3. N. B. Karayannidis and G. W. Mi. Growing Radial Basis Neural Networks: Merging Supervised and Unsupervised Learning with Network Growth Techniques. *IEEE Trans. Neural Networks*, 8(6):1492–1506, Nov. 1997.
4. W. Pedrycz. Conditional Fuzzy Clustering in the Design of Radial Basis Function Neural Networks. *IEEE Trans. Neural Networks*, 9(4):601–612, 1998.
5. H. Pomares, I. Rojas, J. Ortega, J. González, and A. Prieto. A Systematic Approach to a Self-Generating Fuzzy Rule-Table for Function Approximation. *IEEE Trans. Syst., Man, Cyber. Part B*, 30(3):431–447, June 2000.
6. T. A. Runkler and J. C. Bezdek. Alternating Cluster Estimation: A New Tool for Clustering and Function Approximation. *IEEE Trans. Fuzzy Systems*, 7(4):377–393, Aug. 1999.
7. E. L. Sutanto, J. D. Masson, and K. Warwick. Mean-Tracking Clustering Algorithm for Radial Basis Function Center Selection. *Int. J. Contr.*, 67(6):961–977, 1997.

Auto-Adaptive Neural Network Tree Structure Based on Complexity Estimator

Mariusz Rybnik, Abdennasser Chebira, Kurosh Madani

Laboratoire d'Etudes et de Recherches en Instrumentation, Signaux et Systèmes
Division Réseaux Neuronaux
Université PARIS XII, I.U.T. DE CRETÉIL-SENART
Rue Pierre POINT - F-77127 LIEUSAINT - FRANCE

Abstract. We present in this paper an auto-adaptive neural network tree structure that is used to solve classification problems. Two main ideas are behind this neural network tree structure. The first one is that it is easier to solve simple problems than difficult ones, so we divide an initial hard classification problem into several simpler/easier ones in a recursive way: "Divide To Simplify". In order to perform the classification problem decomposition, we need to answer two following questions: is the problem difficult? And then, how to simplify it? The second idea is to use some classification complexity estimation methods to evaluate the problem difficulty. We propose an algorithm which estimates the problem complexity, divides it into several easier ones and builds a self organized neural tree structure that solves this problem efficiently.

Key words: Auto-Adaptive, Multi-Neural Networks, Divide To Simplify, Complexity Estimation, Classification.

1- Introduction

Building Neural Network Model to solve real word problems implies the availability of an important **information** source and an **experienced** system designer. Knowledge a priori about the problem under study is of great importance. This a priori knowledge mainly helps the system designer: to choose the neural network model, to have ideas about the information source quality, to set and fix neural network model parameters. The main goal is the construction of a neural model that conforms to the problem in an acceptable time. Several approaches were suggested in the literature that allow to enhance the generalization factor and to reduce the computing time. Among all the approaches, one can mention the Intelligent Hybrid Systems [1], Neural Network Ensemble concept [2], Dynamic Cell Structure architecture [3] and active learning approaches [4]. The approach we propose in this paper is original in that way that it is based on the following paradigm: « *Divide To Conquer, Julius Cesar* ».

In this paper, we are concerned mainly with classification problems. Our idea is that if we can take advantage from a measure that is proportional to the problem

complexity then we can decide to divide a problem into sub-problems and then solve the generated less complex sub-problems. We will present an algorithm that builds a tree-like neural structure on the basis of a complexity indicator estimation, which decides about the splitting. This structure is called T-DTS or Tree Divide To Simplify. The nodes and leafs of the tree are composed of neural networks. The neural networks in the nodes are used to decompose problem into sub-problems, those in the leafs are used to solve the sub-problems. Use of classification complexity estimation methods gives us an objective criterion that allows us to take the decision if to decompose a problem or not.

In the next section of this paper, we present various classification complexity estimation methods one can find in the literature. We will explain why for an early approach we have decided to use the Fisher discriminant ratio as a measure of complexity. In section three the T-DTS approach is explained. In order to validate our approach, we use a set of benchmarks to test the efficiency of neural structure. Those benchmarks represent classification problems of increasing complexity and are described in section four. The promising results we get are presented in section five. Conclusion ends this paper.

2- Estimation of Problem Complexity

The term “*complexity*” is used in literature mainly for describing the computational effort to perform some task. In this paper by “*classification complexity*” we mean difficulty for a classifier to generate high classification rates [5]. Classification complexity estimation is one of the fundamental steps in pattern recognition in order to understand the behaviour of classifiers by relating their performance to the nature of data used [6] and for feature selection. Ho and Basu [7] describe a number of factors that contribute to classification complexity. Misclassifications may arise from **class ambiguity, imperfectly modelled boundary complexity and small sample sizes with high dimensional feature spaces**. Also, some problems have a **complex decision boundary and subclass structures**. In addition, the **high dimensionality of feature space and sparseness of available samples** adds to the classification difficulty. An **incomplete or sparse sample** adds another layer of complexity. Most obvious measure of complexity is the error rate of classifier itself. Evidently it depends on the classifier method thus it is neither universal nor reliable. Classification complexity estimators can be also used to:

- decide about type of used classifier or its properties,
- decide if the decomposition should take place.

2.1 Measurements of classification complexity

There are plenty of classification complexity measurements, described among others by Singh [8], mostly related with Bayes error.

1. Bayes error is theoretically the best estimate. However it is awkward to use in practice because of its high computational complexity and it is often empirically rather than analytically derived [9]. A number of different approaches have been adopted to circumvent the need to compute Bayes error. Statistical probability distances such as Bhattacharaya [11] and Chernoff [12] provide upper and lower boundaries for Bayes error as a special case of two-class problem.
2. Scatter matrices – they provide a computationally cheap method of computing class separability [10].
3. Boundary methods [13] – data of each class is enclosed in a boundary of specified shape. Compactness may be increased by excluding outliers. Count of the overlapping regions is related to the misclassification rate. Fragmenting boundaries in a structured manner provides a series of values related to classification error (total sum is called *overlap sum*).
4. Correlation based approaches [14] – measure the correlation between classes. For separable data correlation between different classes should be low.
5. Other: purity, neighbourhood separability – complexity measures related to pattern recognition based on space partitioning (PRISM) [5].

2.2 Complexity estimation in T-DTS

T-DTS is an auto-adapting system, i.e. its structure adapts to difficulty of data. So it needs to measure the amount of necessary decomposition. Those measurements allow building tree decomposition structure of appropriate density. By this it can achieve desired quasi-constant performance (in term of classification rate) with datasets of various difficulties.

Our approach may use two types of classification complexity estimators:

- Estimator which evaluates the difficulty of classification of the **whole dataset**;
- Estimator which evaluates the difficulty of classification of the **current subset during decomposition of dataset**, to produce decomposition decision;

These estimators may seem identical, but they are very different. In general, estimation of whole dataset complexity is much more difficult and prone to faults (occurs only once, large size of data) than estimation of subset (occurs multiple times, data size is reduced; especially on lower level of decomposition tree, data complexity is reduced). Estimation of current subset's complexity can be done by less advanced techniques as it is only a small part of system. The estimation occurs for each *subset dividing decision* thus *computational complexity* of the algorithm should be small. Considering this estimation during decomposition has been chosen to achieve the auto-adaptation feature of system. We use a complexity measure which is efficient and computationally cheap – Fisher discriminant ratio.

2.3 Fisher Discriminant ratio

The Fisher discriminant ratio is given as: $f_1 = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}$, where $\mu_1, \mu_2, \sigma_1, \sigma_2$

are the means and variances of two classes respectively. The measure is calculated in each dimension separately and afterwards the maximum of the values is taken. To use it for more than two class problem it is necessary to compute each two-element combination of classes and later average it.

Important feature of the measurement is that it is strongly related to the structure of data. Thus it can be efficiently used as decomposition criterion. Computational complexity of the measure is very low, which is an important feature when taking into account multiple measurements.

3- The T-DTS approach and algorithms

T-DTS (Tree Divide To Simplify) is a Multi-Neural Network based data driven structure which is able to adapt to data classification complexity by building decomposition tree of appropriate complication [15]. T-DTS construct a treelike evolutionary neural architecture, where nodes (called Decomposition Nodes) are intelligent decision units and leafs (called Processing Leafs) correspond to intelligent processing units.

The T-DTS consists of two main phases. The first is the learning phase, when T-DTS system decomposes the input data and provides processing structure and tools for decomposed sets of data. The second phase is in fact the operation phase (using the system to process data). There could be also a pre-processing phase at the beginning, which arranges data to be processed.

Learning Phase

In the learning phase, T-DTS algorithm constructs a tree neural structure dynamically and without the intervention of operator. Given learning dataset is decomposed in recursive way until necessary. The decision if the decomposition should take place is based on the Fisher discriminant ratio. In this way, a complex problem is decomposed recursively into many easier sub-problems: we decompose the initial problem space into K sub-spaces. Subsequently for each subspace i , T-DTS algorithm constructs a simple neural model describing the relations between inputs and outputs. Construction of the neural tree is compounded of the following steps (figure 1):

1. **Preprocessing** of learning database (normalization, reduction of number of features, Principal Component Analysis, etc.)
2. **Decomposition decision.** Computation of the Fisher Discriminant Ratio for a given subset. If this value is lower than a defined threshold, decomposition takes place. For each resulting subset the *decomposition decision* is taken *recurrently*. If this value is greater than threshold then a model is built for the subset.

3. **Decomposition.** Competitive Network is used to learn the distribution of the learning database patterns in input space and decompose initial dataset into K subsets.

4. **Model construction –** learning of neural network model, to learn the relations between input and output for each subset $i \in \{1 \dots K\}$.

An example of T-DTS decomposition tree is depicted on figure 2

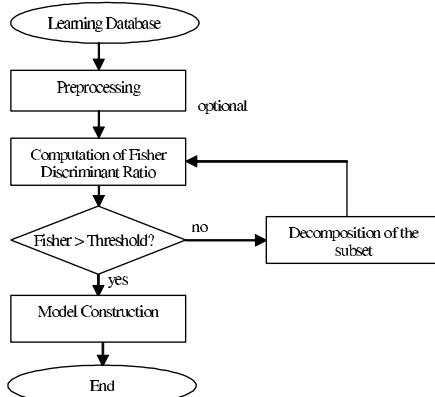


Fig. 1. T-DTS flowchart

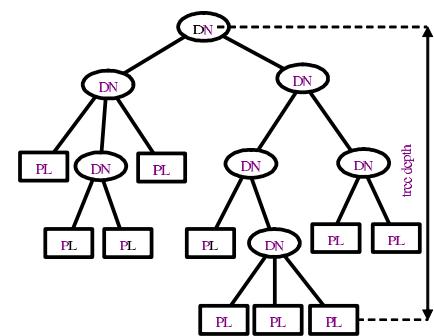


Fig. 2. T-DTS decomposition tree

Decomposition Nodes

The Decomposition Nodes whose estimate the dispersion of data in the problem space and decompose it into many sub-spaces of smaller size (using unsupervised learning) are Competitive Networks or Kohonen Self Organization Maps [16].

Processing Leafs

Various conventional or neural techniques can be used to build a set of models, describing the relations between input and output for each sub-space. The neural network models used at the leaf level (called Processing Leafs) are Learning Vector Quantization, Probabilistic Neural Networks, Multi-Layered Perceptron, etc. [17].

Generalization Phase

During relaxation step, Decomposition Nodes switch the input vector to the Processing Leaf most appropriate to process such cases by following the decomposition tree. At each Decomposition Node the decision is taken to which child pass the vector. When vector reaches a Processing Leaf it is processed by it.

4- Experimental T-DTS Evaluation through an Academic Classification Benchmark

We would like to examine the adapting of the system to data of various difficulties. Thus we have created a sequence of simple datasets of intuitively increasing

complexity. They are presented in the figure 3. In order to validate our assumptions about the increasing difficulty of the benchmark problems we use T-DTS with a static structure. T-DTS generates here approximately the same tree structure and number of processing leafs (measure of *feature space volume* is taken as decomposition criterion to achieve similar degree of decomposition; **not depending on the difficulty**).

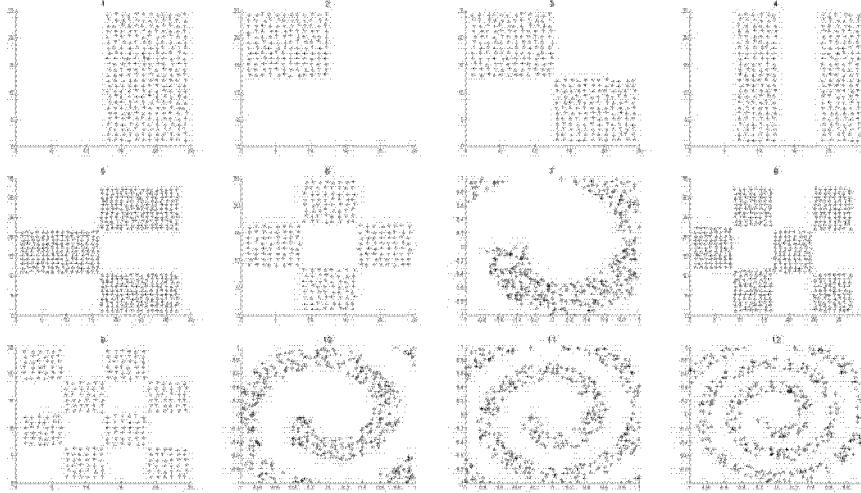


Fig. 3. Sequence of datasets of increasing complexity

On the figure 4 one can see that classification rates drop significantly for more complicated datasets (to right). Processing times are approximately the same for each dataset. This is due to fact that the decomposition tree structures are practically identical and they are too simple for more difficult datasets.

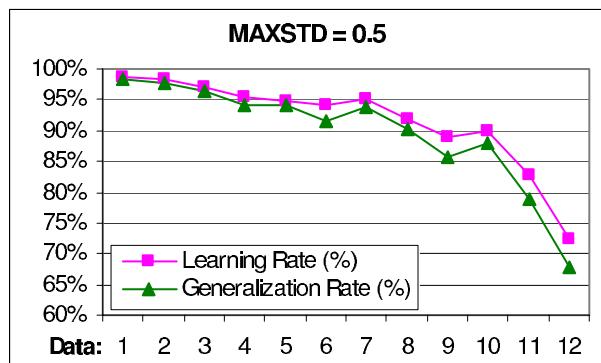


Fig. 4. Classification rates for static structure

Now we will present results when T-DTS adapts to classification complexity. As decomposition threshold we are using Fisher discriminant ratio.

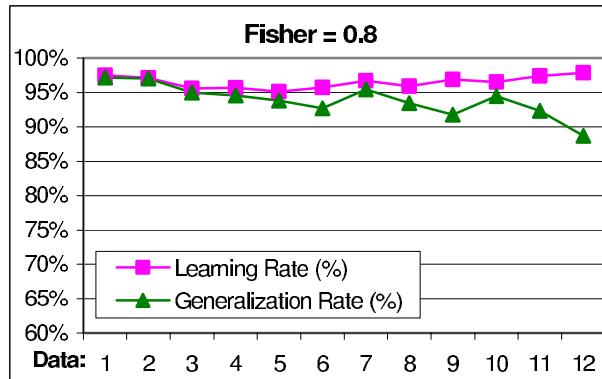


Fig. 5. Classification rates for T-DTS adapting to problem difficulty

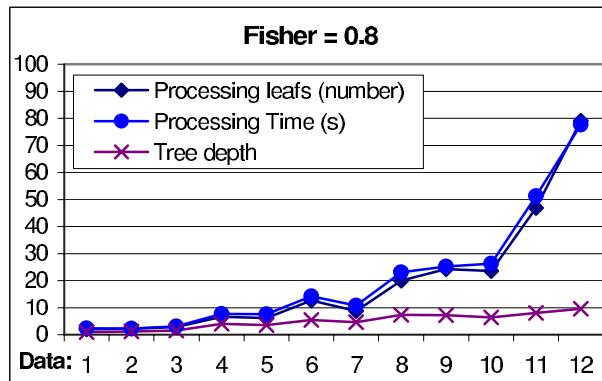


Fig. 6. Statistics of computational effort for T-DTS adapting to problem difficulty

One can notice in figure 5 that classification rates for learning phase are alike, and for generalization rates there is only small dropping tendency. Number of prototypes (related to processing time) significantly increases for more complex datasets as can be seen on figure 6. Thus T-DTS structure adapts to difficulty of dataset, creating corresponding tree structure.

5- Conclusion and future works

We have presented in this article a new method, which simplifies classification problem by dividing it into several easier sub-problems. We decompose the problem in recursive way, obtaining Multi-Neural Network tree. Decomposition is performed by unsupervised competitive Neural Networks. T-DTS uses complexity estimation in order to adapt to problem difficulty. The complexity measurements occur at each decomposition decision, allowing better accuracy than one-time measurement. Decomposed sub-problems are processed by supervised Neural Network classifiers. It

was shown that T-DTS can process data of very different difficulty without neither loss of accuracy nor overcomplicating. Future works are focused mainly on further mathematical T-DTS modeling and enhancement. Expanding T-DTS concept to regression area is also one of our actual major concerns.

References

- [1] S. Goonatilake and S. Khebbal, "Intelligent Hybrid Systems: Issues, Classification and Future Directions", in Intelligent Hybrid Systems, John Wiley & Sons, pp 1-20, ISBN 0 471 94242 1.
- [2] A.Krogh, J. Vedelsby "Neural Network Ensembles, Cross Validation, and Active Learning, in Advances" in Neural Information Processing Systems 7, The MIT Press, Ed by G. Tesauro, pp 231-238, 1995.
- [3] J. Bruske, G. Sommer, "Dynamic Cell Structure", Advances in Neural Information Processing Systems 7, The MIT Press, Ed by G. Tesauro, pp 497-504, 1995.
- [4] K.K. Sang and P. Niyogi, "Active learning for function approximation" in Neural Information Processing Systems 7, The MIT Press, Ed by G. Tesauro, pp 497-504.
- [5] S. Singh, "PRISM - A novel framework for pattern recognition", Pattern Analysis and Applications, vol. 6, 2003 (in press).
- [6] S.Y. Sohn, "Meta analysis of classification algorithms for pattern recognition", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, no. 11, pp. 1137-1144, 1999.
- [7] T.K. Ho and M. Basu, "Measuring the complexity of classification problems", Proceedings of the 15th International Conference on Pattern Recognition, Barcelona, Spain, pp. 43-47, September 3-8, 2000.
- [8] S. Singh, Multi-resolution estimates of classification complexity, IEEE Transactions on Pattern Analysis and Machine Intelligence, (submitted 2003).
- [9] T.K. Ho and H.S. Baird, "Estimating the intrinsic difficulty of a recognition problem", Proceedings of the 12th International Conference on Pattern Recognition, Jerusalem, Israel, 178-183, 1994.
- [10] P.A. Devijver and J. Kittler, "Pattern recognition: a statistical approach", Prentice-Hall, New Jersey, 1982.
- [11] A. Bhattacharya, "On a measure of divergence between two statistical populations defined by their probability distributions", Bulletin of Calcutta Maths Society, vol. 35, pp. 99-110, 1943.
- [12] A. Chernoff, "Estimation of a multivariate density", Annals of the Institute of Statistical Mathematics, vol. 18, pp. 179-189, 1966.
- [13] W.E. Pierson, "Using boundary methods for estimating class separability", PhD Thesis, Department of Electrical Engineering, Ohio State University, 1998.
- [14] A.F.R. Rahman, M. Fairhurst, "Measuring classification complexity of image databases: a novel approach", Proceedings of International Conference on Image Analysis and Processing, pp. 893 -897, 1998.
- [15] M. Rybnik, A. Chebira, K. Madani, "Multi-Neural Networks Approach reducing complexity on both modeling and processing chain levels: application to classification and systems identification", Proceedings of the 17th IAR/ICD Annual Meeting 2002, Grenoble.
- [16] T. Kohonen, "Self-Organization and Associative Memory", Second Edition, Springer-Verlag, New York, 1988.
- [17] M.A. Arbib (ed.), "Handbook of Brain Theory and Neural Networks" 2ed. M.I.T. Press, 2003.

Intelligence and Computation: A View from Physiology

Juan Vicente Sanchez-Andres

Depto. Fisiología, Univ. La Laguna
38320 Tenerife, Spain
j.andres@ull.es

Abstract. It is broadly assumed that the incorporation of intelligence in computer-based devices should increase their capabilities. We lack an operational understanding of the elements that should be implemented in an artificial system to deserve the attribute of intelligent. Meanwhile, the devices are rather characterized by showing an interface able to convince naive users but lacking real intelligence. It could be worth to analyze the physiological properties of natural intelligence to draw conclusions on the operational properties that could be effectively implemented into an artificial system. It is stated that behaviors based just in programs, whether in natural or artificial systems, do not contain signs of intelligence. The emergence of intelligence is dissected and shown that at least in basic conditions this function can be explained in terms of associative learning, leveraged by some properties that result specifically apparent in humans: unlimited exploratory activity, apparent absence of genetically defined aims, transgenerational transference of information, and generalization and symbolic manipulation capabilities. It is proposed that the interest of the field should move towards the definition of rules to instruct an associative learning machine on adaptiveness.

1 Introduction

There is a general agreement concerning that incorporation of intelligence into machines can make them at least more useful en efficient. But the concept of intelligence comes from the natural world, and it does not seem immediate to have a clear definition that can provide and operational specification to be implemented in artificial developments. Frequently, the word intelligence is applied to machines that in fact only show a suggestive appearance based on more or less fortunate interfaces. More frequently, the reason to talk on intelligence rests on marketing strategies addressed to users ignoring the possibilities of the computer-based technology. Then, there is a gap from what is being currently done and the expectances that follow from a better understanding of intelligence. Consequently, it seems reasonable to make a reflection on what can be learnt from nature to be implemented in artificial devices.

2 Some Assumptions

As far as intelligence is assumed to be a brain's superior function, it makes sense to consider the Physiology's point of view. Physiology studies the functions of the organism. Interestingly, there is no a defined opinion at the respect. In fact, the textbooks on Physiology use to lack a chapter on this item. Nevertheless, it is possible to find some specialized books dealing with it (1). The lack of attention in textbooks is associated to the difficulties found in generating a consensus in the understanding of this function. The reason is that intelligence is considered a high level function hard to be approached by the broadly used reductionist procedures that are highly powerful in dissecting micro and nanophenomena but pretty dumb when addressed to answer integrative questions. Approaches from other fields like philosophy or psychology are considered speculative and/or difficult to translate to implementations.

Assuming the limitations in the understanding of the concept of intelligence it is hard to expect a dictionary definition. For the purpose of this paper I will avoid to advance any of the existing definitions to avoid prejudices in the reasoning. The only *a priori* considerations to guide the arguments will be that intelligence is a process that seems to be associated to problem solving, that should imply some predictive capability and increases the probability of making adaptive decisions. Clearly, this is no a definition but a statement speculative and diffuse.

The application of the Physiology's logic implies to limit the question applying a degree of reductionism. The condition to obtain some success from this approach will derive from avoiding missing the very nature of the phenomena or, at least, to keep it at the extent necessary for the application field. In order to do that we will make a first assumption: the interest for intelligence in the technological field is not based in trying to replicate a human being but in building more efficient machines. Maybe a science-fiction writer could not agree with this statement but his opinion would not be relevant here. The assumption implies that endowing machines with something called intelligence will increase their performance. Certainly, nature has generated highly efficient mechanisms through the evolutionary filter. Furthermore, engineers wish to design machines able to replace and automatize human work, then it makes sense to implement capabilities that nature has labeled as relevant. But only in the movies the idea is to build human machines, at least in the next 20-30 years or even in the next centuries. Consequently, it can be better to try to understand intelligence isolated of other processes that could shadow the view. I am talking about emotions, motivation, and consciousness. It can be reasonably argued that it is an impossible aim. The answer is that the argument is correct. But, I am not trying to explain the human intelligence. Rather, the point here is to define the elements that a machine should incorporate to show actually an intelligent behavior. It is a very different question, and we will see at what extent even a so "simplified" version of the question can be convincingly answered. In the best of the cases, provided we will get a satisfactory response, we could walk the way back and incorporate the conclusions into something like a modular model together with the other human functions that now are being separated. This possibility sounds right now, also, other aim in the border with science fiction.

There are two other assumptions to be done: a. *Avoid the anthropocentrism*. Anthropocentrism use to contaminate this kind of analysis, and is completely

unacceptable in Physiology. Furthermore, avoiding anthropocentrism allows taking full advantage on the principle of the evolutionary conservation of functions. This principle states that along evolution there are functions that change gradually being possible to find quantitative differences between species that can be used as models. It can be questioned the validity of this principle for all functions but it cannot be denied that the principle is in the base of a huge amount of successful animal biomedical research. As a rule is accepted that model systems show simplified behaviors, and that this simplicity helps in finding hints that can be generalized to more complex systems. We can use the principle to justify the dissociation above proposed of intelligence from other functions: It is possible to record behaviors that can be considered endowed with some kind of animal intelligence in species (mollusks, insects) where these functions (emotions, consciousness) do not play a known role. b. *Avoid the cultural bias.* The cultural bias is directly associated to the previous point although frequently unnoticed. In fact, sometimes is called “cultural anthropocentrism” or “cultural distortion of the appreciation of human facts”. The more frequent contaminations from this bias result in the use of fortunate metaphors for didactic purposes but highly dangerous for scientific ones (i.e. the computational metaphor for the brain), and the magnification of the actual human realizations. This way of thinking drives to give added value to the job of building today the Empire State than the required for an aborigine some thousand years ago to build a shack. We better accept that there is not a qualitative difference between the man at different times but a transgenerational accumulation of skills. This last assumption is solidly based, as it is known the absence neither of significant changes nor in men genes neither in their phenotype in the last thousand years. Consistently, the differences in the realizations should not rely on intrinsic factors intelligence related.

3 Getting into the Problem

The first real problem comes just at the moment we try to delimit what is known as intelligence. If we observe a given subject along a representative day of his life we can easily conclude that he does some intelligent actions and others that hardly can receive such consideration. These last ones can be divided in two groups: actions that are wrong or inefficient (at least from the observer’s point of view) and others that are mechanics. This requires further clarification that will be done later. What is clear now is that the same subject can develop intelligent and non-intelligent actions. This simple and obvious observation drives us to three significant points: *First.* We are talking about actions that not necessarily have to be visible. Commonly the word “action” is associated to motor activity but not necessarily. For example, a given neuronal activation can induce a movement but if the neural pathway is sectioned or pharmacologically blocked the same neural activation will be unable to reach the muscle and the movement will not happen. Then, the concept of action is more satisfactorily associated to the activation of given neuronal ensembles than with the capability of the signal to reach a target muscle. This idea is under discussion but here help us to extend the concept of action to the general brain processing whether associated or not to movement. *Second.* The intelligence seems to be more than a function an attribute applied to some actions (associated or not to movement) with an

output valued as positive. *Third.* There is a clear generalization from the fact that a subject performs some intelligent actions to accept that he is intelligent. This last point is clearly out of the scope of this paper.

4 Types of Reflex Responses: Relevant Clues

We noted above that some actions could appear intelligently performed but being in fact mechanics. Most of them are genetically programmed or reflex responses. It can be said that most of the animal behavior falls in these categories. Nature has endowed the animals with broad strictly stereotyped behavioral repertoires mechanically executed. It can be considered smart to close the blink if some object approaches de eye but there is not intelligence at all in this action. Neither there is in the complex mating behavior of a lot of species.

The consideration of reflex and programmed actions as empty of any type of intelligence has relevant implications for the field of artificial intelligence. By definition cannot be considered intelligent a given machine if its work is based just in programmed rules does not matter how sophisticated the rules are or how impressive the computational power. In the best of the cases it could be possible to state that the machine has an “intelligent-like” behavior as evaluated by a potential buyer. It is worth to illustrate this with an example. Let’s imagine two squirrels playing a mating game (genetically programmed). Then, a predator appears and the squirrels stop the game and try to escape (genetically programmed reaction). A naïve observer could estimate as intelligent the capacity to replace one behavior for the other. But, there is not a single reason to accept any intelligence in the change. Rather, the replacement happens because the input to raise the second behavior is stronger than those that would keep the first one.

5 Associative Learning and Intelligence

While there is no discussion on the fact that the reflex responses cannot underlie intelligent behavior, they hold a property that can be highly relevant which is its capability to be modified by experience to constitute conditioned responses through associative learning. It is very well known that along conditioning a subject learns to respond to a stimulus unable to raise a response already. It is a new response that emerges after the pairing of the conditioned stimulus to other (unconditioned or reflex) that was naturally (genetically programmed) capable to elicit the response. The conditioned responses play a relevant role in the generation of the individual behavioral repertoires because are based in each individual experience. More, they can be chained up and combined one after another in such a way that in a given moment in the life of the subject can be highly difficult to determine the original reflex that was the base for a particular current association. Under this scope it is tempting to hypothesize a relevant contribution of associative learning to the generation of intelligent behaviors. There is no doubt on that this hypothesis can be criticized with the argument that associative learning is too simple to underlie

complex behaviors. This criticism would be based more in the observation of the experimental paradigms than in the knowledge on the way that associative learning works in nature. It is worth to note some properties of associative learning to emphasize its potentiality. *a.* Associative learning requires a strict temporal contingency between the associated stimuli such that the functionality is equivalent to that of a coincidence detector. *b.* The establishment of associations requires a positive adaptive value in a way such that non-adaptive interactions are not stored as memories. *c.* One of the fundamental properties of associative learning is its predictive role. The subject acquires the capability to anticipate that an event is going to take place and that it will happen in a given temporal window. This is exactly what happened to the Pavlov's dog: the bell's ring provided to him with the clues on the probability of receiving food, in other words provided to him predictive capability on an event completely independent on the genetic programs and acquired through the individual experience. *d.* As a corollary of the previous points, it can be said that associative learning allows the establishment of cause-effect relationships when they have adaptive meaning.

It can be stated that associative learning can explain most of the non-programmed apparently intelligent behaviors in animals. This is the case even in species where emotions and motivation can play a role (it is under discussion where some trace of consciousness appears). Certainly, the called intelligent behaviors in animals use to consist on experience modified reflex responses. Let's use the example of the dog that seems intelligent because carries the newspaper, some keys or a prey.

6 Is there anything More in Humans?

In humans it seems that the question is more complicated. It appears that the repertoire of intelligent actions is close to unlimited. The question arises on what can be the reasons underlying this richness.

To try to answer this question is necessary to address one of the functions I was trying to keep apart: motivation. There is a qualitative difference between humans and other animals at this respect. In other animals the motivation to initiate actions comes from the necessity to satisfy primary necessities. They start actions to look for food or partner. At higher levels of evolution, animals also play games that frequently serve to refine and keep trained the skills useful to satisfy those necessities. It can be said that the aims are programmed, as they are the behaviors. The window to express individual differences is rather narrow. Nevertheless, every environment has particularities to be discovered. The behavior oriented to learn on the environment is called exploratory activity. Frequently, the animals develop a very intense exploratory activity associated to the rise of a necessity. Once the necessity is satisfied the activity declines until the necessity rises again. These cycles of behavior can be understood as negative feedback loops. We do not know the complete neural circuitry involved but it is clear that the animals have sensors that detect, for example, the levels of glucose in the blood that are correlated to the nutritional state. Much less is known on the way this information is processed in the brain to raise the exploratory behavior to search for food. Whatever the circuitry, the fact is that limits the exploratory activity and, as a consequence, the amount of experience to which an animal can be exposed. This

negative feedback loop does not appear to exist in humans. In our specie the exploratory activity is ordinarily called curiosity that drives humans to explore continuously even after having the primary necessities satisfied. More, our exploratory activity frequently is not addressed to objectives that apparently could help in satisfying them. It does not seem that in humans or the behavior neither the aims very programmed. It is tempting to propose that the negative feedback loop that regulates the exploratory activity was lost in some moment of the hominization process. It seems interesting to note that if this was the case could have lethal consequences on this new specie because is a highly unadaptive change: it does not make any adaptive sense to use energy in exploratory activity not oriented to satisfy basic necessities. Under this scope, the lost of the loop was an unexpectedly lucky accident. Nobody intelligent would bet on a specie using no economically its limited resources.

The fact is that the unrestricted exploratory activity provides the humans with the option to be exposed to an unlimited repertoire of experiences, and in the same way to unlimited possibilities of associations and new emergent behaviors.

The lack of the negative feedback loop controlling curiosity provides a plausible explanation for associative learning to play a role in human intelligence that is just marginal in other animals. Furthermore, humans show other convergent strategies.

One of these convergent strategies is the relatively long period of dependent life. If we consider that the dependent life ends when the reproductive capability starts and that the average prolongation of live over the end of the fertile life is a recent acquisition (last century), we can estimate that the proportion of dependent life on the total life is around a 25%. This proportion is extremely high if compared to other species and allows massive transgenerational transference of information. It is worth to remark the meaning of this scenario as far as the implications are by far more than simple leverage. The association process implies trial-and-error sequences. The transgenerational transference of information allows not only teaching previous associations but also minimizing the risk of involvement in dangerous or potentially unfruitful experiences.

The other convergent strategy is the capability of humans to realize generalizations that allow the establishment of associations by analogies. It is highly improbable that a subject will find two identical causal scenarios in its lifetime. The chances to predict an effect when the causes are not identical depend on the accuracy of the causes diagnostics that depend on the previous experiences. The establishment of analogies provides the way of saving trial-and-error assays. This capability appears early in the evolution. For example, a predator can generalize what is a prey. But in humans the generalization mechanisms are exceptionally developed at the extent to generate representations of reality that can be handled symbolically after the emergence of language.

7 Is there a Biological Substrate for the Intelligent Behavior?

In the previous part of the paper has been presented a view of intelligence based in associative learning leveraged by several other convergent strategies. Under this scope, intelligence seems to be a process emerging out of the interaction of several

components that can be distributed in the brain. Hardly, then, can be expected to find a delimited site for intelligence. Better, the question can be formulated as follows: Is it possible to find in the brain conditions compatible with the proposed strategies?

To try to make some progress in this direction can be useful to interpret the proposed mechanisms. It seems that humans show a level of brain plasticity by far more developed than other animals. In the rest of animal prevails the programmed behavior, but in humans the learnt behavior seems unlimited being the learning provided by experience. A neural substrate for this learning capability should develop after birth. In the other extreme, as much mature the neural tissue at birth, less options to take advantage of the experience and more dependence on genetically programmed behaviors. It has been observed that more evolved animals have more cell layers in the brain cortex. Furthermore, it has been described that humans have one layer more (a total of seven) than other mammals (six) (2, 3) and that this layer could be a candidate to explain intelligence based in associative learning. The reasons supporting this proposal are: *a.* Humans and the rest of mammals have six common layers that could be responsible for shared behaviors. *b.* The human-specific seventh layer is undeveloped at birth growing up postnatally being, then, configurable in interaction with the environment.

Implications for the implementation of intelligence in artificial systems

The arguments exposed above support the point that intelligent behaviors in humans are based on the next properties:

- Unlimited exploratory activity
- Apparent absence of genetically defined aims (at least the capability to escape out the genetic determinism)
- Transgenerational transference of information
- Generalization and symbolic manipulation capabilities

It can be accepted that a reasonable part of the called intelligent behaviors could be explained in terms of these properties. In the context of this paper two questions arise: 1. Would it be possible to implement these properties in an artificial system? It seems possible with the current technology under the restrictions imposed by the machines (mobility, inputs nature, etc.). 2. Would it result of any interest a machine endowed with these properties? Probably more for academic than for commercial purposes. It could be highly multitasking but less efficient at the short term than other machines designed for solving specific tasks.

8 Future Trend

Finally, it is worth to consider what can be the future trends in the field provided that the given explanation of intelligence is at some extent close to the reality. A computerized device endowed with the properties that can allow the intelligence emergence should do sequences of associative learning. These sequences would imply trial-and-error series that require the existence of rules specifying what has and what has not adaptive meaning. The definition of that rules is not trivial as far as the concept of adaptation seems at least not obvious for an artificial system. The

engineers will not have thermodynamic or evolutionary clues. It can be predicted that in the middle term the question on intelligence, when concerning implementation in artificial systems, will move from the actual considerations towards the understanding of adaptiveness in computer based devices.

References

- [1] Calvin, W.H. How brains think. Evolving intelligence, then and now. BasicBooks. HarperCollins. New York. 1996
- [2] Marín-Padilla, M. Cajal-Retzius cells and the development of the neocortex. *TINS* 21, 64, 1998
- [3] Marín-Padilla, M. Desarrollo de la corteza cerebral humana. Teoría citoarquitectónica. *Rev. Neurol.* 29, 208, 1999

Image Understanding analysis at the Knowledge Level as a Design Task

M. Rincón¹, M. Bachiller¹, J. Mira¹, R. Martínez¹

¹ Dpto. de Inteligencia Artificial, ETSI Informática, UNED, C/. Juan del Rosal 16, 28040 Madrid, Spain
{mrincon,marga,jmira,rmtomas}@dia.uned.es

Abstract. In this paper we analyse the problem of image understanding at the knowledge level. We treat the problem as a design task and define a generic problem solving method (PSM) which allows us to tackle the task in a hierarchical and recursive way with subsumption. The main advantage of this generic PSM is the possibility to instantiate specific PSMs through parameter space configuration, which makes it possible to reuse this structure both in the task decomposition at different hierarchical levels and in different applications. This generic PSM was implemented following the well established foundations of Knowledge Engineering which prescribe the maintenance of the conceptual structure from the modeling stage at the knowledge level down to the particular implementation. Finally, we apply the proposed framework to the problem of optic nerve head identification in eye fundus images and particular results are presented.

Introduction

The objective of the task of Image Understanding (IU) is to obtain a description of an image with a sufficient level of abstraction for any task that requires it (diagnostic, monitoring, etc.). IU task is a complex one, which requires mutual interaction of processing steps (pre-processing, segmentation and recognition). The problem with visual systems is that there is a loss of information in each of the stages of the process, just like in models that represent domain knowledge (DK), which leads to the appearance of uncertainty throughout the process.

To counteract the uncertainty, we should restrict the generality of our visual system, adapting it to the particular type of information in the domain of application by the injection of DK at the different stages of the process. On this theme [1] shows a good summary of the types of uncertainty which appear in the visual process and the control strategies to counteract them. The knowledge may be used in an implicit or an explicit form. The resultant system is a knowledge based system, and the more the complexity of any system increases, the more evident is the necessity to use Knowledge Engineering as the development method.

We can classify the visual systems into two large groups: *Generic systems*, with sufficient flexibility and representation capacity to be capable of integrating any kind of knowledge into the system [2,3], and *More specific systems*, which implement a method of image interpretation based on some assumptions which simplifies the problem [4,5,6,7,8]. The main criticism which can be made about generic systems is

that, needing to be flexible, a good part of the problem is transferred to the designer, who should program the DK utilised just as much as the method which is utilising it. Furthermore, these describe architectures which enable the knowledge to be computed, but do not help in the definition phase of the visual system, only making that knowledge computable which is going to be used in an explicit form in accordance with the inference strategy. Concerning the second group, they are capable of proposing specific solutions, but do not differentiate between the different types of knowledge in the proposed solution, for which reason they cannot be used as soon as the problem does not adjust itself to the initial assumptions. In this work an attempt is made to solve the difficulties found in the bibliography, modelling at the knowledge level the use of DK in the IU task. Our proposal is to use for this the methodology of Knowledge Engineering.

The task of IU allows two basic formulations. On the one hand, it can be solved like an analysis task, whenever it is possible to associate definite pixel configurations with the objects of interest. On the other hand, the IU task may be considered as a design task, in which the solution should be constructed throughout the process. This is the most common case owing to the uncertainty associated with the visual process. According to [9,10,11], two types of design tasks can be defined, depending on prior knowledge of the components which form part of the solution model: creative design, in which not all the elements which may be used to construct a solution to the problem are known, or routine design, in which all of the elements are known in advance.

In some recent work [10], Motta presents a generic model of the parametric design problem solving method, which subdivides the parametric design problem into a number of subtasks. The claim is that this collection of subtasks defines the knowledge space intensive decision making activities which are carried out when solving parametric design problems. In this paper we are going to describe a generic PSM to model the IU task, following the philosophy developed by Motta. In the bibliography, in no case is a common framework described which allows the distinct methods to be compared from a functional point of view, where the knowledge used to solve the task would be made explicit. This is key to the treatment of complex problems, as it does not directly concern the acquisition of knowledge as a function of a given PSM (thought to be most adequate), but rather to declare the available knowledge, to organise it into types of knowledge related to the task and the method, and as a function of this to select the PSM, group of PSMs or compound PSM most suitable.

In the following of the article, we first present in section 2 a generic model for IU task. In order to validate this generic model, in section 3, we describe its application to identify the optic nerve head and we show some experimental results obtained.

2. A Generic Model for the Image Understanding Task

The IU task may be considered as a design task given that the solution may be constructed throughout the process owing to the associated uncertainty. In general it may be understood as a routine design task since all of the elements which can be utilized to construct the solution are known in advance. In the following the ontology

of the IU task as a design task and the method (generic PSM) used to solve it are described.

2.1 The Ontology of the Design Task

"The solution to the design problem consists of a complete specification of a set of components and their relations that together describe an artifact that delivers the functions and satisfies the constraints." [9]. Viewing the IU task as a design task the following group of entities appears in the IU ontology: objective, design components and their interrelations, requirements, restrictions and preferences. The objective is the artifact which meets the requirements and complies with the restrictions. The design requirements represent the necessity to adjust the interpretation to the characteristics of the image, i.e. the objects of the model should correspond to regions of the image. The design restrictions represent the need for the solution to respect the structure of the model. Finally, the preferences indicate which is the best design with respect to a given criteria. One can distinguish between local preferences, which are associated with partial groupings of components of the domain model, and global preferences, which are related to the global interpretation of the image.

2.2 Generic PSM for Routine Design Problem Solving

The generic PSM proposed in this work is orientated towards the task and is independent of the concrete method, serving as a reference for the description of the distinct types of knowledge utilized in the solution of the task. Furthermore, they serve as a reference for the integration of distinct PSMs in the solution of a complex problem. The way to solve a routine design problem has been characterised as a searching process in the design space [10,11,12,13].

2.2.1 Ontology of the Method

In the dictionary, the terms of the ontology of the method obtained for this work fall into two different types: entities which depend on the domain model and entities characteristic of the method. In tables I and II the groups of entities in both cases are summarised.

2.2.2 Structure of the Method

Given the design space and characterizing the PSM as a search process, three subtasks are defined which should be carried out cyclically. *Select State*, *Design from State* and *Evaluate State*. As additional subtasks in this design cycle we have *Initialise design*, which allows configuration of the design with the objectives of the task and the input data and *Select Solution* which assesses in each design cycle whether the desired solution has been reached and selects the solution states. The diagram of subtasks in this generic PSM is shown in figure 1.

Select State allows the design to continue from the set of best states of those known at each moment. For each one of these states, in *Design from State* the operator is selected which is going to produce the transition to the following state, the selected operator is applied and a new design state is reached. If it is not possible to select a

Design	Assemble of components which describe the current solution to the task
Case Model	Entity which contains all the information associated with a design
Model Components	Objects of the domain model
Input Components	Objects which allow the case to be described as a group of components
Objective components	Group of components which should be requested in the case model
Global preference	Function which evaluates the solution globally
Restriction	Characteristic of a component of the model or relation between components which must be maintained in the interpretation of the image
Operator	Procedure to carry out a transformation of the case model
Simple Operator	Design operator with a simple procedure associated to <i>obtain value</i>
Complex Operator	Design operator with a procedure associated to <i>obtain value</i> which constitutes a new task with which all of a new generic PSM is associated

Table I. Entities which depend on the domain model.

Context	Concept which indicates the objective to be followed in a transition between states: EXTENSION, REPAIR OR IMPROVEMENT
Evaluation	Compound entity which contains the partial and abstract evaluations of the design

Table II. Entities characteristic to the method.

single operator, each one of the selected operators will be applied and afterwards the best of the obtained solutions is chosen. When a new state is reached it will be evaluated to determine whether it is a solution or in order to select the states for the next design cycle. Otherwise, if the operator does not produce a new state it is necessary to go back and select another operator. The selection of the operator depends on the context and on the focus of attention that the operator is going to be centred upon, and may be random (without DK), based in generic heuristics or in DK. Therefore, the subtask *Design from state*, breaks down according to this idea into *Select context*, *Select focus* and *Design in context and focus*. In turn the last of these subtasks divides into others more simple following the inference diagram shown in figure 2.

The inference *Select Applicable Operators*, selects from the list of operators associated with the design, those applicable to the context and focus and which meet the conditions of applicability in the current state of the design. Subsequently a selection is made of the specific operator as a function of the criteria utilised. If this is not possible, in the inference *select operator-value*, the operator-value pair will be selected which has produced the best result. The operator selected is used in two subtasks: *Obtain value* and *Generate states*. *Obtain value* determines the value which is going to be used for the modification of one or several components of the case model by *Generate States*.

2.2.3 Correspondence between Entities of the Method and Roles

In figures 1 and 2 the static and dynamic roles which the DK should play in the method structure are shown. In Table III we show the relationship between the dynamic roles of the method and the entities which play them.

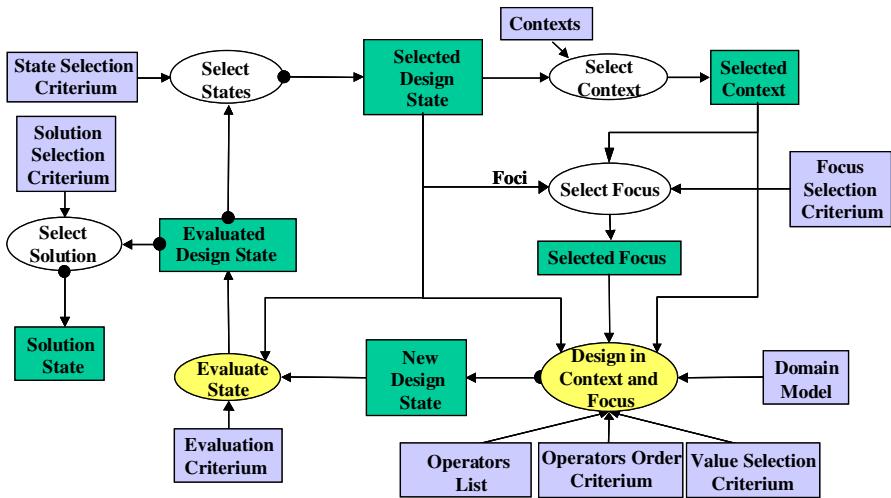


Fig. 1. Diagram of the subtasks of the generic PSM for the design task.

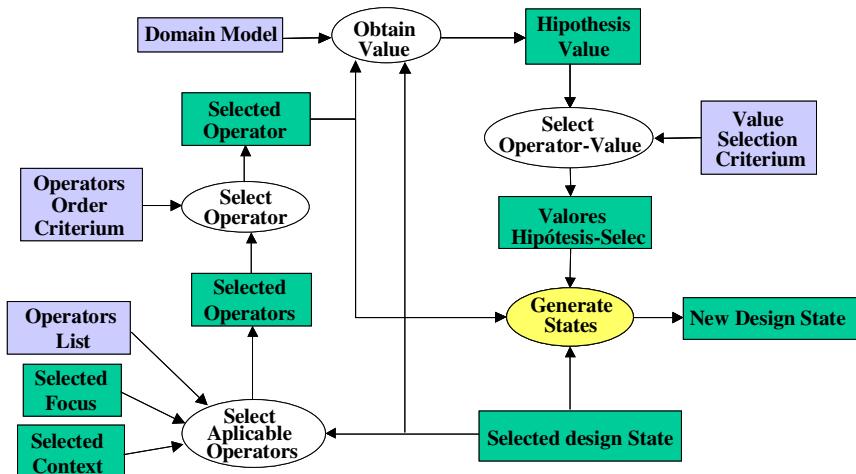


Fig. 2. Diagram of the subtasks of *Design in context and focus*.

2.3 Simplifications of the Task through the Domain Structure

Up to this point, the IU task has been treated in a generic form and the generic PSM which allows it to be resolved as a design task has been described without taking into account possible simplifications derived from the domain structure. Given the complexity of the problem, and the current state of the technology of image processing it is practically impossible to directly generate transitions between states of the space solution, therefore we depend on intermediate spaces for which we can define operators.

Level	Task	Role	Type	Entity
1	Select State	Evaluated Design Estate	I	Case Model
		Selected Design State	O	Case Model
	Select Context	Selected Design State	I	Case Model
		Selected Context	O	Context
	Select Focus	Selected Design State	I	Case Model
		Selected Context	I	Context
			O	Model Component
		Selected Focus	O	Input Component
			O	Restriction
	Design in Context and Focus	Selected Context	I	Context
		Selected Design State	I	Case Model
		Selected Focus	I	Model Component
				Model Component
				Restriction
	Evaluate State	New Design State	O	Case Model
		New Design State	I	Case Model
	Select solution	Evaluated Design State	O	Case Model
		Evaluated Design State	I	Case Model
	Initialise design	Solution State	O	Design
		Case Description	I	Input Component
		Evaluated Design State	O	Case Model

Table III. Correspondence between the roles of PSM and the entities of the Method.

A design task can be simplified when the functional specifications can be broken down into a group of subfunctions. This knowledge may come in the form of part-subpart relations if a direct association exists between functions and components [9]. In the case of IU the desired functionality consists in associating parts of the image with the components of the application domain. The fact that the domain objects are composed by others enables the design task to be broken down into more simple subtasks. In this manner the complexity of the problem is reduced, since the number of parameters which has to be adjusted in each one of the sub-problems is reduced. In other words, the *composition* relation divides the parameter space into subgroups associated with the components, in a way that breaks down the initial design problem into a group of more simple sub-problems. The composition relation establishes a hierarchy between the objects of the domain, which is maintained in the partially associated designs giving way to a *hierarchical recursive design* structure.

Finally, the visual attributes allow the association between the domain model and the image to be established. This relation is necessary for the interpretation, given that our objective is to obtain a description of the image at the level of detail required by the task at the superior level, which respects the structure of the model and which is based on visual characteristics obtained from the image. Therefore, the breakdown of the task into subtasks will end when this union between model and image is established. So, if we take into account the hierarchical structure of the domain and that all of the visual objects of the domain display visual attributes, the association between image and model may be established at any level of the hierarchy of objects. This property means that it is not necessary to keep on breaking down the design to the simplest objects if it is not essential for the interpretation of the image. This brings

us to a design structure which we call *hierarchical recursive design with subsumption*. Each one of the sub-problems is resolved in an independent way utilising for it a PSM, which can be defined on the basis of the structure of the generic PSM described. Nevertheless, there may be dependencies between the distinct sub - problems which conditions the order in which they are resolved.

3. Application to the Identification of the Optic Nerve Head

In this section we are going to apply the generic PSM described to the identification of the Optic Nerve Head (ONH). The objective of the task is to identify the ONH in a 2D RGB image of the eye fundus for its subsequent use in di agnostic tasks.

Figure 3 shows task decomposition according to the *propose-revise-improve* PSM, which has been derived from the generic PSM described. In this case, the entities of the application domain have been defined, which are a particularisation of the entities of the method. The operator *propose-OHN-1* is a complex operator with a complex procedure related to the subtask *obtain value* which we again treat as a design task.

The method was validated over 100 eye fundus images with a 87% identification successful. Figure 4 shows two examples of results obtained.

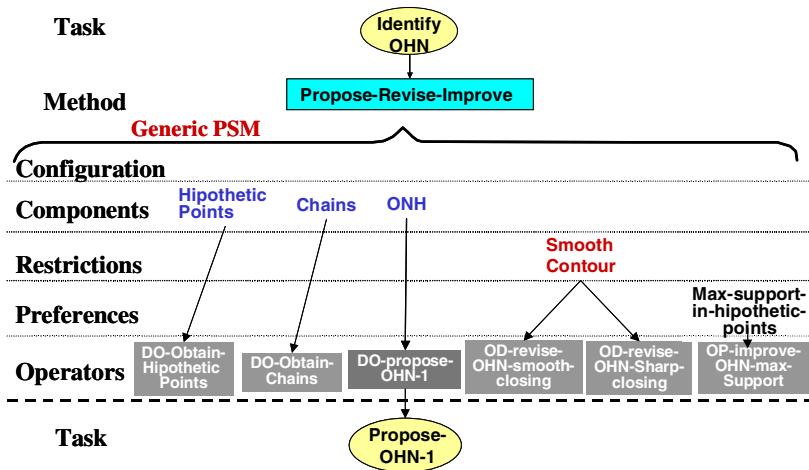


Fig. 3. Results obtained in the identification of the ONH.

4. Conclusions

This work has demonstrated the utility of the methodology of knowledge based systems for formal analysis, at the level of knowledge and in the domain of the observer, for the task of image understanding. This analysis has been done on three levels: task, method and DK, distinguishing the generic components in any IU task, which enables reutilization. In order to confirm t he usefulness of the proposed framework an application has been developed for the identification of the optic nerve head in eye fundus images.

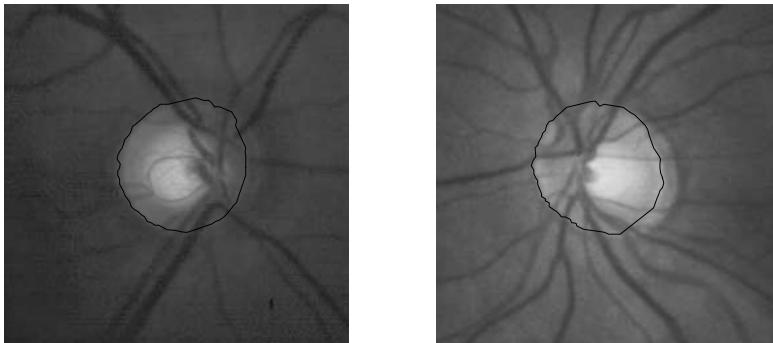


Fig. 4. Results obtained in the identification of the ONH.

Acknowledgments

We would like to thank to the Ophthalmology Service of Miguel Servet Hospital and to the Oftalmology Department of the San Carlos Clinic Hospital for letting us using their images in this study. We would like to thank the CICYT for its support, through the DIAGEN project (TIC-97-0604), in the context of which this case study has been carried out.

References

- [1]Tsotsos, J. *Image understanding*. The Encyclopedia of Artificial Intelligence, pp. 389 -407. S. Shapiro, 1st edition.1987
- [2]Draper, B.A.Brollo, J., Hansen, A.R. and Riseman, E.. *The Schema System*. Int. Jour. of Computer Vision. vol. 2. pp209-2501989
- [3]Matsuyama, T. and Hwang,V. *A Knowledge-Based Aerial Image Understanding System*. Plenum, New York.1990
- [4]Kummert, F., Niemann, H., Prechtel, R. and Sagerer, G.*Control and explanation in a signal understanding environment*. Signal Processing, vol. 3, pp. 111-145. 1993
- [5]Grimnes, M. and Aamodt, A. *A two layer case based reasoning architecture for medical image understanding*. EWCBR96.1996
- [6]Brooks, R. *Model-based three-dimensional interpretations of two-dimensional images*. IEEE Trans. on PAMI, vol. 5(2), pp. 140-150. 1993
- [7]Chan, S.W.K., Leung, K.S. and Wong, W.S.F. *Object-Oriented Knowledge-Based System for Image Diagnosis*. Applied AI 10(5). pp 407-438. 1996
- [8]Gong, L. and Kulikowski, C. *Composition of image analysis processes through object - centered hierarchical planning*. IEEE Trans. on PAMI, vol. 17, pp. 997-1009. 1995
- [9]Chandrasekaran,B.*Design problem solving:A task analysis*. AI Magazine,Winter,59-71.1990
- [10]Motta, E. *Reusable Components for Knowledge Modelling*. IOS Press. 1999
- [11]Wielinga, B., Akkermans, J. and Schreiber, A. *A formal analysis of parametric design problem solving*. Proceedings of the 9th Banff Knowledge Acquisition Workshop.1995
- [12]Wielinga,B. and Schreiber,A. *Configuration design problem solving*. IEEE Expert, vol. 12(2) 1997
- [13]Stefik, M. *Introduction to knowledge systems*. Morgan Kaufman. 1995

Morphological Clustering of the SOM for Multi-dimensional Image Segmentation

Aureli Soria-Frisch and Mario Köppen

Fraunhofer IPK, Dept. Security and Inspection Tech., Pascalstr 8-9, 10587 Berlin,
Germany aureli.soria_frisch@ipk.fhg.de

Abstract.

1 Introduction

New imaging sensors and technologies challenge the application of traditional computer vision methodologies due to an increment of the image dimensionality. Images processed in different application fields are not any more restricted to the grayvalue image domain, but take into consideration images of larger dimensionality. Color, multisensorial and satellite images are some examples being used in different application fields. This increment in the dimensionality of the problems related to the utilization of these larger feature spaces has been already characterized as a problem denoted by the “curse of dimensionality” in pattern recognition. Taking into consideration feature spaces of large dimensions introduce some geometric anomalies , which hinder the interpretability of the results and thus the attainment of the expected ones [3]. The paper presents a hybrid framework, which makes use of a Self-Organizing Map (SOM) [2] and the fuzzy integral [1] in order to cope with the segmentation of images in high-dimensional feature spaces.

SOM is a neural paradigm extensively used as a tool for data visualization and knowledge engineering, where its performance demonstrates at its best in high-dimensional feature spaces [8]. The SOM paradigm seeks the projection of the incoming data set into a discrete grid of nodes. In case the grid is two-dimensional two auxiliary matrices can be defined on the output map, which help in the visualization of the clusters generated on it [8]. These matrices are denoted as *U*- and *hit-matrices* [8].The main contribution of the here presented paper is a methodological approach for prototype selection in the SOM output map. This selection is achieved by clustering the *U-matrix* through so-called morphological reconstruction [4].

Some approaches have been presented for clustering the SOM output map [9]. Vesanto proposes in this work a general framework for clustering that is characterized by the existence of two stages. This is attained in order to improve the performance of clustering in terms of total computational cost. Since the SOM achieves a reduced data representation through the prototypes in the output grid, the second clustering becomes computationally more tractable by taking into consideration the prototype set instead of the original data set [9].

The similarity with this general framework of the here presented approach is the consideration of the two clustering stages. A first one implemented through a SOM and the second one by a fuzzy integral. Nevertheless the here presented approach does not consider the clustering of the prototype set, but of the resulting *U-matrix*. Therefore the first clustering is not undertaken in the original high-dimensional data space, but in the *U-matrix* domain. Thus the clustering of a high-dimensional feature space is reduced to the clustering of a grayvalue image, which allows to attach the problem in the same way without having to take into consideration the dimensionality of the feature space. This point can be well used in the segmentation of multi-dimensional images as those formerly mentioned. The segmentation of multi-dimensional images is achieved through a classification system based on the fuzzy integral [6], where the selected prototypes of the SOM are used in order to parameterize this fusion operator.

The fuzzy integral is a fuzzy fusion methodology specially used in pattern recognition and multi-criteria decision making applications [1]. The fuzzy integral is a weighted operator. So-called fuzzy measures [1] are used in the fuzzy integral for weighting the data to be fused. The fuzzy integral has been successfully used in problems of image segmentation [5]. However the automated determination of the fuzzy measure coefficients is still an open question for the extended utilization of the fuzzy integral in real applications.

The paper is organized as following. First the morphological clustering of the *U-matrix* is presented in Section 2. Thence Section 3 describes the framework for image segmentation. Finally, some results (Sect. 4) and conclusions (Sect. 5) are given.

2 Clustering the SOM output map through morphological reconstruction

Although the SOM paradigm is applied in the image segmentation framework for the automated assessment of the fuzzy measure coefficients, the following section describes the algorithm for the segmentation of the *U-matrix* on its own. As already mentioned (see Introduction) the here presented procedure can be used in order to encompass the clustering of high-dimensional feature spaces by taking into consideration the data projection achieved in the SOM. This data projection is used for the computation of the *U-* and *hit-matrices* [8] after training the neural map.

The *U-matrix* represents the distances between the prototypes of the output map in a grayvalue image space. Such a representation can help in getting a first idea of the cluster distribution [9]. Clusters are characterized in this image through an homogeneous area of low grayvalues separated by edge-wise elongated areas of high grayvalues (see Fig. 1c). The *hit-matrix* is a two-dimensional histogram. Once the output map has been trained, it can be applied to the data set once again in order to obtain the winning neurons of each data point. This information is accumulated in the *hit-matrix* and represented in image form, where the grayvalue will be proportional to the number of times a neuron has won (see

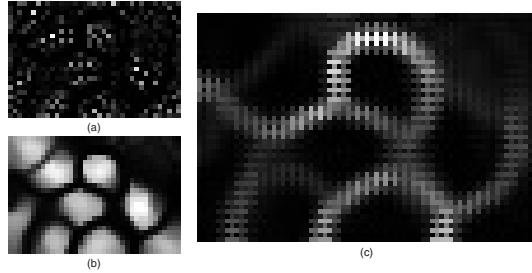


Fig. 1. (a) Exemplary *hit-matrix* of a SOM. Black points represent no hits in the prototype. (b) Exemplary *fuzzy hit-matrix* of a SOM. (c) Exemplary *U-matrix* of a SOM. Grayvalues represent distances between prototypes (white for larger ones). Distance in prototype points are computed as the maximal distance to its neighbors.

Fig. 1a) [8]. Such prototypes with a larger number of hits are the most-frequent winning values, and thus can be considered as the most representative ones. There is also a fuzzy variant (FH) of the hit histogram [8], which is computed through the expression:

$$FH = \sum_{i=1}^n \frac{1}{1 + (d_i/\bar{Q})^2} \quad (1)$$

where d_i is a vector containing the distance from the data sample i to each map unit and \bar{Q} is the average of quantization error achieved in the data reduction (see Fig. 1b).

For clustering the output map of the SOM a methodology based on mathematical morphology that takes into consideration the *fuzzy hit-* and *U-matrices* as grayvalue images is proposed. The methodology, which can be denoted as morphological clustering of SOM, is based on the so-called geodesic transformations [4]. While the basic morphological operations dilation $\delta^{(1)}(f)$ and erosion $\epsilon^{(1)}(f)$ are applied based on the grayvalue image f to be operated on and a particular structuring element, the geodesic dilation $\delta_g^{(1)}(f)$ and geodesic erosion $\epsilon_g^{(1)}(f)$ [4] take into consideration a third element, which receives the name of geodesic mask g . Furthermore f is denoted in this case as marker set. The geodesic mask limits the result of the basic morphological operation on the marker set to a particular image subdomain. These two geodesic operations are defined as:

$$\delta_g^{(1)}(f) = \delta^{(1)}(f) \wedge g, \quad \epsilon_g^{(1)}(f) = \epsilon^{(1)}(f) \vee g. \quad (2)$$

A pictorial description of a geodesic erosion can be observed in Fig. 2. The so-called morphological reconstruction results from the iterative repetition of a geodesic transformation until stability is achieved. Thus the reconstruction by erosion $R_g^*(f)$, which will be used here, can be expressed as [4]:

$$R_g^*(f) = \epsilon_g^{(i)}(f) \quad \forall i / \epsilon_g^{(i)}(f) = \epsilon^{(i+1)}(f). \quad (3)$$

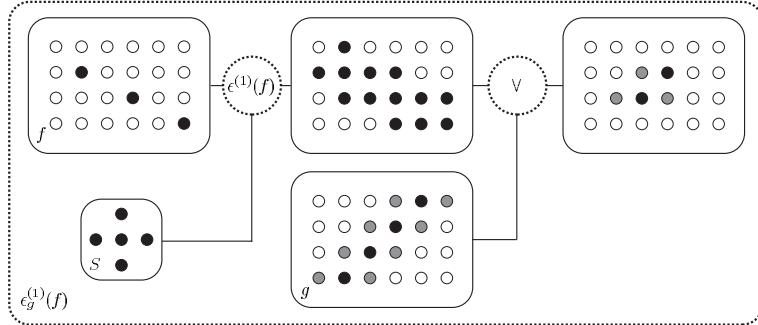


Fig. 2. Pictorial description of the geodesic erosion $\epsilon_g^{(1)}(f)$ with marker set f and geodesic mask g . S : Structuring element. $\epsilon^{(1)}(f)$: Erosion. \vee : Maximum.

In the procedure to be presented (see Alg. 2) the *U-matrix* is used as geodesic mask g after undergoing a binarization through a threshold γ . The goal to be attained by the procedure is the selection of a particular number (N) of prototypes. Moreover just one prototype in each cluster of the *U-matrix* can be selected. The binarization of the *U-matrix* defines a set of the possible areas, from which a prototype can be selected. The *fuzzy hit-matrix* is first binarized through a threshold corresponding to its maximal value. In this way a set of candidate prototypes is defined. Thence a reconstruction by dilation is computed using the candidate set as the marker set and the set of possible areas as geodesic mask. The resulting image is used two-fold. On the one hand its set intersection with the candidate set returns the prototypes to be added to the set of selected prototypes. On the other hand the set difference between the resulting image and the set of possible areas actualize this set. Thus the selection of just one prototype per cluster is achieved. The detailed operation is sequentially repeated by decreasing the threshold of the *fuzzy hit-matrix* until N prototypes have been selected.

Summarizing the procedure determines the clusters on the *U-matrix* through the parameter γ and then select the N larger local maxima of the *fuzzy hit-matrix* that fall within these clusters.

3 Multichannel Image Segmentation based on the Fuzzy Integral

3.1 Theoretical Background on the Fuzzy Integral

There are several types of fuzzy integral [1]. Till now the so-called Choquet Fuzzy Integral showed a better performance in the resolution of classification problems [6]. Thus the here presented framework uses this integral (4). Its mathematical

Algorithm 1 Morphological clustering of the *U-matrix* (U) based on reconstruction by erosion ($R_g^*(f)$). PA : Set of possible areas. γ : Threshold of the *U-matrix*, which determines the cluster structure. SP : Set of selected prototypes. N : Number of prototypes to be selected. FH : *Fuzzy hit-matrix*. CP : Set of candidate prototypes. sp : element of selected prototypes.

```

 $PA \leftarrow \{U > \gamma\}$ 
 $\theta \leftarrow 256$ 
 $SP \leftarrow \{\emptyset\}$ 
while  $\|SP\| < N$  do
5:    $\theta \leftarrow \theta - 1$ 
     $CP \leftarrow \{FH > \theta\} \cap PA$ 
    while  $\|CP\| \neq 0$  do
       $sp \leftarrow CP_i$ 
       $SA \leftarrow R_{PA}^*(sp)$ 
10:     $SP \leftarrow SP \cup \{sp\}$ 
       $PA \leftarrow PA - SA$ 
       $CP \leftarrow CP - SA$ 
    end while
  end while

```

expression is:

$$\mathcal{C}_\mu(\mathbf{x}) = \sum_{i=1}^n [h_{(i)}(x_i) - h_{(i+1)}(x_{i+1})] \cdot \mu(A_{(i)}) \quad (4)$$

Any Fuzzy Integral is capable of aggregating fuzzy information. The data from the different information sources, $\mathbf{x} = x_1, \dots, x_n$, are fuzzified, where h_i represent the fuzzifying functions (see (4)). The fuzzified data is operated against the quantification of *a priori* importance embedded in the fuzzy measure coefficients, $\mu(A_i)$. A_i , $i = 0, \dots, 2^{n-1}$, stands for the different subsets that can be established over the set of information sources. The parentheses in the subindices indicate a sorting operation previous to the aggregation, up to which the fuzzy measure coefficients weighting this aggregation are selected.

The fuzzy measures are functions on fuzzy sets, $\mu : \mathcal{P}(X) \rightarrow [0, 1]$, satisfying the following axiomatic conditions in case of a finite number of information sources: I. $\mu(\emptyset) = 0$; $\mu(X) = 1$ and II. $A \subseteq B \rightarrow \mu(A) \leq \mu(B)$. The consideration of n information sources to be aggregated makes necessary the assessment of $2^n - 2$ coefficients (n for individual sources, $\binom{n}{2}$ for two combinations, etc ...). Thus the number of coefficients to be assessed exponentially increases with respect to the number of information sources being fused. This fact represents a challenge for automatic assessment procedures.

There exists different types of fuzzy measures, whose differentiating characteristic is the kind of relationship between the fuzzy measure coefficients of individual sources and those of their subsets. They are used in different theoretical frameworks. Furthermore when using a particular type of fuzzy measure the construction of the fuzzy measure reduces to the determination of the co-

efficients of the individual sources (n coefficients). Fuzzy λ -measures [1] makes that relationship dependent on a parameter λ as stated by:

$$\mu(\{x_i\} \cup \{x_j\}) = \mu_{ij} = \mu_i + \mu_j + \lambda \mu_i \mu_j \quad (5)$$

There is a standard procedure for finding λ up to the importance of the individual information sources [7].

3.2 Framework for the segmentation of Multi-Dimensional Images

The block diagram of the framework for the segmentation of multi-dimensional images is depicted in Fig. 3. Some details on it are given following.

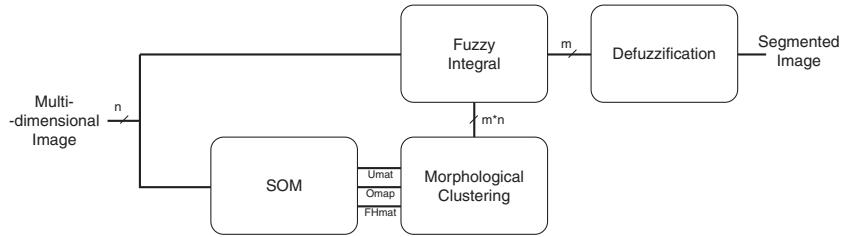


Fig. 3. Block diagram of the here presented framework for the segmentation of multi-dimensional images.

When using the fuzzy integral as classification function m fuzzy measures have to be constructed, where m is the number of classes. The fuzzy integral deliver m different results, one result for each fuzzy measure. Thus a SOM is first trained with the features present in the image to be segmented. After the training phase is completed the *fuzzy-hit* and *U-matrices* are computed. Thence the morphological clustering of the SOM is applied on the *U-matrix* in order to obtain m different vectors of n components, which are the coefficients of the individual information sources. The coefficients of the coalition subsets are computed by applying (5). The m fuzzy measures constructed this way are delivered to the fuzzy integral in order to classify the pixels on the m classes.

In the application of the fuzzy integral for image segmentation an integral is computed for each pixel of the input image. After applying the fuzzy integral with respect to the m fuzzy measures computed this way, the fuzzy integral results have to be defuzzified. This is attained by finding out the argument which delivers a maximum fuzzy integral result. The class of each pixel is finally assigned to this argument.

4 Application Results

The presented framework was first applied for the segmentation of benchmark color images. Preliminary results showed the convenience of considering a 5-

dimensional color space, namely formed by the three color channels of the typical RGB color space channels plus the hue and saturation of the HSI color space.

The “peppers” image is first segmented with the here presented framework. The obtained results can be observed in Fig. 4. The employment of the *fuzzy hit-matrix* instead of the *hit-matrix* improved the classification results (compare Figs. 4a and b). The clusters in the *U-matrix* (see Fig. 4d) can be identified in the grid projection achieved through the SOM, which is depicted in Fig. 4c. The segmentation was achieved by applying $\gamma = 40$ and $N = 6$ in the morphological clustering (see Fig. 4f).

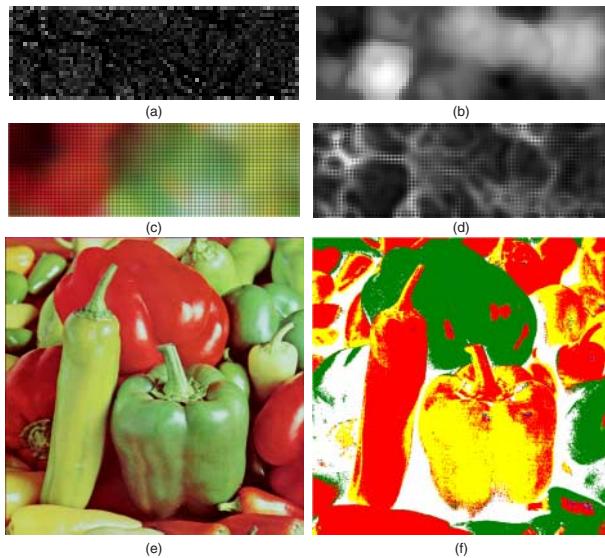


Fig. 4. Segmentation results of the “peppers” image (d) on a 5-dimensional feature space. (a) *Hit-matrix*. (b) *Fuzzy hit-matrix*. (c) Projection of RGB features on the output grid achieved by the SOM. (d) *U-matrix*. (f) Achieved label image.

The framework is tested on the “baboon” image as well. The obtained results are depicted in Fig. 5. In this case the parameters applied were $\gamma = 20$ and $N = 6$. The position in the *U-matrix* of the prototypes selected through morphological clustering can be observed in Fig. 5a.

5 Conclusions

A procedure for the morphological clustering of the *U-matrix*, which is computed in the two-dimensional data projection achieved by a SOM, is presented on this paper. In this way large dimensional feature spaces can be clustered somehow independently from the dimensionality of the feature space, i.e. just the response

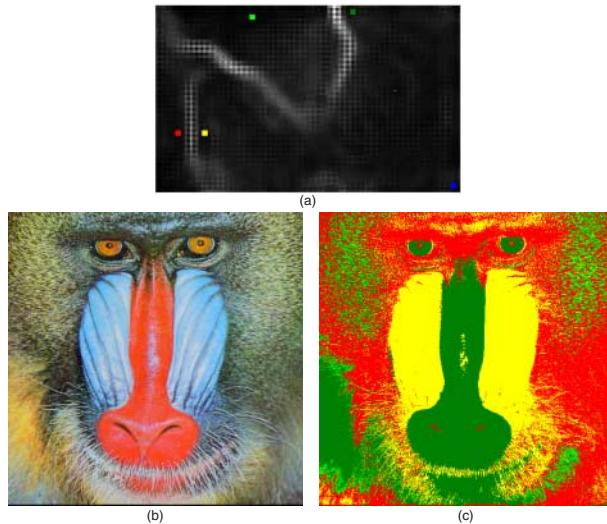


Fig. 5. Segmentation results of the “baboon” image (b) on a 5-dimensional feature space. (a) *U-matrix* with selected prototypes in color. The color of the prototype corresponds to the color in the label image (c).

of the *U-matrix* to the high-dimensional space influences the result. A framework based on this procedure and on the utilization of the fuzzy integral is described. The framework is successfully applied for the segmentation of five-dimensional color features on benchmark images. Future work will consider its application in a practical problem.

References

1. M. GRABISCH, H. T. NGUYEN, AND A. A. WALKER, *Fundamentals of Uncertainty Calculi with Applications to Fuzzy Inference*, Dordrecht: Kluwer Ac. Pub., 1995.
2. T. KOHONEN, *Self-Organizing Maps*, Heidelberg: Springer-Verlag, 1995.
3. M. KÖPPEN, *The curse of dimensionality*, in Proc. WSC5, 5th Online Conference on Soft Computing in Industrial Applications, ISBN: 951-22-5205-8, 2000.
4. P. SOILLE, *Morphological Image Analysis*, Berlin: Springer-Verlag, 1999.
5. A. SORIA-FRISCH, *Soft data fusion in image processing*, in Soft-Computing and Industry: Recent Advances, R. R. et al., ed., Springer-Verlag, 2002, pp. 423–444.
6. ———, *Hybrid som and fuzzy integral frameworks for fuzzy classification*, in To be published in Proc. IEEE International Conference on Fuzzy Systems, 2003.
7. H. TAHANI AND J. M. KELLER, *Information fusion in computer vision using the fuzzy integral*, IEEE Trans. SMC, 20 (1990), pp. 733–741.
8. J. VESANTO, *SOM-Based data visualization methods*, Intelligent Data Analysis, 3 (1999), pp. 111–126.
9. J. VESANTO AND E. ALHONIEMI, *Clustering of the self-organising map*, IEEE Trans. on Neural Networks, 11 (2000).

Introducing Long Term Memory in an ANN based Multilevel Darwinist Brain

F. Bellas and R. J. Duro
Grupo de Sistemas Autónomos
Universidade da Coruña

Abstract. This paper deals with the introduction of long term memory in a Multilevel Darwinist Brain (MDB) structure based on Artificial Neural Networks and its implications on the capability of adapting to new environments and recognizing previously explored ones by autonomous robots. The introduction of long term memory greatly enhances the ability of the organisms that implement the MDB to deal with changing environments and at the same time recover from failures and changes in configurations. The paper describes the mechanism, introduces the long term memory within it and provides some examples of its operation both in theoretical problems and on a real robot whose perceptual and actuation mechanisms are changed periodically.

1 Introduction

The Multilevel Darwinist Brain (MDB) is a proposal for a Darwinist cognitive mechanism intended for autonomous artificial organisms that must generate original solutions and use previously acquired experience when negotiating new environments and situations. The initial ideas for MDB were introduced in [1] and [2], and in this paper we are going to study its capabilities in order to adapt to changing environments.

This cognitive mechanism is based on a series of theories within the field of cognitive science that relate the brain and its operation through a Darwinist process. These theories are: the Theory of Evolutionary Learning Circuits (TELC) [3], [4]; the Theory of Selective Stabilization of Synapses (TSSS) [5], [6]; the Theory of Selective Stabilization of Pre-Representations (TSSP) [7] and the Theory of Neuronal Group Selection (TNGS) or “Neural Darwinism” [8]. Each theory has its own features but they all lead to the same concept of cognitive structure based on the fact that the brain adapts its neural connections in real time through evolutionary selectionist processes.

Taking inspiration from these theories we have developed an operational cognitive mechanism for artificial organisms which does not predetermine their behavior or learning and which allows them to find creative and original solutions in an autonomous way. This mechanism was designed to work in real dynamic environments and, in this kind of problems, it is very important to manage previously learned information efficiently. We have included a Long Term Memory (LTM) block in the basic schema of the MDB, and the study we have carried out on this LTM can be generalized to any kind of learning structure.

Some other authors have made reference to structures that make use of on-line evolutionary techniques, for example [9] and [10], applied to autonomous robotics but in these mechanisms no long term memory was considered.

The paper is structured as follows: in section 2 we will review the main features of the MDB, in section 3 we are going to include a Long Term Memory in the basic schema of the mechanism, in section 4 we will present the results obtained with this LTM and finally some conclusions of the work will be extracted in section 5

2 Multilevel Darwinist Brain

A Multilevel Darwinist Brain (MDB) [1][2], is an approach to implement a mechanism that is able to allow an organism to interact with the environment and learn from it so that it can improve its performance in time without the designer predetermining future behaviours, in a computationally efficient way. To this end we have considered several concepts:

- *Strategies*: sequences of actions applied to the effectors of the agent
- *World model (W)*: function that relates the sensory inputs of the agent in the instant of time t to the sensory inputs in the instant $t+1$ after applying a strategy
- *Internal model (I)*: is a function that relates the sensory inputs in instant of time $t+1$ with the internal state according to the motivation for the agent
- *Action-perception pair*: is a set of values made up by the sensorial inputs and the internal state obtained after the execution of a strategy in the real world. It is used when perfecting world and internal models.

We have connected these elements as shown in the block diagram of figure 1, which represents the whole mechanism. The final objective of the mechanism is to obtain the strategy the agent must execute in the real world to satisfy its motivations. This selected strategy is represented in the block diagram of figure 1 by the block labeled *Current Strategy* and is applied to the *Environment* through the *Effectors* providing new *Sensing* values for the agent. Thus, after each iteration, we have an action-perception pair obtained from the real world. These action-perception pairs are stored in the *Short-Term Memory* and are used as the fitness function for the evolution of the world and internal models.

Three concurrent evolutionary processes take place in our mechanism (figure 1):

- *World Model Evolver*: structure that manages world models through evolution. This structure evolves a population of world models and each moment of time selects the best one according to the information it has about the real world.
- *Internal Model Evolver*: manages an internal model base, evolving it and providing the best internal model available when it is requested.
- *Strategy Evolver*: manages and evolves strategies and when the agent needs one it provides it.

Each one of these evolutionary processes starts from the population stored in a memory (*World Model Memory*, *Internal Model Memory* and *Strategy Memory*). The contents of these memories are random for the first iteration of the mechanism. These contents are maintained from one iteration to the next in order to obtain a generally good population of models and not a superindividual. The individual with the highest fitness value after evolution according to the information of the short-term memory is selected as the current model, both in the case of world and internal models. These

current models are represented in figure 1 by the *Current World Model* and *Current Internal Model* blocks.

In the case of strategies, these are tested during their evolution by implementing a virtual environment using the current models, that is, the current world model provides the sensory inputs in instant $t+1$ for the strategy we want to test. The current internal model uses these predicted values as input to predict the internal state corresponding to the strategy according to the agent's motivation. The strategy with the highest fitness value is selected as the *Current Strategy* and is applied to the *Environment* through the *Effectors*, obtaining again new *Sensing* values and thus a new *action-perception pair* that is stored in the *Short-Term Memory*.

This basic cycle is repeated and, as time progresses, the models become better adapted to the real world and the selected strategies improve in their efficiency to fulfil the agent's motivations.

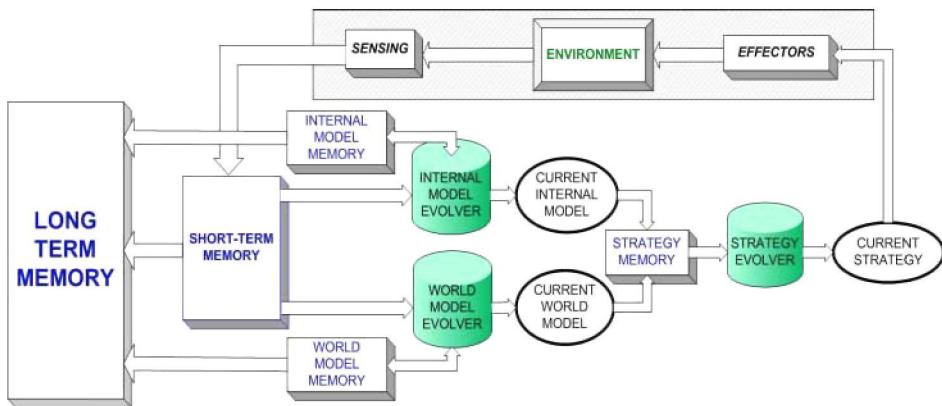


Figure 1. Block diagram of the enhanced MDB

Two elements that must be considered when applying this mechanism are the STM replacement strategy and the evolvers that must be used. Regarding the first of these two points, it must be considered that the information obtained from the real world which is stored in the STM can't, obviously, be infinite. So we need an efficient replacement strategy to manage the stored information.

The main objective must be to store the most relevant data that the environment provides, so the models (world and internal) can predict this environment in the internal operation of the mechanism. The information the agent receives in any given instant of time and which it must use to evaluate its models is partial and, in many cases, noisy or irrelevant. Any model it extracts from this instantaneous information will, in most cases, be useless for other instants of time. To generalize appropriate models the agent will have to consider and relate information obtained in many different instants of time in an efficient manner.

In more mathematical terms and considering a set of data that define a function, the fitness of the model is given by how well it fits the whole function and not by how well it fits the individual point or the subset of points it is considering in a given instant of time. This is what we have called a Sparse Fitness Function (SFF), the

fitness of the individual is given by its response to several different instantaneous local fitness functions, which together conform the desired global one. A more thorough study of this kind of fitness functions is presented in [11], but to summarize we have developed a replacement algorithm for the STM which contains 4 terms:

1. **Distance:** a min-max algorithm is used to decide if an action-perception pair is relevant. This term distributes the information in the problem space.
2. **Functionality:** the information considered is more relevant if it is wrongly-predicted by the current model.
3. **Initial relevance:** if a pair is wrongly-predicted before it is stored in the STM, it must be important to determine the general shape of the generalized function.
4. **Age:** an age term may be included to force the STM to adapt to changes in time.

The replacement mechanism includes these four terms in order to decide if a new pair must replace an existing one in the STM. The weight of each term is adjusted as a function of our requirements of generalization, antiquity, etc. or through the dynamic interaction of the agent with the world, but this is beyond the scope of the present paper.

In addition to managing the STM, when applying the basic MDB to real tasks [2], we have found that our main problem is learning the world models. The required evolutionary process tries to obtain the best ANN to model the environment, and this is complex in real problems where the environment is not regular at all.

We have carried out a lot of experiments with different parameters of the ANNs and GAs in order to find the best solution to a given problem. Finally, we decided to develop a new GA which must autonomously obtain the size of the ANNs representing the world, which must divide a complex function into simple parts automatically obtaining the appropriate number of simple ANNs that make up the complex one and, finally, which must evolve these different models in the same population. The algorithm was called the Promoter Based Genetic Algorithm and was presented in [12].

3 Long Term Memory

At this point, we can include a new term into the basic Block Diagram of the MDB that is shown in figure 1. This new block represents a new memory that stores the world and internal models that were good when used. In more specific terms, when an agent learns to model its environment or just a part of it with accuracy, we would like to store this model in a higher level memory (as compared to the STM). The resulting MDB block diagram is shown in figure 1 labeled as Long Term Memory.

In real problems if the environment where the agent is changes and later it returns to the previous one, it is not necessary to learn the same function again minimizing thus the learning time. We can apply this reasoning to the internal models, because the motivation of a behavior can change in a period of time but older motivations (previously learned internal models) could appear again. In what follows, we will refer just to world models but the results can be generalized to internal models too.

In the current configuration of the MDB, we have a PBGA obtaining models from the information that is stored in the STM using the presented replacement strategy. As

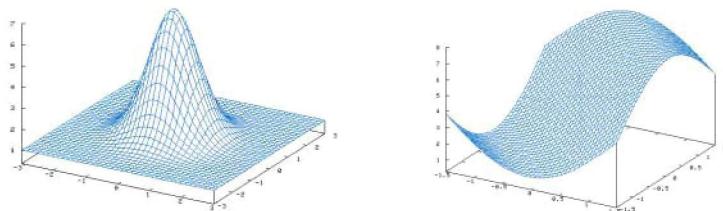


Figure 2. Theoretical functions used to test the behaviour of the LTM

we can see, this is a complex mechanism, where once a model is well-learned we don't want to lose it. To decide if a model must go into the LTM, we have to establish when a model is stable in terms of its error in predicting the pairs arriving at the STM. We just want to store in the LTM models that are really general because we would like to have only one model for each different zone of the environment or circumstance. Thus, we have used a stability criterion related to the error provided by the current world model applied to the whole STM in a given instant of time. If this error value is stable (in a 30% window) during a number of interactions equal to the size of the STM, we assume that the model is stable and can be stored in the LTM.

But what happens later? Do we have to look for more stable models? We want to have the best models in LTM, so each time a new model is stable (in terms of variations of prediction error), it must be stored in the LTM. Each time a model is fit to go into the LTM, we compare it with all those already in LTM. To carry out this comparison, we store the model and the STM the model was stable with. Then, we apply the new model to the STM of the existing models and the existing models to the current STM. If both errors are similar, we assume that the models predict the same zone and we leave the best one in LTM. If the errors are very different with the crossed STMs, we assume that we are dealing with a new model and it is automatically stored in the LTM.

All the models in the LTM in a given instant of time are introduced in the evolving population as seeds, so if the agent returns to a previous learned zone, the model will be present in the population and the prediction will be very good soon. Additionally, if the new area is similar to one the robot has seen before, the fact of seeding the population with the LTM models will allow the evolutionary process to reach a solution very fast.

Another element that must be considered is how to detect zone or circumstance changes so that data from two areas are not mixed in the STM thus forcing the

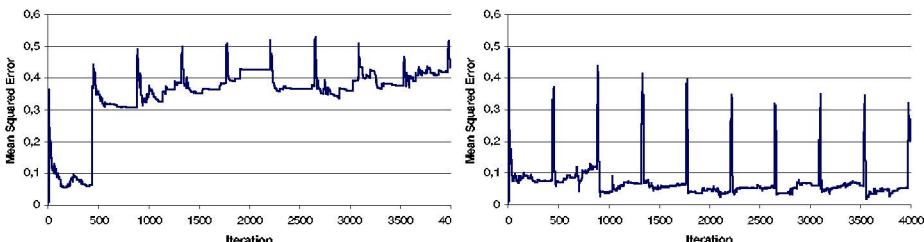


Figure 3. Evolution of the MSE of the prediction made by the current world model. The left figure was obtained without LTM and the right figure using LTM.

existence of intermediate models, which is something undesirable. To do this we have just considered an instability criterion for the predictions. When the predictions suddenly become highly unstable for a given period of time, we assume we are in a new zone and purge the contents of the STM by increasing the relevance of the age factor.

Summarizing, we have developed a mechanism to manage the LTM of an evolutionary process that is evolving ANNs to model the world. This mechanism is based on the detection of stable models that are stored in the LTM after applying them to the STMs of the previously stored models (crossed prediction errors). This way we just have one model (the best) for each stable zone of the environment.

4 Experimental results

To test the relevance of the LTM in our evolutionary processes, we first run the MDB using as environment a theoretical 3D function (Gaussian function). Thus we can control the predictions made by the ANNs obtained. After a period of time with the Gaussian, we make the “world” change and it becomes a 3D sinusoidal function. These two functions are displayed in figure 2.

In figure 3 we show the evolution of the mean squared error (MSE) of the prediction made by the best world model without LTM (left) and with LTM (right). As mentioned before, the evolution starts using the gaussian function and, after 450 iterations of evolution, we change it and start using the sinusoidal one. This change of functions occurs every 450 iterations from this point onwards. As we can see in figure 3, the evolution without LTM seems to adapt well to the first change of function, but in the subsequent ones the error increases and the results of modelling are poor. This is because the STM is not large enough to model both functions simultaneously, and the models obtained from the combination of pairs coming from both zones induce an intermediate model that is not good in either zone.

In the right figure, we show the result obtained using the LTM and the results are very good. The evolution of the error is good in the stable zones, there are clear transitions between zones (large error before instability is detected) and this behaviour is maintained in time. For example, in the first 450 iterations (gaussian

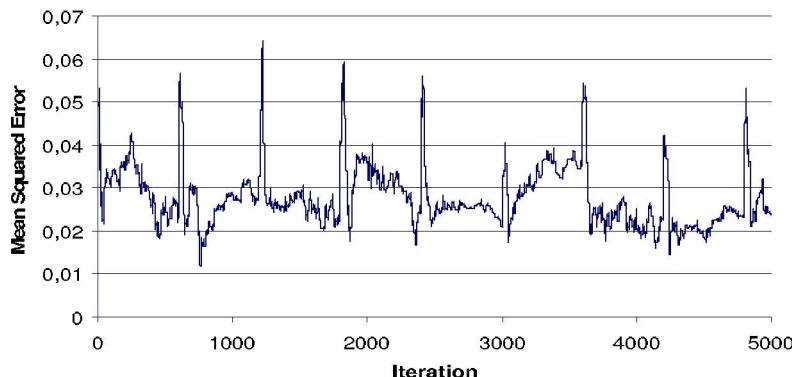


Figure 4. Evolution of the mean squared error of the prediction made by the actual world model in the distance predictions using the LTM in a

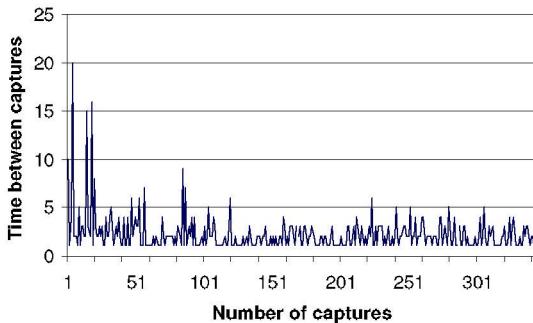


Figure 5. The left graph shows the time between two consecutive captures of the object. The right schema shows the experimental setup

function) the management mechanism stores one model in the LTM and in iteration 900, when we use the gaussian function again; this model is used as a seed in the population. The transition when entering in the gaussian zone the next times is very short because the models were learned before. This way, we reduce learning time maximizing the accuracy of the modeling. The instability detection works as expected and unlike in the case corresponding to the right figure, there is not any mixing of pairs in the STM and thus independent models are obtained automatically for each zone. In addition, it can be observed that as the MDB sees more iterations of the two zones, the models become better, the error is lower, this is an indication that the models corresponding to each zone in the LTM are being changed for better ones when these are available.

The discussion we have presented is focused on applying the MDB to real problems (real robots and real environments). Thus, we think it is necessary to test the whole mechanism in a real task. To do this, we have used the Pioneer 2 robot (a wheeled robot) and considering a simple task of finding an object with its sonar sensors and reaching it, as shown in the right part of figure 5, where we display a real trajectory followed by the robot to reach the objective.

It is a simple task, but the world model must deal with 16 noisy sonar sensors and two noisy actuators (wheels). The world models are represented by ANNs obtained with the PBGA and they have 5 inputs and 3 outputs. The inputs are: distance given by the nearest sonar, angular position of that sonar, a boolean value that permits distinguishing if the sonar is in the front or back of the robot and the two motor values given as linear and angular speed. The outputs are the predicted distance given by the nearest sonar, the predicted angular position of that sonar and the predicted boolean value.

In figure 4 we display the evolution of the mean squared error in the prediction of the first output (distance to the object) made by the current world model in each iteration. Every 600 iterations we assume that a failure in the real robot occurs skewing its perception of the world. Basically the perceptions of distances is inverted in sign and the actions of the wheels are exchanged. Obviously, initially, after the failure a large error appears in the prediction, but after the system has seen the two perception systems once it is able to recover very rapidly and its mean squared error remains between 0,02 and 0,03. In the left part of figure 5 we display the time between two consecutive captures of the objective, initially it is high, but after a while it consistently reaches it in two or three sensings.

Conclusions

In this work we have added a new element, a long term memory, to the Artificial Neural Network based Multilevel Darwinist Brain. This new element increases the power of the cognitive mechanism allowing the agent to remember ANN based world and internal models that have been useful before and incorporate them or even modifications of them into new behaviors due to the fact that the contents of the long term memory participate as seeds in the evolutionary processes within the MDB. The results confirm the usefulness of this strategy within the mechanism in terms of enhancing the capacity of the artificial organism to adapt to new situations and environments it finds itself in as well as to recover from failures and changes in its perceptual and/or actuation capacities.

Acknowledgements

This work was funded by Xunta de Galicia under project PGIDIT02PXIB10501PR the MCYT of Spain under projects TIC2000-0739C0404 and REN2000-0204-P4-0.

References

1. F. Bellas, J.A. Becerra, R. J. Duro, Using evolution for thinking and deciding, Proceedings WSES2001, pp 6161-6166.
2. F. Bellas, A. Lamas, R.J. Duro, Multilevel Darwinist Brain and Autonomously Learning to Walk, Proceedings CIRAS2001, pp 392-398.
3. M. Conrad, Evolutionary Learning Circuits, *J. Theor. Biol.* 46, 1974, pp. 167.
4. M. Conrad, Complementary Molecular Models of Learning and Memory, *BioSystems* 8, 1976, pp. 119-138.
5. J.P. Changeux, P. Courrèges, A Theory of the Epigenesis of Neural Networks by Selective Stabilization of Synapsis, *Proc. Natl. Acad. Sci. USA*, 70, 1973, pp. 2974-2978.
6. J.P. Changeux and A. Danchin, Selective Stabilization of Developing Synapsis as a Mechanism for the Specification of Neural Networks, *Nature* 264, 1976, pp. 705-712.
7. J.P. Changeux, T. Heidmann., and P. Patte, Learning by Selection. The Biology of Learning, P. Marler and H.S. Terrace (Eds.), Berlin, 1984, pp. 115-133.
8. G.M. Edelman, Neural Darwinism. The Theory of Neural Group Selection. Basic Books 1987.
9. P. Nordin, W. Banzhaf and M. Brämeier., Evolution of a World Model for a Miniature Robot Using Genetic Programming, *Robotics&Autonomous Systems*, Vol. 25, pp. 105-116.
10. L. Steels, Emergent Functionality in Robotic Agents through On-line Evolution, *Artificial Life IV*, 1995, pp. 8-14.
11. F. Bellas, J.A. Becerra, R.J. Duro, Sampled Fitness Functions in Complex Problems (parts I and II), Proceedings JCIS 2002, pp 631-639.
12. F. Bellas, R.J. Duro, Modelling the world with statistically neutral PBGAs. Enhancement and real applications, Proceedings ICONIP 2002, pp 2093-2098.

New Directions in Connectionist Language Modeling*

María José Castro and Federico Prat

Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València
E-46022 València, SPAIN
mcastro@dsic.upv.es

Departament de Llenguatges i Sistemes Informàtics
Universitat Jaume I de Castelló
E-12071 Castelló, SPAIN
fprat@lsi.uji.es

Abstract. In language engineering, language models are employed in order to improve system performance. These language models are usually N -gram models which are estimated from large text databases using the occurrence frequencies of these N -grams. An alternative to conventional frequency-based estimation of N -gram probabilities consists in using neural networks to this end. These “connectionist N -gram models”, although their training is very time-consuming, present a pair of interesting advantages over the conventional approach: networks provide an implicit smoothing in their estimations and the number of free parameters does not grow exponentially with N .

Some experimental works provide empirical evidence on the capability of multilayer perceptrons and simple recurrent networks to emulate N -gram models, and proposes new directions for extending neural networks-based language models.

1 Introduction

Language modeling is the attempt to characterize, capture and exploit regularities in natural language. In problems such as automatic speech recognition, machine translation or other pattern recognition tasks, it is useful to adequately restrict the possible or probable sequences of units which define the set of sentences (*language*) allowed in the application task. Under a statistical framework [1], if the system is used to recognize, the decision is usually based on the *maximum a posteriori* rule. The best sentence W^* is chosen so that the probability of the sentence W , knowing the observation X , is maximized:

$$W^* = \operatorname{argmax}_{W \in \Omega^+} \Pr(W|X) = \operatorname{argmax}_{W \in \Omega^+} \Pr(X|W) \Pr(W), \quad (1)$$

* Work partially supported by the Spanish CICYT under contract TIC2002-04103-C03-03.

where Ω is a vocabulary of words and $\Pr(X|W)$ is the conditional probability of the observation X when sentence W is generated. The a priori probability $\Pr(W)$, which represents the knowledge that the system has about the composition of correct language sentences in terms of the units in vocabulary Ω , must be estimated using a previously learned stochastic *language model*. In general, the incorporation of a language model reduces the complexity of the system (it guides the search for the optimal response) and increases its success rate.

These language models are unusually N -gram models and the value of N is usually small (2 or 3) in order to reduce the number of parameters of the model, which are learned from text using the occurrence frequencies of these N -grams, that is, subsequences of N word units. These N -gram models are popular in spite of some well-known weaknesses, such as the fact that large text databases are needed for their learning to be reliable and the way they ignore long-term dependencies between linguistic units. We feel that N -gram-like connectionist models will be able to override some of the flaws of conventional N -gram models.

In the following section, we give a brief introduction to statistical language modeling with N -grams. The connectionist approach with feedforward and recurrent neural networks is described in Sections 3 and 4, along with some results and analysis of their performance in different tasks. Finally, we draw some conclusions and propose new directions in connectionist language modeling in Sections 5 and 6.

2 Statistical language modeling with N -grams

Under a statistical framework, for each sentence, a language model must provide an estimation of its a priori probability of being the correct system response, as exemplified by Equation (1). Statistical language models are usually based on the prediction of each linguistic unit¹ in the sequence given the preceding ones [1]:

$$\begin{aligned} \Pr(W) &= \Pr(w_1 w_2 \dots w_{|W|}) \\ &= \Pr(w_1) \Pr(w_2|w_1) \dots \Pr(w_{|W|}|w_1 w_2 \dots w_{|W|-1}) \\ &= \prod_{i=1}^{|W|} \Pr(w_i|h_i), \end{aligned} \tag{2}$$

where $h_i = w_1 w_2 \dots w_{i-1}$ (the subsequence of units preceding w_i) is known as the *history* from which unit w_i has to be predicted. For further discussion, it is useful to assume a function Φ such that, given any history h , $\Phi[h]$ preserves all the information relevant for predicting the next linguistic unit. Thus, the probability $\Pr(W)$ can be expressed, without loss of generality, as

$$\Pr(W) = \prod_{i=1}^{|W|} \Pr(w_i|\Phi[h_i]). \tag{3}$$

¹ Depending on the kind of model, these units can be words, characters, phones, or other linguistic units.

The number of parameters to estimate becomes intractable as the length of the sentence increases. N -gram models [2] are the most extended method to reduce this number approximating the probability of a word as if only the last $N - 1$ words have influence. Hence, we can choose $\Phi[h_i] = w_{i-N+1} \dots w_{i-1}$, and taking into account that an approximation has been made, we can write

$$\Pr(W) \approx \prod_{i=1}^{|W|} \Pr(w_i | w_{i-N+1} \dots w_{i-1}). \quad (4)$$

The rationale behind this assumption is that it provides a model that can be automatically learned from a sample of linguistic sequences: for each possible N -gram, the model has a parameter representing the probability of the last unit given the $N - 1$ previous ones, and whose value can be estimated in a maximum-likelihood fashion by simply dividing the occurrence count of the N -gram in the sample between the sum of the counts corresponding to N -grams beginning with the same $N - 1$ units.

Deciding a value for N requires a trade-off between detail and reliability: larger values of N would provide more detailed models (the simplifying assumption would be less crude); however, more parameters implies that more training data is needed for the parameter values to be reliably estimated (so, for a given set of training data, less reliability in the estimation). Note that the number of parameters grows very fast (exponentially) with N (there are $|\Omega|^N$ N -grams), so typical values for N are two and three, and may even be four.

The detail-versus-reliability problem mentioned above does not usually allow for a good solution for the available training data, giving rise to the weaknesses of conventional N -gram models:

- As practical use of N -gram models is limited to low values of N , these models cannot exploit long-term dependencies between linguistic units.
- Even when using trigram models, maximum-likelihood estimation of parameters can be unreliable (for instance, an important fraction of possible trigrams is usually not present in the available training data). In order to alleviate this problem, some techniques can be applied, such as smoothing or clustering techniques [1].

In spite of these flaws, N -gram models are the most popular language models in automatic language processing models, for several reasons: they can be automatically learned from samples using a very simple parameter estimation procedure; they are easily integrated into automatic speech recognition systems using a unified statistical framework; and they are very successful in modeling short-term dependencies between linguistic units.

3 Statistical language modeling with feedforward neural networks

A different approach to statistical language models based on N -grams consists in using neural networks. A first step in this direction was given in 1989 by Naka-

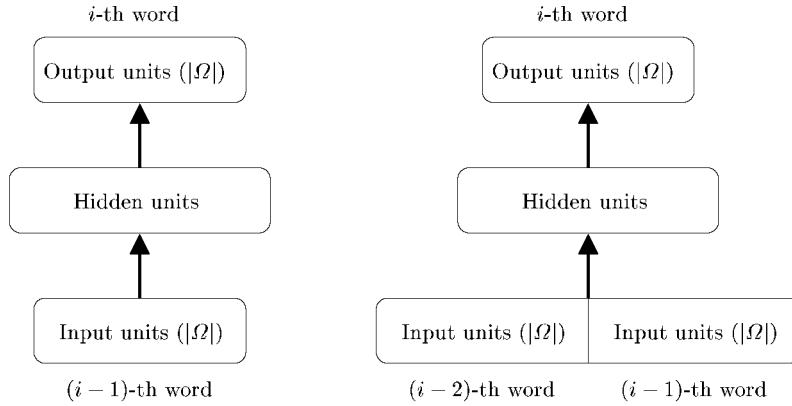


Fig. 1. Connectionist N -gram models for bigrams (*left*) and trigrams (*right*) by using multilayer perceptrons. In the input layer, sequences of $N - 1$ words are represented using a local coding for each word; in the output layer, aimed at predicting the word that follows the input sequence, a unit is devoted to each word in the vocabulary. After training the network as a classifier (each sequence of $N - 1$ words must be labelled with the word that follows it), outputs are considered as estimations of posterior probabilities of the kind N -gram models need. The bigram net predicts the i -th word given the $(i - 1)$ -th word. The input of the trigram net is composed of the $(i - 1)$ -th and $(i - 2)$ -th words.

mura and Shikano [3] who empirically showed how multilayer perceptrons (called “NETgrams” by the authors) can emulate N -gram model predictive capabilities with additional generalization features.

In their work on an N -gram word category prediction task, the authors empirically demonstrated that the connectionist model is equivalent to the conventional N -gram model, but one advantage of the NETgram over the conventional model is that its number of free parameters increases linearly with N instead of exponentially. Another advantage which has been empirically demonstrated by the authors is that the connectionist model directly performs a smoothing of the predictions (just like the deleted interpolation), which does not occur with the conventional model (it usually needs a posterior smoothing). The major disadvantage of the NETgram versus the conventional approach is the high temporal cost of training the model.

We will call “connectionist N -gram model” to a statistical language model which follows Equation (4) and where the probabilities that appear in that expression are estimated with a neural network (a multilayer perceptron [4, 5], for example). The model naturally fits under the probabilistic interpretation of the outputs of the neural networks: if a neural network is trained as a classifier, the outputs associated to each class are estimations of the posterior probabilities of the defined classes. The demonstration of this assertion can be found in a number of places, for example, in [5]. An illustration of two multilayer perceptrons to estimate bigrams and trigrams is shown in Figure 1.

Experimental results of these kinds of models can be found in [6, 7]. Our experimental work provides empirical evidence, on a small-vocabulary Spanish text corpus, on multilayer perceptron capability to emulate N -gram models. In the input layer, sequences of $N - 1$ words are represented using a local coding for each word; in the output layer, aimed at predicting the word that follows the input sequence, a unit is devoted to each word in the vocabulary. After training the network as a classifier (each sequence of $N - 1$ words must be labelled with the word that follows it), outputs are considered as estimations of posterior probabilities of the kind N -gram models need.

In order to compare conventional and connectionist models, we estimated a bigram and trigram model for the task using the “CMU-Cambridge Statistical Language Modeling Toolkit”². Finally, we calculated the perplexity of a test set in order to measure the quality of the estimated models. Perplexity can be intuitively interpreted as the geometric mean of the branch-out factor of the language [1]: a language with perplexity x has roughly the same difficulty as another language in which every word can be followed by x different words with equal probabilities.

Our results on bigrams and trigrams favorably compares connectionist models with conventional ones: in both the bigram and the trigram case, connectionist models achieved a lower test-set perplexity. Specially the neural network for trigrams performed much better than the conventional trigram. Moreover, in the trigram case, the connectionist model implied a large reduction (more than 95%) in the number of free parameters over the conventional one.

Similar conclusions were obtained by Xu and Rudnicky [8]. Their experimental results (with a moderate vocabulary size task and by using a single layer network) showed that the artificial neural network could learn a language model with a performance even better than standard N -gram models.

4 Statistical language modeling with recurrent neural networks

Recurrent neural networks can also be applied directly to the connectionist statistical language modeling problem. One possibility would be to use Elman networks [9], where the activations at the first hidden layer may be fed back to the input layer using context units. The context units could be fully connected to the first hidden layer via the context weights. Elman style networks are multilayer networks with simple recurrences, that is, with back-connections whose weights are fixed to value one. These recurrences attempt to catch the dynamicity in parsing natural language and have the potential to learn temporal dependencies of unspecified length. This architecture might develop an internal temporal representation which encodes the number of previous linguistic units needed to predict the following one; that is, the hidden layer might be expected to encode

² <http://svr-www.eng.cam.ac.uk/~prc14/toolkit.html>

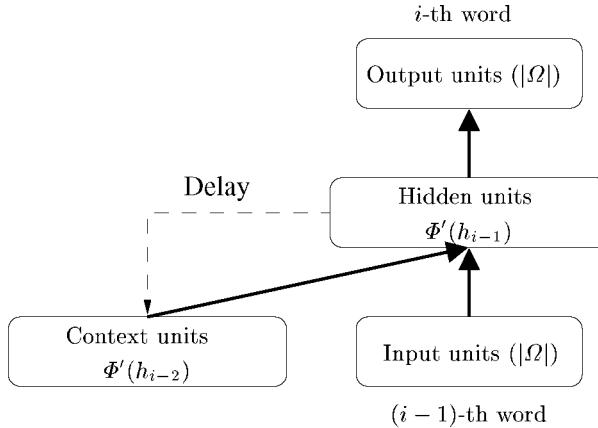


Fig. 2. Connectionist N -gram model by using simple recurrent neural networks. As in Figure 1, local coding is used to represent each word in the input and the prediction in the output. After training, the recurrent net predicts the i -th word given the $(i - 1)$ -th word and the information of the context units $\Phi'(h)$.

a representation of $\Phi[h]$ in Equation (3).³ Therefore, we could avoid the design decision to select the value of N , as the recurrent architecture would not need the histories to be previously categorized into equivalence classes. An illustration of a connectionist N -gram model by using a simple recurrent neural network is shown in Figure 2.

In a very recent work by Rodriguez [10], experiments with a simple recurrent network are reported. The recurrent network is trained on a large corpus in order to examine the ability of the network to learn bigrams, trigrams, etc., as a function of the size of the corpus. The author performed experiments with the Wall Street Journal database in order to make predictions of the next possible letter. Thus, a vocabulary of 28 symbols (letters from the English alphabet) was used. With enough training data and hidden units, the network was capable to learn 5 and 6-gram dependencies [10].

5 Discussion and conclusions

The advantages of the connectionist approach to language modeling are due to their automatic estimation (the same feature as with statistical language models), the lowest (in general) number of parameters of the obtained models and the automatic smoothing performed by the neural networks estimators.

³ Actually, the hidden layer of such recurrent network must fulfilled something more: function $\Phi[h]$ from Equation (3) must provide information to predict the following linguistic unit; the hidden layer of the recurrent network (we will call this representation $\Phi'(h)$) must provide the same information to predict the following unit and also the information needed to predict future units.

Estimation of connectionist N -gram models in more ambitious tasks is needed in order to fulfill these advantages. For example, a trigram model (estimated with a multilayer perceptron of two hidden layers of 100 units each using a local coding for each word) for a restricted-semantic domain task with a vocabulary of 2,000 words would have around 600,000 parameters. We realize that the scaling of these ideas is not trivial: the larger the lexicon is, the larger the number of parameters the neural network needs. Problem becomes even harder when using recurrent neural networks.

In this line, similar ideas proposed in this work have been presented by Bengio et al. [11, 12]. Their work shows on two large text corpora that a neural probabilistic model very significantly improves on a state-of-the-art trigram model. Their model learns simultaneously a distributed representation for each word of the vocabulary with the probability function for word sequences. Generalization is obtained because a sequence of words that has never seen before gets high probability if it is made of words that are similar to words forming an already seen sentence [11, 12]. Other recent works extend the same approach to other large vocabulary continuous speech recognition tasks [13].

We think that the use of such distributed representation for each word of the vocabulary, along with its combination of a connectionist language model for the same task which uses categorized lexicon, can be a successful approach for more realistic tasks with large lexica.

6 New directions in connectionist language modeling

This work is a step towards a better understanding of how neural networks can perform language modeling applications. We have investigated an alternative approach to build statistical language models. Experimental results for a variety of authors [3, 6–8, 10–13] show that artificial neural networks can learn language models which have performances comparable or better than standard statistical models based on N -grams.

We also propose new directions for extending language models based on artificial neural networks:

1. By using a distributed representation of words instead of a local one, large-vocabulary tasks could be easily tackled.
2. By using categories of words (semantical or syntactical) in restricted-semantic domain tasks, much more compact artificial neural networks could be used.
3. By using recurrent networks, the estimation of the probability of a word could depend on previous context without fixing a priori its amount, but learning it from data.
4. By using committees of artificial neural networks [5]. For instance, by using committees of multilayer perceptrons, each network could be trained to estimate N -gram models of increasing values of N , and a combination of the predictions of each network could be used as the estimator of the searched probability.

5. By mixing the connectionist language model with standard N -grams (or other language models), perplexity could also be reduced.

References

1. F. Jelinek. *Statistical Methods for Speech Recognition*. Language, Speech, and Communication. The MIT Press, 1997.
2. L. R. Bahl, F. Jelinek, and R. L. Mercer. A Maximum Likelihood Approach to Continuous Speech Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):179–190, 1983.
3. M. Nakamura and K. Shikano. A study of English word category prediction based on neural networks. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP'89)*, pages 731–734, Glasgow (Scotland), May 1989.
4. D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *PDP: Computational models of cognition and perception, I*, chapter Learning internal representations by error propagation, pages 319–362. MIT Press, 1986.
5. C. M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1995.
6. M. J. Castro, F. Prat, and F. Casacuberta. MLP emulation of N -gram models as a first step to connectionist language modeling. In *Proceedings of the International Conference of Artificial Neural Networks (ICANN'99)*, pages 910–915, Edinburgh (UK), September 1999.
7. María José Castro, Vicent Polvoreda, and Federico Prat. Connectionist N -gram Models by Using MLPs. In *Proceedings of the Second Workshop on Natural Language Processing and Neural Networks (NLPNN'01)*, pages 16–22, Tokyo (Japan), November 2001.
8. Wei Xu and Alex Rudnicky. Can Artificial Neural Networks Learn Language Models? In *Proceedings of the 6th International Conference in Spoken Language Processing (ICSLP'00)*, Beijing (China), 2000.
9. J. L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.
10. Paul Rodriguez. Comparing Simple Recurrent Networks and n -grams in a Large Corpus. *Journal of Applied Intelligence*, 2003. In press.
11. Yoshua Bengio, Rejean Ducharme, and Pascal Vincent. A Neural Probabilistic Language Model. In *Advances in Neural Information Processing Systems*, volume 13, pages 932–938, 2001.
12. Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.
13. Holger Schwenk and Jean-Luc Gauvain. Connectionist language modeling for large vocabulary continuous speech recognition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP'02)*, pages 765–768, Orlando, Florida (USA), May 2002.

Rules and Generalization Capacity Extraction from ANN with GP

Juan R. Rabuñal, Julián Dorado, Alejandro Pazos, Daniel Rivero

Univ. da Coruña, Facultad Informática, Campus Elviña, 15071 A Coruña, Spain
{ juanra, julian, apazos, danielrc } @udc.es

Abstract. Different techniques for extracting Artificial Neural Networks (ANN) rules have been used up to the present time, but most of them have focused on certain types of networks and their training. However, there are practically no methods which deal with ANN rule-discovery as systems that are independent from their architecture, training, and internal distribution of weights, connections, and activation functions. This paper proposes a method based on Genetic Programming (GP) with the purpose of achieving the generalization capacity characteristic of ANNs, by means of symbolic rules which can be understood by human beings.

1 Introduction

Artificial Neural Networks (ANN) are systems which can be easily implemented and used. They also have a number of features which make them ideal for solving problems in various fields. However, many developers tend not to use them because they are considered as “black boxes”. For this reason, there are certain fields, such as medicine, for instance, who do not consider ANNs as totally reliable. One example is a medical diagnosis: any computational system which is designed to provide a diagnosis should be able to justify how and why it has achieved that diagnosis.

The system’s design must fulfil certain requirements [1]:

1. It must work with any ANN architecture.
2. It must work with any type of ANN training.
3. It must be correct.
4. It must have a highly expressive level.

2 State of the Art

2.1 Genetic Programming

Genetic Programming (GP) appeared as an evolution of Genetic Algorithms (GA). Two papers can be found from the first two conferences on GA (International Conferences on Genetic Algorithms-ICGA) [2][3]. Many authors consider that both articles are the heralds of GP, however, Friedberg [4] started to work on the automatic creation of programmes in machine language in 1958 and 1959. But it was John R. Koza who coined the term which gave title to his book “Genetic Programming” [5]. It provides the formal basis of GP which is used in today’s works.

2.2 ANN rules Extraction

Several authors have done research on the equivalence between ANNs and fuzzy rules [6][7][8]. Apart from being purely theoretical solutions, they require a great number of rules in order to approach the ANN's functioning.

Andrews [9] identifies three techniques for discovering rules: "decompositional", "pedagogical" and "eclectic". The first one refers to discovery at the neural level. This makes the methods related to this technique totally dependent from the architecture of the ANN. The second one deals with the ANN as a "black box", where, by means of applying inputs, a forward analysis of the neurons in the hidden layers of the network is carried out. The third technique uses the ANN's architecture and the input-output pairs as a complement to a symbolic training algorithm.

Towell and Shavlik [10] apply the first technique using the connections among neurons as rules, based on a simple transformation. This limits the discovery to networks with a certain multilayer architecture and few process elements [9].

Thrun [11] has presented the main approach using the second technique, titled "Validity Interval Analysis" (VI- Λ). The algorithm uses linear programming (SIMPLEX), applying value intervals to each neuron's activations in each layer. This algorithm has, in the worst scenario, an exponential complexity.

Other approaches using the second technique are the algorithms RULENEG [12] and DEDEC [13]. These use an ANN in order to extract rules from another ANN starting from the training set. Some other rule-discovery techniques are [14][15][1].

GA have recently been used for searching and discovering ANN rules [16]. It uses a GA where chromosomes are multicondition rules based on value intervals or ranges applied to ANN inputs. Wong and Leung have presented LOGENPRO (Logic Grammar Based Genetic Programming) [17]. It uses GP. More recently, Engelbrecht, Rouwhorst and Schoeman [18] have applied the GP technique and decision trees in order to discover knowledge in DB, designing the algorithm called BGP (Building-Block Approach to Genetic Programming).

3 Description of the System

The algorithm's design may be tackled from the viewpoint of adjusting as faithfully as possible the four features explained by Tickle [1], which were commented in the paper's introduction. GP complies with all of these requirements [20]. This paper has applied GP as an algorithm for building a syntactic tree which reflects a set of rules as similar as possible to an ANN's functioning. Symbolic regression applied to input-output patterns has been used for this purpose. These patterns are input sequences applied to an ANN and the outputs produced by it.

The purpose is to extract the knowledge which the ANN obtains from the training patterns and from test cases which have not been provided during the learning process. For this purpose, we must design an algorithm for the creation of new test cases which are going to be applied to the ANN, allowing the analysis of its behaviour in the face of new situations. With regard to discovering the generalisation capacity, an algorithm called "algorithm for the creation of new patterns" has been designed. New test cases can be created by means of this algorithm, allowing to discover rules and expressions of those parts of the ANN's internal functioning which have not been explored by the learning patterns. For this purpose, we start from the

learning patterns. Once it is considered that a sufficient number of rules have been discovered, 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. In order to avoid having an unlimited number of test cases, the number of new test cases is set, establishing an elimination rule for test cases: when there is a new test case where the ANN’s outputs coincides with the outputs of those produced by the best combination of rules obtained, then that test case is eliminated, given that it is considered that the rules obtained have acquired the knowledge which is represented by that test case.

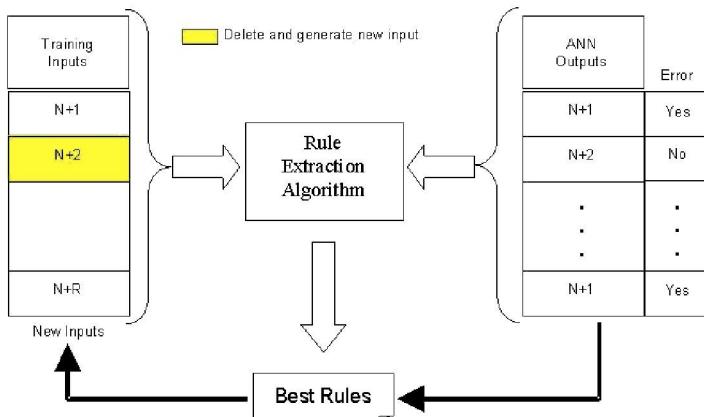


Fig. 1. Generalization Rule Extraction process

The following parameters are involved in the algorithm for creating new patterns:

- **Percentage of new patterns:** the number of sequences is extended in a percentage which relates to the number of learning patterns.
- **Probability of change:** a sequence of new inputs is chosen at random from the training file. For each input the same value stays, or a new one is generated at random, following a certain probability.

4 Results

ANNs have proven enormously useful at tasks where, starting from an input sequence, it must be decided whether these values correspond to a given classification. The following example will serve to test the system’s functioning when dealing with multiple classification tasks. It has been observed and carefully analysed in fields such as ANNs and rule discovery [19].

The task consist of identifying a plant’s species: *Iris setosa*, *Iris versicolor* and *Iris virginica*. This example has 150 cases with four continuous variables which stand for 4 features about the flower’s shape. The four input values represent measurements in millimetres of the following characteristics: Petal Width (X_1), Petal Length (X_2), Sepal Width (X_3) and Sepal Length (X_4).

The learning patterns have 4 inputs and 3 outputs. The three outputs are Boolean ones, representing each *Iris* species. By doing it in this manner (three Boolean outputs instead of a multiple one) additional information can be provided about whether the system's outputs are reliable or not (only one of the three is true) [20].

Figure 2 shows the space distribution for variables X_1 and X_2 . According to authors such as Duch, Adamczak and Grabczewski [21], a finer classification for the three classes is achieved by these variables. Specifically, a 98% of correct answers is achieved by using only these two variables. Therefore, they are an important reference point in order to compare the results obtained graphically.

The values corresponding to the four input variables have been normalised in the interval [0-1] so that they are dealt with by the ANN. An ANN has been trained later on, so that using its outputs we can discover the rules which the ANN has learnt, thus checking the differences and similarities to what it should learn.

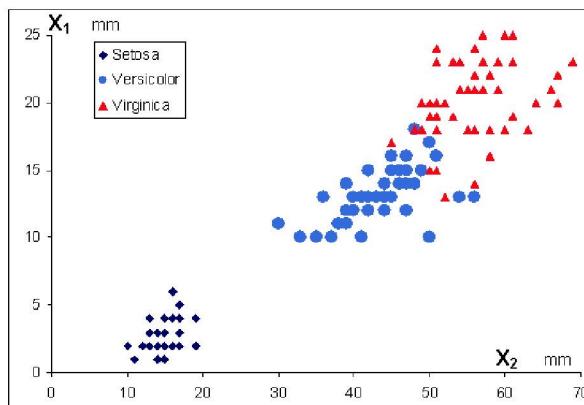


Fig. 2: Original distribution of all three classes (real values)

4.1 Rules extraction using only GP

The parameters that the best results produce are: 20 random constant values in the range [0-1], $<$, $>$, AND, OR, NOT operators, tournament selection algorithm, 95% of crossover rate, 4% of mutation rate, 500 individuals of population size and 0.0001 of parsimony level.

The rule obtained from the original (normalised) patterns for *Iris setosa* produces an adjustment of 100% correct answers, being the following one:

$$(X_1 < 0.3141)$$

For *Iris versicolor* produces an adjustment of 100% correct answers:

$$\begin{aligned} & (((0.6773 > X_3) \text{ OR } (0.5266 < X_2 < (0.7364))) \text{ AND } \\ & (((0.6104 < X_1 < 0.7217) \text{ OR } ((0.3360 < X_1 < 0.5266) \text{ OR } \\ & (0.5266 < X_2 < 0.7217))) \text{ AND } ((X_3 > X_1) \text{ OR } (0.6773 > X_1))) \end{aligned}$$

For *Iris virginica* produces an adjustment of 99.33% correct answers:

$$\begin{aligned} & (((X_1 > X_2) \text{ OR } (X_2 > 0.7182)) \text{ AND } ((X_2 > X_4) \text{ OR } \\ & (((0.7390 < X_2 < 0.7650) \text{ OR } (X_4 > 0.9028)) \text{ OR } (X_1 > X_3)))) \end{aligned}$$

It should be noted that the error made by the latter expression produces as system output (in the three outputs) the classification of *Iris versicolor* and *Iris virginica* (“false-true-true” output) which is a null output, being eliminated.

In order to carry out a deep analysis of the behaviour of the rules obtained, an analysis file has been built with all the possible input values for the four variables, taking regular intervals: for X_1 each 7 mm, for X_2 each 2.5 mm., for X_3 each 4.4 mm. and for X_4 each 7.9 mm. With these intervals, the analysis file has 14641 examples, and the whole possible range of classifications that the rules carry out can be analysed.

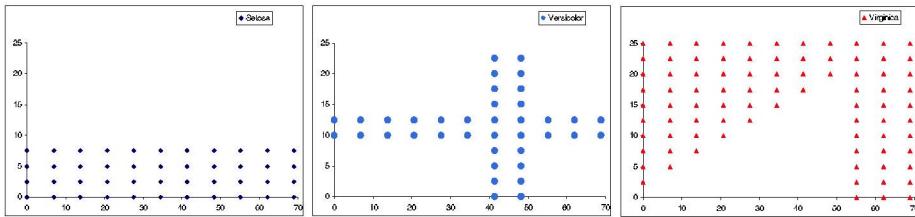


Fig. 3: Distribution obtained of the three classes

4.2 ANN rules extraction

Martinez and Goddard [22] have proven that a maximum adjustment of 98.67% correct answers (2 errors) is achieved with 6 neurons in only 1 hidden layer and 3 output neurons. In the cases with *Iris setosa*, no error is obtained, in the cases with *Iris versicolor* and *Iris virginica* 2 errors are made, which are not detected because the ANN produces a valid classification (only one true output), but an erroneous one.

The rules obtained from the inputs-outputs of the ANN for *Iris setosa* produces an adjustment of 100% correct answers, being the following:

$$(X_1 < 0.3116)$$

For *Iris versicolor* produces an adjustment of 100% correct answers:

$$\begin{aligned} & ((0.2892 < X_1 < 0.5316) \text{ OR } ((X_3 > X_2) \text{ OR } ((X_4 > 0.7643) \\ & \quad \text{AND } (X_2 > X_1))) \text{ AND } (0.5316 < X_2 < 0.7268))) \end{aligned}$$

For *Iris virginica* produces an adjustment of 100% correct answers:

$$\begin{aligned} & (((X_1 > X_3) \text{ AND } (X_1 > X_2)) \text{ OR } (((0.5497 < X_1) \text{ AND } \\ & \quad (0.5497 > X_3) \text{ OR } (0.7279 < X_2))) \text{ AND } (0.6787 < X_2))) \end{aligned}$$

If we analyse the results obtained, we may observe the distributions carried out by the ANN, and the rules obtained by using the analysis file. It is shown in Figure 4.

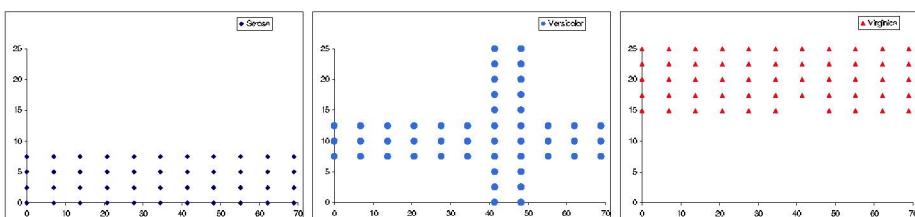


Fig. 4: Distribution obtained of the three classes produced by the rules

4.3 Generalization of ANN

If we apply the rule-extraction mechanism combined with the algorithm for creating new patterns, the result is that with a value between 20% and 35% of new patterns and between 20% and 25% of probability of change, the values are the ones with a better behaviour in the algorithm. A dynamic adjustment of 96.27% was obtained for *Iris setosa*, 89.17% for *Iris versicolor*, and 93.13% for *Iris virginica*. This is considered as a dynamic adjustment since it depends upon the new test cases which are dynamically generated, and that adjustment took place at the moment when the process was detained. Obviously, these adjustment values do not correspond to the real ones. The analysis file is transferred to the rules obtained in order to achieve the real adjustments. The real adjustment for the *Iris setosa* is 95.45%, for *Iris versicolor* is 88.81% and for *Iris virginica* is 77.88%.

The rules obtained for the classification of *Iris setosa* are:

$$((-(0.1941) > (X_2 - X_3)) \text{ AND } ((X_3 - X_2) > (0.6381 * X_1)))$$

For *Iris versicolor* are:

$$\begin{aligned} &((X_4 \% 0.8032) > X_3) \text{ AND } (((X_1 + (X_2 \% 0.7875)) > X_3) \text{ AND } \\ &(X_2 < ((X_4 \% 0.7875) - (0.4966 * X_1) \% 0.8896))) \end{aligned}$$

For *Iris virginica* are:

$$\begin{aligned} &(X_4 < ((X_1 - (-(0.4925 * (X_2 * 0.5280)))) - (X_4 - X_2))) \text{ AND } \\ &((X_3 < X_2) \text{ OR } (X_1 > (X_3 - X_2))) \end{aligned}$$

The distribution of the rules obtained may be seen in the following figures. They show that the visual aspect of the distributions in Figure 5 and Figure 6 are very similar. This leads to believe that the final behaviour of the rules is very similar to the one produced by the ANN and the algorithm has done a good job.

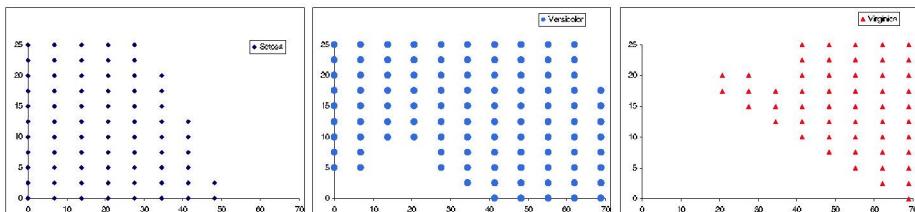


Fig. 5: Distribution obtained of the three classes produced by the ANN

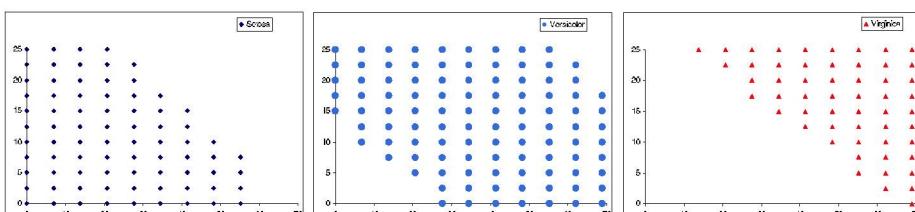


Fig. 6: Distribution obtained of the three classes produced by the rules

The table 1 draws a comparative chart of the adjustments achieved by using only the learning patterns, and those achieved by using the algorithm for creating new patterns. It shows the adjustment values produced by the rules obtained with both methods when the analysis file patterns are evaluated (14641 cases).

Table 1. Comparative between using only the learning patterns and using the algorithm for creating new patterns

<i>Method</i>	<i>Iris setosa</i>	<i>Iris versicolor</i>	<i>Iris virginica</i>
Without dynamic creation	63.34%	56.97%	57.23%
With dynamic creation	95.45%	88.81%	77.88%

5 Conclusions

As shown by the results which have been presented, the rule-discovery algorithm based on GP can be compared to other existing methods, although it offers the great advantage of being capable of application to any ANN architecture with any activation function.

We may conclude that a rather accurate simulation of the ANN's behaviour has been achieved with regard to the possible combinations of values which may occur in the inputs. Therefore, we may state that the knowledge treasured by the ANN has been obtained in an explicit and comprehensible way. It has also been possible to express that ANN's generalisation capacity by means of symbolic rules.

6 Future Works

As described throughout the article, the algorithm which has been presented works on data sequences generated by the ANN. Thus, it could also be applied to ANNs with recurrent architecture and to dynamic problems, where the RANN outputs depend upon the present inputs and upon the inputs in N previous cycles.

In order to accelerate the rule-discovery process, it is possible to use a network of computers with the purpose of proceeding with the search in a distributed and recurring manner, exchanging rules (subtrees) among them.

Acknowledgements

This work has been supported in part by the University of A Coruña project "Extracción de reglas de RNA mediante CE".

References

1. Tickle, A.B., Andrews, R. Golea, M. Diederich, J. "The truth will come to light: directions and challenges in extracting the knowledge embedded within trained artificial neural networks". IEEE Transaction on Neural Networks, vol. 9, nº 6, pag. 1057-1068, 1998.
2. Cramer, N.L. "A Representation for the Adaptive Generation of Simple Sequential Programs", Grefenstette: Proceedings of First International Conference on Genetic Algorithms, 1985.

3. Fujiki C. "Using the Genetic Algorithm to Generate Lisp Source Code to Solve the Prisoner's Dilemma", International Conf on GAs, pp. 236-240, 1987.
4. Friedberg R.M. "A learning machine: Part I", IBM Journal of Research and Development, 2(1) pag. 2-13, 1958.
5. Koza J. "Genetic Programming. On the Programming of Computers by means of Natural Selection". The Mit Press, Cambridge, Massachusetts, 1992.
6. Jang J., Sun C. "Functional equivalence between radial basis function networks and fuzzy inference systems". IEEE Transactions on Neural Networks, vol. 4, páginas 156-158, 1992.
7. Buckley J.J., Hayashi Y., Czogala E. "On the equivalence of neural nets and fuzzy expert systems", Fuzzy Sets Systems, Nº 53, pag. 129-134, 1993.
8. Benítez J. M., Castro J. L., Requena I. "Are artificial neural networks black boxes ?". IEEE Transactions on Neural Networks, vol. 8, nº 5, pag. 1156-1164, 1997.
9. Andrews R. Diederich J. & Tickle A. "A Survey and Critique of Techniques For Extracting Rules From Trained Artificial Neural Networks". Knowledge Based Systems vol. 8, pag. 373-389, 1995.
10. Towell G., Shavlik J. W. "Knowledge-Based Artificial Neural Networks". Artificial Intelligence, 70, pag. 119-165.
11. Thrun S. "Extracting rules from networks with distributed representations". Advances in Neural Information Processing Systems (NIPS) 7, G. Tesauro, D. Touretzky, T. Leen (eds), MIT Press.
12. Pop E., Hayward R., Diederich J. "RULENEG: Extracting Rules from a Trained ANN by Stepwise Negation". Queensland University of Technology, Neurocomputing Research Centre. QUT NRC Technical report, 1994.
13. Tickle A. B., Orlowski M., Diederich J. "DEDEC: A methodology for extracting rules from trained artificial neural networks". Queensland University of Technology, Neurocomputing Research Centre. QUT NRC Technical report, 1996.
14. Chalup S., Hayward R., Diederich J. "Rule extraction from artificial neural networks trained on elementary number classification task". Queensland University of Technology, Neurocomputing Research Centre. QUT NRC Technical report, 1998.
15. Visser U., Tickle A., Hayward R., Andrews R. "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, pag. 133-139. 1996.
16. Keedwell E., Narayanan A., Savic D. "Creating rules from trained neural networks using genetic algorithms". In the International Journal of Computers, Systems and Signals (IJCSS), vol. 1, Nº 1, pag. 30-42. 2000.
17. Wong M.L., Leung K.S. "Data Mining using Grammar Based Genetic Programming and Applications", Kluwer Academic Publishers, 2000.
18. Engelbrecht A.P., Rouwhorst S.E., Schoeman L. "A Building Block Approach to Genetic Programming for Rule Discovery", Data Mining: A Heuristic Approach, Abbass, R. Sarkar, C. Newton editors, Idea Group Publishing, 2001.
19. Fisher R.A. "The use of multiple measurements in taxonomic problems", Annals of Eugenics, vol. 7, pag. 179-188, 1936.
20. Rivero D., Rabuñal J., Dorado J., Pazos A., Pedreira N. "Extracting knowledge from databases with Genetic Programming: Iris flower classification problem", Proceedings of CIMCA and IAWTIC, 2003
21. Duch W., Adamczak R., Grabczewski K. "A new methodology of extraction, optimisation and application of crisp and fuzzy logical rules", IEEE Transactions on Neural Networks, vol. 11, nº 2, 2000.
22. Martínez A., Goddard J. "Definición de una red neuronal para clasificación por medio de un programa evolutivo", revista mexicana de ingeniería biomédica, vol. 22, 1, pp. 4-11, 2001.

Numerosity and the Consolidation of Episodic Memory

J.G. Wallace¹, K. Bluff¹

¹ Swinburne University of Technology, P.O. Box 218, Hawthorn, Victoria, Australia 3122.

Abstract.

Consolidation, the process by which episodic and other memories acquire permanence, is currently a particularly significant research topic. The final aspect of consolidation is the process culminating in the fixation of links between neocortical representations. Fixation is determined by interaction between the significance level and frequency of repetition of episodic sequences. The process of assessing frequency, as distinct from setting the frequency criterion, provides the focus of our current work. A Ca^{2+} based signaling mechanism has potential relevance to the design of a model of a temporally extended consolidation process. Data on the transition from egg to embryo suggest the existence of mechanisms capable of extended numerosity detection. A model of frequency detection in consolidation based on the registration of Ca^{2+} transients or waves is presented and illustrated in a loose simulation of embryonic signaling. Its features are consistent with the wide, temporal flexibility required to fit data from animal and human subjects.

1. Introduction

The locus and mode of operation of episodic memory are currently, highly active research topics. As Eichenbaum [1] points out, consolidation, the process by which episodic and other memories acquire permanence, is a particularly significant topic. Empirical evidence indicates that consolidation occurs gradually over days to years and involves two-way interaction between the hippocampal region, including entorhinal cortex (EC), and widespread areas of the neocortex (NC).

Analysis of the consolidation process at the metalevel indicates the essential involvement of at least four elements. They have not attracted equal research attention. Arranging episodic information in appropriate temporal sequences has been the focus of much activity centred on the operation of the hippocampus.

A second, essential process is the matching of segments in episodic sequences to detect repetition and identify consistency in the stream of experience. Matching does not involve the same representation of episodic information throughout the cortical-hippocampal interaction. Consensus favours the use of a more sparse or attenuated representation in the hippocampus.

To conform with the extended time scale of consolidation the matching process must include highly stable memory storage of outcomes. Structural and strengthening

modifications of synaptic connections between neurons is the most obvious possibility. The ability of synaptic plasticity to satisfy the long term stability requirement has been questioned, [2].

The third essential process in consolidation involves the determination of the relative significance or importance of stretches of episodic information. This arises from the sheer volume of episodic memory data to be processed. Among the criteria employed in determining the most significant information are novelty, motivational and emotional importance and current contextual relevance. In our model the stream of episodic information is subjected to a matching process first at the neocortical level and then in entorhinal cortex. In each case, only unmatched episodic information proceeds to the next level as potentially novel. Novel stretches passed from EC to the hippocampus are assessed for motivational and emotional significance through interaction between the dentate gyrus and brain stem [3]. Significant episodic information is finally reviewed for contextual relevance in CA3, [4].

The surviving information enters the fourth and final aspect of consolidation, the process culminating in the fixation of links between neocortical representations.

Fixation is determined by interaction between the significance level and frequency of repetition of episodic sequences. This results in a current setting of the frequency required for progress towards fixation. The frequency criterion is set and modified within the hippocampus and communicated from CA1 to EC with the results of hippocampal processing of episodic sequences. Subsequent matching of the incoming episodic information stream in EC proceeds with updated episodic sequence representation and a revised frequency criterion. Similarly, communication of the results of EC processing to NC produces updating of the episodic sequences and frequency criterion used in the final phase of the fixation process.

The process of assessing frequency, as distinct from setting the frequency criterion, provides the focus of our current work. This is an aspect largely ignored in current research on episodic memory. Our approach is derived from sources far removed from concern with consolidation in episodic memory.

2. Frequency Assessment in Consolidation

Dehaene and Changeux [5] have suggested a parallel accumulator model to account for elementary number processing abilities in humans and animals. The model is designed to cope with both visual and auditory input. Since our focus is on internal processing of sequences and episodic information we will consider how it deals with sequential auditory input. At each moment in time, one of 15 input clusters coding for a given auditory object can be activated. Each cluster simulates several hundred or thousand neurons with common response properties. Each cluster is implemented as a single McCulloch-Pitts sigmoid unit, with an autoexcitatory connection simulating the

various synapses linking individual neurons within each cluster. The output of the unit represents the average number of active neurons within the cluster. These input clusters project to an intermediate layer where further clusters with strong recurrent connections keep a long lasting remnant activity which provides a short term “echoic” memory of recent auditory stimuli. Echoic memory clusters project with equal strength to every unit in an array of 15 clusters with increasing threshold. By responding whenever the total activity exceeds a threshold, these ‘summation clusters’ in effect respond whenever the input numerosity exceeds a certain limit. Finally, summation clusters project to an array of ‘numerosity clusters’. Connection strengths are organized with central excitation and lateral inhibition set to ensure that each numerosity cluster only responds if its corresponding summation cluster is active and others with a higher threshold are inactive.

The model’s performance was simulated on a variety of tasks with input numerosities in the 1-5 range. Although its numerosity detection scheme does not limit its applicability to small numbers of objects, the model was designed to conform to Fechner’s Law with the difficulty of discriminating numerosity increasing with rising numerosity. The results in a same-different task involving comparisons in the 1-5 range indicated close to chance performance for 3 vs. 4 and 4 vs. 5. This is attributable to increases in the variance of the model’s numerosity estimate with a rise in the number of objects involved. The change in performance efficacy at the 3-4 numerosity level is designed to reflect the results of the extensive literature on fast numerosity apprehension or ‘subitizing’ [6].

More recently, Ahmad et al. [7], in contrast to Dehaene and Changeux’s ‘hardwired’ or innate approach, have explored the possibility that ontogenesis could lead to internalized representations of numerical quantities. Subitizing is simulated using unsupervised, self-organising neural networks in a multi-net system. For our purposes the key feature is the magnitude representation system. This takes the form of a Kohonen self-organising map (SOM). Training results indicated that the SOM was capable of discriminating patterns representing numerosities from 1 to 5. Their positions on the SOM were consistent with the distance effect and Fechner’s Law. Once again, the numerosities 1 to 3 are learnt easily but intermediate numerosities in the range above 3 are learned with difficulty.

The Dehaene and Changeux model and subsequent work provide a stimulating starting point in designing a frequency detector mechanism for inclusion in consolidation. It does not, however, contribute to devising an approach consistent with the days to years temporal extension required in a model of consolidation. For this input we have turned to the work of Ducibella et al. [8].

Ca^{2+} oscillations and signaling represent a basic mechanism for controlling many cellular events. This includes the transition from egg to embryo. Activation of mouse eggs, for example, is followed by a temporal series of Ca^{2+} dependent events resulting in the process of cellular differentiation which converts the egg into the zygote. Prior

to Ducibella's work, the relative dependency of individual cellular events on Ca^{2+} oscillation number was unknown. His results identify three categories of events respectively requiring 1-4, 4-8 and 8-24 Ca^{2+} oscillations for their initiation. Once begun, many of the events do not proceed to completion without additional Ca^{2+} transients suggesting that continued stimulation of a Ca^{2+} dependent enzyme(s) activity or signaling pathway is necessary. The Ca^{2+} signaling system does not appear to function as a digital on/off switch in which exceeding a single threshold results in both the initiation and completion of a cellular activity. The extent of the changes display a graded response with respect to the increases in the Ca^{2+} stimulus.

As Ducibella points out, an advantage of using Ca^{2+} as a driver, working periodically in the form of transients, is that the temporal order of cellular events can be specified by a single master regulator. "Since the time between each Ca^{2+} transient cycle does not widely vary, each Ca^{2+} transient encodes a specified period of time. Thus, for cellular events that depend on the level or length of time of a Ca^{2+} dependent enzyme activity, transient number, amplitude, and frequency could play a role in timing the initiation and completion of these events." Given that the electric field pulses used to create the Ca^{2+} transients in the study occurred at the rate of one pulse per eight minutes and that the targeted cellular events extend to eight hours after the commencement of stimulation, a Ca^{2+} based signaling mechanism has potential relevance to the design of a model of a temporally extended consolidation process. The three bandwidths of numerosity discrimination identified in the cellular event responses, also, suggest the existence of mechanisms capable of extended numerosity detection. These ideas are incorporated in our model of frequency detection in consolidation.

3. A Model

In our model no attempt is made to reflect the operations of perception. Segments of sequential episodic records are the basic processing units. Each segment occupies a 'place field' modeled on the place fields composed of place cells in the hippocampus. Although originally regarded as being confined to processing external spatial information place fields are now viewed as being involved in processing a more general range of segmented information. Successful activation of a place field in the hippocampus, EC or NC indicates a matching of a stored episodic segment with a segment of the current stream of episodic information transiting the area. This results in the generation of a Ca^{2+} transient or wave that registers on a local activity accumulator module (LAAM).

Each LAAM consists of a set of N leaky accumulator units (AU) linked in series.

For $n \in \{1 \dots N\}$, if we define the input to the n^{th} AU (AU_n) at time t as,

$$I_n(t) = \begin{cases} \text{external input} & \text{if } n = 1 \\ O_{n-1}(t) & \text{if } n \neq 1 \end{cases}$$

and also define

$$\Phi_n(t) = \alpha a_n(t-1) + I_n(t)$$

where α is a decay constant,

a_n is the accumulated activity level of accumulator AU_n ,
with all AU_n having the same maximum accumulated activity level A

$$\text{and } O_n(t) = \max [\Phi_n(t) - A, 0]$$

$$\text{then } a_n(t) = \min [\Phi_n(t), A]$$

Further, if $\theta \in \{1 \dots N\}$ represents a module trigger parameter then module output is triggered when $a_\theta(t) = A$.

Current research on the signaling roles of Ca^{2+} provides numerous possibilities for implementation of such an accumulator [9]. Particularly promising is the increasing evidence that the functions of the capacity of mitochondria for accumulating Ca^{2+} extend to signalling that modulates the spatio-temporal pattern of cytosolic Ca^{2+} increases, [10]. The locus of mitochondrial involvement in an accumulator mechanism may extend beyond neurons to the interaction between neurons and surrounding astrocytes. Two-way, Ca^{2+} based signaling has already been suggested as a means of maintaining Ca^{2+} records in astrocytes of neural activity relevant to the slow learning of neural connections, [11].

The discrimination capability of each LAAM reflects the results of Dehaene's simulation and research on subitizing in being restricted to the results of processing by 3 AUs ($N = 3$ above). A LAAM can over time register up to 3A units of activity representing variations in accumulated Ca^{2+} . θ determines the number of AUs contributing to the LAAM's activity registration.

Modelling of consolidation does not require the translation of accumulator states into numerosity clusters and, thus, cardinal numbers. The key requirement is to produce a discrimination capability consistent with the extension of the consolidation process over a protracted time period as the results of accumulation transit the hippocampus – EC-NC sequence. Our approach to meeting this requirement, is to introduce linkages between the local activity accumulators associated with place fields representing the same episodic information (although in different representational forms) at each of the three levels. The activity level required for processing of a place field to be transferred from hippocampus to EC is, as already indicated, determined by the significance level assigned to the episodic information located in the field, which sets

the trigger parameter θ . Successful detection of 1, 2 or 3 units of activity as required initiates the registration process in the corresponding local accumulator in EC, setting θ to the same value there. In EC the AUs require three times more activity to respond (from above, if $A = 1$ in the hippocampus it will have the value 3 in EC). Also, in EC the decay half-life for the LAAM is larger. In numerical terms this produces an activity discrimination range of 3, 6 or 9 units for a LAAM in EC. Iteration of the same process accords a discrimination range of 9, 18 or 27 activity units to a LAAM in NC. The increase in the gaps between discriminable units with the move from the hippocampus to EC and from EC to NC offsets the increase in difficulty of discrimination with rising numerosity.

Three linked LAAMs, as described, are sufficient to loosely model the Ca^{2+} oscillations discrimination requirements of the three categories of cellular events identified by Ducibella and extending over eight hours after the commencement of stimulation. Systems of 1, 2 and 3 LAAMs, linked as described above, were subjected to activity pulses at a mean frequency of 4 pulses per hour. Both the inter-pulse interval and the pulse size had Gaussian noise added and 1000 trials were conducted for each configuration with each trigger parameter (θ). The total activity required to cause triggering was recorded. Figure 1 shows the range of activity recorded for each configuration and indicates the discriminability achieved by this model.

The range of temporal data relevant to evaluation of a model of consolidation of episodic memory is complex and highly variable. The main source is studies of amnesia following hippocampal system damage and, in particular, temporally graded retrograde amnesia. The results of studies of humans and a variety of animals are summarized and discussed by Eichenbaum (2002). They support the existence of temporally graded retrograde amnesia and in animals a duration of the hippocampal phase of consolidation ranging from around 8 days for rodents to around 8 weeks for monkeys. The human data are much more complex since they reflect the results of hippocampal and other related forms of accidental brain damage that are extremely difficult to define in detail. Where temporally graded retrograde amnesia can be demonstrated its extent appears to be associated with the anatomical extent of damage within the hippocampal region. Cell loss confined to the CA1 field, for example, produces retrograde amnesia extending 1-2 years while damage throughout the hippocampus and part of the entorhinal cortex extends it back 15-25 years.

Our model and, more specifically, quantification mechanism are compatible with this view of the temporal relationships in consolidation. The need for incorporating temporal flexibility can be met by varying the number of LAAMs linked in sequence and/or adjusting the activity level defining the unit of activation at the basal HC level (A in our model). In addition the time decay parameter can be adjusted to extend or contract the accumulation period available for episodic information in a place field. The asymptotic effect in NC can be achieved by the combination of detailed, specific representation of the ‘old’ information to be matched with the stream of current

experience and the employment of a high recurrence criterion for consolidation delivered via the temporal flexibility mechanisms.

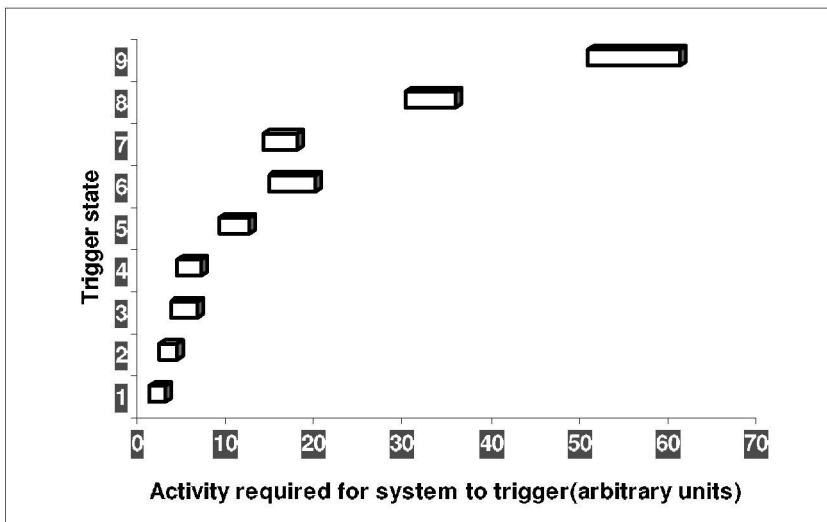


Figure 1: The range (over 1000 trials) of accumulated activity required to trigger system output in a loose simulation of Ducibella's experiment. Trigger states 1 to 3 correspond to a 1-LAAM system (θ values 1 to 3), 4 to 6 to a 2-LAAM system (θ values 1 to 3) and 7 to 9 to a 3-LAAM system (θ values 1 to 3). All LAAMs have $N = 3$ and in the linked multi-LAAM systems each LAAM has A set to 3 times the size of its predecessor. Its decay parameter α is set to give a corresponding decay half-life relationship.

The implausibility of a consolidation process for episodic information extended over such a lengthy time period and with a hippocampal role terminating as information is transferred to EC and later NC has produced alternatives such as multiple trace theory which accords a continuing function to the hippocampus as episodic memories are activated in NC. We favour an approach that recognizes the limited storage capacity of the hippocampus but, also, its architectural suitability for the relatively rapid analysis of episodic information. NC, in contrast, has abundant storage capacity but contains representations that typically change only slowly and incrementally, [12]. As Eichenbaum [1] points out, NC is the location of ongoing interaction between reactivation of existing or 'old' memories and new information emerging from the processing of novel information in the episodic memory chain. Consequently, consolidation should be viewed as a lifelong process that only asymptotically approaches a state where more interaction cannot significantly alter NC representation.

References

1. Eichenbaum, H.: The Cognitive Neuroscience of Memory – an Introduction: Oxford University Press, NY (2002)
2. de Ortiz, S.P., Arshavsky, Y.I. DNA recombination as a possible mechanism in declarative memory: A hypothesis. *Journal of Neuroscience Research*, Vol. 63, Issue 1, 72-81 (2001)
3. Wallace, J.G. and Bluff, K.: The Borderline Between Subsymbolic and Symbolic Processing: A Developmental Cognitive Neuroscience Approach. *Proceedings of the 20th Annual Conference of the Cognitive Science Society 1998*, Lawrence Erlbaum Associates, New Jersey 1108-1112 (1998)
4. Lisman, J.E.: Relating hippocampal circuitry to function: recall of memory sequences by reciprocal dentate-CA3 interactions. *Neuron*, Vol. 22, 233-242 (1999)
5. Dehaene, S., Changeux, J.-P.: Development of elementary numerical abilities: a neuronal model, *Journal of Cognitive Neuroscience* 54, 390-407 (1993)
6. Karmiloff-Smith, A. Beyond Modularity: A Developmental Perspective on Cognitive Science. Cambridge, MA: MIT Press/Bradford Books (1992)
7. Ahmad, K., Casey, M., Bale, T.: Connectionist simulation of quantification skills: in *Connection Science*, Vol. 14, No. 3, 2002, 165-201 (2002)
8. Ducibella, T., Huneau, D., Angelichio, E., Xu, Z., Schultz, R.M., Kopf, G.S., Fissore, R., Madoux, S., Ozil, J-P.: Egg-to-embryo transition is driven by differential responses to Ca²⁺ oscillation number. *Developmental Biology* 250, 280-291 (2002)
9. Cowan, W.M., Sudhof, T.C., Stevens, C.F. (Eds) *Synapses*, Baltimore (2001)
10. Pozzan, T., Rizzuto, R.: The renaissance of mitochondrial calcium transport. *Eur.J. Biochem.* 267, 5269-5273 (2000)
11. Wallace, J.G. and Bluff, K.: Astrocytes and Slow Learning in the Formation of Distal Cortical Associations In Mira, J., Moreno-Diaz, R., Cabestany, J. (Eds.) *Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence* 360-369, Springer (1997)
12. Alvarez P, Squire, L.R. Memory consolidation and the medial temporal lobe: a simple network model. *Proc Natl. Acad Sc USA* 91:7041-7045 (1994)

Rule Extraction from a Multilayer Feedforward Trained Network via Interval Arithmetic Inversion^{*}

Carlos Hernández-Espinosa, Mercedes Fernández-Redondo,
and Mamen Ortiz-Gómez

Universidad Jaume I, Campus de Riu Sec, D. de Ingeniería y Ciencia de los
Computadores 12071 Castellón, Spain
espinosa@icc.uji.es

Abstract. In this paper we propose a new algorithm for rule extraction from a trained Multilayer Feedforward network. The algorithm is based on an interval arithmetic network inversion for particular target outputs. The types of rules extracted are N-dimensional intervals in the input space. We have performed experiments with four databases and the results are very interesting. One rule extracted by the algorithm can cover 86% of the neural network output and in other cases 64 rules cover 100% of the neural network output.

1 Introduction

Neural networks have been applied to a great number of applications and one of the most widely used neural network paradigms is Multilayer Feedforward. However, in same applications it is not only sufficient a correct classification of an input, it is also necessary an explanation of the classification [1]. One example is the medical diagnosis field, in this case, we need to provide a correct classification of the symptoms (the disease) and an explanation of the classification for the doctor. The intelligent systems in this field are conceived as an aid for the doctor and not as a substitution of the doctor. Therefore, they should provide an explanation capability.

A fundamental problem of neural networks is that the information they encode can not be easily understood by humans, for example, it is difficult to give an explanation on how they solve a particular problem.

One of the methods to solve this problem is rule extraction from a trained neural network. With this method, we tray to convert the information contained in a neural network in a set of rules that can be understood by a person.

There are many algorithm for rule extraction [2-8]. They differ in the type of rules extracted and many other characteristics. However, they lack from a common problem, the computational cost of the extraction of rules increases exponentially with the number of parameters in the neural network (weights or neurons). So, it is usually of crucial importance the application of pruning algorithms to reduce the size of the network previously to the application of the rule extraction method.

* This research work was supported by a Spanish CICYT project number TIC2000-1056.

In this paper, we propose a new algorithm for rule extraction from a trained Multilayer Feedforward network based on interval arithmetic. The algorithm is based in a network inversion for a particular target using the interval arithmetic properties. The type of rules extracted are N-dimensional intervals in the input space.

This new algorithm has the problem of an exponential computational cost increase with the number of inputs in the network, but other parameters like the number of weights or hidden units does not affect significantly the computational cost.

The organization of the paper is the following. In section two we describe the interval arithmetic basis, the neural network inversion method and the rule extraction algorithm. In section three we present the experimental results with four databases and finally the conclusions are in section four.

2 Theory

This section is divided in four subsection. The first one reviews the basic properties of interval arithmetic. The second explains how to calculate the output of a neural network for an interval input. Subsection three describes the interval arithmetic inversion algorithm, and finally in the fourth we explain the rule extraction algorithm.

2.1 Interval Arithmetic Basis

First, we will review the basic operations of interval arithmetic used in this paper. They are sum of intervals, multiplication of an interval by a number and the exponential [9].

The sum of two intervals is an interval whose upper limit is the sum of the upper limits of the intervals and whose lower limit is the sum of the lower limits of the intervals. See equation 1.

$$A + B = [a^L, a^U] + [b^L, b^U] = [a^L + b^L, a^U + b^U] \quad (1)$$

Where the superscripts L and U denote the lower and upper limits of the interval.

The second property is the product of a real number by an interval. In this case the final interval depends on the sign of the real number. See equation 2.

$$m \cdot A = m \cdot [a^L, a^U] = \begin{cases} [m \cdot a^L, m \cdot a^U] & \text{if } m \geq 0 \\ [m \cdot a^U, m \cdot a^L] & \text{if } m < 0 \end{cases} \quad (2)$$

Another interesting property is the exponential function of an interval. It is interesting because it is often used in the transfer function of a neural network.

Since the exponential function is monotonically increasing the result is an intervals whose lower limit is the exponential of the lower limit and whose upper limit is the exponential of the upper limit. See equation 3.

$$\exp(A) = \exp([a^L, a^U]) = [\exp(a^L), \exp(a^U)] \quad (3)$$

2.2 Interval Output of Multilayer Feedforward for an Input Interval

With these three basic properties, we can calculate the output of the usual Multilayer Feedforward for an input interval, i.e., in the case we use intervals as the inputs of the neural network.

In this situation the output of the neural network becomes an N-dimensional interval where N is the number of output units.

Also, all the intermediate values of the neural network, like the output of the hidden units, becomes intervals.

The interval outputs of the hidden units can be calculated with equation 4.

$$H_{P,j} = [H_{P,j}^L, H_{P,j}^U] = f(Net_{P,j}) = f([net_{P,j}^L, net_{P,j}^U]) = [f(net_{P,j}^L), f(net_{P,j}^U)]$$

$$\text{where } Net_{P,j} = \sum_{i=1}^{N\text{inputs}} w_{j,i} \cdot I_{P,i} + \theta_j \quad Net_{P,j} = [net_{P,j}^L, net_{P,j}^U]$$

$$\text{where } net_{P,j}^L = \sum_{i=1, w_{j,i} \geq 0}^{N\text{inputs}} w_{j,i} \cdot I_{P,i}^L + \sum_{i=1, w_{j,i} < 0}^{N\text{inputs}} w_{j,i} \cdot I_{P,i}^U + \theta_j \quad (4)$$

$$net_{P,j}^U = \sum_{i=1, w_{j,i} \geq 0}^{N\text{inputs}} w_{j,i} \cdot I_{P,i}^U + \sum_{i=1, w_{j,i} < 0}^{N\text{inputs}} w_{j,i} \cdot I_{P,i}^L + \theta_j$$

Where f is the standard sigmoid function, $I_{P,i} = [I_{P,i}^L, I_{P,i}^U]$ is the input interval and $H_{P,j} = [H_{P,j}^L, H_{P,j}^U]$ are the output intervals of the hidden units.

In equation 4, we have to distinguish between positive and negative weights because of the same distinction in the property of multiplication of a real number by an interval.

Analogously for the interval outputs of the neural network we have the equation 5.

$$O_{P,k} = [O_{P,k}^L, O_{P,k}^U] = f(Net_{P,k}) = f([net_{P,k}^L, net_{P,k}^U])$$

$$\text{where } net_{P,k}^L = \sum_{j=1, w_{k,j} \geq 0}^{N\text{hidden}} w_{k,j} \cdot H_{P,j}^L + \sum_{j=1, w_{k,j} < 0}^{N\text{hidden}} w_{k,j} \cdot H_{P,j}^U + \xi_k \quad (5)$$

$$\text{and } net_{P,k}^U = \sum_{j=1, w_{k,j} \geq 0}^{N\text{hidden}} w_{k,j} \cdot H_{P,j}^U + \sum_{j=1, w_{k,j} < 0}^{N\text{hidden}} w_{k,j} \cdot H_{P,j}^L + \xi_k$$

Where f is the standard sigmoidal function and $O_{P,k} = [O_{P,k}^L, O_{P,k}^U]$ is the interval output of the neural network.

With these equations we can calculate the transformation of an interval in the inputs into an interval at the outputs across the neural network structure.

2.3 Network Interval Arithmetic Inversion

The aim of a neural network inversion algorithm is to fix a particular output of the network and obtain an input or set of inputs whose output is the previously fixed output.

In the case of interval arithmetic inversion the objective is the same, but in this case the fixed output is an interval and obviously the obtained input will be an interval.

The algorithm for interval arithmetic inversion is basically the same algorithm of neural network inversion [10], but in this case, the target will be an interval vector and the error function will be the one of equation 6.

$$E_p = \frac{1}{4} \sum_{k=1}^{\#Noutput} \left\{ (t_{p,k}^U - o_{p,k}^U)^2 + (t_{p,k}^L - o_{p,k}^L)^2 \right\} \quad (6)$$

After we fix an interval output, the inversion is accomplished by selecting and initial interval vector as the initial input $\{[i^L_1(0), i^U_1(0)], [i^L_2(0), i^U_2(0)], \dots, [i^L_N(0), i^U_N(0)]\}$ and applying an iterative gradient descent algorithm similar to Backpropagation that will minimize the error value by changing the initial input. The equations are basically in 7.

The process is iterative, we calculate the interval output of the neural network for the initial interval input. After that, we apply equations 7 and we obtain a new input interval. With this new input interval we can calculate again the output interval and iterate in this way the process.

$$i_{P,k}^L(n) = i_{P,k}^L(-1) - \eta \frac{\partial Error}{\partial i_{P,k}^L} \quad ; \quad i_{P,k}^U(n) = i_{P,k}^U(n-1) - \eta \frac{\partial Error}{\partial i_{P,k}^U} \quad (7)$$

The values of the partial derivates of equation 7 are in equation 8 and 9:

$$\begin{aligned} \frac{\partial Error}{\partial i_{P,k}^L} &= -\frac{1}{2} \sum_{k=1}^{\#Noutput} \left(t_{p,k}^L - o_{p,k}^L \right) o_{p,k}^L (1 - o_{p,k}^L) \sum_i^{w_{k,i} \geq 0, w_{i,l} \geq 0} w_{k,i} \cdot H_{P,i}^L \cdot (1 - H_{P,i}^L) \cdot w_{i,l} + \\ &\quad \sum_i^{w_{k,i} < 0, w_{i,l} < 0} w_{k,i} \cdot H_{P,i}^U \cdot (1 - H_{P,i}^U) \cdot w_{i,l} + \sum_{k=1}^{\#Noutput} (t_{p,k}^U - o_{p,k}^U) \cdot o_{p,k}^U \cdot (1 - o_{p,k}^U) \cdot \\ &\quad \sum_i^{w_{k,i} \geq 0, w_{i,l} < 0} w_{k,i} \cdot H_{P,i}^U \cdot (1 - H_{P,i}^U) \cdot w_{i,l} + \sum_i^{w_{k,i} < 0, w_{i,l} \geq 0} w_{k,i} \cdot H_{P,i}^L \cdot (1 - H_{P,i}^L) \cdot w_{i,l} \end{aligned} \quad (8)$$

$$\begin{aligned} \frac{\partial Error}{\partial i_{P,k}^U} &= -\frac{1}{2} \sum_{k=1}^{\#Noutput} \left(t_{p,k}^U - o_{p,k}^U \right) o_{p,k}^U (1 - o_{p,k}^U) \sum_i^{w_{k,i} \geq 0, w_{i,l} < 0} w_{k,i} \cdot H_{P,i}^L \cdot (1 - H_{P,i}^L) \cdot w_{i,l} + \\ &\quad \sum_i^{w_{k,i} < 0, w_{i,l} > 0} w_{k,i} \cdot H_{P,i}^U \cdot (1 - H_{P,i}^U) \cdot w_{i,l} + \sum_{k=1}^{\#Noutput} (t_{p,k}^L - o_{p,k}^L) \cdot o_{p,k}^U \cdot (1 - o_{p,k}^U) \cdot \\ &\quad \sum_i^{w_{k,i} \geq 0, w_{i,l} > 0} w_{k,i} \cdot H_{P,i}^U \cdot (1 - H_{P,i}^U) \cdot w_{i,l} + \sum_i^{w_{k,i} < 0, w_{i,l} \leq 0} w_{k,i} \cdot H_{P,i}^L \cdot (1 - H_{P,i}^L) \cdot w_{i,l} \end{aligned} \quad (9)$$

2.4 Rule Extraction Algorithm

The type of rules we want to obtain are N-dimensional intervals in the input space like the following:

If $x_1 \subset [a_1^L, a_1^U]$, $x_2 \subset [a_2^L, a_2^U]$, ..., $x_N \subset [a_N^L, a_N^U]$ then $\{x_1, x_2, \dots, x_N\} \in \text{Class K}$.

If the input is contained in the N-dimensional interval $[a_i^L, a_i^U]$ the output is a particular class (class K in the example).

We should obtain the limits of the intervals a_i , b_i . They limit a N-dimensional interval in the input space and the whole N-dimensional interval has to be included in a classification class. An interval neural network inversion is used to get the intervals.

In order to obtain a rule, first, we will select a target value of the following type for one classification class (for example, class number 2): $\{[0,0.5]_{\text{Class1}}, [0.5,1.0]_{\text{Class2}}, [0,0.5]_{\text{Class3}}, \dots, [0,0.5]_{\text{ClassN}}\}$. An output vector inside the above interval suppose a correct classification inside the class number 2 because neuron number 2 is activated and the rest neurons are not.

Second, we will apply the inversion algorithm for the target. We expect that the initial input interval will evolve to give an interval whose output is inside the target interval selected, in this case, the final input interval will correspond to a valid rule.

We have performed simulations with three types of initial intervals in several two dimensional examples. And the conclusion is that if the initial interval is a point correctly classified by the target interval, during the inversion, the point will expand to an interval, the final output limits of the interval input will generally touch the borders of the classification class and the final results will normally correspond to a valid rule. We can see several examples in Fig. 1.

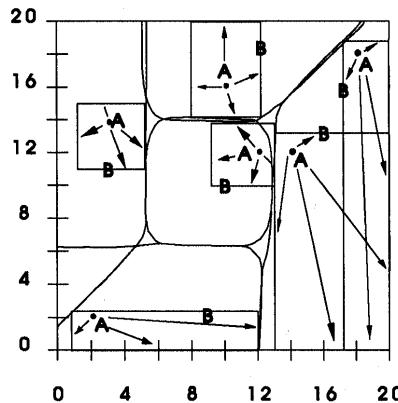


Fig. 1. Example of point expansion by interval arithmetic inversion. A is the initial point, B is the final interval.

It is obvious that we can exploit this behavior of this type of initial intervals in order to propose an algorithm, which can convert the information contained in the neural network into rules. The algorithm can be resumed as follows:

- (a) Select an initial point and calculate the output of the neural network for this input.
- (b) Select a target of the type described above, this target should agree with the classification class of the output of the neural network for the initial point.
- (c) Apply the inversion algorithm and extract a rule.
- (d) Select a new point which is not included in the rules we have obtained before, and calculate the output of the neural network for this new point.
- (e) Select a target which agrees with the output of the neural network for this new point.

- (f) Apply the inversion algorithm and extract a new rule.
- (g) If we have not cover the whole input space go to step d).

In order to test whether we have covered the input space and select a new initial point we can scan the input space with equally spaced points and test if the points are included in the rules. The space between the points will also influence the final accuracy of the set of rules.

The problem of this scanning method is that its yields a computational complexity increase with the increase of the number of inputs and it can not be used with a high number of inputs. The methods of input or feature selection will play an important role to apply this algorithm [11].

There are other specific characteristics of this method. A rule will always have an output interval inside the target described above, so there will not be incorrect classification of points by the rule. Also another consequence is that there will not be overlapping among rules of different classes. And finally, the set of rules will not usually cover the total space of the neural network input, for example, the output $\{0.1, 0.8, 0.7\}$ is not in the initial target intervals because two output units are activated and if an input has this output it will not be covered. We can say that the points where the classification of the neural network is not clear are not covered by the rules.

3 Experimental Results

We have tested the neural network rule extraction algorithm with four database from the UCI repository of machine learning databases. The databases are Balance Scale (BALANCE), Liver Disorders (BUPA) and two of the Monk's Problems (MONK1 and MONK 2). We have selected these databases because the input dimensionality is at most six (<http://www.ics.uci.edu/~mlearn/MLRepository.html>).

We have applied the rule extraction algorithm to ten networks for each database which were trained with different random initialization of weights and different partition of data among training, cross-validation and test.

In a rule extraction algorithm of this type we think that the two most important criterion for the results are the fidelity of the rules and the number of rules extracted. By fidelity of rules, we should understand how the results reproduced the behavior of the neural network.

In table 1 we have the results for the four databases.

The second column (Prec. Space) is the distance in the input space between the points generated to construct the rules. A lower number means a higher number of points and therefore a higher number of rules.

We have randomly generate 10.000 point inside the input space with the condition that only one output unit is activated. The third column (Percentage) is the mean percentage of point covered by the rules and the fourth column (Not Cover) the mean percentage of points which were not cover for the ten networks of each database.

Columns five and six are two mean percentages, in this case we have generated randomly 10.000 points without restriction inside the input space. The fifth column (Total Percentage) is the mean percentage of points cover by the rules and the sixth column (Total Not Cover) is the mean percentage of points not covered by the rules.

The seventh column (Number of Rules, Nrule) is the mean number of rules generated by the algorithm for the ten networks of each database. Column number eight is the minimum number of rules generated for a network and column number nine is the percentage of covering of this minimum number of rules.

Table 1. Results of the rule extraction algorithm

Database	Prec. Space	Percent- age	Not Cover	Total Percentag e	Total Not Cover	Number of Rules (Nrule)	Nrule minimum	Percentag e Nrule minimum
BALANCE	0.2	74.15	25.84	65.57	34.42	331.3	323	74.96
BALANCE	0.13	83.14	16.86	73.75	26.25	745.9	671	82.04
BUPA	0.2	92.59	7.41	92.59	7.41	6357.1	64	100
BUPA	0.13	85.11	14.89	85.11	14.89	19289.5	1	89.27
MONK1	0.2	81.10	18.90	81.10	18.90	8899.5	5036	82.44
MONK1	0.13	81.31	18.69	81.31	18.69	47211.4	35842	82.56
MONK2	0.2	93.21	6.79	93.21	6.79	7288	64	100
MONK2	0.13	83.16	16.84	82.69	17.31	12943.1	1	87.57

We can see that, in general, the number of rules increases with the precision of the scanning of the input space as it was expected. We can see very interesting results in the minimum number of rules, for example for the databases BUPA and MONK2 sixty four rules are enough to completely cover the input space. Also, in BUPA and MONK2 one rule cover more than 85% of the input space.

As we commented before, the rules will only cover the input space where only one output unit is activated and the rest are not. We can evaluate the maximum percentage of points not covered by subtracting the columns “Percentage” and “Total Percentage”, they are 8.58% for the database BALANCE, 0% for BUPA, 0% for MONK1 and 0.47% for MONK2. As we can see this effect is not so important in the experimental results.

The results of mean percentage of covering are in general good. But if we want to increase the percentage of covering we can generate random points outside the covering of the rules and extract new rules from this points. In a rule the initial point is usually covered by the rule (see Fig. 1), therefore the new rule will usually cover part of the input space not cover by the rest of the rules (the initial point is not contained in any rule).

We have applied this technique with the database BALANCE which got the lower percentage of covering and the results are in Table 2, using 5000 new points.

As we can see the mean percentage of covering by the rules (column three) increases from 74.15 to 95.15, so this is a good technique to increase the performance of our rule system.

Table 2. Results of the rule extraction algorithm

Database	Number of Points	Percent- age	Not Cover	Total Percentag e	Total Not Cover	Number of Rules (Nrule)	Nrule minimum	Percentag e Nrule minimum
BALANCE	5.000	95.19	4.81	85.45	14.54	2963	2706	95.87

4 Conclusions

We have presented a new algorithm for rule extraction from a trained Multilayer Feedforward neural network. The algorithm is based on an interval arithmetic network inversion for particular interval target outputs. The type of rules extracted are N-dimensional intervals in the input space. The experimental results are encouraging, one rule extracted by the algorithm can cover 86% of the input space of the neural network, and in other cases 64 rules cover 100% of the neural network.

References

- [1] A. Maren, C. Harston y R. Pap, *Handbook of Neural Computing Applications*, Academic Press Inc., 1990.
- [2] Lu, H., Setiono, R., Liu, H., "Effective Data Mining Using Neural Networks", *IEEE Trans. on Knowledge and Data Engineering*, vol. 8, no. 6, pp.957-961, 1996.
- [3] Thrun, S., "Extracting Rules from Artificial Neural Networks with Distributed Representations", *Advances in Neural Information Processing Systems 7*, pp. 505-512, 1995.
- [4] Gupta, A., Lam, S.M., "Generalized Analytic Rule Extraction for Feedforward Neural Networks", *IEEE Trans. on Knowledge and Data Engineering*, vol. 11, no. 6, pp. 985-991, 1999.
- [5] Narazaki, H., Shigaki, I., Watanabe, T., "A Method for Extracting Approximate Rules from Neural Network", *Proc. of the IEEE Int. Conf. on Fuzzy Systems*, vol. 4, pp. 1865-1870, 1995.
- [6] Palade, V., Neagu, D.C., Puscasu, G., "Rule extraction from neural networks by interval propagation", *Fourth Int. Conf. on Knowledge-Based Intelligent Engineering Systems and Alllied Technologies*, pp. 217-220, 2000.
- [7] Greczy, P., Usui, S., "Rule extraction from trained artificial neural networks", *Behaviormetrika*, vol. 26, no. 1, pp. 89-106, 1999.
- [8] Taha, I.A., Ghosh, J., "Symbolic interpretation of artificial neural networks", *IEEE Trans. on Knowledge and Data Engineering*, vol. 11, no. 3, pp. 448-463, 1999.
- [9] Alefeld, G., Herzberger, J., *Introduction to Interval Computations*, Academic Press, New York, 1983.
- [10] Linden, A. and Kinderman, J., "Inversion of Multilayer Nets", in *Proc. of the Int. Conf. on Neural Networks*, Washington D.C., vol. 2, pp. 425-30, 1989.
- [11] Fernandez, M., Hernandez, C., "Analysis of Input Selection Methods for Multilayer Feedforward", *Journal Neural Network World*, vol. 10, no. 3, pp. 389-406, 2000.

Necessary First-Person Axioms of Neuroconsciousness

Igor Aleksander and Barry Dunmall

Intelligent and Interactive Systems Group, Imperial College, London SW7 2BT, UK
{i.aleksander,b.dunmall}@imperial.ac.uk

Abstract. A new approach that identifies necessary design requirements for neural machinery that one could claim is conscious is presented. Usually such designs stem from a third-person inference of suitable mechanisms which are thought to underpin conscious behaviour. The novelty here comes from using the first person (introspection) to identify a set of necessary axiomatic principles that the neuro-machinery of an agent must deliver and turn these into architectural design requirements for a conscious system. It also leads to the major conclusion that a full description of the neural mechanisms is also a full description of what it is to be conscious.

1 Introduction

This paper relates to a formal statement of the mechanisms that are thought minimally necessary to underpin consciousness. This is expressed in the form of axioms which stem directly from an inspection of first person sensations. As usual, axioms are ways of making formal statements of intuitive beliefs and looking, again formally, at the consequences of such beliefs.

The extended scope of this approach is to lay down some essential properties that should be considered when designing machines that could be said to be conscious. The approach is meant to be open-ended so that others can build further axiomatic clarifications that address the very large number of questions which, in the search for a formal basis for consciousness, still remain to be answered. The outcome has been positive as it formalises an intuition about those objects which could, potentially, make an agent conscious. So if some object does not possess some of the major mechanisms indicated by the minimal set it is unlikely that it could be considered to be conscious. For example, a computer *per se* has no mechanisms that are indicated by the axioms and is therefore not conscious. But a virtual machine hosted by an unconscious computer that does obey the axioms might be potentially conscious of, say, a virtual world for which the axioms apply.

2 Scope of the paper

A set of axioms is proposed which aim to define a minimal set of *necessary* material conditions for cognitive mechanisms that would support consciousness in an agent. For the purpose of this paper ‘Agent’ is used in the sense of an active object that can sense its environment, have a purpose, plan according to that purpose and then choose to act purposefully. An agent can be biological, non-biological and, indeed, virtual

(i.e. software running on a host computer). However, casting this material in the domain of physically realisable robot-like neuro machinery, is exploited as a hook on which to hang the more general axiomatic theory. The axioms are about perception, imagination, contemplation, planning, acting and emotions. Hypotheses and predictions that arise from these axioms are examined in the light of findings in biological systems. It will be seen that these indicate that, in order to support consciousness, mechanisms need to be cellular and influenced not only by external perceptual events, but also by the actions needed to gather and organise the perceptual sensations. The collation of axioms and corollaries leads to a final hypothesis about what objects may be said to be conscious. Also, as this theory introduces a variant form of functionalism, the paper concludes with a consideration of some philosophical issues that might be perturbed by this approach.

3 Discerning consciousness in an agent

Given are two very similar artificially built agents (A and B) that perform a set of tasks competently. Say that someone claims that one of these is conscious (A) while the other (B) is pre-programmed using conventional computing techniques to cover all eventualities that are likely to be encountered in the performance of the task. Central to our theory is a question we simply call Q.

Q: Is there a set of tests that will substantiate the claim that (A) performs consciously while (B) does not?

To answer this even a rudimentary working definition D of ‘being conscious’ makes the point.

D: “Being conscious is defined by having a private sense: of an ‘out-there’ world , of a self , of contemplative planning and of the determination of whether, when and how to act.”

There appear to be two major approaches for tackling Q: behavioural tests and tests based on an introspective knowledge of consciousness . We discard behavioural tests as it has already been said that the two agents perform roughly the same activity. It is also generally accepted that while consciousness may be attributed to agents with an impressive behaviour it cannot be unequivocally inferred from such behaviour.

Here we take the view that it is possible to start with an internal, first-person set of axioms that can be translated into necessary mechanisms that, as usual, are viewed as third person objects.

4 The Axiomatic Approach

Let A be an agent in a sensorily-accessible world S . For A to be conscious of S it is necessary that five axioms to be described below hold.

4.1 Depiction

Axiom 1 : I feel to be in the centre of a world ‘out there’ therefore A has *perceptual states* that *depict* parts of S .

We shall pursue the notion that S is a conjunction of minimally perceptible events. This is familiar in vision, for example, where it is possible to build a visual world out of dots as is done in newspapers or television. We further assume that there is an ‘energy’ involved in any event that can be perceived. In general terms we define a minimally perceptible event δSj as a minimal element of the sensory world S which, were it to have less energy, would not be perceived at all. That is, δSj indicates the minimum event in the world of which an organism can become conscious. We then write S as:

$$S = \bigcup_j (\delta Sj)$$

where \bigcup_j is an assembly (or, in logic, a conjunction) over all values of j , that is of all the minimal events of which the organism could become conscious. Now we assume that just one minimally perceptible element in S has just changed. In order for this to be perceived, some functional support in A has to change. We define this by means of a corollary related to Axiom 1.

COROLLARY 1.1: To be conscious of δSj , it is necessary for A to have a functional support, say

$$N(\delta Sj).$$

By functional support we mean that the state of some part of the machinery of A must have changed. In other words, for an organism to perceive a minimal event in the world requires a related physical state of some state variables of that organism. In neural systems $N(\delta Sj)$ is the state of firing of neuronal groups. In the abstract sense $N(\delta Sj)$ is a unique set of values of a number of state variables related to that event. For a different event, that is, a different j , a different set of state variables would be affected.

On notation: there could be conditions under which the state variables which support $N(\delta Sj)$ may fail to hold the world-related state. That is, the mechanisms which are necessary for being conscious may not be operating in a conscious mode. In this paper this applies to all cases where the notation $N(x)$ is used.

Assuming a physical world of three spatial dimensions, j has three spatial coordinates (x,y,z) . It is assumed that $(x,y,z)=(0,0,0)$ is the ‘point of view’ of the organism (or, in more philosophical terms, the location of the observing self), giving all other world events ‘out there’ measurements at some distance from this point of view. Clearly j needs to be encoded in $N(\delta Sj)$ in order to retain ‘out-thereness’ in the inner functional support and this leads to the next corollary.

COROLLARY 1.2: To achieve $N(\delta Sj)$ it is necessary for A to have a functional support for the measurement of j . We call this measurement the ‘ j referent’.

In living organisms j is often provided by motor action such as the movement of the eyes when foveating on a minimal visual event, or, in some animals, the ability to locate the source of an odour through head movement. In human audition, phase

differences and the shape of the ear locate sound. The general point made in corollary 1.2 is that the encoding of the ‘where’ of a minimal event is as necessary as the qualitative encoding of the sensory event itself in order to give this encoding uniqueness. This leads to an important hypothesis.

HYPOTHESIS 1: *It is sufficient* for each state variable to be indexed by j for the grouping into $N(\delta Sj)$ to be uniquely identified with δSj , irrespectively of the physical location of the variables.

In other words the minimal event is fully encoded internally by the indexing of its state variables irrespectively of their physical location. This hypothesis has been discussed elsewhere [1]. It resolves the ‘binding’ problem by eliminating it. So in the primate visual system this suggests that neurons in disparate areas such as V3, V4 and V5 may combine to support a single minimal conscious event (e.g. the perception of a small red diamond moving in some direction) with no binding other than the j – indexing of specific neurons. This also leads to a prediction

PREDICTION 1.1: j -indexed neurons will be found in living conscious organisms.

This prediction should be seen in retrospect as motor-indexed neurons have actually been first found in primates since 1989 [2] later confirmed by many others. In fact it is quite remarkable how widely spread such discoveries are in the brain.

Given all this it is now possible to define $N(S)$, a *depiction* of S , as the functional support required to be conscious of S , where

$$N(S) = \bigcup_j N(\delta Sj)$$

This suggests that all the values of j can be gathered in parallel which somewhat contradicts the practicality of generating values of j through muscular action (such as eye movement). So in living systems, during a perceptual act, internally, j has a sequential nature $j(t)$, which makes it necessary for $N(\delta Sj(t))$ to have persistence in time (see: Attention, below). In summary, the importance of Axiom 1 is that it suggests a method (the depiction) of internal support for the sensation of an ‘out-there’ world. It is noted that this strongly suggests a cellularity that determines minimally perceivable events. The support indexes cells to encode the ‘out-there’ coordinates of minimal events and needs to be compositional through cell interaction, to match the composition of non-minimal events in the world from minimal ones.

4.2 Imagination

Axiom 2: I can recall the ‘out there world’ as it appeared to me, therefore A has internal *imaginational states* that *recall* parts of S or *fabricate* S -like sensations. Axiom 2 comes from the fact that any organism that is conscious must be conscious of imagined as well as perceived events. Here follow some implied mechanisms that support imagination.

COROLLARY 2.1: The state of a set of cells which supports conscious sensation has necessarily at least two components

$$N(C) = N(S), N(I),$$

where $N(I)$ is a j -referenced set of distributed cell states of a dynamic system to which significant states of $N(S)$ are transferred (see corollary 2.2) becoming *attractors* in this set of variables. Attractors in $N(I)$, in psychological language, would be called recalled memories of past experience, and, $N(I)$ in neurological language is a neural memory site. So the process of acquiring experience that may be ‘replayed internally’ as indicated in the above corollary is a sequence of collective cell states with input $N(S)$ and state $N(I)$. It is suggested that if both are j -indexed they will automatically overlap in consciousness. The next corollary suggests one way in which this transfer may be achieved.

COROLLARY 2.2: One possible way for $N(I)$ to inherit the j -referent is by transfers from $N(S)$ to $N(I)$ through *Iconic learning*

Iconic learning in a cellular or neural system is the process whereby the states of a network copy the pattern on the input variables of the machine while learning is taking place. Learning is the process whereby each cell associates with its own current state with the inputs derived from the outputs of other cells in the system and external inputs (see, ‘iconic learning’, p. 122 in [3]).

We note that what we have described is a system of cells each of which determines its state from external input and the states of other cells, that is, a *state machine*.

COROLLARY 2.3: Time and sequentiality are the properties of any state machine, therefore $N(I)$ can be the imagination of a sequence.

COROLLARY 2.4: $N(S)$ is not the only input to the state machine. Other inputs could be internal emotional E (see ‘Emotion’), internal sensations H (such as pain in living organisms) or verbal input V (in human systems). In very general terms then the state equation for Imagination becomes

$$N(I)' = N(I) \times N(S) \times E \times H \times V \times \dots$$

where $N(I)'$ is the next internal state to $N(I)$ and ‘ \times ’ indicates a Cartesian product, that is a combined influence of E , H , V ... This means that imagination sequences may be initiated from a variety of inputs. For example, a sensory state that indicates that a problem has to be solved (a ball is rolling off a table and has to be caught), a state of hunger that needs to be satisfied, or a verbal request has to be understood and acted upon. In summary, the concept of imagination that is thought to be essential for an agent to be conscious, has been linked to the necessity for a state-machine structure whose states inherit the ‘out-thereness’ afforded by j -referencing at its controlling inputs.

4.3 Attention

Axiom 3: I attend to the out-there world selectively therefore A is capable of selecting which parts of S to depict or what to imagine.

Attention is necessary whenever the build-up of $N(S)$ from S cannot be done in parallel. It addresses the question of what generates the j -referent. It is a search

process made necessary by any restriction in the degree of parallelism which is available at the input of a system. It is initially driven by agent actions. These actions create the j -referent that enable $N(S)$ to be internally reconstituted. It applies to $N(I)$ as well as a way of bringing detail into consciousness. That is, there is a $N(\delta I j)$ extracted from $N(I)$ which of necessity parallels the $N(\delta S j)$ related to $N(S)$.

COROLLARY 3.1: Attention is a search process made necessary by any restriction in the capacity of a sensory input channel. It is initially driven by agent actions to make depiction as complete as possible. These actions create the j -referent that enable $N(S)$ to be internally reconstituted.

The best known example of external attention (i.e. applied to $N(S)$) is eye movement and saccade. This draws attention to the fact that j -referencing can take place for a group of minimal events simultaneously (i.e. whatever is fixated for one eye position).

COROLLARY 3.2: Attention is a search process that can be context independent or context dependent on S or $N(S)$ or $N(I)$.

COROLLARY 3.3: Attention with respect to $N(I)$, attention is the inverse process to depictive construction.

Again in primate vision, these three modes can be observed. The neurological system that involves the frontal eye fields and the superior colliculus is the evidence that all three modes indicated by corollaries 3.1, 3.2, 3.3 are possible for perception. This is the system which acts directly on eye muscles, receives input directly from the retina as well as higher visual areas. It moves the eye to areas of high change in the retinal image, or, when it receives inputs from higher areas so it can move the eyes to where important information is predicted to occur.

HYPOTHESIS 2: Inner attention uses the j -referent mechanisms that generate $N(S)$ to select parts of $N(I)$.

In other words, selection mechanisms which drive action in external attention (such as the activation of gaze locked cells in vision) are at work when attending to imagined sensations. They (i.e. the j -referent) may be a crucial indexing set of signals which position elements of $N(I)$ in the world even when the world is not being perceived. Actuators may or may not be stimulated. If they are not, a clear vetoing mechanism needs to be found.

4.4 Planning

Axiom 4: I can plan for the future by imagining the consequences of my actions, therefore A has means of control over imaginational state sequences *to plan actions*.

So far $N(I)$ has been described as a passive recall process, possibly indexed by some sort of need. Here we see the same state machine mechanisms working against needs that eventually lead to motion or other forms of action on the part of the agent.

DEFINITION 4.1: Planning is the traversing of trajectories in $N(I)$ under the input control of a need, or a desire or, no input at all but, in all cases leading to the selection of one among several possible outward actions.

COROLLARY 4.1: During exploration, action results in j -referents, to have planning, it is necessary for the j -referents linked to state trajectories in $N(I)$ also to link to potential action sequences.

The implication of this definition and corollary is that a trajectory in $N(I)$ is a depiction of ‘if I do Y will it result in X?’ where X is a need or a desire and Y is a set of j -referents on the input of the state machine with the $N(I)$ states. However this allows that X need not be fully defined, leading to something that could be called internal brainstorming. The importance of the role of the j -referent is stressed, as 4.1 closes the loop between exploratory mechanisms, the accumulation of experience, and the use of this experience in generating actions. The need for ‘gating’ (see ‘emotions’, below) is also highlighted, in the sense that plans need to be ‘approved’ before they lead to action, as will be seen in the next section. This gating has been postulated in primate neurology in cortico-thalamic loops that are ‘vetoed’ by the action of the anterior cingulate areas [4]. In summary, planning is the primary task for the $N(I)$ state machine, and good planning is a property which aids evolutionary survival.

4.5 Emotion

Axiom 5: When I plan, emotions guide me to find a good plan therefore A has additional *affective states* that evaluate planned actions and determine the ensuing action .

As agreed by many commentators (e.g. Freeman, [5]), emotion is not just a passive change of internal state in response to perceptual events. It is closely linked to the perception - planning - action link. Plans need to be evaluated, at times very quickly. This implies a mapping from a prediction in $N(I)$. Indeed, to recognise that emotion may be constantly active in consciousness, the following corollary extends corollary 2.1 as follows.

COROLLARY 5.1: The total support of consciousness $N(C)$ contains an emotional element $N(E)$ where this is an evaluation of the states of $N(I)$. That is,

$$N(C) = N(S), N(I), N(E)$$

Where $N(E) = f_e N(I)$
 f_e being an evaluative function operating on $N(I)$.

We note that f_e may be both inborn (say, fear and pleasure) or more refined by being acquired through experience (e.g. envy, appreciation of beauty).

5 The concluding hypothesis and its implication

The rudimentary definition D given at the beginning of this paper, uses the words ‘having a private sense of ...’. The five axioms, driven by first-person arguments and their corollaries have suggested the necessity of specific mechanisms without which an agent’s consciousness would be incomplete or not existing at all. The axioms all present necessary conditions while it could be argued that they are not sufficient. The key lack of sufficiency lies (as in the argument, by David Chalmers [6] for example) that an additional factor , say factor F , is required to map from $N(C)$ into \mathcal{C} , the *sensation of being conscious*. In our terminology:

$$\mathcal{C} = F(N(C)),$$

where F is a mapping which (some argue, [7]) cannot be defined using known science. In contrast with this, the axiomatic approach suggests the following hypothesis

HYPOTHESIS 3: For every event in \mathcal{C} there is a corresponding event in $N(C)$. That is, in the formulation

$$\mathcal{C} = F(N(C)),$$

the mapping F is one to one making \mathcal{C} a complete description of $N(C)$ and $N(C)$ a complete description of \mathcal{C} . The implication of this is that in order to have a complete theory of consciousness \mathcal{C} it is sufficient to have a complete theory of $N(C)$ where the latter may be derived from introspective arguments.

Of course, it is stressed that the five axioms and their consequent mechanisms are what we consider to be the minimally *necessary* elements for an agent to be conscious. This opens the way for the application of this introspective methodology to a sequence of further axioms that may lead to *sufficiency*.

References

1. Aleksander, I. and Dunmall, B.: An extention to the hypothesis of the asynchrony of visual consciousness. Proc. R. Soc. Lond B, **267** (2000) 197-200
2. Galletti, C. and Battaglini, P. Gaze-Dependent Visual Neurons in Area V3A of Monkey Prestriate Cortex. Journal of Neuroscience, **6**, 1112-1125 (1989),
2. Aleksander, I.: Impossible Minds: My Neurons, My Consciousness, IC Press London (1996)
3. Cotterell, R.: The Enchanted Loom. Routledge London (1998)
4. Freeman, W. J.: How Brains Make Up Their Minds: Weidenfeld and Nicolson London (1999)
5. Chalmers, D. J.: The Conscious Mind: In Search of a Fundamental Theory, (Oxford Universiy Press (1996)
6. Nagel, T.: What is it like to be a bat? Philosophical Revue, (1974) **83** 435-50.

An Agent Based Approach of Collective Foraging

Marian Gheorghe^{1*}, Carlos Martín-Vide², Victor Mitrana^{3**}, and Mario J. Pérez Jiménez^{4 ***}

¹ Department of Computer Science, University of Sheffield,
S1 4DP, Sheffield, United Kingdom.

m.gheorghe@dcs.shef.ac.uk

² Research Group in Mathematical Linguistics, Rovira i Virgili University
Pça. Imperial Tàrraco 1, 43005 Tarragona, Spain.

cmv@correu.urv.es

³ Faculty of Mathematics and Computer Science, University of Bucharest
Str. Academiei 14, 70109, Bucharest, Romania,

vmi@f11.urv.es

⁴ Dpto. Ciencias de la Computación e Inteligencia Artificial
University of Sevilla, Spain.

Mario.Perez@cs.us.es

Abstract. In this paper the behaviour of a bee colony is modeled as a society of communicating agents acting in parallel and synchronizing their behaviour. Two computational models for defining the agents behaviour are introduced and compared and tools developed for these models are briefly illustrated.

1 Introduction

An agent is an encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives [11]. There are two fundamental concepts associated with any dynamic or reactive system, such as an agent, that is situated in and reacting with some environment [9]:

- the environment itself must be defined in some precise, mathematical way, something which involves identifying the important aspects of the environment and the way in which they may change in accordance with the activities of the agent;
- the agent will be responding to environmental changes by changing its basic parameters and possibly affecting the environment as well. Thus, there are two ways in which the agent reacts, i.e. it undergoes internal changes and it produces outputs that affect the environment.

* Work supported by EPSRC grant GR/R84221/01

** Present address: Ramón y Cajal Position at Research Group in Mathematical Linguistics, Rovira i Virgili University, Pça. Imperial Tàrraco 1, 43005 Tarragona, Spain.

*** Work supported by the project TIC2002-04220-C03-01 of the Ministerio de Ciencia y Tecnología of Spain, cofinanced by FEDER funds

Agents, as highly dynamic systems, are concerned with three essential factors:

- a set of appropriate environmental stimuli or inputs;
- a set of internal states of the agent;
- a rule that relates the two above and determines what the agent state will change to if a particular input arrives while the agent is in a particular state.

One of the challenges that emerge in intelligent agent engineering is to develop agent models and agent implementations that are *correct*. According to [9], the criteria for *correctness* are:

- the initial agent model should match with the requirements;
- the agent model should satisfy any necessary properties in order to meet its design objectives;
- the implementation should pass all tests constructed using a complete functional test generation method.

All the above criteria are closely related to three stages of agent system development, i.e. *modelling, verification and testing*.

In agent oriented engineering, there have been several attempts to use formal methods, each one focusing on different aspects of agent systems development. One of them was to formalize *PRS* (Procedural Reasoning System), a variant of the *BDI* architecture [15] with the use of Z, in order to understand the architecture in a better way, to be able to move to the implementation through refinement of the specification and to be able to develop proof theories for the architecture [10]. Trying to capture the dynamics of an agent system, [16] viewed an agent as a situated automaton that generates a mapping from inputs to outputs, mediated by its internal state. [6] developed the *DESIRE* framework, which focuses on the specification of the dynamics of the reasoning and acting behaviour of multi-agent systems. In an attempt to verify whether properties of agent models are true, work has been done on model checking of multi-agent systems with reuse of existing technology and tools [5], [14]. Towards implementation of agent systems, [2] focused on program generation of reactive systems through a formal transformation process. Finally, in a less formal approach, extensions to *UML* to accommodate the distinctive requirements of agents (*AUML*) were proposed [17].

A specific class of agents, namely cooperative ones, have been used to modelling the collective foraging behaviour of a colony of honey-bees [8]. This class of agents is fully compatible with the rules used by foraging honey-bees that include specifications for [18]:

- travelling from the nest to the source;
- searching for the source;
- collecting nectar from the source;
- travelling back to the nest;
- transmitting the information about the source - the dancing of the returning bee;
- the reaction of a bee in the nest to the dancing of a nest mate.

In this paper, we intend to show how simple agents motivated from biology can be modelled as P systems [13] and X-machines [9]. Such modelling will facilitate both verification and testing of an agent model, since appropriate strategies for model checking and testing are already developed around these methods. In addition, modular construction of agent models is feasible since these models are provided with communicating features, which allow simple models to interact. Finally, tools developed for these models are briefly presented in order to demonstrate the practicality of the approach. The two models present many similarities, but also important differences making them complementary approaches which is an important characteristic of the models developed in the area of biochemical systems [7].

2 P systems and stream X-machines. Basic definitions

Definition 1. A P system $\Pi = (V, T, \mu, M_1, \dots, M_m, R_1, \dots, R_m)$ is a construct, where:

- V is an alphabet; its elements are called objects;
- $T \subseteq V$ is the output alphabet;
- μ is a membrane structure (a rooted tree where the nodes are called membranes and the root is called skin) consisting of m membranes, with the membranes and the regions labeled in a one to one manner with elements $1, \dots, m$;
- M_1, \dots, M_m are multisets over V associated with the regions $1, 2, \dots, m$;
- R_1, \dots, R_m are finite sets of rewriting (evolution) rules of the form $a \longrightarrow x$ where a is a word over V and x is a word over $\{a_{\text{here}}, a_{\text{in}}, a_{\text{out}} \mid a \in V\}$.

Usually the membrane structure is materialized by matching pairs of parentheses. The multisets M_i and the rule sets R_i are associated with the regions of μ . When a rule $a \longrightarrow x$ is used in a region then the objects designed by a are removed, and the objects designed by x are sent to the regions indicated by tar . When $\text{tar} = \text{here}$, the object is kept in the same region; when $\text{tar} = \text{out}$, the object leaves the current region and goes into the outer one; when $\text{tar} = \text{in}$, the object is sent to one of the directly included regions, if any exists, otherwise the rule can not be applied. The target indication *here* is in general omitted. A computation is defined as follows: the process starts with the multisets present in the initial configuration and proceeds iteratively by applying in parallel the rules in each region to all multisets that can be rewritten. The result (the number of objects) is collected outside of the system at the end of the halting computation. The language generated by a system Π is denoted by $L(\Pi)$ and consists of all numbers that are computed during halting computations. Other ingredients are also considered alongside of such a definition: priority rules among the components of the sets R_i , membrane polarization or the permeability (thikness) of a membrane etc., [13]. Also the symbol-objects with no internal structure may be replaced by string-objects and in this case we deal with usual languages over a given alphabet.

Definition 2. A stream X-machine $X = (\Sigma, \Gamma, Q, M, \Phi, F, I, T, m_0)$ is a system, where:

- Σ and Γ are finite sets called the input and the output alphabets, respectively;
- Q is the finite set of states;
- M is a (possibly infinite) set of memory symbols;
- Φ is a set of basic partial functions of the form $f : \Sigma \times M \longrightarrow M \times \Gamma^*$;
- F is the next state function $F : Q \times \Phi \longrightarrow 2^Q$;
- I and T are the sets of initial and final states;
- m_0 is the initial memory value.

A computation in X is defined as follows: the process starts with the initial configuration (m_0, q_0, s, ϵ) , where $q_0 \in I$, $s \in \Sigma^*$, ϵ denotes the empty output, and proceeds iteratively by processing the current input symbol - the symbol on the left of the input string - and the current memory value by a function f emerging from the current state q , producing an output symbol going to the right of the output string and updating the memory value; the next state is one belonging to $F(q, f)$. The machine will stop when arrives in a final state and has processed all the input symbols of s . The output string obtained in this state is the result of the computation. The set of all output strings obtained by using all the possible computations in X starting with an input set Inp is denoted by $L_X(Inp)$.

Comparisons involving these models based on their computational power, when they both are dealing with strings, has been investigated [1], and ways of combining them into a hybrid model called Eilenberg P system have been proposed [4]. For such a model has been shown that NP-complete problems like SAT can be solved in linear time [4].

3 Two computational models of collective foraging

In this section an agent based approach using both P systems and X-machines will be used in order to specify the collective foraging behaviour of a colony of honey-bees.

The P system model has the following elements organized on three layers:

- the environment (M_1) containing the nectar source;
- the nest (M_2);
- the bees (M_3 to M_m).

M_1 will contain information about the amount of nectar carried by a bee, its current position, and information about the source. M_2 will have for each bee in the hive, the amount of nectar carried, its current position, a memory value identifying the nectar source position as well as an identification of each bee. M_i , $1 \leq i \leq m$ will give the position of a bee, the amount of nectar carried by a bee and a position of the source as a memory value; when some nectar will be transferred to another bee, the position and amount to be transferred will be also specified.

The membrane structure is: $\mu = [1[2[i_1]_{i_1} \dots [i_p]_{i_p}]_{i_2}[i_{p+1}]_{i_{p+1}} \dots [i_{m-2}]_{i_{m-2}}]_1$. The following rules are applied:

- 1. $(nectar, p, m) \longrightarrow (nectar, p', m')$, where p, p' are positions, $nectar$ is an arbitrary amount of nectar from a finite set of values and m is the memory content defining the position of the source; this rule mentions that a bee changes its position and may update its memory;
- 2. $(0, p_source, m) \longrightarrow (nectar_taken, p_source, m)$ - this rule describes what a bee is doing once she has arrived at the source: the current amount of nectar she carries is 0, the source position is p_source , the bee picks up the amount $nectar_taken$ and keeps the position;
- 3. $(nectar, p, m) \longrightarrow (nectar, p, m, i)_1$ - according to this rule a foraging bee i being in the environment communicates its load, position and identity to the environment (the element on the right hand side is sent to the environment, 1); similarly we have 3'. $(nectar, p, m) \longrightarrow (nectar, p, m, j)_2$ - the bee j communicates with the hive;
- 4. $(nectar, p, m, i) \longrightarrow (nectar, p, m, i)_2(0, p, m, i)_2^k$ according to this rule, from the environment it passes to the hive the nectar load of the foraging bee i and k copies indicating the source position simulating the information that is passed through the waggle dance to k bees (the amount of nectar is not considered);
- 5. $(nectar, p_1, m_1, i)(nectar, p_2, m_2, j) \rightarrow (nectar'_1, p_1, m_1, i)_{(1)}(nectar'_2, p_2, m_2)_{(j)}$ - this rule shows that the bees i and j exchange nectar; the first object could go to the environment or rest in the hive (the notation (1) means either 1 i.e. the environment or nothing which means the objects remains in the hive);
- 6. $(nectar, p, m, i) \longrightarrow (nectar, p, m)_i$ - this rule shows that the bee i (re)starts foraging after nectar has been given to honey-bees in the hive;
- 7. $((*, p_1, source_info, i), (nectar_2, p_2, m, j)) \longrightarrow (nectar_2, p_2, new_m)_j$ - according to this rule the bee j who attends the dance will refresh her memory with the position of the source; depending on the distance between the two bees i and j (the difference between p_1 and p_2), a transmission error may occur [18].

The rules 1, 2, 3 or 3' are in the sets R_i , $3 \leq i \leq m$, whereas the rules 5, 7 are only in R_2 and 4, 6 in R_1 . The sets R_i , $i \geq 1$ formally define the six requirements stated in [18]. In the next section we will show how these rules are codified in a software tool which helps to simulate the behaviour of this system. V contains elements $(nectar, p, m)$ or $(nectar, p, m, i)$ with $0 \leq nectar \leq Nectar_max$, $0 \leq p \leq Position_max$, $0 \leq memory \leq Postion_max$, $3 \leq i \leq m$; where $Nectar_max$ and $Position_max$ are two positive integers.

The next model will approach the foraging problem from a different perspective. In this case instead of modeling the whole problem, we will approach different aspects and model them as separate X-machines. At the end we will have a set of such machines. In [12] it is proposed a way of aggregating these components by providing a protocol for communication among them. We will illustrate here only the X-machine which models the dancing behaviour.

The X-machine X will have the following elements:

- $\Sigma = \{f_{bee}, space, nest, source\}$;
- $\Gamma = \{"dancing", "flying\ out", "flying\ in", "source\ found", "keep\ flying"\}$;
- $Q = \{in\ the\ nest, out\ of\ nest\}$;
- $M = \{(bee_pos, source_pos) \mid bee_pos, source_pos \in Fin\}$, where Fin is a finite set of integer values $\{0, \dots, Position_max\}$;
- Φ contains
 - $dancing(f_{bee}, (bee_pos, source_pos)) = ((bee_pos, source_pos), "dancing")$;
 - $fly_out(space, (bee_pos, source_pos)) = ((bee_pos', source_pos), "flying\ out")$;
 - $fly_in(nest, (bee_pos, source_pos)) = ((bee_pos', source_pos), "flying\ in")$;
 - $find_source(source, (bee_pos, source_pos)) = ((source_pos, source_pos), "source\ found")$;
 - $keep_fly_out(space, (bee_pos, source_pos)) = ((bee_pos', source_pos), "keep\ flying")$.
- $F(in\ the\ nest, dancing) = in\ the\ nest; F(in\ the\ nest, fly_out) = out\ of\ nest;$
 $F(out\ of\ nest, fly_in) = in\ the\ nest;$
 $F(out\ of\ nest, find_source) = out\ of\ nest;$
 $F(out\ of\ nest, keep_fly_out) = out\ of\ nest;$
- $I = \{in\ the\ nest\}; F = Q$.

In both systems either their rules (in the first case) or the X-machine components (in the second case) run in parallel approaching the honey-bees behaviour as cooperative agents.

4 Tools

In this section we will show how the formal models defined in the previous section might be transformed into an executable code such as to be able to simulate the behaviour of the defined system.

A P system simulator has been implemented; it allows to input a P system specification and then to simulate its nondeterministic and highly parallel behaviour [3].

The MzScheme code for the P system model defined in the previous section, is presented below (only the rules for the environment are provided):

```
(define A (vector (n1 p1 m1) ... (nk pk mk) (n1 p1 m1 3)... (n1 p1 m1 m)
                  ...
                  (nk pk mk 3)... (nk pk mk m)))
      ;the alphabet codifying the system objects

(define MS ((1 2) (1 3) (1 4)... (1 p)
            (2 p+1)... (2 m)); the membrane structure

(define objects
  (vector ((n1 p1 m1 3)... (nh ph mh j)))
          ; initial configuration

(define rules
  (vector (((n p m i)->((n p m i) 2)((n p m i) 2) ... ((n p m i) 2))
          ((n p m i)->((n p m) i)) ; rules corresponding to 4 and 6
          ))
```

The X-machine model is supported by a mark-up language called XMDL [12]. An XMDL fragment code of the X-machine specification of the model defined in the previous section is given below:

```
#input = {fbee, space, nest, source}.
#output = {"dancing", "flying out", "flying in", "source found",
           "keep flying"}.
#memory = {(b,s) | 0<=b<=Position, 0<=s<=Position}.
#state = {in_the_nest, out_of_nest}.
#fun dancing(?inp,(?b_p,?s_p)) =
    if ?inp==fbee then ((?b_p,?s_p),"dancing").
#fun fly_out(?space,(?b_p,?s_p)) =
    if next(?b_p,?s_p,?b_p',?s_p') then ((?b_p',?s_p'),"flying out").
```

From the above sample code we may notice the fair similarity between their formats and the formalisms used in these models. Both tools are able to animate the specified systems and XMDL is also providing a sort of formal analysis of the system by generating test sets and supporting model checking analysis.

5 Conclusions

The work reported in this paper refers to two computational models that are proposed as cooperative agent paradigms suitable to modelling the behaviour of social insects, in particular honey-bee colonies. The models are parallel distributed paradigms with specific communication mechanisms and are supported by software tools able to animate the systems specified in these frameworks.

These approaches have some similarities: they are computational devices with components running in a fully parallel manner and acting as a society of agents. For both models the design is flexible and reusable as the whole system may be built from individual components (multisets and rules in the membrane approach and component X-machines in the second case) that are assembled and enriched with the communication aspects. There are also some important differences between the two approaches. In the case of the membrane systems the outputs are defined only at the end of the computation and in some specified components; the inputs are defined only in some variants, the memory is implicitly defined as being the set of symbols associated to each component. For X-machines the inputs and the outputs are explicitly associated to every basic function and a memory element is part of any computation.

The theoretical study of these models has been developed independently and somehow on different aspects, computational power for P systems [13] and testing issues for X-machines [9]. Consequently some more work is expected in order to combine these approaches [4]. On the practical side of modeling the behaviour of different social insects, more powerful and flexible tools are requested to simulate and animate various aspects related to the evolution of these colonies and to capture a broader range of system specifications.

References

1. J. Aguado, T. Bălănescu, T. Cowling, M. Gheorghe, M. Holcombe, F. Ipate, P systems with replicated rewriting and stream X-machines (Eilenberg machines), *Fundamenta Informaticae* 49(2001), 1–17.
2. A. Attoui, A. Hasbani, Reactive systems developing by formal specification transformations. In *Proceedings of the 8th International Workshop on Database and Expert Systems Applications* (DEXA 97), 1997, 339–344.
3. D. Balbotín Noval, M. J. Pérez Jiménez, F. Sancho Caparrini, A MzSceme Implementation of Transition P Systems. In *Membrane Computing* (Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron, eds.), LNCS 2597, Springer-Verlag, 2003, 58–73.
4. T. Bălănescu, M. Gheorghe, M. Holcombe, F. Ipate, Eilenberg P systems. In *Membrane computing* (Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron, eds.), LNCS 2597, Springer-Verlag, 2003, 43–57.
5. M. Benerecetti, F. Giunchiglia, L. Serafini, A model checking algorithm for multiagent systems. In *Intelligent Agents V* (J. P. Muller, M. P. Singh, A. S. Rao, eds.), LNAI 1555, Springer-Verlag, 1999, 163–176.
6. F. Brazier, B. Dunin-Keplicz, N. Jennings, J. Treur, Formal specification of multi-agent systems: a real-world case. In *Proceedings of International Conference on Multi-Agent Systems* (ICMAS'95), MIT Press, 1995, 25–32.
7. L. Clark, R. Paton, Towards computational models of chemotaxis in Escherichia coli. In *Information Processing in Cells and Tissues* (M. Holcombe, R. Paton, eds.), Plenum Press, 1998, 39–46.
8. M. Gheorghe, M. Holcombe, P. Kefals, Computational models of collective foraging, *BioSystems*, 61(2001), 133–141.
9. M. Holcombe, F. Ipate, *Correct systems: Building a Business Process Solution*, Springer Verlag, London, 1998.
10. M. d'Inverno, D. Kinny, M. Luck, M. Wooldridge, A formal specification of dMARS. In *Intelligent Agents IV* (M.P.Singh, A.Rao, M.J.Wooldridge, eds.), LNAI 1365, Springer-Verlag, 1998, 155–176.
11. N. R. Jennings, On agent-based software engineering, *Artificial Intelligence*, 117(2000), 277–296.
12. E. Kapetis, P. Kefalas, A design language and tool for X-machines specification. In *Advances in Informatics* (D. I. Fotiadis, S. D. Nikolopoulos, eds.), World Scientific Publishing Company, 2001, 134–145.
13. Gh. Păun, *Membrane Computing. An Introduction*, Springer, Berlin, 2002.
14. A. S. Rao, M. P. Georgeff, A model-theoretic approach to the verification of situated reasoning systems. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence* (IJCAI'93) (R. Bajcsy, ed.), 1993, 318–324.
15. A. S. Rao, M. P. Georgeff, BDI Agents: from theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems* (ICMAS95), 1995, 312–319.
16. S. R. Rosenschein, L. P. Kaelbling, A situated view of representation and control, *Artificial Intelligence*, 73(1995), 149–173.
17. J. Odell, H. V. D. Parunak, B. Bauer, Extending UML for agents. In *Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence*, 2000.
18. H. de Vries, J. C. Biesmeijer, Modelling collective foraging by means of individual behaviour rules in honey-bees, *Behavioral Ecology and Sociobiology*, 44(1998), 109–124.

Hybrid Architecture Based on Support Vector Machines

Haydemar Núñez^{1,3}, Cecilio Angulo^{1,2}, and Andreu Català^{1,2}

¹Dept. of Systems Engineering, Polytechnical University of Catalonia
Avda. Víctor Balaguer s/n E-08800 Vilanova i la Geltrú, Spain

{hnunez,cangulo}@esaii.upc.es

²LEA-SICA. European Associated Lab. Intelligent Systems
and Advanced Control
Rambla de l'Exposició s/n E-08800 Vilanova i la Geltrú. Spain
andreu.catala@upc.es

³Universidad Central de Venezuela. Facultad de Ciencias.
Escuela de Computación. Caracas, Venezuela
hnunez@strix.ciens.ucv.ve

Abstract. An hybrid SVM-symbolic architecture for classification tasks is proposed in this work. The learning system relies on a support vector machine (SVM), meanwhile a rule extraction module translate the embedded knowledge in the trained SVM in the form of symbolic rules. The new representation is useful to understand the nature of the problem and its solution. Moreover, a rule insertion module in the hybrid architecture allows incorporate the available prior domain knowledge into the machine expressed in the form of symbolic rules. Thus, it is render possible the integration of SVMs with symbolic AI systems.

1 Introduction

The support vector machine (SVM) is a learning technique based on statistical learning theory [3],[5],[14], which has been applied successfully in a variety of classification and regression problems. Nevertheless, a possible limitation of the SVMs is that, like the artificial neuronal networks (ANN), they generate black box models: the solution provided by them is a linear combination of kernel functions which is difficult understand.

In the field of artificial neuronal networks there has been widespread activity aimed at redressing this situation, by translating the embedded knowledge in trained ANN in an easier to interpret representation [1],[4],[10],[12],[15]; this new representation can be useful to understand the nature of the problem and its solution.

On the other hand, hybrid neuro-symbolic systems are computational systems integrating both, symbolic and connectionist approaches, exploiting the complementarities between these knowledge representation and acquisition techniques [8],[15]. Some integration schemes include a neuronal and an interpretation modules (based on rule extraction methods); other ones can also include an insertion module making possible the use of prior domain knowledge in the neural network, expressed like a set of rules.

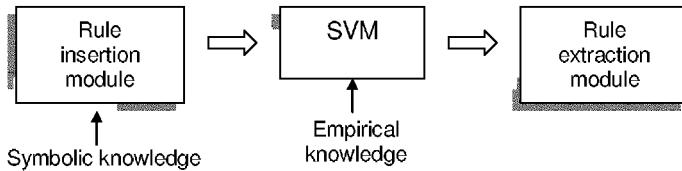


Fig.1. Hybrid SVM-symbolic architecture.

A possible solution to the lack of transparency of the SVM models for classification problems is the development of rule extraction techniques for these machines. In this work an hybrid architecture based on support vector machines is proposed (Fig 1). The rule extraction module translates the learned knowledge by the SVM into a set of rules. The rule insertion module incorporates the available prior domain knowledge expressed like symbolic rules, into these machines. The hybrid SVM-symbolic architecture to be developed should allow the integration of the SVM with symbolic components, like in the neuronal case.

This paper is organized as follows: The rule extraction from SVM method is introduced in the next section, along with experimental results on standard benchmarks. Section 3 describes the knowledge integration into a SVM method. Finally, we present conclusions and future work.

2 Rule Extraction from Support Vector Machines

In order to develop a rule extraction from SVM technique, it is fundamental to identify how the knowledge is codified by this machine when an hypothesis is generated. The solution provided by SVMs is a weighted combination of kernel functions built on the basis of the support vectors

$$f(\mathbf{x}) = \text{sign}(\sum_{i=1}^{sv} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b) \quad (1)$$

Therefore, captured knowledge is expressed through:

- The support vectors (*sv*), which are the nearest patterns to the separation limit between classes.
- The weights α_i , associated to the patterns: if $\alpha_i < C$ the pattern is a support vector without error; it is a support vector with error when $\alpha_i = C$; and the pattern is not considered in the decision function when $\alpha_i = 0$.
- The kernel function K and its parameters.

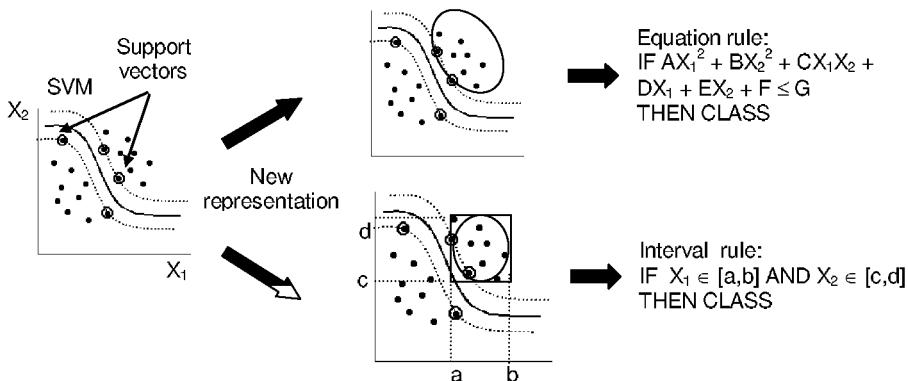


Fig. 2. Types of rules generated by the rule extraction method.

Support vectors, although they conform a small group of the learning set, are the most informative samples for the classification task. It could be advantageous to use them in the rule extraction technique. In our method, these vectors determine the boundaries of the regions built in the input space (Fig.2). These regions can be of two types: (a) ellipsoids, and (b) hyper-rectangles. Ellipsoids produce equations rules whose antecedent is the mathematical equation of the ellipsoid. Hyper-rectangles produce interval rules.

In order to construct a model with the new representation, it is necessary to construct a set of ellipsoids (or hyper-rectangles) that show a minimum of overlapping among them. The α_i parameters will help in this task. Moreover, dependency of the rule extraction process on the kernel function parameters must be reduced at minimum, so it can be applied to several SVM models.

Ellipsoids must adjust to the form of the decision limit. In order to obtain an ellipsoid with these characteristics, support vectors are used to determine the axes and vertices of them, by means of geometric methods, as follows:

First, the algorithm determines one prototype vector per class by using a clustering algorithm. This vector will be the center of the ellipsoid. Then, the support vector with associated α_i parameter being smaller than C and maximum distance to the prototype is chosen. The straight line defined by these two points is the first axis of the ellipsoid. The rest of the axes and the associated vertices are determined by simple geometry. Three possibilities exist to define these vertices: with the support vectors themselves, derived from the support vectors, or with the farthest point to the prototype. Selected support vectors are those having associate weight smaller than C. A similar procedure is followed to construct hyper-rectangles, with the only difference that parallel lines to the coordinate axes are used to define the axes of the associated regions.

The number of necessary regions to describe the SVM model will depend on the form of the decision limit. It can happen that only one ellipsoid is not sufficient to describe the data from a class with a minimum overlapping between classes. Fig.3(a) shows an ellipsoid generated by using the described procedure; it overlaps the other

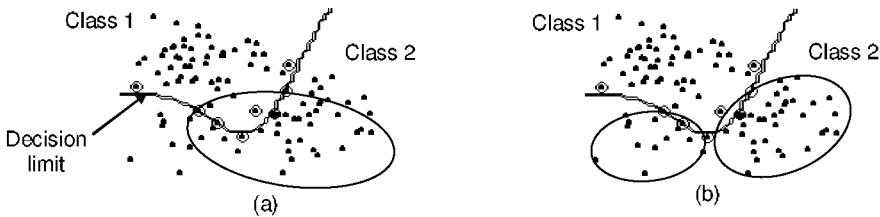


Fig. 3. (a) Generation of one ellipsoid. (b) Dividing to the ellipsoid, the overlapping between classes is reduced

class due to the form of the separation limit. By dividing the ellipsoid (Fig.3(b)), overlapping is reduced and the two new regions adjust better to this separation surface. It can be observed in Fig.3(a) that support vectors from the another class exists within the ellipsoid, providing information about the separation surface form. Therefore, support vectors help to determine when to divide an ellipsoid.

This criterion constitutes a partition test, which will be applied to the ellipsoids. A partition test has a positive result when some support vector from another class exists within the region. Additionally, the partition test will be positive if one of the generated vertices or prototypes belongs to another class. The class label for these vectors is determined by using the SVM function. If the test result is positive for one ellipsoid, the same will be divided.

In order to define the number of ellipsoids per class, the algorithm follows an incremental scheme. Beginning with a single prototype, the associated ellipsoid is generated. Next, the partition test is applied on this region. If it is negative, the region is translated to a rule. Otherwise, new regions are generated as follows: Each iteration produces m regions with positive partition test and p regions with negative partition test. The latter ones are translated to rules. In the next iteration, the data of the m regions is used to determine $m+1$ new prototypes and to generate $m+1$ new ellipsoids. This procedure is stopped when all the partition tests are negative or the maximum number of iterations is reached. This process allows to control the number of generated rules.

After rules are extracted, the system classifies an example by assigning it to the class of the nearest rule in the knowledge base (following the nearest-neighbor philosophy) by using the Euclidian distance [6]. If an example is covered by several rules, we choose the class of the most specific ellipsoid or hyper-rectangle containing the example; that is, the one with the smallest volume [11].

2.1 Experimental results

In order to evaluate the performance of the rule extraction algorithm, we applied it to several data sets obtained from the UCI repository [9]. Table 1 shows their characteristics. To generate the prototype vectors the k-means clustering algorithm [7] was used. The performance of the generated rules was quantified with the following measures:

- *Error (Err)*: Classification error provided by the rules on test set.
- *Consistency (Co)*: Percentage of the test set for which the SVM and the rule base output agree.
- *Coverage (Cv)*: Percentage of examples from a test set covered by the rule base.
- *Overlapping (Ov)*: Percentage of examples from a test set covered by several rules
- *Number of extracted rules (NR)*.

Table 2 shows the prediction error of the SVM and the performance values of the extracted rule base for each data set. These measures were obtained by averaging over stratified ten-fold cross-validation. Our results show a high agreement between the performance values obtained from the rule base and those obtained from the SVM. It should be emphasized that the consistency percentage between the rule base and the SVM is very high. Only in the case of the Soybean data set and the Spect data set, the equation rules provide a low percentage of covering.

Table 1. Data sets and their characteristics

Data sets		<i>Samples</i>	<i>Attributes</i>	<i>Classes</i>
<i>Cod.</i>	<i>Name</i>			
1	Iris	150	4	3
2	Wisconsin	699	9	2
3	Wine	178	13	3
4	Soybean	47	35	4
5	New-Thyroid	215	5	3
6	Australian	690	14	2
7	Spect	267	23	2
8	Monk1	432	6	2
9	Monk2	432	6	2
10	Monk3	432	6	2

Table 2. Performance values from data sets.

Data set	SVM Error	Equation rules					Interval rules				
		<i>Err</i>	<i>Co</i>	<i>Cv</i>	<i>Ov</i>	<i>NR</i>	<i>Err</i>	<i>Co</i>	<i>Cv</i>	<i>Ov</i>	<i>NR</i>
1	0.033	0.040	98.00	72.00	0.67	7.0	0.040	99.33	68.00	0.67	4.7
2	0.047	0.039	97.06	92.51	6.73	3.0	0.062	93.54	87.01	4.97	5.8
3	0.022	0.017	98.30	67.49	0.55	5.9	0.023	96.07	69.89	2.28	8.2
4	0.000	0.000	100.0	9.00	0.00	4.9	0.040	96.00	75.00	0.00	6.4
5	0.032	0.032	97.21	80.09	0.00	7.1	0.032	96.30	70.58	2.72	9.2
6	0.127	0.133	93.60	65.70	2.86	18.4	0.137	93.20	87.66	3.18	21.6
7	0.102	0.117	96.26	21.39	0.53	14.0	0.112	96.26	40.11	0.00	22.0
8	0.051	0.091	85.18	33.56	0.00	24.0	0.056	92.59	59.49	0.00	33.0
9	0.178	0.211	76.39	32.87	0.46	60.0	0.219	75.95	63.19	5.78	84.0
10	0.034	0.053	94.44	50.00	0.00	5.0	0.027	97.45	95.37	0.00	4.0

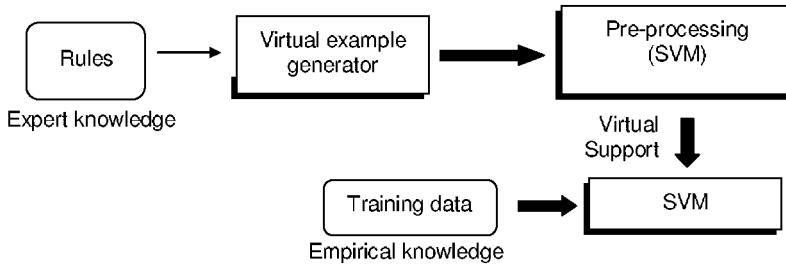


Fig. 4. Rule insertion module.

3 Integrating Prior Knowledge in Support Vector Machines

Different techniques have been used in the literature to incorporate prior knowledge in the support vector machines: By means of virtual examples generated from transformation functions applied to the data [17] or the support vectors [13]; designing kernel functions that adapt to the problems [2],[5],[18]; adding new restrictions to the optimization problem [16]. Nevertheless, sometimes it can be difficult to formalize the prior knowledge in the form of a transformation or a kernel function. But, it is even possible to express knowledge like a set of rules provided by an expert. In this case, how we could integrate this knowledge into a SVM? We propose do it by means of a strategy similar to the virtual support method.

The prior knowledge, expressed as a rule, defines a convex region in the input space (in the form of a hyper-rectangle). This convex region is defined by a set of vertices, which provide information about the limits of the associated rule. These vertices can be used like virtual examples and added to the training set. Fig. 4 describes the proposed rule insertion method.

The virtual example generator determines the associated vertices of the rules. Then, we train a SVM using the vertices of the rules of one class and data from the other classes. This stage of pre-processing generates a set of *virtual supports* for each class. Finally, we train a SVM with the overall learning set and the generated virtual supports. In this form, the part of the knowledge that could be more informative for the classification task is added to the data (the nearest vertices to the separation limit between classes). To maintain the relative proportions of the classes in the data base, only a part of the virtual supports could be added, otherwise we could obtain a data set that does not exactly represent the original problem. A reasonable measure could be to increase the proportion of a class with respect to the other classes in a 5%, at maximum. The α_i parameters can help to select the virtual supports to be added.

3.1 Experimental results

In order to evaluate the rule insertion method, we applied it to the three MONK problems of the UCI repository, because they have a defined domain theory. To verify

the improvement on the SVM performance when the prior knowledge is added, the following procedure was implemented: First, a SVM without prior rule based knowledge was trained. After, we train the same SVM using the rule insertion method. This procedure was repeated 50 times and we determined the average values on the following parameters:

- *Training set error (ErrEnt)*
- *Data test error (TestErr).*
- *Number of support vectors from MONK class (Sv1)*
- *Number of support vectors from NO-MONK class (Sv2)*

In addition, a SVM was trained adding a number of virtual supports to the data, so that the proportion of the MONK class with respect to the NO-MONK class increased in a 5%. Tables 3, 4 and 5 show the obtained results.

Tables below show that the rule insertion method improves the original SVM performance with a not excessive increasing of the number of supports vectors.

Table 3. Results from MONK1 data sets

MONK1	ErrEnt	ErrTest	Sv1	Sv2
Without knowledge	0.00	0.0516	31.64	26.52
With knowledge	0.00	0.0328	44.32	25.34
With knowledge (5% added)	0.00	0.0439	38.90	25.80

Table 4. Results from MONK2 data sets

MONK2	ErrEnt	ErrTest	Sv1	Sv2
Without knowledge	0.00	0.1811	41.36	40.14
With knowledge	0.00	0.1179	62.34	47.26
With knowledge (5% added)	0.00	0.1377	53.12	44.76

Table 5. Results from MONK3 data sets

MONK3	ErrEnt	ErrTest	Sv1	Sv2
Without knowledge	0.0507	0.0330	15.78	19.76
With knowledge	0.0523	0.0265	18.88	19.94
With knowledge (5% added)	0.0424	0.0278	16.62	21.86

4 Conclusions and future work

The rule extraction from SVM method is based on the combination of support vectors and prototype vectors, by means of geometric procedures, to determine a set of ellipsoids that represent a class with minimum overlapping between classes. It

allows transform the SVM into a rule-based classifier, making transparent the knowledge learned by this machine.

On the other hand, the rule insertion method allows incorporate to the SVM prior knowledge expressed like a set of rules. The proposed hybrid architecture makes possible the integration of a SVM with symbolic components.

In the future, proposed algorithms will be validated on a larger variety of data sets, in order to that the conclusions are even more consistent.

References

1. Andrews, R., Diederich, J., Tickle, A.: A Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks. *Knowledge-Based Systems*, 8(6) (1995)373-389.
2. Brailovsky, V., Barzilay, O., Shahave, R.: On global, local, mixed and neighborhood kernels for support vector machines. *Pattern Recognition Letters*, 20 (1999) 1183-1190
3. Cortes, C., Vapnik, V.: Support-Vector Networks. *Machine Learning*, Kluwer Academic Publisher, Boston, 20 (1995) 237-297
4. Craven, M., Shavlik, J.: Using Neural Networks for Data Mining. *Future Generation Computer Systems*, 13 (1997) 211-229
5. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and other kernel-based learning methods, Cambridge University Press, Cambridge, (2000)
6. Domingos, P.: Unifying Instance-Based and rule-Based Induction. *Machine Learning*, Kluwer Academic Publisher, Boston, 24 (1991)141-168
7. Duda, R. O., Hart, P. E., Stork, D. G.: *Pattern Recognition*, John Wiley & Sons, Inc., New York, (2001)
8. McGarry K., Wermter, S., MacIntyre, J.: Hybrid Neural Systems: From Simple Coupling to Fully Integrated Neural Networks. *Neural Computing Surveys*, 2 (1999).
9. Merz, C. J., Murphy, P. M. Murphy.: UCI Repository for Machine Learning Data-Bases. Irvine, CA: University of California, Department of Information and Computer Science, <http://www.ics.uci.edu/~mlearn/MLRepository.html>, (1998)
10. Mitra, S., Pal, S.K., Mitra, P.: Data Mining in Soft Computing Framework: A survey. *IEEE Transactions on Neural Networks*, 13(1) (2002) 3-14
11. Salzberg, S.: A Nearest Hyperrectangle Learning Method. *Machine Learning*, Kluwer Academic Publisher, Boston, 6 (1991) 251-276.
12. Tickle, A., Andrews, R., Mostefa, G., Diederich, J.: The Truth will come to light: Directions and Challenges in Extracting the Knowledge Embedded within Trained Artificial Neural Networks. *IEEE Trans. on Neural Networks*, 9(6) (1998)1057-1068
13. Schölkopf, B., Burges, C., Vapnik, V.: Incorporating Invariances in Support Vector Learning Machine. *ICANN'96. Lecture Notes in Computer Science*, 1112 (1996) 47-52
14. Vapnik, V.: *Statistical Learning Theory*. John Wiley&Sons, Inc., New York, (1998)
15. Wermter, S., Sun, R. (Eds): *Hybrid Neural Systems*. Springer-Verlag, Berlin, (2000)
16. Zhang, X.: Using Class-Center vectors to build Support Vector Machines. *Proc. IEEE Conference on Neural Networks for Signal Processing*, (1999) 3-11
17. Zhao, Q., Principe, J.C.: Improving ATR Performance by Incorporating Virtual negative examples. *Proc. International Joint Conference on Neural Networks*. 5 (1999) 3198-3203
18. Zien, A., Rätsch, G., Mika, S., Schölkopf, B., Lengauer, T., Müller, K.-R.: Engineering Support Vector Machine kernels that Recognize Translation Initiation Sites. *Bioinformatics*. 16(9) (2000) 799-807

A neural approach to extended logic programs*

Jesús Medina, Enrique Mérida-Casermeiro, and Manuel Ojeda-Aciego

Dept. Matemática Aplicada. Univ. Málaga, Spain.

{jmedina,merida,aciego}@ctima.uma.es

Abstract. A neural net based development of multi-adjoint logic programming is presented. Transformation rules carry programs into neural networks, where truth-values of rules relate to output of neurons, truth-values of facts represent input, and network functions are determined by a set of general operators; the output of the net being the values of propositional variables under its minimal model. Some experimental results are reported.

1 Introduction

One of the advantages of the use of neural networks is its massively parallel architecture-based dynamics which are inspired by the structure of human brain, adaptation capabilities, and fault tolerance. The latter provides the ability of dealing with modeling and control aspects of complex processes, as well as with uncertain, incomplete and/or inconsistent information, being fuzzy logic a powerful mathematical tool for its study.

Fuzzy logic systems are capable to express nonlinear input/output relationships by a set of qualitative if-then rules, and to handle both numerical data and linguistic knowledge, especially the latter, which is extremely difficult to quantify by means of traditional mathematics. Neural networks, on the other hand, has an inherent learning capability, which enables the networks to adaptively improve their performance. In this work, we introduce a hybrid approach to handling uncertainty, which is expressed in the rich language of multi-adjoint logic but is *implemented* by using ideas borrowed from the world of neural networks.

Multi-adjoint logic programming, which was introduced in [6] as a refinement of residuated logic programming, allows for very general connectives in the body of the rules; moreover, sufficient conditions for the continuity of its semantics are known. The handling of uncertainty inside our logic model is based on the use of a generalised set of truth-values, usually a (finite or infinite) subset of the real unit interval $[0, 1]$, instead of the Boolean constants $\{v, f\}$. Such an approach is interesting for applications, for instance, consider a situation in which connectives are built from the users preferences, it is likely that knowledge is described by a many-valued logic program where connectives have many-valued truth functions and aggregation operators (such as arithmetic mean or weighted sum) where different implications could be needed for different purposes, and different aggregators are defined for different users, depending on their preferences.

* Partially supported by Spanish DGI project BFM2000-1054-C02-02.

In this paper, following ideas in [7], we present a neural net based implementation of the fixpoint semantics of multi-adjoint logic programming, introduced in [6], with the advantage that, at least potentially, we can calculate in parallel the answer for any query. The implementation using neural networks needs some preprocessing of the initial program to transform it in a *homogeneous* program; the ideas under this definition are based on the results in [1].

2 Preliminary definitions

Multi-adjoint logic programming is a general theory of logic programming which allows the simultaneous use of different implications in the rules and rather general connectives in the bodies; a preliminary version was presented in [6], where models of these programs were proved to be post-fixpoints of the immediate consequences operator, which turned out to be monotonic under very general hypotheses. In addition, the continuity of the immediate consequences operator was studied, and some sufficient conditions for its continuity were obtained.

To make this paper as self-contained as possible, the necessary definitions about multi-adjoint structures are included in this section. For motivating comments on the multi-adjoint stuff the interested reader is referred to [6].

The first interesting feature of multi-adjoint logic programs is that a number of different implications are allowed in the bodies of the rules. Formally, the basic definition is given below:

Definition 1. Let $\langle L, \preceq \rangle$ be a complete lattice. A multi-adjoint lattice \mathcal{L} is a tuple $(L, \preceq, \leftarrow_1, \&_1, \dots, \leftarrow_n, \&_n)$ satisfying the following items:

1. $\langle L, \preceq \rangle$ is bounded, i.e. it has bottom and top elements;
2. $\top \&_i \vartheta = \vartheta \&_i \top = \vartheta$ for all $\vartheta \in L$ for $i = 1, \dots, n$;
3. $(\leftarrow_i, \&_i)$ is an adjoint pair in $\langle L, \preceq \rangle$ for $i = 1, \dots, n$; i.e.
 - (a) Operation $\&_i$ is increasing in both arguments.
 - (b) Operation \leftarrow_i is increasing in the first argument and decreasing in the second.
 - (c) For any $x, y, z \in P$, $x \preceq (y \leftarrow_i z)$ holds if and only if $(x \&_i z) \preceq y$ holds.

The need of the monotonicity of operators \leftarrow_i and $\&_i$ is clear, if they are to be interpreted as generalised implications and conjunctions. The third property in the definition can be adequately interpreted as a generalised modus-ponens rule.

Originally, the multi-adjoint paradigm was developed for multi-adjoint lattices, however, for the sake of simplicity, in this specific implementation we will restrict our attention to $[0, 1]$. As example of adjoint pairs in this lattice, we have the *product* ($\leftarrow_P, \&_P$), *Gödel* ($\leftarrow_G, \&_G$) and *Lukasiewicz* ($\leftarrow_L, \&_L$) adjoint pairs; which are defined as

$$\begin{aligned} x \leftarrow_P y &= \min(1, x/y) & x \&_P y &= x \cdot y \\ x \leftarrow_G y &= \begin{cases} 1 & \text{if } y \leq x \\ x & \text{otherwise} \end{cases} & x \&_G y &= \min(x, y) \\ x \leftarrow_L y &= \min(1 - x + y, 1) & x \&_L y &= \max(0, x + y - 1) \end{aligned}$$

Definition 2. A multi-adjoint program is a set of weighted rules $\langle F, \vartheta \rangle$ satisfying the following conditions:

1. F is a formula of the form $A \leftarrow_i B$ where A is a propositional symbol called the head of the rule, and B is a well-formed formula built from propositional symbols B_1, \dots, B_n ($n \geq 0$) by the use of monotone operators in the multi-adjoint Ω -algebra, which is called the body formula.
2. The weight ϑ is an element (a truth-value) of $[0, 1]$.

Facts are rules with body \top (which usually will not be written),¹ and a query (or goal) is a propositional symbol intended as a question $?A$ prompting the system.

Regarding the implementation as a neural network, it will be useful to give a name to a specially simple type of rule: the *homogeneous rules*.

Definition 3. A weighted formula is homogeneous if it has one of the following forms:

$$\langle A \leftarrow_i \&_i (B_1, \dots, B_n), \vartheta \rangle \quad \langle A \leftarrow_i @ (B_1, \dots, B_n), \top \rangle \quad \langle A \leftarrow_i B_1, \vartheta \rangle$$

where the B_i are propositional symbols.

The homogeneous rules represent exactly the simplest type of (proper) rules we can have in our program. In some sense, homogeneous rules allow a straightforward generalization of the standard logic programming framework, in that no operators other than \leftarrow_i and $\&_i$ are used.

Definition 4. An interpretation is a mapping I from the set of propositional symbols Π to the lattice $\langle L, \preceq \rangle$.

The fixed point semantics provided by the immediate consequences operator, given by van Emden and Kowalski [8], is generalised to the framework of multi-adjoint logic programs, as shown below:

Definition 5. Let \mathbb{P} be a multi-adjoint program. The immediate consequences operator, $T_{\mathbb{P}}^{\mathcal{L}} : \mathcal{I}_{\mathcal{L}} \rightarrow \mathcal{I}_{\mathcal{L}}$, maps interpretations to interpretations, and for $I \in \mathcal{I}_{\mathcal{L}}$ and $A \in \Pi$ is defined by

$$T_{\mathbb{P}}^{\mathcal{L}}(I)(A) = \sup \left\{ \vartheta \&_i \hat{I}(B) \mid \langle A \leftarrow_i B, \vartheta \rangle \in \mathbb{P} \right\}$$

As usual, it is possible to characterise the semantics of a multi-adjoint logic program by the post-fixpoints of $T_{\mathbb{P}}$; that is, an interpretation I is a model of a multi-adjoint logic program \mathbb{P} iff $T_{\mathbb{P}}(I) \sqsubseteq I$. The $T_{\mathbb{P}}$ operator is proved to be monotonic and continuous under very general hypotheses, see [6].

Once we know that $T_{\mathbb{P}}$ can be continuous under very general hypotheses, then the least model can be reached in at most countably many iterations beginning with the least interpretation denoted Δ , that is, the least model is $T_{\mathbb{P}}^\omega(\Delta)$.

In the next section we present a model of neural network which allows to evaluate the $T_{\mathbb{P}}$ operator and, therefore, by iteration will be able to approximate the actual values of the least model up to any prescribed precision.

¹ We will consider one *designated implication* to be used for the representation of facts, which is denoted \leftarrow . This designated implication will be also used in the procedure of translation of a program into a homogeneous one.

3 Obtaining a homogeneous program

In this section we present a procedure for transforming a given multi-adjoint logic program into a homogeneous one.

Handling rules. We will state a procedure for transforming a given program in another (equivalent) one containing only facts and homogeneous rules. It is based on two types of transformations:

- T1. A weighted formula $\langle A \leftarrow_i \&_j (\mathcal{B}_1, \dots, \mathcal{B}_n), \vartheta \rangle$ is substituted by the formulas:

$$\langle A \leftarrow_i A_1, \vartheta \rangle \quad \langle A_1 \leftarrow_j \&_j (\mathcal{B}_1, \dots, \mathcal{B}_n), \top \rangle$$

where A_1 is a fresh propositional symbol, and $\langle \leftarrow_j, \&_j \rangle$ is an adjoint pair.

For the case $\langle A \leftarrow_i @(\mathcal{B}_1, \dots, \mathcal{B}_n), \vartheta \rangle$ in which the main connective of the body of the rule happens to be an aggregator, the transformation is similar:

$$\langle A \leftarrow_i A_1, \vartheta \rangle \quad \langle A_1 \leftarrow @(\mathcal{B}_1, \dots, \mathcal{B}_n), \top \rangle$$

where A_1 is a fresh propositional symbol, and \leftarrow is a designated implication.

- T2. A weighted formula $\langle A \leftarrow_i \Theta(\mathcal{B}_1, \dots, \mathcal{B}_n), \vartheta \rangle$, where Θ is either $\&_i$ or an aggregator, and a component \mathcal{B}_k is assumed to be either of the form $\&_j(\mathcal{C}_1, \dots, \mathcal{C}_l)$ or $@(\mathcal{C}_1, \dots, \mathcal{C}_l)$ is substituted by the following pair of formulas, respectively:

- $\langle A \leftarrow_i \Theta(\mathcal{B}_1, \dots, \mathcal{B}_{k-1}, A_1, \mathcal{B}_{k+1}, \dots, \mathcal{B}_n), \vartheta \rangle$ and $\langle A_1 \leftarrow_j \&_j(\mathcal{C}_1, \dots, \mathcal{C}_l), \top \rangle$
- $\langle A \leftarrow_i \Theta(\mathcal{B}_1, \dots, \mathcal{B}_{k-1}, A_1, \mathcal{B}_{k+1}, \dots, \mathcal{B}_n), \vartheta \rangle$ and $\langle A_1 \leftarrow @(\mathcal{C}_1, \dots, \mathcal{C}_l), \top \rangle$

The procedure to transform the rules of a program so that all the resulting rules are homogeneous, is based in the two previous transformations as follows:

1. Apply T1 to rules $\langle A \leftarrow_i \Theta(\mathcal{B}_1, \dots, \mathcal{B}_n), \vartheta \rangle$ such that either $\Theta = \&_j$ with $i \neq j$, or $\Theta = @$ and $\vartheta \neq \top$.
2. Apply T2 to rules $\langle A \leftarrow_i \Theta(\mathcal{B}_1, \dots, \mathcal{B}_n), \vartheta \rangle$ such that either $\Theta = \&_i$, or $\Theta = @$ and $\vartheta = \top$.

Handling facts. After the exhaustive application of the previous procedure we can assume that all our rules are homogeneous. Regarding facts, it might happen that the program contained facts about the same propositional symbol but with different weights.

Assume all the facts about A are $\langle A \leftarrow \top, \vartheta_j \rangle$, con $j \in \{1, \dots, l\}$, then the following fact is substituted for the previous ones: $\langle A \leftarrow \top, \vartheta \rangle$ where $\vartheta = \sup\{\vartheta_j \mid j \in \{1, \dots, l\}\}$.

The new program obtained from \mathbb{P} after the homogenization of rules and facts is denoted \mathbb{P}^* . Note that in this new program there are new propositional symbols.

Preservation of the semantics. It is necessary to check that the semantics of the initial program has not been changed by the transformation. The following results will show that every model of \mathbb{P}^* is also a model of \mathbb{P} and, in addition, the minimal model of \mathbb{P}^* is also the minimal model of \mathbb{P} .

Theorem 1. *Every model of \mathbb{P}^* is also a model of \mathbb{P} .*

Theorem 2. *The minimal model of \mathbb{P}^* when restricted to the variables in Π is also the minimal model of \mathbb{P} .*

4 Model of neural network

Using neural networks in the context of logic programming is not a novel idea; for instance, in [1] it is shown how fuzzy logic programs can be transformed into neural nets. In [2] the fixed point of the $T_{\mathbb{P}}$ operator for acyclic logic programs is constructed. This result is later extended in [3] to deal with the first order case. Our approach in this paper is interesting since our logic is much richer than classical or the usual versions of fuzzy logic in the literature, although we only consider the propositional case.

The set of operators to be implemented will consist of the three most important adjoint pairs defined previously. Note that every continuous t-norm is expressible as an ordinal sum of them [4]. We will implement a family of weighted sums defined as:

$$@_{(n_1, \dots, n_m)}(p_1, \dots, p_m) = \frac{n_1 p_1 + \dots + n_m p_m}{n_1 + \dots + n_m}$$

Each process unit is associated to either a propositional symbol of the initial program or an homogeneous rule. The state of the i -th neuron at time t is expressed by its output $S_i(t)$ and the state of the network is expressed by the state vector $S(t) = (S_1(t), S_2(t), \dots, S_N(t))$. The initial state for neurons associated to propositional symbols is ϑ_A if there is a fact $\langle A, \vartheta_A \rangle$ in the program and zero for any other component.

Regarding the user interface, there are two types of neurons, visible or hidden, the output of the visible neurons is the output of the net, whereas the output of the hidden neurons is only used as input values for other neurons. The set of visible neurons is formed by those associated with propositional symbols of the initial program, the others are hidden neurons.

The connection between neurons is denoted by a matrix of weights \mathbf{W} , in which w_{ij} indicates the existence or absence of connection from unit i to j ; if the neuron i is associated with a weighted sum then w_{ij} represents the weights associated to input j . In the internal register of neuron i are allocated the i -th row of the matrix \mathbf{W} , the initial truth-value v_i , together with a signal m_i that indicates whether the neuron is associated to either a fact or a rule. So the net is a distributed information system.

Therefore, we have two vectors: one storing the truth-values v of atoms and homogeneous rules, and another m storing the type of the neurons in the net.

The signal m_i indicates the functioning mode of the neuron. If $m_i = 1$, then the neuron is associated to a propositional symbol (visible neuron). Its next state is the maximum value among all the operators involved in its input and the initial truth-values v_i . More precisely:

$$S_i(t+1) = \max \left\{ v_i, \max_{k / w_{ik} > 0 \{S_k(t)\}} \right\}$$

When a neuron is associated to the product, Gödel, or Łukasiewicz implication, respectively, then the signal m_i is set to 2, 3, and 4, respectively.

The output of the neuron mimics the behaviour of the implication in terms of the adjoint property when a rule of type m_i has been used; specifically, the output in the next instant will be:

- Product implication, $m_i = 2$: $S_i(t+1) = v_i \prod_{k / w_{ik} > 0} S_k(t)$

- Gödel implication, $m_i = 3$: $S_i(t+1) = \min \{v_i, \min_{k/w_{ik} > 0} \{S_k(t)\}\}$
- Łukasiewicz implication, $m_i = 4$: $S_i(t+1) = \max\{v_i + \sum_{k/w_{ik} > 0} (S_k(t) - 1), 0\}$

Neurons associated to aggregation operators have signal $t_i = 5$. Its output is:

$$S_i(t+1) = \sum_k w'_{ik} S_k(t) \quad \text{where} \quad w'_{ik} = \frac{w_{ik}}{\sum_r w_{ir}}$$

Example 1. The non homogeneous rule $\langle p \leftarrow_P @_{(3,7)}(q, r), 0.5 \rangle$ is decomposed into:

$$\alpha = \langle h \leftarrow @_{(3,7)}(q, r), 1 \rangle \quad \beta = \langle p \leftarrow_P h, 0.5 \rangle$$

The neurons corresponding to the new propositional symbol h and to homogeneous rules α and β are hidden neurons.

Note that in the n -th iteration, the output of neuron of the rule α is $S_\alpha(n) = @_{(3,7)}(S_q(n-1), S_r(n-1))$, and this output is used by the rule β in the next iteration to obtain $S_\beta(n+1) = 0.5 \cdot S_\alpha(n)$. \square

The following result relates the behavior of the components of the state vector with the immediate consequence operator.

Theorem 3. *Given a homogeneous program \mathbb{P} , we have that $T_{\mathbb{P}}^n(\Delta)(A) = S_A(2n-2)$ for all propositional symbol A and $n \geq 1$.*

5 Representing a homogenous program by a neural net

Each neuron in the net represents either a symbol of the initial program \mathbb{P} or a new rules of the homogeneous program \mathbb{P}^* . The different types of neurons are described below:

1. *A propositional symbol:* Its type is $m_i = 1$. The initial truth-value v_i is set either to 0 (by default) or to the truth-value if A is a fact.
The i -th row of the matrix of weights has all components set to 0 but those corresponding to rules whose head is the given propositional symbol, in which case has value 1.
2. *Product implication:* These neurons correspond to a homogeneous product rule. Its internal registers are $m_i = 2$, v_i uses the truth-value of the rule in v_i , and the corresponding row in the matrix of weights is fixed with all components 0, except those assigned to propositional symbols involved in the body, which are set to 1.
3. *Gödel implication:* Similar to the previous case with $m_i = 3$.
4. *Łukasiewicz implication:* Similar to the previous case with $m_i = 4$.
5. *Weighted sums:* These neurons are related to rules of the type:

$$\langle p \leftarrow @_{(n_1, n_2, \dots, n_k)}(q_1, q_2, \dots, q_k), 1 \rangle$$

So the truth-value is always one and it is unimportant which type of implication is used since all of them assign the same value to the head.

The register of this type of neurons is set with truth-value $v_i = 1$, its type is $m_i = 5$, and the vector $w_{i\bullet}$ indicates the weights ($w_{ij} \geq 0$) of the rest of neurons on the output of the weighted sum.

In order to show the power of the neural implementation of the $T_{\mathbb{P}}$ -operator, we present a more complex example.

Example 2. Consider the program with the following rules

$$\begin{array}{lll} \langle s_1 \leftarrow_P t_1, 0.6 \rangle & \langle p_1 \leftarrow_G r_1 \&_L s_1, 0.9 \rangle & \langle x_1 \leftarrow_P @_{(5,2,1)}(p_1, x_2, x_3), 0.8 \rangle \\ \langle r_1 \leftarrow_G q_1, 0.7 \rangle & \langle p_2 \leftarrow_P r_2 \&_P s_2 \&_P t_2, 0.8 \rangle & \langle x_2 \leftarrow_P @_{(5,3,1)}(p_2, x_1, x_3), 0.9 \rangle \\ \langle r_2 \leftarrow_P q_2, 0.6 \rangle & \langle p_3 \leftarrow_P r_3 \&_P s_3, 0.8 \rangle & \langle x_3 \leftarrow_P @_{(5,3,3)}(p_3, x_1, x_2), 0.9 \rangle \\ \langle r_3 \leftarrow_P q_3, 0.7 \rangle & & \langle x \leftarrow_P @_{(4,3,2)}(x_1, x_2, x_3), 0.9 \rangle \end{array}$$

and facts $\langle t_1, 0.4 \rangle, \langle q_1, 0.5 \rangle, \langle s_2, 0.5 \rangle, \langle q_2, 0.5 \rangle, \langle t_2, 0.7 \rangle, \langle q_3, 0.5 \rangle, \langle s_3, 0.6 \rangle$.

In this example the propositional symbols p_1, p_2 and p_3 can be considered as the same economic characteristic in each country when only are considered internal market information and x_1, x_2 and x_3 can be considered as that economic characteristic when relationship among countries are being considered. Finally x can be shown as a global economic one.

Since there exist non homogeneous rules the program is transformed into:

$$\begin{array}{ll} \langle p_1 \leftarrow_L r_1 \&_L s_1, 0.8 \rangle & \langle r_1 \leftarrow_L q_1, 0.7 \rangle \\ \langle s_1 \leftarrow_P t_1, 0.6 \rangle & \langle h_1 \leftarrow_P @_{(5,2,1)}(p_1, x_2, x_3), 1 \rangle \\ \langle x_1 \leftarrow_P h_1, 0.8 \rangle & \langle p_2 \leftarrow_P r_2 \&_P s_2 \&_P t_2, 0.8 \rangle \\ \langle r_2 \leftarrow_P q_2, 0.6 \rangle & \langle h_2 \leftarrow_P @_{(5,3,1)}(p_2, x_1, x_3), 1 \rangle \\ \langle x_2 \leftarrow_P h_2, 0.9 \rangle & \langle p_3 \leftarrow_P r_3 \&_P s_3, 0.8 \rangle \\ \langle r_3 \leftarrow_P q_3, 0.7 \rangle & \langle h_3 \leftarrow_P @_{(5,3,3)}(p_3, x_1, x_2), 1 \rangle \\ \langle x_3 \leftarrow_P h_3, 0.9 \rangle & \langle h \leftarrow_P @_{(4,3,2)}(x_1, x_2, x_3), 1 \rangle \\ \langle x \leftarrow_P h, 0.9 \rangle & \end{array}$$

So, the network will have thirty-three neurons. The first eighteen are associated to propositional symbols: $p_1, q_1, r_1, s_1, t_1, x_1, p_2, q_2, r_2, s_2, t_2, x_2, p_3, q_3, r_3, s_3, x_3, x$, and the remaining ones are associated to the homogeneous rules.

The internal registers of the network are fixed as follows:

$$\begin{aligned} \mathbf{m} = & (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 5, 5, 5, 5, 3, 3, 2, 2, 2, 2, 2, 2, 4, 2, 2) \\ \mathbf{v} = & (0, 0.75, 0, 0, 0.8, 0, 0, 0.8, 0, 0.8, 0.7, 0, 0, 0.7, 0, 0.8, \\ & 0, 0.1, 1, 1, 1, 0.9, 0.7, 0.8, 0.9, 0.8, 0.75, 0.9, 0.8, 0.95, 0.9, 0.9) \end{aligned}$$

W is a 33×33 matrix with all components zeroes except those below:

$$\begin{aligned} w_{1,23} = & 1 \quad w_{3,24} = 1 \quad w_{4,25} = 1 \quad w_{6,26} = 1 \quad w_{7,27} = 1 \quad w_{9,28} = 1 \quad w_{12,29} = 1 \\ w_{13,30} = & 1 \quad w_{15,31} = 1 \quad w_{17,32} = 1 \quad w_{18,33} = 1 \quad w_{19,1} = 5 \quad w_{19,12} = 2 \quad w_{19,17} = 1 \\ w_{20,6} = & 3 \quad w_{20,7} = 5 \quad w_{20,17} = 1 \quad w_{21,6} = 3 \quad w_{21,12} = 3 \quad w_{21,13} = 5 \quad w_{22,6} = 4 \\ w_{22,12} = & 3 \quad w_{22,17} = 2 \quad w_{23,3} = 1 \quad w_{23,4} = 1 \quad w_{24,2} = 1 \quad w_{25,5} = 1 \quad w_{26,19} = 1 \\ w_{27,9} = & 1 \quad w_{27,10} = 1 \quad w_{27,11} = 1 \quad w_{28,8} = 1 \quad w_{29,20} = 1 \quad w_{30,15} = 1 \quad w_{30,16} = 1 \\ w_{31,14} = & 1 \quad w_{32,21} = 1 \quad w_{33,22} = 1 \end{aligned}$$

The network gets stabilized in 130 steps, giving the values of the minimal model $T_{\mathbb{P}}^{\omega}(\Delta)$ for each propositional symbol of the initial program.

6 Concluding remarks and future work

A new neural model has been introduced, which implements the fixpoint semantics of the multi-adjoint logic programming paradigm, which is a new approach to the treatment of reasoning under fuzzy data and/or uncertainty. As a result, it is possible to obtain the truth-values of all propositional symbols involved in the program in a parallel way. Due to space limitations, only a subset of connectives are implemented, but the framework can easily be modified to deal with other types of fuzzy rules and/or connectives.

As future work, we will extend the framework by adding learning capabilities to the net, so that it will be able to adapt the truth-values of the rules in a given program to fit a number of observations. Following this idea, a neural net implementation for abductive multi-adjoint logic programming [5] is planned.

References

1. P. Eklund and F. Klawonn. Neural fuzzy logic programming. *IEEE Trans. on Neural Networks*, 3(5):815–818, 1992.
2. S. Hölldobler and Y. Kalinke. Towards a new massively parallel computational model for logic programming. In *ECAI'94 workshop on Combining Symbolic and Connectionist Processing*, pages 68–77, 1994.
3. S. Hölldobler, Y. Kalinke, and H.-P. Störr. Approximating the semantics of logic programs by recurrent neural networks. *Applied Intelligence*, 11(1):45–58, 1999.
4. E.P. Klement, R. Mesiar, and E. Pap. *Triangular norms*. Kluwer academic, 2000.
5. J. Medina, M. Ojeda-Aciego, and P. Vojtáš. A multi-adjoint logic approach to abductive reasoning. . Lect. Notes in Computer Science 2237, pages 269–283, 2001.
6. J. Medina, M. Ojeda-Aciego, and P. Vojtáš. Multi-adjoint logic programming with continuous semantics. Lect. Notes in Artificial Intelligence 2173, pages 351–364, 2001.
7. E. Mérida-Casermeiro, G. Galán, and J. Muñoz Pérez. An efficient multivalued Hopfield network for the traveling salesman problem. *Neural Processing Letters*, 14:203–216, 2001.
8. M. H. van Emden and R. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742, 1976.

Solving SAT in Linear Time with a Neural-like Membrane System

Juan Pazos, Alfonso Rodríguez-Patón*, and Andrés Silva

Facultad de Informática, Campus de Montegancedo s/n,
Boadilla del Monte - 28660 Madrid (SPAIN).
arpaton@fi.upm.es

Abstract. We present in this paper a neural-like membrane system solving the SAT problem in linear time. These neural P systems are nets of cells working with multisets. Each cell has a finite state memory, processes multisets of symbol-impulses, and can send impulses (“excitations”) to the neighboring cells. The maximal mode of rules application and the replicative mode of communication between cells are at the core of the efficiency of these systems.

1 Introduction

The present paper deals with a topic for further research (open problem) addressed in the paper [11]. In that paper a parallel and distributed computational model called *tissue P System* (in short, *tP system*) was introduced and defined. The efficient resolution of NP-complete problems over graphs was proposed like one possible application of these new systems. For example, it was proved that the Hamiltonian Path Problem can be solved in linear time with these tP systems (details in [11]). However, it was also proposed in that paper like a “topic for further research” the search for any other type of problems outside graph theory that could be efficiently solved by the tP systems.

In this paper we show that tP systems (also called *neural P systems* or *nP systems* in [15]) can solved in linear time a very general and classical NP-complete problem: the SAT problem.

tP systems can be seen at the same time as a contribution to neural networks (of a symbolic type), to membrane computing (with cells arranged in “tissues”), to finite automata networks (working not with strings, but with multisets of symbols), to multiset processing, to (distributed) automata and language theory. The motivation is two-fold: the inter-cellular communication (of chemicals, energy, information) by means of complex networks of protein channels (see, e.g., [1], [10]), and the way the neurons co-operate, processing impulses in the complex net established by synapses (see, e.g., [1], [2]).

The common mathematical model of these two kinds of symbol-processing mechanisms is a net of finite state devices: networks of finite-automata-like processors, dealing with symbols, according to local states (available in a finite

* Work partially supported by the Spanish Ministry of Science and Technology under Project TIC2002-04220-C03-03.

number for each “cell”), communicating through these symbols, along channels (“axons”) specified in advance. Note that the neuron modelling was the starting point of the theory of finite automata ([13], [8]), that symbol processing neural networks have a rich (and controversial) history (see [4] and its references), and that networks of string-processing finite automata have appeared in many contexts ([5], [7], [12], etc), but our models are different in many respects from all these previous models.

Having in mind the bio-chemical reality we refer to, a basic problem concerns the organization of the bunch of symbols available in each node, and the easiest and most natural answer is: no organization. Formally, this means that we have to consider *multisets* of symbols, sets with multiplicities associated with their elements. In this way, we need a kind of finite automata dealing with multisets of symbols, a topic which falls into an area of (theoretical) computer science not very much developed, although some recent (see, e.g., [6]), or not so recent (see, e.g., [3]) approaches can be found in the literature. Actually, most of the vivid area of membrane computing (*P* systems) [14, 15] is devoted to multiset processing (details at <http://psystems.disco.unimib.it/>).

The computing models proposed in [11], under the name of *tP systems*, consist of several *cells*, related by protein channels. In order to preserve also the neural intuition, we will use the suggestive name of *synapses* for these channels. Each cell has a state from a given finite set and can process multisets of *objects*, represented by symbols from a given alphabet. The standard rules are of the form $sM \rightarrow s'M'$, where s, s' are states and M, M' are multisets of symbols. Some of the elements of M' may be marked with the indication “go”, and this means that they have to immediately leave the cell and pass to the cells to which we have direct links through synapses. This communication (transfer of symbol-objects) can be done in a replicative manner (the same symbol is sent to all adjacent cells), or in a non-replicative manner; in the second case we can send all the symbols to only one adjacent cell, or we can distribute them, nondeterministically. One more choice appears in using the rules $sM \rightarrow s'M'$: we can apply such a rule only to one occurrence of M (that is, in a sequential, *minimal* way), or to all possible occurrences of M (a *parallel* way), or, moreover, we can apply a maximal package of rules of the form $sM_i \rightarrow s'M'_i, 1 \leq i \leq k$, that is, involving the same states s, s' , which can be applied to the current multiset (the *maximal* mode). By the combination of the three modes of processing objects and the three modes of communication among cells, we get nine possible behaviors of our machinery.

A way to use such a computing device is to start from a given initial configuration (that is, initial states of cells and initial multisets of symbol-objects placed in them) and to let the system proceed until reaching a halting configuration, where no further rule can be applied, and to associate a result with this configuration. Because of the nondeterminism, starting from one given initial configuration we can reach arbitrarily many different halting configurations, hence we can get arbitrarily many outputs. Another possibility is to also provide *inputs*, at various times of a computation, and to look for the outputs related to

them. Here we will consider only the first possibility, of *generative* tP systems, and the output will be defined by sending symbols out of the system.

At the first sight, such a machinery (a finite net of finite state devices) seems not to be very powerful, e.g., as compared with Turing machines. Thus, it is rather surprising to find that tP systems with a small number of cells (two or four), each of them using a small number of states (resp., at most five or four) can simulate any Turing machine, even in the non-cooperative case, that is, only using rules of the form $sM \rightarrow s'M'$ with M being a singleton multiset; moreover, this is true for all modes of communication for the minimal mode of using the rules, and, in the cooperative case, also when using the parallel or the maximal mode of processing objects. When the rules are non-cooperative and we use them in the maximal mode, a characterization of Parikh images of ET0L languages is obtained, which completes the study of the computing power of our devices (showing that in the *parallel* and *maximal* cases we do not get computational universality).

The above mentioned results obtained in [11] indicate that our cells are “very powerful”; as their power lies in using states, hence in remembering their previous work, a natural idea is to consider tP systems with a low bound on the number of states in each cell. In view of the previously mentioned results, tP systems with at most 1, 2, 3, or 4 states per cell are of interest. We only briefly consider this question here, and we show that even reduced tP systems as those which use only one state in each cell can be useful: using such a net we can solve the Satisfiability Problem in linear time (this is a direct consequence of the structure of a tP system, of the maximal mode of processing objects, and of the power of replicating the objects sent to all adjacent cells); remember that SAT is an NP-complete problem.

The power of tP systems with a reduced number of states per component remains to be further investigated. Actually, many other natural research topics can be considered, with motivations from automata and language theory (variants, power, normal forms), neural networks (learning, dynamic sets of neurons, dynamic synapses), computability (other NP-complete problems treated in this framework), dynamic systems (reachable configurations), etc.

2 Tissue P Systems

We now pass to the definition of our variant of membrane (P) systems, which can also be considered as a model of a symbolic neural net. We introduce it in the general form, then we will consider variants of a restricted type.

A *tissue P system* or a *neural P system* depending the motivation, in short, a tP or nP system, of *degree* $m \geq 1$, is a construct

$$\Pi = (E, \sigma_1, \dots, \sigma_m, syn, i_{out}), \text{ where}$$

1. E is a finite non-empty alphabet (of *chemical objects*, but we also call them *excitations/impulses*);

2. $syn \subseteq \{1, 2, \dots, m\} \times \{1, 2, \dots, m\}$ (*synapses among cells*);
3. $i_{out} \in \{1, 2, \dots, m\}$ indicates the *output cell*;
4. $\sigma_1, \dots, \sigma_m$ are *cells*, of the form $\sigma_i = (Q_i, s_{i,0}, w_{i,0}, P_i)$, $1 \leq i \leq m$, where:
 - (a) Q_i is a finite set (of *states*);
 - (b) $s_{i,0} \in Q_i$ is the *initial state*;
 - (c) $w_{i,0} \in E^*$ is the *initial multiset* of impulses;
 - (d) P_i is a finite set of *rules* of the form $sw \rightarrow s'xy_{go}z_{out}$, where $s, s' \in Q_i$, $w, x \in E^*$, $y_{go} \in (E \times \{go\})^*$ and $z_{out} \in (E \times \{out\})^*$, with the restriction that $z_{out} = \lambda$ for all $i \in \{1, 2, \dots, m\}$ different from i_{out} .

A TP system as above is said to be *cooperative* if it contains at least a rule $sw \rightarrow s'w'$ such that $|w| > 1$, and *non-cooperative* in the opposite case.

Any m -tuple of the form (s_1w_1, \dots, s_mw_m) , with $s_i \in Q_i$ and $w_i \in E^*$, for all $1 \leq i \leq m$, is called a *configuration* of Π ; $(s_{1,0}w_{1,0}, \dots, s_{m,0}w_{m,0})$ is the *initial configuration* of Π .

Using the rules from the sets P_i , $1 \leq i \leq m$, we can define *transitions* among configurations. To this aim, we first consider three *modes of processing the stimuli* and three *modes of transmitting excitations* from a cell to another one. Let us denote $E_{go} = \{(a, go) \mid a \in E\}$, $E_{out} = \{(a, out) \mid a \in E\}$, and $E_{tot} = E \cup E_{go} \cup E_{out}$. For $s, s' \in Q_i$, $x \in E^*$, $y \in E_{tot}^*$, we write

$$sx \Rightarrow_{min} s'y \text{ iff } sw \rightarrow s'w' \in P_i, w \subseteq x, \text{ and } y = (x - w) \cup w',$$

$$sx \Rightarrow_{par} s'y \text{ iff } sw \rightarrow s'w' \in P_i, w^k \subseteq x, w^{k+1} \not\subseteq x,$$

$$\text{for some } k \geq 1, \text{ and } y = (x - w^k) \cup w'^k,$$

$$sx \Rightarrow_{max} s'y \text{ iff } sw_1 \rightarrow s'w'_1, \dots, sw_k \rightarrow s'w'_k \in P_i, k \geq 1,$$

$$\text{such that } w_1 \dots w_k \subseteq x, y = (x - w_1 \dots w_k) \cup w'_1 \dots w'_k,$$

$$\text{and there is no } sw \rightarrow s'w' \in P_i \text{ such that } w_1 \dots w_k w \subseteq x.$$

In the first case, only one occurrence of the multiset from the left hand side of a rule is processed (replaced by the multiset from the right hand of the rule, at the same time changing the state of the cell), in the second case a maximal change is performed with respect to a chosen rule, in the sense that as many as possible copies of the multiset from the left hand side of the rule are replaced by the corresponding number of copies of the multiset from the right hand side, while in the third case a maximal change is performed with respect to all rules which use the current state of the cell and introduce the same new state after processing the impulses.

We also write $sx \rightarrow_\alpha sx$, for $s \in Q_i$, $x \in E^*$, and $\alpha \in \{min, par, max\}$, if there is no rule $sw \rightarrow s'w'$ in P_i such that $w \subseteq x$. This encodes the case when a cell cannot process the current impulses in a given state (it can be “unblocked” after receiving new impulses from its ancestors).

The multiset w' from a rule $sw \rightarrow s'w'$ contains symbols from E , but also symbols of the form (a, go) (or, in the case of cell i_{out} , of the form (a, out)). Such symbols will be sent to the cells related by synapses to cell σ_i where the rule $sw \rightarrow s'w'$ is applied, according to the following modes:

- *replicative* (indicated by *repl*): each symbol a , for (a, go) appearing in w' , is sent to each of the cells σ_j such that $(i, j) \in syn$;
- *unique destination* (indicated by *one*): all symbols a appearing in w' in the form (a, go) are sent to one of the cells σ_j such that $(i, j) \in syn$, nondeterministically chosen; more exactly, in the case of modes *par* and *max* of using the rules, we first perform all applications of rules, and after that we send all obtained symbols to a unique descendant of the cell (that is, we do not treat separately the impulses introduced by each rule, but all of them in a package);
- *non deterministic distribution* (indicated by *spread*): the symbols a appearing in w' in the form (a, go) are non-deterministically distributed among the cells σ_j such that $(i, j) \in syn$.

In order to formally define the transition among the configurations of Π we need some further notations. For a multiset w over E_{tot} , we denote by $go(w)$ the multiset of symbols $a \in E$ appearing in w in the form (a, go) , and by $out(w)$ the multiset of symbols $a \in E$, appearing in w in the form (a, out) . Clearly, $go(w)(a) = w((a, go))$ and $out(w)(a) = w((a, out))$, $a \in E$. Moreover, for a node i in the graph defined by syn we denote $ant(i) = \{j \mid (j, i) \in syn\}$ and $succ(i) = \{j \mid (i, j) \in syn\}$ (the ancestors and the successors of node i , respectively).

During any transition, some cells can do nothing: if no rule is applicable to the available multiset of impulses in the current state, then a cell waits until new impulses are sent to it from its ancestor cells.

A sequence of transitions among configurations of the tP system Π is called a *computation* of Π . A computation which ends in a configuration where no rule in no cell can be used, is called a *halting* computation. Assume that during a halting computation the tP system Π sends out, through the cell $\sigma_{i_{out}}$, the multiset z . We say that the vector $\Psi_E(z)$, representing the multiplicities of impulses from z , is *computed* (or *generated*) by Π .

Rather surprising, if we take into consideration the apparently weak ingredients of our models, when using the mode *min* of applying the rules, even the non-cooperative tP systems turn out to be computationally universal. More results about the computational power of the different variants of tP systems are in [11] and [15].

3 Solving SAT in Linear Time

Problems related to paths in a (directed) graph can be easily solved by a tP system, just by constructing a net with the synapses graph identical to the graph we deal with, constructing all paths in the graph with certain properties by making use of the maximal mode of applying the rules and of the replicative communication, and checking the existence of a path with a desired property. The HPP is solved in this way in [11].

The architecture of tP systems and their way of working (especially the fact that in the maximal mode of using the rules we can process all impulses which

may be processed in such a way that the same next state is obtained, irrespective which rules are used, and the fact that in the replicative mode one can send the same impulses to all successors of a cell) have an intrinsic computational power. We will show this power with the resolution of the SAT problem.

Let $F = C_1 \wedge C_2 \wedge \dots \wedge C_m$, where each clause $C_i, 1 \leq i \leq m$, is a disjunction $C_i = y_1 \vee y_2 \vee \dots \vee y_r$, with each y_j being either a propositional variable, x_s , or its negation, $\neg x_s$, for $s \in \{1, 2, \dots, n\}$. The SAT problem ask whether or not there is a truth-assignment of the variables that makes the formula true. Note: We will use t_i (respectively f_i) to represent $x_i = \text{true}$ (resp. $x_i = \text{false}$).

We construct the tP system Π with the following components:

$$\Pi = (E, \sigma_S, \sigma_{t_1}, \sigma_{f_1}, \dots, \sigma_{t_n}, \sigma_{f_n}, \sigma_E, \sigma_{y_{j,i}}, \sigma_O, syn, O),$$

where $y_{j,i}$ are the variables y_j in the clause C_i for $j \in \{1, 2, \dots, n\}$ and $i \in \{1, 2, \dots, m\}$.

$$\begin{aligned} E &= \{z \mid z = \{\lambda\} \text{ or } z = g_1 g_2 \cdots g_i, \text{ for } g_i = t_i, f_i, 1 \leq i \leq n\}, \\ \sigma_S &= (\{s\}, s, \lambda, \{s\lambda \rightarrow s(\lambda, go)\}) \\ \sigma_{t_i} &= (\{s\}, s, t_i, \{sz \rightarrow s(zt_i, go)\}, \text{ for each } i = \{1, 2, \dots, n\}, \text{ and} \\ &\quad z = \lambda \text{ or } z = g_1 g_2 \cdots g_i, \text{ for } g_i = t_i, f_i, 1 \leq i \leq n\}), \\ \sigma_{f_i} &= (\{s\}, s, f_i, \{sz \rightarrow s(zf_i, go)\}, \text{ for each } i = \{1, 2, \dots, n\}, \text{ and} \\ &\quad z = \lambda \text{ or } z = g_1 g_2 \cdots g_i, \text{ for } g_i = t_i, f_i, 1 \leq i \leq n\}), \\ \sigma_E &= (\{s\}, s, \lambda, \{sz \rightarrow s(z, go)\} \mid z = g_1 g_2 \cdots g_n, |z| = n), \\ \sigma_{y_{j,i}} &= (\{s\}, s, \lambda, \{sz \rightarrow s(z, go)\} \text{ if } t_j \in z \text{ and } C_i \text{ contains } x_j, \\ &\quad \text{or if } f_j \in z \text{ and } C_i \text{ contains } \neg x_j\}), \\ \sigma_O &= (\{s\}, s, \lambda, \{sz \rightarrow s(z, out)\}), \\ syn &= \{(S, g_1), (g_k, g_{k+1}), (g_n, E), (E, y_{j,1}), (y_{j,i}, y_{j,i+1}), (y_{j,m}, O) \mid \\ &\quad g_k = t_k, f_k \text{ for } k = \{2, \dots, n-1\}, i = \{2, \dots, m-1\} \text{ and } j = \{1, 2, \dots, n\}\}, \\ O &= \text{ Output membrane.} \end{aligned}$$

It is easy to see that $N_{max,repl}(\Pi) \neq \emptyset$ if and only if the SAT problem has a solution. This system works as follows. There are two phases. In the first one the system generates all the truth-assignments in the form of strings of length n composed of $t_i, f_i, 1 \leq i \leq n$ in all possible combinations (this takes n steps). The strings are constructed starting in membrane S with sequential concatenations. In each membrane g_i the corresponding symbol g_i for $g_i = t_i, f_i, 1 \leq i \leq n$ is concatenated and the resulting strings are replicated to the following membranes g_{i+1} . The generation phase ends with the addition of the last symbol $g_n = t_n, f_n$ forming the 2^n truth-assignments z that reach the membrane E . Note that the strings z are manipulated as symbols through the tP system. In figure 1 we can see the topology of the generative membranes.

In membrane E starts the second phase: the computational or "filtering" phase (resembling the filter steps of the Lipton's algorithm, see [9]). This second phase is a sequence of m steps or filters, one for each clause of the formula. In

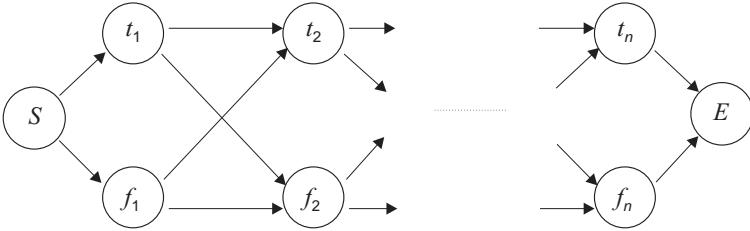


Fig. 1. Generation of the the 2^n truth-assignments z of length n .

the first step (membranes $y_{j,1}$) only the strings z that satisfies the first clause C_1 are allow to go to the next level. In other words, only the strings z such that $t_j \in z$ and C_1 contains x_j , or the strings z such that $f_j \in z$ and C_i contains $\neg x_j$ are allow to travel to the second level (filter). These filters are repeated until the level m . If some z reaches the output membrane O , the formula is satisfiable and the symbol z sent out of the system encodes a SAT assignment. If no symbol z is sent out after $n+m$ steps, the formula is no satisfiable. For example, the filter membranes for the formula $F = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$ are shown in figure 2.

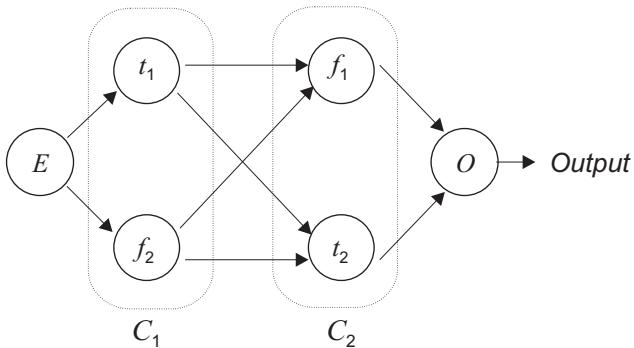


Fig. 2. Computation phase: two "filtering" steps across membranes.

Therefore, our tP system (or nP system) can solve the SAT problem in linear time: n steps for generating all truths-assignments and m steps to check the truth value of each of the m clauses. (Note that in our calculations of time steps we have not count the two steps to/from cell E . We could suppress the E cell but for sake of clarity we have maintained it in the system acting like delimiter between the two phases).

4 Conclusions

We have shown that neural-like membrane systems can solve in linear time not only graph theory problems but also the SAT problem. There are a lot of topics for further research pointed in [11]. One more topic for further research could be the interpretation of the SAT bio-algorithm presented here like a special ballistic computation. In some sense the objects z cross the obstacles imposed by the clauses. The SAT problem has solution if some z arrives at the end of the “clause labyrinth”.

Acknowledgments. Thanks to Prof. Gheorghe Păun for his clever comments and suggestions.

References

1. B. Alberts et al., *Essential Cell Biology. An Introduction to the Molecular Biology of the Cell*, Garland Publ. Inc., New York, London, 1998.
2. M.A. Arbib, *Brains, Machines, and Mathematics*, second ed., Springer-Verlag, Berlin, 1987.
3. J.P. Banatre, P. Fradet, D. LeMetayer, Gamma and the chemical abstract reaction model: fifteen years after, in vol. *Multiset Processing. Mathematical, Computer Science, and Molecular Computing Points of View* (C.S. Calude, Gh. Păun, G. Rozenberg, A. Salomaa, eds.), *Lecture Notes in Computer Science*, 2235, Springer-Verlag, 2001, 17–44.
4. D.S. Blank *et al* (24 co-authors), Connectionist symbol processing: Dead or alive?, *Neural Computing Surveys*, 2 (1999), 1–40.
5. C. Choffrut, ed., *Automata Networks*, *Lecture Notes in Computer Science*, 316, Springer-Verlag, Berlin, 1988.
6. E. Csuha-J-Varju, C. Martin-Vide, V. Mitrana, Multiset automata, in vol. *Multiset Processing. Mathematical, Computer Science, and Molecular Computing Points of View* (C.S. Calude, Gh. Păun, G. Rozenberg, A. Salomaa, eds), *Lecture Notes in Computer Science*, 2235, Springer-Verlag, 2001, 67–82.
7. F. Gecseg, *Products of Automata*, Springer-Verlag, Berlin, 1986.
8. S.C. Kleene, Representation of events in nerve nets and finite automata, *Automata Studies*, Princeton Univ. Press, Princeton, N.J., 1956, 2–42.
9. R. J. Lipton, Using DNA to solve NP-complete problems, *Science*, 268 (April 1995), 542–545.
10. W.R. Loewenstein, *The Touchstone of Life. Molecular Information, Cell Communication, and the Foundations of Life*, Oxford Univ. Press, New York, Oxford, 1999.
11. Carlos Martín-Vide, Gheorghe Păun, Juan Pazos and Alfonso Rodríguez-Patón, Tissue P systems, *Theoretical Computer Science*, vol. 296, issue 2, (2003), 295–326.
12. A. Mateescu, V. Mitrana, Parallel finite automata systems communicating by states, *Intern. J. Found. Computer Sci.*, to appear.
13. W.S. McCulloch, W.H. Pitts, A logical calculus of the ideas immanent in nervous activity, *Bull. Math. Biophys.*, 5 (1943), 115–133.
14. Gh. Păun, Computing with membranes, *Journal of Computer and System Sciences*, 61, 1 (2000), 108–143 (see also *Turku Center for Computer Science-TUCS Report No 208*, 1998, www.tucs.fi).
15. Gh. Păun, *Membrane Computing*. Springer-Verlag, 2002.

An Exponential-Decay Synapse Integrated Circuit For Bio-inspired Neural Networks.

Ludovic Alvado, Sylvain Saïghi, Jean Tomas, Sylvie Renaud

Laboratoire IXL, CNRS UMR 5818, ENSEIRB-Université Bordeaux 1
351 cours de la Libération, 33405 Talence, France
alvado@ixl.u-bordeaux.fr

Abstract. Biologically-realistic artificial neurons and synapses are implemented on analog specific integrated circuits, to offer real-time computation. We present here specific circuits where synaptic interactions are modeled using exponential-decay synapses, which are digitally controlled. The design method is detailed, and experimental measurements on fabricated circuits are presented to illustrate the synaptic mechanisms. We finally propose to exploit the synapses to build a mixed analog-digital neural simulator, where an analog computing core is digitally controlled to simulate synaptic plasticity or learning algorithms.

1 Introduction

Experimentally-based neuron models are now currently used in computational neurosciences. However, numerical simulation systems (software or hardware) present a limited processing speed. Analog computation is an interesting alternative: it runs in real time and computes analog currents and voltages, like real neurons. We designed specific analog integrated circuits (ASICs), which exploit the intrinsic voltage-current relationships of individual transistors (bipolar and MOSFET) to simulate the membrane equation of neurons, including voltage-dependent conductances [1]. Synaptic conductances can also be modeled using a similar representation. The model variables (state variables, conductances, ionic currents, membrane voltage) are identified to electrical variables in the circuit (currents, voltages). These artificial neurons were used for computational neuroscience studies at single-cell level, as well as for hybrid experiments, in which hardware artificial neurons are connected to living neurons, via artificial synapses and intracellular micro-electrodes [2].

To address neural computation at the network level, we now integrate specific circuits where synaptic interactions are modeled using exponential-decay synapses. In that case, the synaptic conductance increases instantaneously of a given step value when a pre-synaptic spike occurs, then relaxes exponentially to zero; the neural activity and the pre-synaptic information are digitally coded. The synaptic circuits are to be integrated in a set-up where they will be associated to artificial conductance-based neurons we already developed [3]. A digital interface will read the neuron's activity

events (spikes), and dispatch them as inputs to other neurons according to a flexible connection matrix. Software computed algorithms eventually modulate in real-time the value of the conductance step. This system is specially relevant to the study of plasticity and learning rules, for which it will provide large flexibility.

2 Model

Studies showed that the macroscopic behavior of voltage-dependent ionic currents can be captured using kinetic models that describe the transitions between states of the ion channels [4]. This class of models is known as “Markov models”, among which one finds the Hodgkin-Huxley model. If we focus on synaptic interactions, detailed kinetic models (see [5] for a review) exist for synaptic release and for different types of synaptic currents and receptors. Simplified models with fewer states however exhibit essential properties of synaptic currents (for example the summation of post-synaptic currents). They are computationally efficient, and are useful in accurately representing synaptic transmissions in network simulations.

We chose to implement a model in which both the receptor kinetic model and the release process that determines the transmitter concentration T are simplified. T is pulse-shaped, assuming that the transmitter is released when a pre-synaptic action potential occurs. The receptor kinetic scheme is a simple two-state (open-close) pattern. This model was shown for example to represent correctly the AMPA and GABA_A receptors schemes [6].

The associated post-synaptic current I_{SYN} is given in equation (1), where g_{MAX} is the maximal conductance, E_{SYN} the reverse synaptic potential, V_{MEM} the post-synaptic membrane potential, r the fraction of receptors in open state, α and β voltage-independent forward and backward rate constants, $[T]$ the transmitter concentration.

$$\begin{aligned} I_{SYN} &= g(V_{MEM} - E_{SYN}) = g_{MAX}r(V_{MEM} - E_{SYN}) \text{ with} \\ \frac{dr}{dt} &= \alpha[T](1-r) - \beta r \end{aligned} \quad (1)$$

The following figure illustrates the time-variation of the synaptic conductance g when a transmitter pulse occurs. As the quantum Δg is proportional to the $[T]$ pulse width, this later parameter will be exploited to modulate Δg . Furthermore, synaptic summation, which occurs when multiple pre-synaptic spikes are simultaneously presented, will be handled naturally by the integration of successive transmitter pulses.

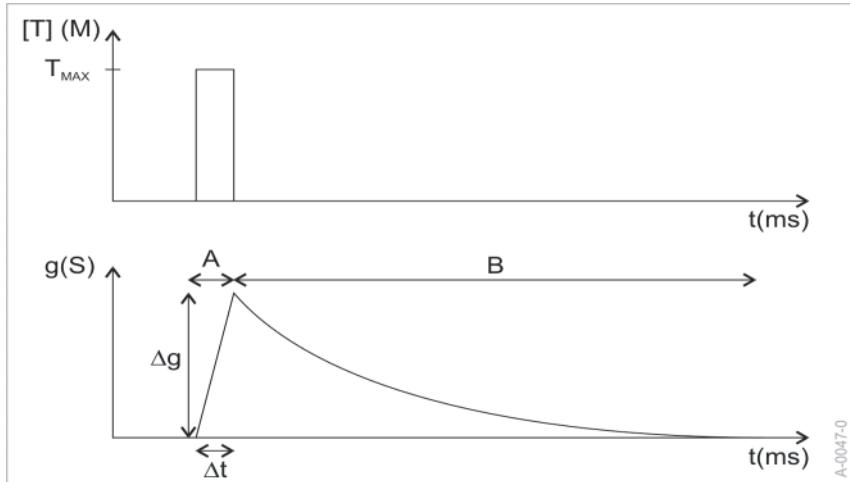


Fig. 1. Phase A: $t \in [0, \Delta t]$; a $[T]$ transmitter pulse occurs during Δt ($[T] = T_{MAX}$, $r_\infty = r_A$), the state variable r increases by Δr , which is proportional to the pulse width and results in a step Δg of the synaptic conductance g . Phase B: $t > \Delta t$; $[T]=0$, $r_\infty=0$; g then decays exponentially with a $\tau=\tau_B$ time constant.

3 Method

A specific integrated circuit was designed to generate the synaptic currents of an exponential-decay synapse. To ease electronics integration, we chose to write equation (1) as follows, where r_∞ is null except during the transmitter pulse.

$$\tau \frac{dr}{dt} = r_\infty - r \text{ with } \tau = \frac{1}{\alpha[T] + \beta} \text{ et } r_\infty = \frac{\alpha[T]}{\alpha[T] + \beta} \quad (2)$$

Equation (2) represents the transfer function of a first-order circuit, which implementation methods are well-known. We chose to use a RC structure to generate the exponential-decay effect; during the $[T]$ pulse, the RC sub-circuit is loaded by a digitally-controlled current source. After the pulse, C discharges through the resistor R (see figure 2). The voltage value across R is proportional to the state variable r , later amplified and multiplied by $(V_{MEM} - E_{SYN})$ and g_{MAX} to generate I_{SYN} .

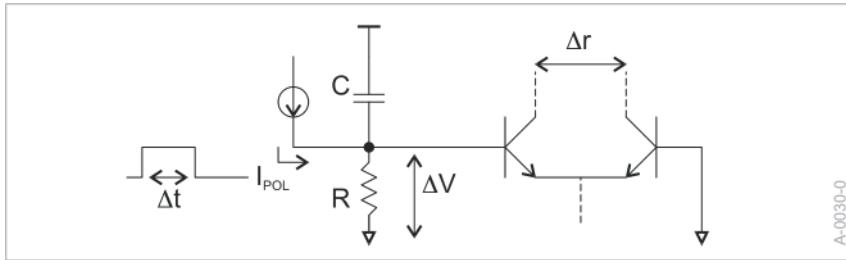


Fig. 2. The exponential decay is realized by a RC circuit. During the transmitter pulse, the current source provides a non-zero current I_{POL} , that quickly increases the voltage ΔV . After the pulse, I_{POL} goes to zero, and ΔV exponentially decreases as C discharges. The bipolar differential pair provides Δr proportional to ΔV . Δr will be later multiplied by the voltage difference ($V_{MEM}E_{SYN}$) and amplified with a gain proportional to g_{MAX} .

At $t = \Delta t$, $[T] = T_{MAX}$, $r_\infty = r_{\infty A}$, $\tau = \tau_A$ (end of Phase A, see figure 1), equation (2) gives:

$$r(t) = r_{\infty A} \left(1 - e^{-\frac{t}{\tau_A}} \right) \quad (3)$$

$$\Delta r = r(\Delta t) = r_{\infty A} \left(1 - e^{-\frac{\Delta t}{\tau_A}} \right) \approx r_{\infty A} \left(1 - 1 + \frac{\Delta t}{\tau_A} \right) = r_{\infty A} \frac{\Delta t}{\tau_A} \quad (\Delta t \ll \tau_A) \quad (4)$$

$$\text{Then, with (2):} \quad \Delta r = \alpha \Delta t T_{MAX} \quad (5)$$

Now, from the circuit in figure 2, still at $t=\Delta t$:

$$\Delta V = RI_{POL} \left(1 - e^{-\frac{\Delta t}{RC}} \right) \approx RI_{POL} \left(1 - 1 + \frac{\Delta t}{RC} \right) \quad (\Delta t \ll RC) \quad (6)$$

$$\Delta V = \frac{I_{POL}}{C} \Delta t \quad (7)$$

ΔV is the input of the bipolar differential pair [7]. The output Δr is:

$$\Delta r = \tanh \left(\frac{\Delta V}{V_t} \right) \quad (8)$$

The I_{POL} source is a proportional to absolute temperature current-source [8]:

$$I_{POL} = \frac{V_t}{R_{POL}} \ln(n) \quad (9)$$

$$\Delta r = \tanh\left(\frac{\Delta V}{V_t}\right) = \tanh\left(\frac{V_t}{R_{POL}} \ln(n) \frac{\Delta t}{C} \frac{1}{V_t}\right) \approx \frac{\Delta t \ln(n)}{R_{POL} C} \quad (10)$$

The C value comes from identification of (5) and (10):

$$\Delta r = \frac{\ln(n)}{R_{POL} C} \Delta t = \alpha T_{MAX} \Delta t \Rightarrow C = \frac{\ln(n)}{R_{POL} \alpha T_{MAX}} \quad (11)$$

For $t > \Delta t$, $[T] = 0$, $r_\infty = 0$, $\tau = \tau_B$ (Phase B, see figure 1), we deduce R :

$$\tau_B = \frac{1}{\beta} = R C \Rightarrow R = \frac{R_{POL} \alpha T_{MAX}}{\beta \ln(n)} \quad (12)$$

The whole circuit is integrated on a chip, except for R and C which are discrete elements connected externally, due to their prohibitive silicon area. Parameters are tuned to fit the synapse biological model (equation (1), numerical values deduced from [1]). Currents have a $\times 1000$ gain and voltages a $\times 10$ gain to be compatible with the integrated circuit technical constraints. The resulting variation range is $\pm 10\mu A$ for the currents and $\pm 1V$ for voltages.

4 Result

The circuit was integrated as an ASIC (application specific integrated circuit) using the 0.8 microns BiCMOS process of Austriamicrosystems. This process provides both bipolar and MOS transistors, and has already been used for previous ASICs we implemented. The layout of a single synapse is presented on figure 3. It comprises 300 transistors, within an area of $460\mu m \times 200\mu m$. Various layout techniques were applied to overcome process unrepeatability, which can cause discrepancies between circuits and so be the cause of a loss of accuracy in the I_{SYN} calculation. Particular attention was paid to the matching rules between sensible elements: we integrated common-centroïd blocks and interdigitated devices [9]. We can observe in figure 3 the high level of symmetry of each internal block.

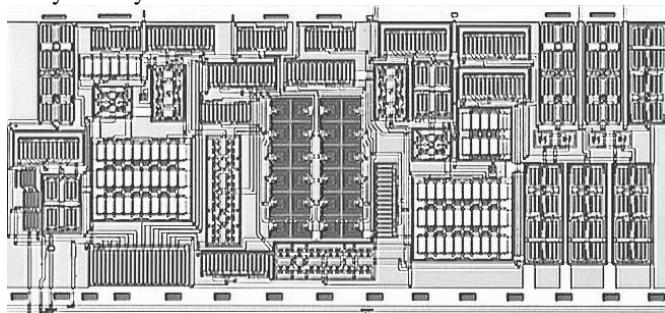


Fig. 3. Microphotograph of a synapse ASIC (area: $460\mu m \times 200\mu m$, 300 transistors).

Figures 4 and 5 present experimental measurements on prototype synapse circuits, with $R_{POL}=348\Omega$, $n=4$, $R=1k\Omega$ and $C=1\mu F$. The corresponding decay time constant is 1ms. Measurements show the synapse input and the resulting ΔV signal, which represents the synaptic conductance. The synaptic current is not shown here, as we don't consider a post-synaptic neuron. In figure 4, the conductance presents a typical exponential decay response to a single digital pulse. In figure 5, two successive pulses of different width simulate two pre-synaptic events of different strength. A cumulative effect appears on the synaptic conductance, and demonstrates its “integrative” property. In both cases, we verify that the kinetic of the decay is 1ms.

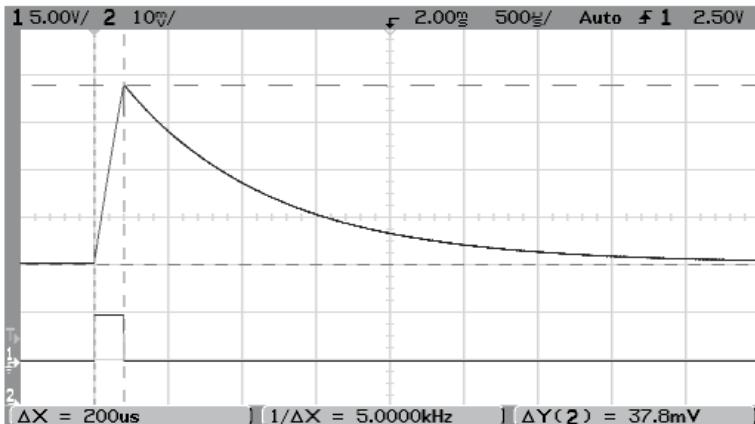


Fig. 4. Measurement on the synapse circuit: the conductance response (channel 2, highest trace) to a single pulse (channel 1, lowest trace)

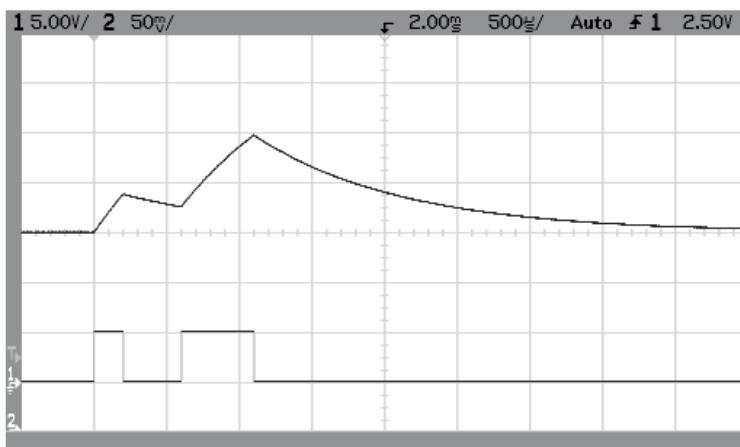


Fig. 5. Measurement on the synapse circuit: the conductance response (channel 2, highest trace) to successive pulses (channel 1, lowest trace)

5 Conclusion

To evaluate the synaptic circuit performances when integrated in a neural network, we integrated in an ASIC a minimum system: a Morris and Lecar 3-conductances neuron [10] receives two synaptic currents, one from an excitatory synapse and one from an inhibitory synapse. The pre-synaptic inputs are digital pulses generated by a digital microcontroller. The pulse pattern can then be easily programmed to study the neuronal events in different synaptic configurations. The neuron voltage membrane enters an integrated comparator with hysteresis; digital outputs of the comparator are available to represent a spiking event. The whole system runs in real time.

Figure 6 plots the Morris and Lecar neuron membrane voltage, with the associated pulse pattern applied on the excitatory synapse. Synaptic events are set to be strong enough (by tuning Δt) and spaced enough to trigger an action potentials.

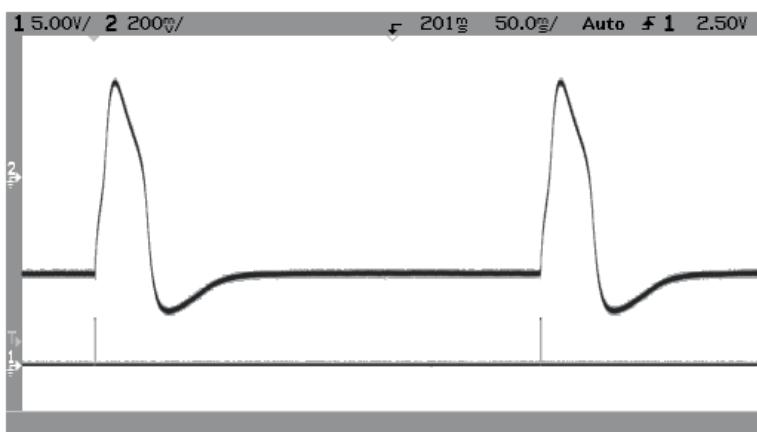


Fig. 6. A Morris and Lecar neuron with an excitatory synapse input: the membrane voltage (channel 2, highest trace) and the pre-synaptic events (channel 1, lowest trace). Pulses are spaced out of 300ms, to ensure they are not applied during the action potential refractory period.

For more complex applications, a few neurons and synapses will be implemented on the same chip, with a programmable connectivity. A digital interface will control both inputs and outputs, with possible back-propagation or synaptic weights modulation. The whole system will form a mixed analog-digital neural simulator, where the analog computing core will digitally controlled to simulate synaptic plasticity or learning algorithms.

Exponential decay synapses provide a unique opportunity to integrate digitally controlled synaptic interactions while keeping a good accuracy for biologically-realistic modeling. Moreover, those circuits keep the real-time computation property which is a key point to run simulations at network level.

6 References

- [1] V. Douence, S. Renaud-Le Masson, S. Saïghi, G. Le Masson, A field-programmable conductance array IC for biological neurons modeling , *IWANN'2001*, Grenada (Espagne), 2001.
- [2] G. Le Masson, S. Renaud-Le Masson, D. Debay, T. Bal, Feedback inhibition controls spike transfer in hybrid thalamic circuits, *Nature*, 417, 854-858, 2002.
- [3] S. Le Masson, A. Laflaquerre, D. Dupeyron, T. Bal, G. Le Masson, Analog circuits for modeling biological neural networks: design and applications, *IEEE Transactions on Biomedical Engineering*, 46- 6, 638-645, 1999.
- [4] L.F. Abbott, Single neuron dynamics: an introduction, in *Neural modeling and neural networks*, Pergamon Press, Oxford, 57-78, 1994.
- [5] A. Destexhe, Z.F. Mainen, T.J. Sejnowski, Kinetic models of synaptic transmission, in *Methods in Neuronal Modeling*, Eds C. Koch and I. Segev, MIT Press, Cambridge (MA-USA), 1-26, 1999.
- [6] Destexhe, A., Mainen, Z., Sejnowski, T., An efficient method for computing synaptic conductances based on a kinetic model of receptor binding, *Neur. Comp.*, 6, 14-18 (1994).
- [7] P.G. Gray, P.J. Hurst, S.H. Lewis, R.G. Meyer, *Analysis and design of analog integrated circuits*, Ed. John Wiley and Sons, New York, 2001.
- [8] C. Toumazou, F.J. Lidgey, D..G. Haigh, *Analogue IC design: the current-mode approach*, Peter Peregrinus, London, 1990.
- [9] C. Saint, J. Saint , *IC Mask Design*, McGraw-Hill, New-York, 2002.
- [10] C. Morris, H. Lecar, Voltage oscillations in the barnacle giant muscle fiber, *Biophys. J.*, 35, 193-213.

A high level synthesis of an auditory mechanical to neural transduction circuit.

Ferrández J.M.^{1,2}, Sacristán M.A^{2,3} Rodellar V.³, Gomez P.³

¹Instituto de Bioingeniería, U. Miguel Hernández, Alicante,

²Dept. Electrónica y Tecnología de Computadores, Univ. Politécnica de Cartagena,

³Lab. Com. Oral R.W. Newcomb, Univ. Politécnica Madrid

Corresponding Author: jm.ferrandez@upct.es

Abstract. In this paper implementation of an inner hair cell model, including macromechanics and mechanical to neural transduction process is presented and discussed. The well-known Meddis model will be used as the reference system, and a high level synthesis will provide a parametrizable implementation and a reusability code, which allows future refinements on an FPGA system.

1. Introduction

Speech processing has undergone an important step forward during the last years. Nevertheless it is well known that the most efficient techniques and algorithms for speech processing and recognition do not reach the performance attainable by biological systems. The development of algorithms with a clear biological parallelism may be useful not only in artificial prosthesis design, but also in the study of the basic principles involved and the production of new and better man-machine interfaces. Artificial bio-inspired systems behave adequately in general but demand very high computational costs, which render the inadequate for many real-time applications. The work herein described is substantially focused to the implementation of an ASIC showing the behavior of the inner hair cells on the inner auditory system, with the function of converting arriving acoustical stimuli into trains of neural pulses to be carried to the higher auditory centers. The proposed system may be used as a front-end processor in bio-inspired speech processing and recognition applications.

2. Auditory system processing

Sound is produced by vibrations-for example the movement of speakers' diaphragms, piano strings or vocal cords-that result in the alternating compression and rarefaction (increased or decreased pressure) of the surrounding air. Sounds reach the middle ear through the auditory canal, causing the tympanic membrane to vibrate. This vibration is then conveyed through the middle ear by a series of small bones, one of which, the malleus is attached to the tympanic membrane. The function of the middle ear is

matching impedances. It ensures that sounds from a gaseous medium (air) that fills the auditory canal are transmitted efficiently to the fluid that replenish the cochlea (perilymph). If the middle ear were absent, only 3% of the signal energy would arrive to the cochlea.

The vibration of the middle ear is transmitted to an opening in the cochlea, the oval window, which oscillate, and these oscillations transmit energy to each of the three compartments (scala tympani, scala vestibuli and scala media) of the cochlea. This energy cause vibration of the basilar membrane, the floor of the scala media, where a sensory apparatus (organ of Corti) transforms these vibrations into electrical impulses.

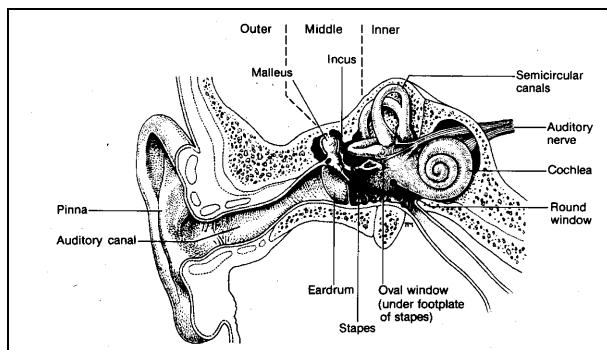


Fig. 1 Auditory periphery

The basilar membrane has cross striations, much like the strings of a piano, and its apical end is much wider than the basal one, so striations resonate with different frequencies, but the overall behaviour is in the form of a travelling wave, so a single frequency stimulation cause a very broad area displacement in the basilar membrane.

It is important to note that different frequencies of sound produce different travelling waves with different peak amplitudes. These peaks code different frequency stimuli in the basilar membrane. Also these peak amplitudes will excite different hair-cell at different positions in the cochlea which will be responsible for the mechanical to neural transduction process that propagates electrical impulses to higher neural centers through a tonotopically organized array of auditory nerve fibers. Each auditory nerve fiber is specialized in the transmission of a different characteristic frequency and the rate of the transmitted pulses by these pathways code not only the frequency intensity information, but also certain features of the signal relevant for discrimination purposes.

One important aspect of transduction is adaptation. It is important for sensory systems to ignore strong and continuous stimuli, and to be very sensitive to small ones, which will be more relevant for recognition. Initially there is a certain amount of

neurotransmitter substance (approximately 10 % of the total amount) in the synaptic cleft that will cause the spontaneous firing of the fiber. When the stimulus begins, a peak is formed in the amount of neurotransmitter released, followed by an adapting curve caused by re-capture of substance by the cell and by hydrolysis process in the medium. When the stimulus ends, there is a transient decrease in the firing rate, below the spontaneous level, followed by a new adapting curve that reaches spontaneous level again. This mechanism is common to all sensory systems.

The set of auditory nerve fibers, jointly with the vestibular fibers are grouped in the eighth auditory nerve, which conducts the set of pulses to the central nervous system. The next processing center is the cochlear nucleus where several actions must take place. First, input information is mapped in different areas, where different kinds of neurons are specialized in different kinds of processing (some of them segment the signals, others detect the onset of stimulus in order to locate it by interaural differences, others delay the information to detect the temporal relationship...etc.). Second, it sends information back to the cochlea in order to sharp the response and attenuate and protect the organ of Corti from overstimulation. The last function is to feedforward information to the Olivary Complex, where sounds are located, and the Inferior Colliculus, where the duration of signals is determined. This center sends information to the thalamus (medial geniculated nucleus) which acts as a relay station for prior representations, and as a tonotopic mapper of arriving information to cortex. And because it is the last step to the cortex in the auditory pathway, we can switch off the information from sensory receptors arriving to the cortex in deep sleep (theta waves in EEG) just by inhibiting this area.

3. The bioinspired model

The cochlear mechanics behavior is represented by means of a transmission line model which is linear and unidimensional. It has been properly transformed in a time domain digital filter [1] in order to allow its computation by numeric processors taken into account the need of having a certain flexibility of altering the biological parameters of the model. The resulting digital filter is responsible of separating the different frequencies present in the speech trace [2]. This separation is done according similar principles to the ones which take place in the real ear, converting the speech trace V_n to a set of 6 dimensional spectral vectors X_n , n being the time index. The model reduces the redundancy present in the spectral characteristics of the speech input signal by bank-filtering the speech trace and detecting the envelope of its main spectral components present in the filtered output. The envelopes once resampled constitute the 6-dimensional output vector X_n .

The transduction model is based on the work from Meddis [3] as it was mentioned before. The main characteristics of the model are that the relation-firing rate/intensity and temporal adaptation are in agreement with experimental results and it also reproduces the phase-locking effect.

The model receives as input the envelope of the volumetric velocity of the basilar membrane and according to it calculates the postsynaptic-potential depending on the hair cell response in terms of changes in the permeability of the membrane and from it a train of spikes which are the electrical response of the auditory nerve fibers are generated. The main idea consists in using a probabilistic model for the neurotransmitter release from the hair cell, auditory neurons and discharge patterns.

The model considers three neurotransmitters storing areas, the pre-synaptic area where the vesicles are full of neurotransmitters, the cellular body where the neurotransmitters are recollected and encapsulated for its posterior use and the synaptic cleft. The amount of neurotransmitters present in the synaptic cleft is basically a function of stimulus intensity, the neurotransmitters taken back to the cell and the neurotransmitters lost in the cleft (Figure2).

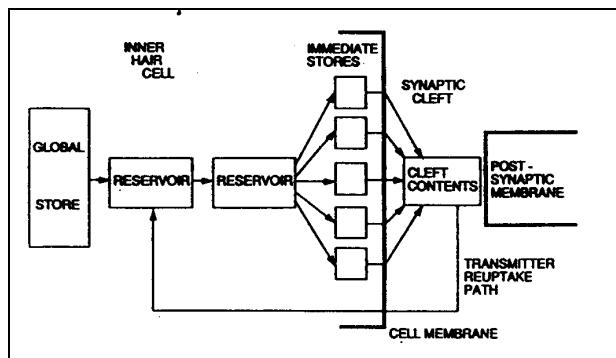


Fig. 2 Representation of the model

The results presented below can be considered as an extension of the ones presented in the original work by Meddis, because he uses 1 KHz. sinusoids and sinusoid bursts with silences of variable intensity as inputs to the model and we use the spectral characteristics of the sound. Simulations agree with the experimental data obtained by Furukawa [4], that is, this number represents the amount of neurotransmitters in synaptic cleft.

4. Model implementation

Circuit design methodology applied has followed the ideas of design for reusability. The idea of reusability allows the parameterisation of all constants of the model, which permits the study of the model behavior dependency on this parameters, with no changes on the code, just setting up the new values. Another important aspect to be taken into account is the compromise between the precision of the results and the sources involved in the solution. Having in mind a fixed point arithmetic implementation, the two key points in this compromise are word lengths and fixed

point data formats (position of the decimal point). Both aspects have been parameterized in the design. The last aspect considered deals with the performance of the design in terms of area and the critical pathway delay. The parameter area is difficult to predict when automatic design tools are used. Concerning time delays special care should be taken in the functional unit strategy implementation to try maintaining the maximum independence between the delay increments regarding word length.

VHDL has been used as the hardware description language. A circuit involving the behavior expressed in Meddis equations has been implemented. The units required were adders, subtractors, multipliers and dividers. To proceed with the design, the basic functional units were reusable ones with parameterisation of the word length, data format and the delay path of each unit. To maintain some degree of independence between the delay and the word length, redundant binary adders were included for the addition of the partial products in the multiplier as these are carry free adders [6]. The operation of division is based on successive products [7], and is physically designed re-using a multiplier as library element.

The main characteristics of the electronic circuit were a data format with 26 bits, with 10 bits for the integer part and 16 bits for the fraction. The resulting structure is shown on Figure 3. It is composed of one divisor, two adders, one adder/substractor, and six multipliers. The integration of a fiber requires 9592 core cells, as a result of direct automatic synthesis. The critical pathway delay is 62.386 nsec which allows a clock speed of 16.029 MHz.

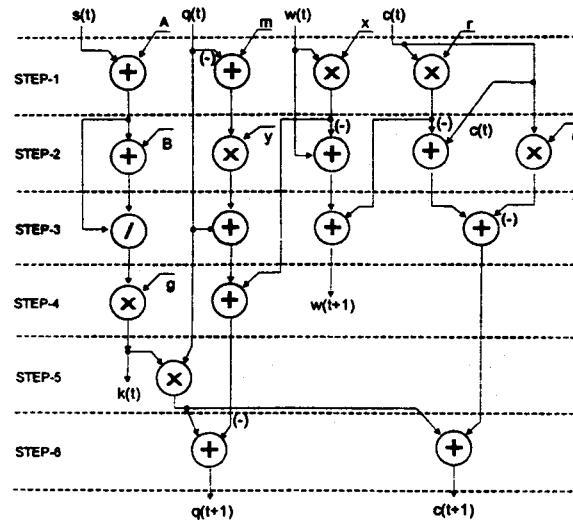


Fig. 3 Data Flow representation.

5. Results

The test data are obtained from the model response to the basilar membrane displacement. This model has been fed with speech traces containing the five vowels of Spanish (/a/, /e/, /i/, /o/, and /u/) sampled at a frequency of 22050 HZ. These sounds are characterized by spectral bands (the pitch given by f_0 and the first two formants, f_1 and f_2). The pitch is not relevant to speech processing but f_1 and f_2 are crucial ones. The result obtained in processing vowel /e/ with the basilar membrane model are shown in Figure 4. The x axis lists the sections, while the y and z axis represents the amplitude and displacement respectively.

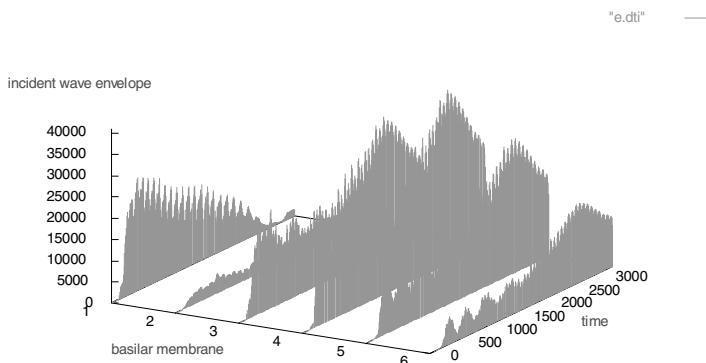


Fig. 4 Temporal evolution of the basal membrane activity for Spanish vowel /e/

It may be observed that the main activation are sections 3, 4 and 5, where lie the main formants, which consists in 530 Hz, and 1840 Hz. It may be seen that the first section responds, as it includes the round window where incident waves arrives. The second section does not show any relevant activity, while third section models second formant , and sections 4 and 5 represent first formant altogether in a temporal (phase-locking) code. Finally section 6 is activated by pitch. These results are in agreement with experimental data.

Figure 5 shows the results of the circuit implemented for the data presented in Figure 4. The response of the six fibers corresponding with the prior six sections may be observed. In this case axe y represents the amount of neurotransmitter responsible of generating the postsynaptic potential. In the proximity of $t=0$ a very low amount of neurotransmitters are present in the cleft $c(t)$, which will cause the spontaneous firing of the post-synaptic neuron. When the envelope reaches a certain level the maximum

neurotransmitter released can be observed, the value and stability of the peak will discriminate among the different vowels and after the peak the temporal adaptation is reached as observed experimentally. Different values of the peaks will reveal the energy of the formants. Each peak represents an increasing firing rate of the fibers. According with Secker works [5], intervals between these peaks may code the basic formants of phonemes and their transitions may be represented by the relative behavior of peaks between different fibers, so these intervals may provide the key for discriminating phonemes. Another characteristic observed in mechanical to neural transduction process is noise reduction. So the main peaks and its intervals remain uncorrupted under the speech recognition point of view.

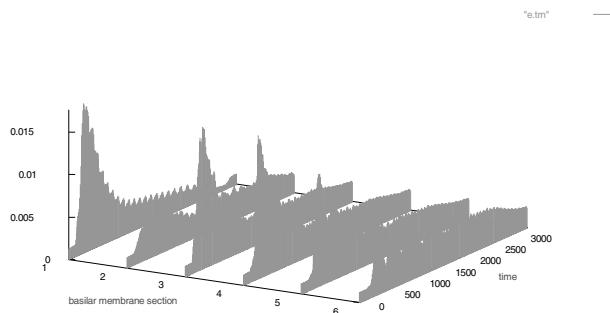


Fig. 5 Transmitter amount in the synaptic cleft for Spanish vowel /e/

6. Conclusions

Through this work the digital implementation of a circuit supporting the model for the hair cells has been described. The design methodology being followed has been based on Design with Reusability, and the implementation technique based on FPGAs allow rapid prototyping of electronic systems.

The results obtained from the implementation of the Meddis model behave reasonably well, as they are essentially in good agreement with the ones obtained experimentally. It is expected that these will be refined in the near future adding more sections to the model without increasing the computational requirements, to better emulate both cochlear macro and micromechanics. Similarly, it is also expected that the new implementations will reduce the size and the response time of the resulting circuits, reaching higher clock speeds.

Acknowledgements

We would like to thank J. Sanchez for the VHDL simulation. This research is being funded by F. Seneca OI-26/00852/FS/01 to JM.F and MA.S

References

- 1 Rodellar V., Gómez P., Hermida M. and Moreno A. 1993 "A numerical method based on Padé's Approximation to simulate and design a low-cost auditory filter for speech processing." *Simulation Practice and Theory*, Elsevier Publishers, Vol.1, No.1, pp. 17-29.
- 2 Gómez P., Rodellar V., Hermida M., and Díaz A. 1986 "VLSI implementation of a Digital Filter for Hearing Aids." *Proceedings of the 1987 International Conference on Digital Signal Processing*, V. Capellini et. al. Eds., North Holland, pp. 341-345.
- 3 Meddis R. 1986 "Simulator of mechanical to neural transduction in the auditory receptor." *J. Acoust. Soc. Am.* 79 (3), pp. 702-711.
- 4 Furukawa T. and Matsuura S. 1978 "Adaptive Rundown of excitatory post-synaptic potentials at synapses between hear-cells and eighth nerve fibers in goldfish." *J. Physiol.* Vol . 276, 1978, pp. 193-209.
- 5 Secker H. and Searle C. 1990 "Time domain analysis of auditory-nerve fibers firing rates" *J. Acoust. Soc. Am.* 88 (3), pp. 1427-1436.
- 6 Sacristan M., et al. "A reusable Inner Product Unit for DSP applications" *Proc. Euromicro 99*.
- 7 Cavanagh J.F. "Digital Computer Aritmeto" *Mc Graw Hill*, 1985.

Restriction Enzyme Computation

Olgierd Unold¹, Maciej Troć¹

¹ Institute of Engineering Cybernetics, Wroclaw University of Technology
Wyb. Wyspianskiego 27, 50-370 Wroclaw, Poland
phone: (+4871) 320 20 15, fax: (+4871) 321 26 77
e-mail: unold@ict.pwr.wroc.pl, <http://www.ict.pwr.wroc.pl/~unold>

Abstract. In this paper we explore the molecular computation model based on a splicing system. A simulator of the programmable, two-input symbol finite automaton is described. The molecular finite state machine is implemented with three enzymes of the class IIS restriction enzymes: FokI, BseMII, and BseXI.

1. Introduction

DNA Computing is a method of solving mathematical problems with the help of biological operations on DNA strands. It is interdisciplinary by nature, lying in the interface between biochemistry and computer science. The advantages of DNA Computing are its high, energy efficiency, and economical information storing. There are many potential applications of DNA Computing. The paradigm of DNA computing, which was first proposed by Adleman [1] and later expanded by Lipton [7], is one of data-parallel computation using DNA molecules. Within a test tube used in ordinary molecular biology experiments, about 10^{12} DNA molecules are generated each molecule represents one candidate for a solution. Since each operation is simultaneously applied to all of the molecules in the test tube, this process can be classified as a data-parallel computation. This approach may prove to be the best way to solve *NP* complete problems. DNA Computing has the potential to provide huge memories. Each individual DNA strand can encode binary information. DNA Computing also has the potential to supply massive computational power. DNA in one liter of water can encode the state of about 10^{18} processors!

Up to now one can distinguish a few models of DNA Computing: computing inside a single molecule, computing by interactions among molecules, computing with membranes and computing with geometry [4]. Regardless of applied DNA model in order to construct a general molecular computer some universal model of computation must be expressed in chemistry, such as a Turing Machine (TM). In this paper, we explore the computation model proposed by Ehud Shapiro [2], which is in fact a kind of splicing system [5].

A simulator of the programmable, two-input symbol finite automaton is proposed. The molecular finite state machine was implemented with three enzymes of the class IIS restriction enzymes: FokI, BseMII, and BseXI - in comparison to only one enzyme used in Shapiro's model.

This paper is organized as follows. In the next section, we summarize the basic biological concepts necessary to understand DNA Computing. In Section 3 we introduce finite state machines. Splicing systems are introduced in Section 4. In Section 5 we describe shortly Shapiro's model of computation. Some changes to the model of DNA computation were proposed in Section 6. Section 7 concludes the paper. The paper is supplemented by two Appendices, which present the details of working of molecular finite state machines.

2. Biological Preliminaries

In DNA Computation, the instances of problem are encoded in oligonucleotides, or strands, of DNA. There are four types of nucleotides which differ in the chemical group called *base*. The four bases are *adenine* (A), *guanine* (G), *cytosine* (C), and *thymine* (T), which bind according to the Watson-Crick complement condition. The pairs A,T and G,C are called complementary. The nucleotides form DNA strands which possess polarity, it means that beside the sequence also the direction of the strand is important. For example, CTT and TTC are different strands. *Hybridization* is a chemical process that joins two complementary single strands into a double strand. *Ligation* is a chemical process whereby two double strands are joined into one double strand. A restriction enzyme (such as EcoRI, FokI, BseMII) is characterized by double strand that it recognizes. For example the enzyme FokI cleaves the input molecule far

away from the recognition site $\begin{array}{c} \text{CGATG} \\ \text{GCTAC} \end{array}$ and works according to the definition

FokI: GGATG(N)9/13↓ where $N \in \{\text{A,C,G,T}\}$. See [11] for further background in molecular biology.

3. Finite State Machine

We assume that the reader is familiar with finite-state machines (FSM - for background see [6]). Here we give only a cursory glance, avoiding any formal model. A FSM is a subclass of TM. A FSM has only an internal memory determined by its finite state set; the input tape is not used as an additional memory. A FSM just reads the input in one sweep from the left to the right.

The machine can be in one of a finite number of internal states of which one is designed an initial state and some are designed accepting states. Its software consists of transition rules, each specifying a next state based on the current symbol. It is initially positioned on the leftmost input symbol in the initial state. In each transition the machine moves one symbol to the right, changing its internal state according to one of the applicable transition rules. Alternatively, it may suspend without completing the computation if no transition rule applies. A computation terminates on processing the last input symbol. An automaton is said to accept an input if a computation on this input terminates in an accepting final state.

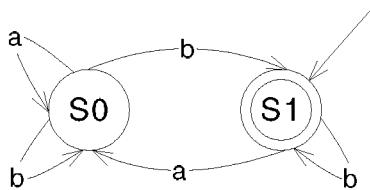


Fig. 1. Finite automaton with two states (S0 and S1) and two input symbol (a and b)

Figure 1 represents diagram of two-state, two-input symbol automaton. Incoming unlabelled arrow represents the initial state (S1), labeled arrows represent 5 transition rules (T1: $S_0 \rightarrow aS_0$, T3: $S_0 \rightarrow bS_0$, T4: $S_0 \rightarrow bS_1$, T5: $S_1 \rightarrow aS_0$, T8: $S_1 \rightarrow bS_1$), and the double circle represents an accepting state (S1).

4. Splicing Systems

Splicing is a paradigm for DNA Computing which provides a theoretical model of enzymatic systems operating on DNA strands. Splicing models a solution with enzymatic actions (restriction, hybridization, and ligation) operating on DNA in parallel in a common test tube. The DNA strands are modeled as strings over a finite alphabet. Splicing allows for the generation of new strings from an initially chosen set of strings, using a specified set of splicing operations. The splicing operations on the DNA are string editing operations such as cutting, appending, etc. These operations can be applied in any order, and thus the splicing system can be considered to be autonomously controlled. Also, the operations may be nondeterministic, and a large of possible results may be obtained from a small number of operations. The splicing system was introduced as a generative formalism in 1987 by Head [5]. The splicing systems and their extensions have been studied by many authors (for example [3], [8], [9]), and Rothenmund [10] proved that a universal TM can be simulated by recombinant DNA operations in splicing models. The restriction enzymes that Rothenmund suggests to use are a subclass of the II endonucleases, that is the IIS restriction endonucleases, so called nonpalindromic enzymes.

Recently an experimental test of splicing was done by Shapiro [2], who used biological molecules to create DNA computer based - as a Rothenmund's TM - on a restriction nuclease FokI.

5. Shapiro's Programmable and Autonomous Computing Machine

In [2] Shapiro and his coworkers have described a programmable finite two-state, two-input symbol machine comprising DNA and DNA-manipulating enzymes. This kind of the automaton can have 8 possible transition rules and programming amounts to selecting some of these transition rules and deciding which internal states are accepting. There are 255 possible transition-rule selections and 3 possible selections of ac-

cepting states (either S0, or S1 or both), resulting in 756 syntactically distinct programs. A few of these programs were tested in the lab by Shapiro's team.

Hardware of the molecular Shapiro's computer consists of restriction enzyme FokI, T4 DNA ligase and ATP, while the software comprises 8 short double-stranded DNA molecules – the transition molecules, which encode the 8 possible transition rules.

A double strand DNA molecule encodes the initial state of the automaton and the input, with 6 base pair coding for one input symbol. The system also contains 'peripherals', two output-detection molecules of different lengths, each of which can interact selectively with a different output molecule to form an output-reporting molecule that indicates a final state and can be readily detected by gel electrophoresis.

The computation starts when the hardware, software and input are all mixed together and runs autonomously, if possible till termination. If the peripherals are also mixed then output reporters are formed *in situ* in termination.

6. Simulator of Restriction Enzyme Computation

Many DNA computing algorithms have been proposed so far. Some of them have been actually implemented in *in vitro* experiments (as a Shapiro's experiment), but most remain mere proposals and their feasibility has not been verified yet. Since it is almost impossible to check all possible molecular reactions in advance, computer simulators for DNA computing are needed. We have developed a simulator for a programmable two-state, two-input symbol finite automaton proposed by Shapiro and his associates. The simulator is able to check all automata programs, not only used to test in a tube by Shapiro's team.

First the user must describe the shape of automaton he wants to simulate. He is supposed to define: initial state, final states and transition rules (example image of the program shown in Figure 2). There are 8 possible transition rules. Each of them has its own DNA representation. After that it is necessary to give input word consist of appropriate symbols (*a* or *b*). Initial state and input word are encoded by the program, as DNA molecules. Transition molecules, input molecules and output-detection molecules are stored in the memory as the special data structures. When every element of automaton is encoded, the simulation can be started. Program is checking all possible paths in the automaton defined by the user. It operates on the data structures representing DNA molecules by subprograms (functions) simulating working of enzymes: ligase and restrictase. During the simulation virtual DNA molecules are hybridized and divided like in the normal, biological experiment. In the memory of computer, the subprogram (function) called "restriction" divides virtual DNA molecules, the intermediate configurations are being created. The subprogram "ligation" try to hybridize intermediate configuration molecules with individual transition molecules or with output-detection molecules. All sequences of transition rules (all paths) have been checked in the program. By the virtual ligation with output-detection molecules, there is possible to get to know, if concrete path ends in one of the final states of the automaton. The program can determine if the input word is acceptable by the automaton or not.

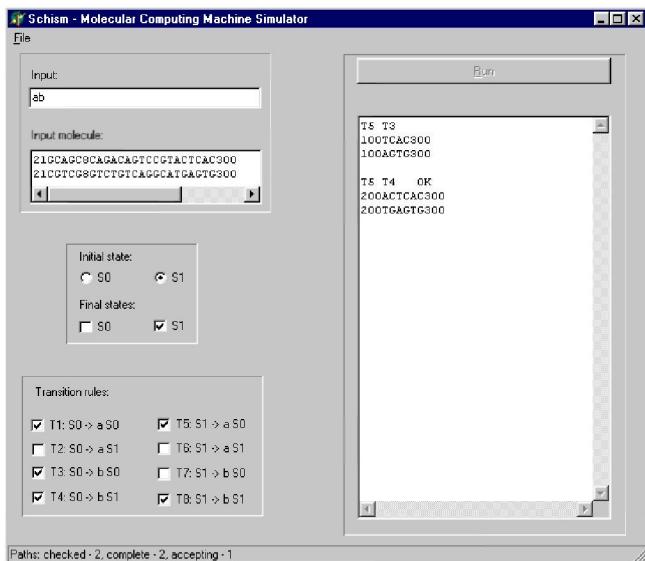


Fig. 2. Simulating the automaton shown in Figure 1

6.1 BseMII Enzyme Computation

After we developed a prototype of the simulator, we examined it by simulating all of the experiments performed by Shapiro, in which enzyme FokI was used. The encoding we used for the input symbols, the terminator, the transition molecules, the states and output-detection molecules is identical as in [2]. We also simulated automata not reported in [2].

In the next step we modified molecular computing machine by using new restriction endonuclease – BseMII: CTCAG(N)10/8↓ where $N \in \{A,C,G,T\}$. In Figure 3 the new encoding of an automaton is presented. A sample processing of an input molecule *ab* by the automaton defined in Figure 1 one can find in Appendix A.

6.2 BseXI Enzyme Computation

In the last step we modified molecular computing machine by using one more new restriction endonuclease – BseXI: GCAGC(N)(8/12)↓ where $N \in \{A,C,G,T\}$. In Figure 4 the new encoding of an automaton is presented. A sample processing of an input molecule *ab* by the automaton defined in Figure 1 is placed in Appendix B.

	a	b	t
	CAC	CTC	CGC
<S0,symbol>	AC	TC	GC
<S1,symbol>	CA	CT	CG

a) Symbols and states encoding

S0: CTCAG (7) **S1:** CTCAG (8)
 GAGTC (7) GAGTC (8)

b) Initial states encoding

T1: S0 → a S0
CTCAG (5) AC
GAGTC (5)

T5: S1 → a S0
CTCAG (4) CA
GAGTC (4)

e) Transition molecules

21CTCAG8CACCTCCGC300
21GAGTC8GTGGAGGCG300

c) Example input molecule *ab* with terminator, starting from S1

S0 - D: (100) GC **S1 - D:** (200) CG
 (100) (200)

d) Output-detection molecules

Fig. 3. Design details of molecular finite automaton with BseMII

	a	b	t
	CAGACA	GTCCGT	ACTCAC
<S0, symbol>	GACA	CCGT	TCAC
<S1, symbol>	CAGA	GTCC	ACTC

a) Symbols and states encoding

GCAGC (6) CAGACAGTCCGTACTCAC300
CGTCG (6) GTCTGTCAGGCATGAGTG300

c) Example input molecule *ab* with terminator, starting from S1

T1: S0 → a S0
GCAGC (2)
CCTCC (3) CTCT

T5: S1 → a S0
GCAGC
CGTCGGTCT

e) Transition molecules

S0: GCAGC (4)
CGTCG (4)

S1: GCAGC (6)
CGTCG (6)

b) Initial states encoding

S0 - D: (100)
 (100) AGTGA
S1 - D: (200)
 (200) TGAG

d) Output-detection molecules

T3: S0 → b S0 **T4: S0 → b S1**
 GCAGC (2) GCAGC (4)
 CGTCG (2) GGCA CGTCG (4) GGCA

T7: S1 → b S0
GCAGC
CCTGGGAGG

T8: S1 → b S1
GCAGC (2)
CCTCG (2) CAGG

Fig. 4. Design details of molecular finite automaton with BseXI

7. Conclusions

We developed simulator for DNA computing model proposed by Shapiro in [2]. The model of computation was experimental proved to be valid for FokI and extended by using new restriction enzymes: BseMII, and BseXI.

Although many issues remain for the future investigation, we have started the first set of experiments *in vitro* with the scientists of Medical University of Wrocław.

References

1. Adleman, L.M.: Molecular Computation of Solutions to Combinatorial Problems, *Science* 266 (1994) 1021-1024
2. Benenson, Y., Paz-Elitzur, T., Adar, R., Keinan, E., Livneh, Z., and Shapiro, E.: Programmable Computing Machine Made of Biomolecules, *Nature* 414 (2001) 430-434
3. Freund, R., Kari, L., Paun, G.: DNA Computing Based on Splicing; the Existence of Universal Computers, *Theory Comput. Syst.* (1999) 32, 69-112
4. Hagiya, M.: From Molecular Computing to Molecular Programming, *DNA Computing*, 6th International Workshop on DNA-Based Computers, DNA 2000, In: Condon, A., Rozenberg, G. (eds.): *Lecture Notes in Computer Science*, Vol.2054 (2001) 89-102
5. Head, T.: Formal Language Theory and DNA: an Analysis of the Generative Capacity of Specific Recombinant Behaviors., *Bull. Math. Biol.* 49 (1987) 6, 737-759
6. Hopcroft, J.E., Ullman, J.F.: *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, Reading MA (1979)
7. Lipton, R.J.: DNA Solution of Hard Computational Problem, *Science* 268 (1995) 542-545
8. Mateescu, A., Paun, G., Rozenberg, G., and Salomaa, A.: Simple Splicing Systems, *Discrete Appl. Math.* (1998) 84, 145-163
9. Pixton, R.: Splicing in abstract families of languages, in SNAC Working Material, TUCS General Publ. (1997) 6, 513-540
10. Rothemund, P.W.K.: A DNA and Restriction Enzyme Implementation of Turing Machines, In: Lipton, R.J., and Baum, E.B. (eds.): *DNA Based Computers*, American Mathematical Society, Providence, RI (1996) 1-22
11. Streyer, L.: *Biochemistry*, Freeman & Co. (1995)

Appendix A

A sample output of the simulator for a DNA computing with BseMII

Sequence of the input symbols: ab
 Initial state: S1
 Input molecule:

```

21CTCAC8CACCTCCCC300
21GAGTC8GTGGAGGCG300
Final states: S1
-----
Sequence of the transition rules: T5 T4   OK
A next intermediate configuration formed upon restriction BseMII:
21CTCAC8CA          CCTCCCC300
21GAGTC8           GTGGAGGCG300
Ligation with T5 transition rule:
  28CTCAC4CACCTCCGC300
  CCT28CACTC4CTCCAGCCCC300
A next intermediate configuration formed upon restriction BseMII:
  28CTCAC4CACCTC      CCC300
  GGT28GAGTC4GTGG     AGGCG300
Ligation with T4 transition rule:
  15CTCAC4ATTCCGC300
  CCT15CACTC4TAACCCC300
A next intermediate configuration formed upon restriction BseMII:
  15CTCAC4ATTCCC      C300
  GGT15GAGTC4TAAG      GCG300
Ligation with output detector S1-D:
200CGC300
200CCC300

```

Appendix B

A sample output of the simulator for a DNA computing with BseXI

```

Sequence of the input symbols: ab
Initial state: S1
Input molecule:
21GCAGC8CAGACAGTCCGTACTCAC300
21CCTCC8CTCTCTCAGGCCATCACTC300
Final states: S1
-----
Sequence of the transition rules: T5 T4   OK
A next intermediate configuration formed upon restriction BseXI:
21GCAGC8          CAGACAGTCCGTACTCAC300
21CGTCG8GTCT      GTCAGGCATGAGTG300
Ligation with T5 transition rule:
  28CCACCCACACACTCCCTACTCAC300
  GGT28CGTCGGTCTGTCAGGCCATGAGTG300
A next intermediate configuration formed upon restriction BseXI:
  28GCACCCAGACAGT      CCGTACTCAC300
  CCT28CCTCCCTCTCTCACCCA      TCACTC300
Ligation with T4 transition rule:
  15CCACCCATTACCCCTACTCAC300
  GGT15CGTCGTAATGGCATGAGTG300
A next intermediate configuration formed upon restriction BseXI:
  15GCAGCATTACCGT      ACTCAC300
  CCT15CCTCTTAATCCCATCAC      TC300
Ligation with output detector S1-D:
200ACTCAC300
200TGAGTG300

```

Neurally Inspired Mechanisms for the Dynamic Visual Attention Map Generation Task

Maria T. López¹, Miguel A. Fernández¹, Antonio Fernández-Caballero¹,
and Ana E. Delgado²

¹ Departamento de Informática

E.P.S.A., Universidad de Castilla-La Mancha, 02071 – Albacete, Spain
`{mlopez, miki, caballer}@info-ab.uclm.es`

²Departamento de Inteligencia Artificial

Facultad de Ciencias and E.T.S.I. Informática, UNED, 28040 - Madrid, Spain
`{adelgado}@dia.uned.es`

Abstract. A model for dynamic visual attention is briefly introduced in this paper. A PSM (problem-solving method) for a generic “Dynamic Attention Map Generation” task to obtain a **Dynamic Attention Map** from a dynamic scene is proposed. Our approach enables tracking objects that keep attention in accordance with a set of characteristics defined by the observer. This paper mainly focuses on those subtasks of the model inspired in neuronal mechanisms, such as accumulative computation and lateral interaction. The subtasks which incorporate these biologically plausible capacities are called “Working Memory Generation” and “Thresholded Permanency Calculation”.

1 Introduction

Visual attention models may be classified as space-based models – e.g., spotlight metaphor [12] and zoom-lens metaphor [1] – and object-based models – for instance, discrete objects [11, 14]. Our approach falls into the models based in objects, where visual attention always focuses on discrete objects or coherent groups of visual information. In order to select an object that captures our attention, we previously have to determine which image features should be combined to obtain the object. This process is carried out in two stages. Firstly, obtaining features to generate simple shapes, and, secondly, combining these simple shapes into more complex ones. Once the objects have been obtained, the next process –attention based on objects– consists in selecting one of the shapes generated.

Vecera [14] introduced a model to obtain objects separated from the background in static images by combining bottom-up (scene-based) and top-down (task-based) processes. The bottom-up process gets the borders to form the objects, whereas the top-down process uses known shapes stored in a database to be compared to the shapes previously obtained in the bottom-up process.

A first plausible neuronal bottom-up architecture was proposed by Koch and Ullman [9]. Their approach is also based in features integration [13]. In Itti, Koch and

Niebur [10] a visual attention system inspired in the neural architecture of the early visual system and in the architecture proposed by Koch and Ullman [9] was introduced. This system combines multi-scale image features in a unique saliency map. The most salient locations of the scene are selected from the highest activity of the saliency map using a winner-take-all (WTA) algorithm.

Another example based in the saliency map idea is the guided search (GS) model proposed by Wolfe [15]. This model incorporates two visual selection stages. The first one, the pre-attention stage, with a great spatial parallelism, performs the computation of simple visual features. The second stage is performed spatially in a sequential way, using more complex visual representations – including the combination of features - to perform the calculations. The value of the features obtained is a combination of bottom-up and top-down knowledge. Attention is directed to the location that shows the highest value in the **Dynamic Attention Map**. The guided search ends if the location contains the target looked for. If this is not the case, the search continues in the **Dynamic Attention Map** in a descending order, finishing when the target is found or when attention falls under a threshold.

In this paper we introduce a model of dynamic visual attention that combines bottom-up and top-down processes. Bottom-up is related to the first step of the architectures proposed, where the input image is segmented using dynamic criteria by means of neurally inspired accumulative computation [2-4] and lateral interaction [5-8]. The observer may indicate how to tune system parameters to define the attention focus using top-down processes. These processes are of a static nature, during the configuration of the features selection and the attention focus processing, or dynamic, when the observer modifies parameters to centre the focus on the interest elements.

2 Model of dynamic visual attention

Our approach defines a PSM for the generation of a **Dynamic Attention Map** on a dynamic scene, which is able to obtain the objects that will keep the observer's attention in accordance with a set of predefined characteristics. Figure 1 shows the result of applying our model to the “Dynamic Attention Map Generation” task, where attention has been paid on moving elements belonging to class “car”.

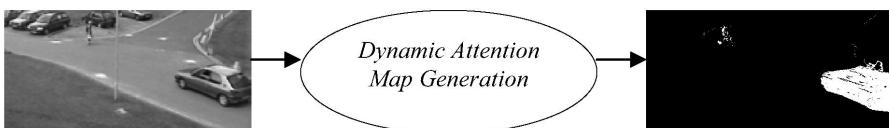


Fig. 1. Input and output of the “Dynamic Attention Map Generation” task

The different subtasks proposed to obtain the **Dynamic Attention Map** are depicted on figure 2, whilst figure 3 shows the inferential scheme of the model introduced. Next these subtasks are briefly described:

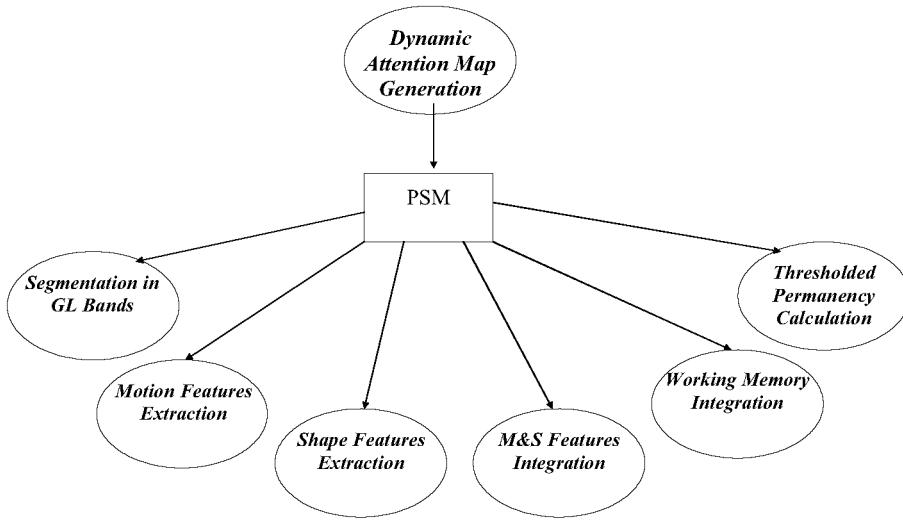


Fig 2. Decomposition of the “Dynamic Attention Map Generation“ task into subtasks

- Subtask “Segmentation in Grey Level Bands” is thought to segment the 256 grey level value input images into n (static role) grey level bands.
- Subtask “Motion Features Extraction” calculates for each pixel a set of dynamic features, e.g. the presence or absence of motion, the speed, or the acceleration. The output of this subtask is formed by those pixels that deserve some dynamic features. Parameters M_i (static roles) indicate the range of values for motion features that pixels must possess to be marked as *activated* at the output of the subtask.
- Now, subtask “Shape Features Extraction” calculates certain shape features such as the size, the width, the height, the relation width to height, the value size/(width*height), and so on. It obtains those elements that deserve a set of predefined features. Parameters S_i (static roles) indicate the range of values for shape features that elements must possess to be labelled as *activated* at the output of this subtask. The elements that do not have the features defined will be marked as *inhibited*. All the rest of pixels which do not belong to elements are labelled as *neutral*.
- The idea of subtask “Motion & Shape (M&S) Features Integration“ is to generate a unique **Motion and Shape (M&S) Interest Map** by integrating motion features and shape features in accordance with the criteria defined by the observer. These criteria correspond to C_i (static role).
- Subtask “Working Memory Generation” segments the image in regions composed of connected pixels whose brightness pertains to a common interval. Each of these regions or silhouettes of a uniform grey level represents an element of the scene.

- f) In subtask “Thresholded Permanency Calculation“ firstly the persistency of its input is accumulated. Then, a further threshold is performed. P_i are the static roles for this subtask.

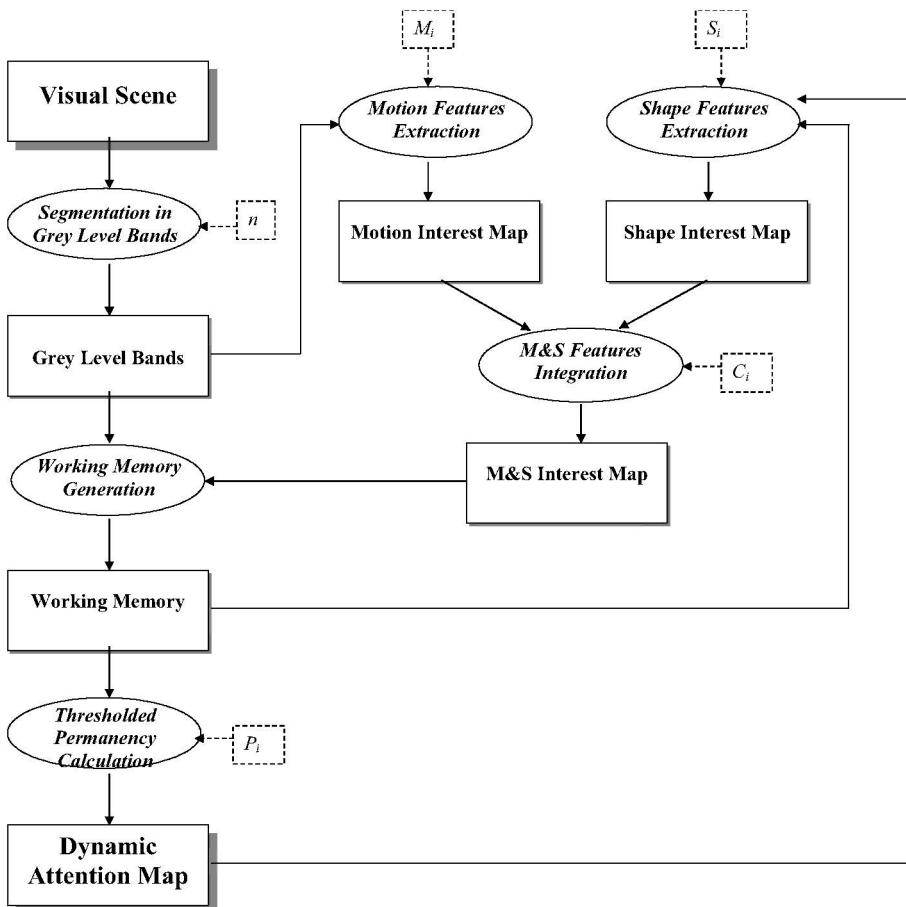


Fig 3. Inferential scheme of the model

From now on, this paper focuses on the generation of the working memory and the **Dynamic Attention Map** from the image segmented in grey level bands and the **Motion and Shape Interest Map**. As already mentioned the **Motion and Shape Interest Map** is calculated frame to frame, and is composed of *activated* pixels pertaining to pixels or elements of interest, *inhibited* pixels included in pixels or elements of no interest, and *neutral* pixels. In other words, this paper will show the processes related to lateral interaction and accumulative computation, whose neuronal basis has already been demonstrated in some previous papers of the same authors [2-8]. Hence, these subtasks may be implemented with a neural inspiration. Next section is devoted to describe in extensive both subtasks.

3 Accumulative computation and lateral interaction subtasks

3.1 Subtask “Working Memory Generation”

The process of obtaining the **Working Memory** consists in superimposing, just as done with superimposed transparencies, the image segmented in grey level bands of the current frame (at time instant t), that is to say, **Grey Level Bands** (dynamic role of inference “Segmentation in Grey Level Bands”) with image **Motion and Shape Interest Map** (dynamic role of inference “Motion and Shape Features Integration”) constructed at the previous frame (at time instant $t-1$). Thus, in the **Working Memory**, at time instant t all scene elements associated to pixels or elements of interest will appear. This process may be observed in more detail in figure 4, where in image **Motion and Shape Interest Map** a black pixel means *neutral*, a grey pixel means *inhibited* and a white pixel means *activated*.

Coming from the image segmented in bands, some processes are performed in parallel for each band. So, there will be as many images as number of bands the image has been segmented in. These images, $B_i(x,y,t)$, are binary: 1 if the brightness of the pixel belongs to band i , and 0 if it does not belong to band i .

For each one of the binary images, $B_i(x,y,t)$, a process is carried out to fill the shapes of any band that includes any value *activated* in the classified **Motion and Shape Interest Map**, $MS(x,y,t)$. For it, initially $S_i(x,y,t)$, the value of the filled shape at band i for coordinate (x,y) at time instant t , is obtained as explained next. It is assigned the value of the band if the corresponding coordinate in the **Motion and Shape Interest Map** is activated:

$$S_i(x,y,t) = \begin{cases} i, & \text{if } B_i(x,y,t) = 1 \text{ and } MS(x,y,t) = \text{activated} \\ 0, & \text{otherwise} \end{cases}$$

Afterwards, by means of lateral interaction, the same value i is assigned to all positions connected to $B_i(x,y,t)$, - using a connectivity of 8 pixels -, whenever its value in the **Motion and Shape Interest Map** does not indicate that it belongs to a non valid shape.

$$\forall(\alpha, \beta) \in [x \pm 1, y \pm 1], \quad S_i(\alpha, \beta, t) = \begin{cases} i, & \text{if } B_i(\alpha, \beta, t) = 1 \text{ and } MS(\alpha, \beta, t) \neq \text{inhibited} \\ 0, & \text{otherwise} \end{cases}$$

Finally, to generate the **Working Memory**, all obtained images are summed up, as in:

$$W(x, y, t) = \sum_{i=1}^n S_i(x, y, t)$$

Notice that, in a given time instant t , scene elements whose shape features do not correspond to those defined by the observer may appear in the **Working Memory**. This may be due to the fact that these elements have not yet been labelled. But, if the

features of these elements do not correspond to those indicated as interesting by the observer, these elements at $t+1$ will appear as *inhibited* in the **Motion and Shape Interest Map**. This way, at $t+1$ these elements will disappear from the **Working Memory**. In order to only obtain elements that really fulfil features defined by the observer, some accumulative computation mechanisms are introduced, as explained in section 3.2.

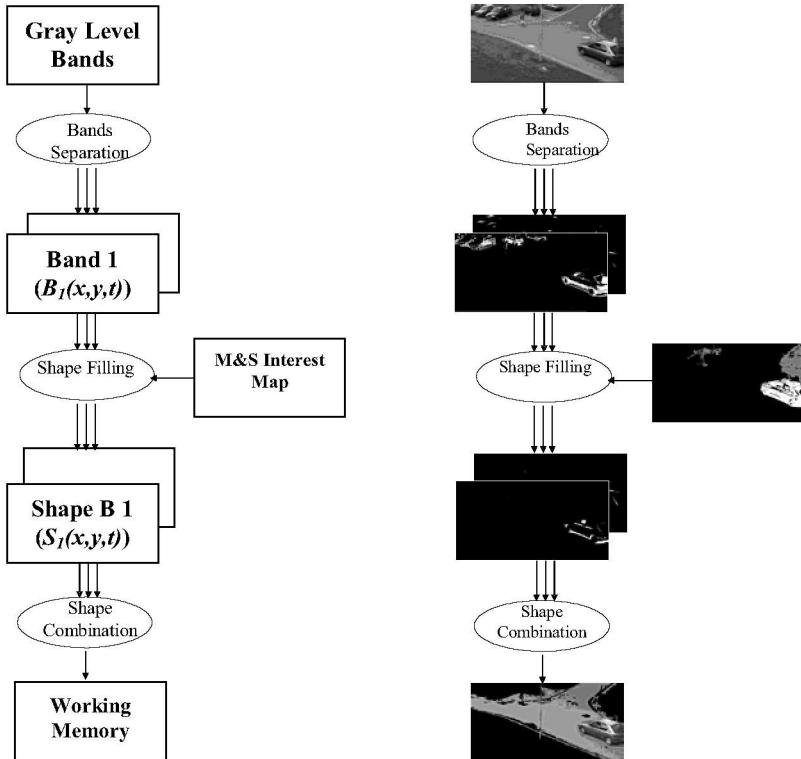


Fig. 4. Subtask “Working Memory Generation”

3.2 Subtask “Thresholded Permanency Calculation”

Subtask “Thresholded Permanency Calculation” uses as input the **Working Memory** and gets as output the final **Dynamic Attention Map**. This subtask firstly calculates the so called permanency value associated to each pixel. If the value of the **Working Memory** is activated, the charge value is incremented at each image frame by a value of δc up to a maximum. If this is not the case, the charge value is decremented by a value of δd down to a minimum. In a second step, the charge values are thresholded to get the **Dynamic Attention Map**. Thus, the **Dynamic Attention Map** will only have activated those pixels which have been *activated* in the **Working Memory** during various successive time instants, namely “threshold / charge” (see figure 5). This is shown by means of the following formulas:

$$Q(x, y, t) = \begin{cases} \min(Q(x, y, t - \Delta t) + \delta c, Q_{\max}), & \text{if } W(x, y, t) \equiv 1 \\ \max(Q(x, y, t - \Delta t) - \delta d, Q_{\min}), & \text{if } W(x, y, t) \equiv 0 \end{cases}$$

$$A(x, y, t) = \begin{cases} 1, & \text{if } Q(x, y, t) \geq \theta \\ 0, & \text{otherwise} \end{cases}$$

where δc is the charge constant, δd is the discharge constant, Q_{\max} is the maximum charge value, Q_{\min} is the minimum charge value, θ is the threshold value and $A(x, y, t)$ is the value of the **Dynamic Attention Map** at coordinate (x, y) at time instant t .

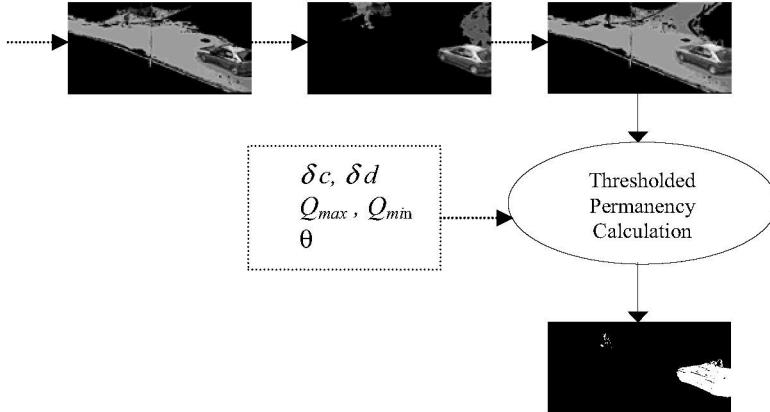


Fig.5. Subtask “Thresholded Permanency Calculation”

The accumulative computation process, followed by the threshold stage, enables to maintain active in a stable way a set of pixels which pertain to a group of scene elements of interest to the observer. Thus, the state of this “memory” defines the attention focus of the system, and is the input to the observer’s system. The observer will modify the parameters that configure the mechanisms of extraction and selection of shapes and/or pixels of interest.

4 Conclusions

A model of dynamic visual attention capable of segmenting objects in a scene has been introduced in this paper. The model enables focusing the attention at each moment at shapes that possess certain characteristics and eliminating shapes that are of no interest. The features used are related to motion and shape of the elements present in the dynamic scene. The model may be used to observe real environments indefinitely in time.

In this paper we have highlighted those subtasks of the generic “Dynamic Attention Map Generation” task related to plausible biologically inspired methods. These mechanisms, namely accumulative computation and lateral interaction, have so far

been proven to be inherently neuronal. The subtasks which incorporate these biologically plausible capacities, called “Working Memory Generation” and “Thresholded Permanency Calculation”, because of the fact of making use of accumulative computation and lateral interaction processes, enable maintaining a stable system of dynamic visual attention.

References

1. Eriksen, C. W., St. James, J. D.: Visual attention within and around the field of focal attention: A zoom lens model. *Perception and Psychophysics* 40 (1986) 225-240
2. Fernández, M.A., Mira, J.: Permanence memory: A system for real time motion analysis in image sequences. IAPR Workshop on Machine Vision Applications, MVA'92 (1992) 249-252
3. Fernández, M.A., Mira, J., López, M.T., Alvarez, J.R., Manjarrés, A., Barro, S.: Local accumulation of persistent activity at synaptic level: Application to motion analysis. In: Mira, J., Sandoval, F. (eds.): From Natural to Artificial Neural Computation, IWANN'95, LNCS 930. Springer-Verlag (1995) 137-143
4. Fernández, M.A.: Una arquitectura neuronal para la detección de blancos móviles. Unpublished Ph.D. dissertation (1995)
5. Fernández-Caballero, A., Mira, J., Fernández, M.A., López, M.T.: Segmentation from motion of non-rigid objects by neuronal lateral interaction. *Pattern Recognition Letters* 22:14 (2001) 1517-1524
6. Fernández-Caballero, A., Mira, J., Delgado, A.E., Fernández, M.A.: Lateral interaction in accumulative computation: A model for motion detection. *Neurocomputing* 50C (2003) 341-364
7. Fernández-Caballero, A., Fernández, M.A., Mira, J., Delgado, A.E.: Spatio-temporal shape building from image sequences using lateral interaction in accumulative computation. *Pattern Recognition* 36:5 (2003) 1131-1142
8. Fernández-Caballero, A., Mira, J., Fernández, M.A., Delgado, A.E.: On motion detection through a multi-layer neural network architecture. *Neural Networks* 16:2 (2003) 205-222
9. Koch, C., & Ullman, S.: Shifts in selective visual attention: towards the underlying neural circuitry. *Human Neurobiology* 4 (1985) 219-227.
10. Itti, L., Koch, C. Niebur, E.: A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20 (1998) 1254-1259
11. Moore, C. M., Yantis, S., Vaughan, B.: Object-based visual selection: Evidence from perceptual completion. *Psychological Science* 9 (1998) 104-110
12. Posner, M.I., Snyder, C.R.R., Davidson, B.J.: Attention and the detection of signals. *Journal of Experimental Psychology: General* 109 (1980) 160-174
13. Treisman, A.M., Gelade, G.: A feature-integration theory of attention. *Cognitive Psychology* 12 (1980) 97-136
14. Vecera, S.P.: Toward a biased competition account of object-based segregation and attention. *Brain and Mind* 1 (2000) 353-385
15. Wolfe, J.M.: Guided Search 2.0.: A revised model of visual search. *Psychonomic Bulletin & Review* 1 (1994) 202-238

A Model of Dynamic Visual Attention for Object Tracking in Natural Image Sequences

Nabil Ouerhani and Heinz Hügli

Institute of Microtechnology, University of Neuchâtel
Rue A.-L. Breguet 2, CH-2000 Neuchâtel, Switzerland
{Nabil.Ouerhani,Heinz.Hugli}@unine.ch

Abstract. Visual attention is the ability to rapidly detect the interesting parts of a given scene on which higher level computer vision tasks can focus. This paper reports a computational model of dynamic visual attention which combines static and dynamic features to detect salient locations in natural image sequences. Therefore, the model computes a map of interest - saliency map - related to static features and a saliency map derived from dynamic scene features and then combines them into a final saliency map, which topographically encodes stimulus saliency. The information provided by the model of attention is then used by a tracking method to attentively track the interesting features in the scene. The experimental results, reported in this work refer to real color image sequences. They clearly validate the reported model of dynamic visual attention and show its usefulness in guiding the tracking task.

1 Introduction

Human vision relies extensively on a visual attention mechanism which selects parts of the scene, on which higher vision tasks can focus. Thus, only a small subset of the sensory information is selected for further processing, which partially explains the rapidity of human visual behavior.

Like in human vision, visual attention represents a fundamental tool for computer vision. Thus, the paradigm of computational visual attention has been widely investigated during the last two decades. Numerous computational models have been therefore reported [1–3]. Most of them rely on the feature integration theory presented by Treisman *et al.* in [4]. The saliency-based model of Koch and Ullman which is one of the most prominent computational models of attention was first presented in [5] and gave rise to numerous software and hardware implementations [6–8].

Most of these works aimed, however, at computing visual attention from static scene images. Little effort has been devoted so far to model dynamic visual attention. Some of the rare attention models that took in consideration the dynamic features of scenes were presented in [9, 3, 10].

This paper reports a computational model of dynamic visual attention which combines static and dynamic features to detect salient locations in natural image sequences. Therefore, the model computes a map of interest - saliency map -

related to static features and a saliency map derived from dynamic scene features and then combines them into a final saliency map, which topographically encodes stimulus saliency. The static saliency map is computed from two color-based features and the intensity of each frame, whereas the dynamic saliency map is based on the normal component of the motion vector which is computed using a multiresolution, gradient-based method. The most salient locations of the scene are selected by detecting the spots with the highest activity on the final saliency map using a Winner-Take-All algorithm.

The information provided by the dynamic model of attention, like the location of the salient points and their characteristics are exploited by a tracking algorithm. The detected salient points are first characterized, by determining their Most Discriminating Feature (MDF), that is the feature that distinguishes a region from its surrounding. This characterization is then used to track the detected spots over time.

The remainder of this paper is organized as follows. Section 2 presents the model of dynamic visual attention. Section 3 describes, how the model of dynamic visual attention guides the tracking task and presents some experimental results. Finally, the conclusions are stated in Section 4.

2 Model of dynamic visual attention

The proposed model of dynamic visual attention computes a static saliency map which discriminates salient scene locations based on static features and a dynamic saliency map that highlights moving scene constituents. The two saliency maps are then combined into a final map of attention also called the final saliency map. The different steps of our model are illustrated in Figure 1.

2.1 The static saliency map

The computation of the static saliency map S_s is achieved in three main steps.

Feature maps

First, a number of features ($1..j..n$) are extracted from the scene by computing the so called feature maps F_j . Such a map represents the image of the scene, based on a well-defined feature. This leads to a multi-feature representation of the scene. This work considers three different features which are computed from RGB color images.

- Intensity feature

$$F_1 = (R + G + B)/3 \quad (1)$$

- Two chromatic features based on the two color opponency filters R^+G^- and B^+Y^- where the yellow signal is defined by $Y = \frac{R+G}{2}$. Such chromatic opponency exists in human visual cortex [11].

$$F_2 = R - G \quad \text{and} \quad F_3 = B - Y \quad (2)$$

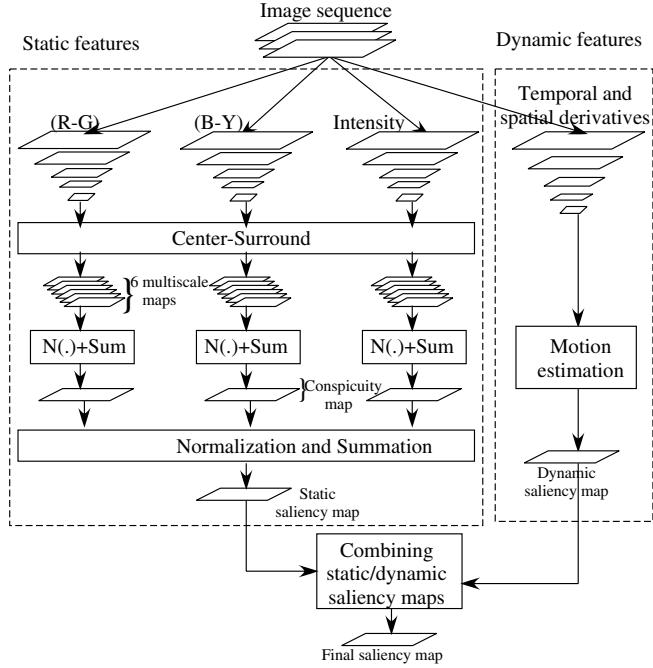


Fig. 1. Model of dynamic visual attention which combine static and dynamic scene features to detect salient locations in image sequences.

Before computing these two features, the color components are first normalized by F_1 in order to decouple hue from intensity.

conspicuity maps

In a second step, each feature map F_j is transformed in its conspicuity map C_j which highlights the parts of the scene that strongly differ, according to a specific feature, from their surroundings. Multiscale *difference-of-Gaussians*-filters, which can be implemented using gaussian pyramids, are suitable means to implement the conspicuity transformation. Practically, For each feature j , a nine scale gaussian pyramid \mathcal{P}_j is created by progressively lowpass filter and subsample the feature map F_j , using a gaussian filter G (see Eq. 3).

$$\mathcal{P}_j(0) = F_j, \quad \mathcal{P}_j(i) = \mathcal{P}_j(i-1) * G \quad (3)$$

Center-Surround is then implemented as the difference between fine and coarse scales. For each feature j , six intermediate multiscale conspicuity maps $M_{j,k}$ ($1..k..6$) are computed according to equation 4, giving rise to 18 multiscale maps for the considered three static features.

$$M_{j,1} = |\mathcal{P}_j(2) - \mathcal{P}_j(5)|, \quad M_{j,2} = |\mathcal{P}_j(2) - \mathcal{P}_j(6)|$$

$$\begin{aligned} M_{j,3} &= |\mathcal{P}_j(3) - \mathcal{P}_j(6)|, & M_{j,4} &= |\mathcal{P}_j(3) - \mathcal{P}_j(7)| \\ M_{j,5} &= |\mathcal{P}_j(4) - \mathcal{P}_j(7)|, & M_{j,6} &= |\mathcal{P}_j(4) - \mathcal{P}_j(8)| \end{aligned} \quad (4)$$

Note that these intermediate multiscale conspicuity maps are sensitive to different spatial frequencies. Fine maps (e.g. $M_{j,1}$) detect high frequencies and thus small image regions, whereas coarse maps, such as $M_{j,6}$, detect low frequencies and thus large regions.

For each feature j , the six multiscale maps $M_{j,k}$ are then combined, in a competitive way into a unique feature-related conspicuity map C_j :

$$C_j = \sum_{k=1}^6 w_k M_{j,k} \quad (5)$$

The weighting function w , which simulates the competition between the different scales, is described below.

The static saliency map

Finally, the three conspicuity maps C_j are integrated, in a competitive way, into the static saliency map S_s in accordance with equation 6.

$$S_s = \sum_{j=1}^3 w_j C_j \quad (6)$$

The weights w_i are determined according to a weighting function $w = (M - \bar{m})^2$, where M is the maximum activity of the conspicuity map and \bar{m} is the average of all its local activity maxima. Indeed, this weighting function promotes conspicuity maps in which a small number of strong peaks of activity is present. Maps that contain numerous comparable peak responses are demoted. It is obvious that this competitive mechanism is purely data-driven and does not require any a priori knowledge about the analyzed scene.

2.2 Dynamic saliency map

The dynamic saliency map S_d should discriminate moving objects in the scene. Since we assume that we deal with image sequences acquired with a static camera, a map related to optical flow approaches well the required map. Therefore, we use a gradient-based method to compute optical flow, which is fairly robust and simple to compute. A drawback of the gradient approach is that it may only be used for small displacements. To overcome this difficulty, we compute optical flow at different scales, taking advantage of gaussian pyramids. This multiscale concept allows the detection of small displacements at fine scales, whereas large displacements are detected at coarser scales [12]. A combination of the optical flow maps, computed at different scales, gives rise to a motion map that clearly discriminates moving scene objects.

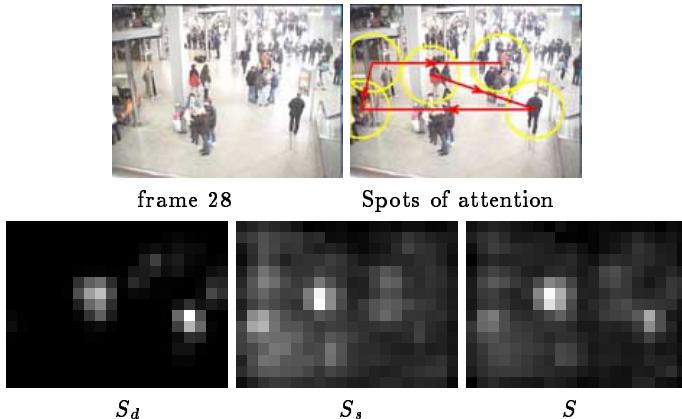


Fig. 2. Detecting salient spots using static and dynamic features. From the frame 28 of our test sequence, we compute a static (S_s) and a dynamic (S_d) saliency maps which are combined into the final saliency map (S). WTA is then applied to detect the five most salient locations (spots of attention) in this frame.

2.3 The final saliency map

In this section, we aim at integrating the static saliency map S_s and the dynamic one S_d . The basic idea is that the two cues should compete for saliency. The purely data-driven competition mechanism presented above (Section 2.1) is a suitable integration concept of both cues. Thus, the final saliency map S is computed according to equation 7.

$$S = w_s S_s + w_d S_d \quad (7)$$

Where w_s and w_d are computed by the weighting function w presented above.

2.4 Selection of salient locations

The most salient locations of the scene are selected by applying a Winner-Take-All (WTA) network on the final saliency map S . The number ($1..i..m$) of the detected locations can be either set by the user or determined automatically through the activities of the saliency map.

Figure 2 shows an example of a dynamic, static and final saliency maps. It also illustrates the selection of the most salient locations (spots of attention).

3 Attentive tracking

The basic idea is exploit the information provided by the visual attention algorithm about the scene to achieve the tracking of the salient features or locations. Therefore, the detected spots of attention are first characterized by determining their Most Discriminating Feature and then tracked over time.

3.1 Characterization of the spots of attention

In a tracking context, it's important to attribute robust features to the tracked objects. The Most Discriminating Feature j^* of a salient location fulfills this criterion. Since equation 6 can be rewritten as follows:

$$S_s = \sum_{j=1}^3 \sum_{k=1}^6 w_{jk} M_{j,k} \quad (8)$$

j^* can be computed according to equation 9.

$$j^* = \operatorname{argmax}_j(M_{j,k}(\mathbf{x})) \quad (9)$$

Where \mathbf{x} is the spatial location of the considered spot of attention. Thus, each of the m ($1..i..m$) spots detected in frame t can be denoted by $P_{t,i}(\mathbf{x}, j^*)$, where \mathbf{x} is the spatial coordinate of the spot.

Figure 3.1 shows an example of spot characterization.

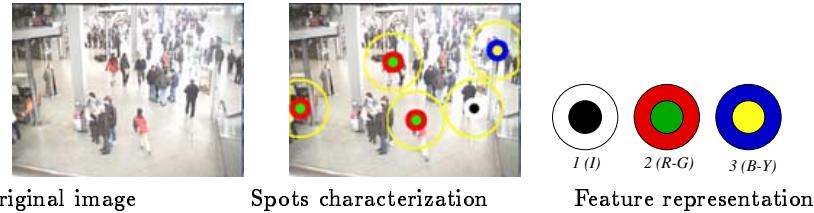


Fig. 3. Spots characterization. Each of the five detected spots is assigned one of the three static features (intensity, $(R - G)$ and $(B - Y)$). A color representation of these features is used (right image).

3.2 Tracking of the spots of attention

This section presents an attentive tracking method that tracks the salient locations of the scene based on Most Discriminating Features (j^*). The tracking algorithm starts with creating m initial trajectories, each of which contains one of the m detected spots of attention in the first frame. A new detected spot of attention is either inserted into an existing trajectory or gives rise to a new one, depending on its similarity with the last inserted spot - the head element- of already existing trajectories. Formally, let $P_{t_1,i_1}(\mathbf{x}_1, j_1^*)$ the actual detected spot of attention and T a trajectory whose head element is the spot $P_{t_2,i_2}(\mathbf{x}_2, j_2^*)$. The decision whether the actual spot is inserted to T is taken according to equation 10.

$$T = \begin{cases} T \cup \{P_{t_1,i_1}(\mathbf{x}_1, j_1^*)\} & \text{if } j_1^* = j_2^* \& \|\mathbf{x}_1 - \mathbf{x}_2\| < \epsilon \\ T & \text{otherwise} \end{cases} \quad (10)$$

Where $\|.\|$ is an euclidean distance and ϵ is a threshold that can be set manually or learned automatically. Thus, the matching between a given spot and the head element of a given trajectory is based on feature similarity and spatial proximity. As mentioned above, if a detected spot does not correspond to any existing trajectory, then it initializes a new one. In a postprocessing step those trajectories which contain only few spots (less 10% of the total number of frames) are discarded. This postprocessing allows to keep only persistent trajectories. An example of attentive tracking is illustrated in figure 3.2.

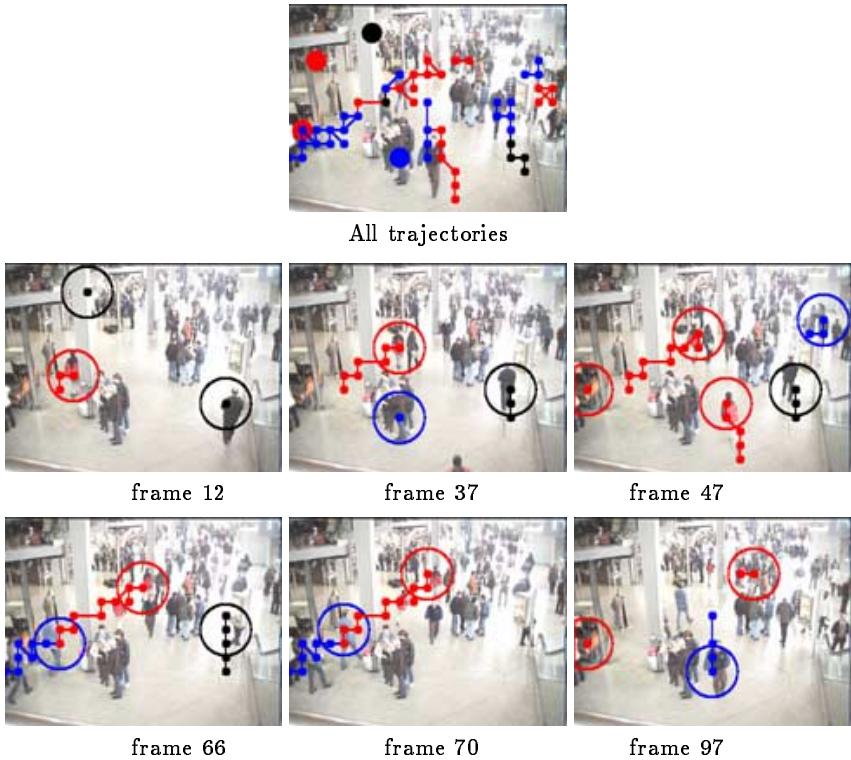


Fig. 4. Attentive tracking. The top image represents all tracked trajectories after the postprocessing step. The large colored circles on this image represent salient static locations. The other images represent the detected spots of attention and the corresponding trajectories in sample frames of the test sequence.

4 Conclusion

This paper presents a computational model of dynamic visual attention which combines static and dynamic features in order to detect the most salient loca-

tions in dynamic scenes. Therefore, a static and a dynamic saliency maps are first computed and then integrated, in a competitive manner, into a final map of attention, the saliency map on which a Winner-Take-All algorithm is applied to select the most visually salient parts of the scene. Tracking-relevant information are extracted from the hierarchical structure of the attention model in order to guide an attentive tracking algorithm which is able to track salient scene constituents over time. The examples presented in this work illustrate the different stages of our attention model and show the usefulness of attention-based scene information to carry on the tracking task. In future work, effort will be devoted to the improvement of the tracking algorithm by allowing the tracking of feature vectors instead of single features.

References

1. S. Ahmed. VISIT: An Efficient Computational Model of Human Visual Attention. *PhD thesis, University of Illinois at Urbana-Champaign*, 1991.
2. R. Milanese. Detecting Salient Regions in an Image: from Biological Evidence to Computer implementation. *PhD thesis, Dept. of Computer Science, University of Geneva, Switzerland*, Dec. 1993.
3. J.K. Tsotsos. Toward computational model of visual attention. *In T. V. Papathomas, C. Chubb, A. Gorea & E. Kowler, Early vision and beyond*, pp. 207-226. Cambridge, MA: MIT Press, 1995.
4. A.M. Treisman and G. Gelade. A feature-integration theory of attention. *Cognitive Psychology*, pp. 97-136, Dec. 1980.
5. Ch. Koch and S. Ullman. Shifts in selective visual attention: Towards the underlying neural circuitry. *Human Neurobiology* (1985) 4, pp. 219-227, 1985.
6. L. Itti, Ch. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 20(11), pp. 1254-1259, 1998.
7. N. Ouerhani and H. Hugli. Computing visual attention from scene depth. *Proc. ICPR 2000, IEEE Computer Society Press*, Vol. 1, pp. 375-378, Barcelona, Spain, Sept. 2000.
8. N. Ouerhani, H. Hugli, P Y. Burgi, and P F. Ruedi. A real time implementation of visual attention on a simd architecture. *Proc. DAGM 2002, Springer Verlag, Lecture Notes in Computer Science (LNCS) 2449*, pp. 282-289, 2002.
9. R. Milanese, S. Gil, and T. Pun. Attentive mechanisms for dynamic and static scene analysis. *Optical Engineering*, Vol. 34, (8), pp. 2428-2434, 1995.
10. A. Maki, P. Nordlund, and J.O. Eklundh. Attentional scene segmentation: Integrating depth and motion from phase. *Computer Vision and Image Understanding*, vol. 78, pp. 351-373, June, 2000.
11. S Engel, X. Zhang, and B. Wandell. Colour tuning in human visual cortex measured with functional magnetic resonance imaging. *Nature*, Vol. 388, no. 6637, pp. 68-71, Jul. 1997.
12. E. Simoncelli. Coarse-to-fine estimation of visual motion. *Proceedings, Eighth Workshop on Image and Multidimensional Signal Processing. Cannes France*, sept., 1993.

Neural competitive structures for segmentation based on motion features

Javier Díaz¹, Sonia Mota¹, Eduardo Ros¹ and Guillermo Botella¹

¹ Departamento de Arquitectura y Tecnología de Computadores, E.T.S.I. Informática, Universidad de Granada, Periodista Daniel Saucedo Aranda s/n, 18071 Granada, Spain
{jdiaz, smota, eros, gbotella}@atc.ugr.es

Abstract. Simoncelli & Heeger studied how the motion is processed in humans (V1 and MT areas) and proposed a model based on neural populations that extract the local motion structure through local competition of MT like cells. In this paper we present a neural structure that works as dynamic filter on the top of this MT layer and can take advantage of the neural population coding that is supposed to be present in the MT cortical processing areas. The test bed application addressed in this work is an automatic watch up system for the rear-view mirror blind spot. The segmentation of overtaking cars in this scenario can take full advantage of the motion structure of the visual field provided that the ego-motion of the host car induces a global motion pattern whereas an overtaking car produces a motion pattern highly contrasted with this global ego-motion field.

1 Introduction

The work presented in this paper has been developed in the framework of ECOVISION European Research project [1]. One of the objectives of ECOVISION consortium is the development of a pre-cognitive visual model that can be useful for real world problems. In particular, the rear-view mirror blind spot monitor is presented as a feasible problem in which motion processing can provide useful information for the overtaking car segmentation.

One of the motion processing models studied in the first stage of the project is the Simoncelli and Heeger model (S&H) that constitutes a model with strong neurophysiological bases. The S&H work proposes a model of how the cortical areas (V1 and MT cells) can extract the motion structure through neural local computation and competition [2,3]. The output layer uses neural population coding, which can be seen as an inefficient code but represents an advantage if the post-processing is done through neural computation as presented in this paper. This is not the case if the optic flow extraction is done through more mathematical based algorithms.

The MT cells specialization on specific movement direction and velocity modules enables a connectivity that can embody perspective deformation correction and rigid

body motion enhancement. This can be achieved by a collective-competitive connectivity pattern as described in section 3. Motion processing is normally very noisy and needs of further post-processing before addressing image segmentation. In this paper we describe how a simple connectivity pattern can be of interest for neural computation of noisy motion information. This connection pattern forces individual cells to behave as dynamic filters that are sensitive to more reliable movement features than simple spatio-temporal correlations.

This post-processing layer is composed by cells that collect the cell activity from MT cells sensitive to similar motion primitives, and compete between themselves to impose a movement feature locally in each visual field point. Furthermore, we also describe how this can enhance the rigid body motion segmentation capabilities of the processing layer by connecting MT cells of local neighbourhoods to facilitate the detection of movement features of rigid bodies through the visual field.

The application of the neural processing strategy presented in this paper in real world problems is also addressed. In particular, promising results have been obtained for the segmentation of overtaking cars in the rear-view mirror blind spot. This application is currently being addressed by many application driven research groups [4]. Besides, in this application the motion processing plays an important role, since an overtaking car exhibits a forward motion pattern clearly contrasted against the global backward motion pattern observed in the rear-mirror due to the ego-motion of the host car.

2 Bio-inspired model for computing optical flow

The Simoncelli & Heeger model [2,3], consists of two primary stages corresponding to cortical areas V1 and MT. The computation form is highly parallel and regular.

A linear model is used for V1 simple cells. This explains simple cell selectivity for stimulus orientation and spatial frequency, and the cells respond to both opposite polarities contrast stimulus.

V1 complex cells sum weighted simple cell afferents distributed over a local spatial region, each of them having the same space-time orientation and phase. Their receptive fields are modeled using only edge detectors. Each V1 neuron squared and normalizes its inputs, and V1 outputs with the same space-time orientation are spatially combined using positive weights to get V1 complex cells receptive field.

MT cells are modeled combining the outputs of a set of direction-selective V1 complex cells, whose preferred space-time orientations are consistent with the MT cells characteristic velocity. The mechanism for velocity selectivity can be described in the spatio-temporal domain easily. The power-spectrum of translational pattern lies on a plane, and the tilt of the plane depends on the velocity; in this way a MT cell detects the tilted plane with maximum response [5] and different combinations of V1 cells can be used to get the MT receptive field [6,7]. Finally, a Winner Takes All configuration among the MT population selects only the MT cells with higher input, i.e., the one that best matches the local motion pattern.

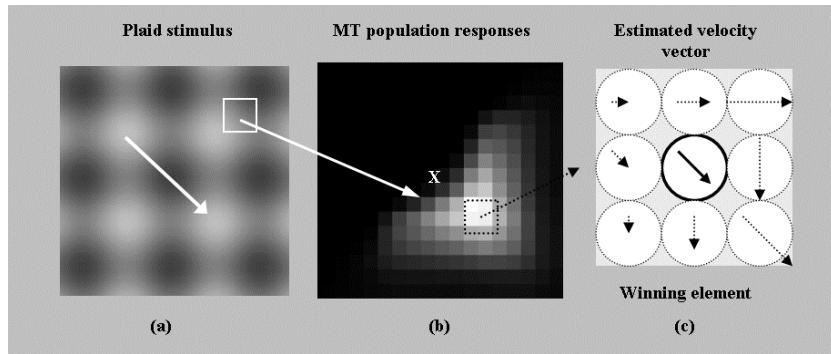


Fig. 1. Population response example. The result of a plaid stimulus composed by a sinusoidal grating moving rightwards and another moving downwards is a moving pattern toward the right-bottom corner (a). A set of MT neurons responses are codified using grey levels. The relative position of the winner element with respect to the center of the population, showing the velocity module and direction. Maximum responses are given at the best tuned MT neuron for that stimulus, but MT cells tuned at near velocities are not zero (b). Finally, the winner element indicate the estimate velocity (c).

The implementation of S&H model, showed in fig. 2, can be summarized in 4 steps:

1. Computing local contrast stimulus.
2. Modeling V1 simple and complex neurons, using spatio-temporal third Gaussian derivatives and spatial pooling.
3. Modeling MT neurons summing the weighted responses of V1 cells which lie on its characteristic plane.
4. Computing winner element for each pixel in the visual field

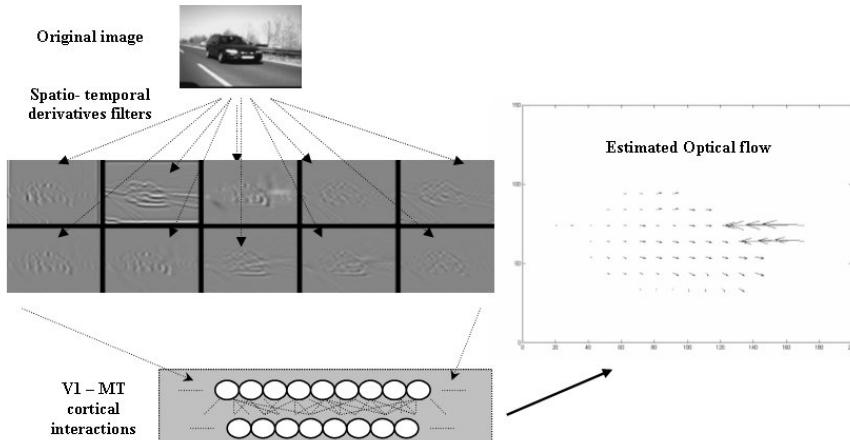


Fig. 2. S&H Model. An overtaking car sequence is used to evaluate the model. The basic set of third Gaussian derivatives are formed by G_{xxx} , G_{yyy} , G_{ttt} , G_{xxy} , G_{xyy} , G_{xxt} , G_{ytt} , G_{ytt} and G_{xyt} . This pre-filtered images are combined to get V1 spatio-temporal orientation cells and their combinations give us the MT receptive field. Finally, for each pixel the winner take all neuron is considered the correct vector velocity value.

The basic model used to describe V1 simple cell receptive field is an edge detector that uses spatio-temporal third Gaussian derivatives [8]. For our considerations we do not take into account other possible receptive fields such as bars detectors or DOG's like. Our implementation uses a basic spatio-temporal set of 40 filters with only one spatial scale to interpolate the MT characteristic orientation because biological systems have only a limited set of V1 orientations [9]. As other energy models [10], the contrast problem is hard to solve and some kind of normalization techniques [11] are used to minimize this effect. Finally, the MT neurons sum the weighted contribution of V1 cells near its preferred velocity.

One model limitation is the detection of second order motion. This kind of motion has power spectrum that lies out of the origin so the proposed filters can not detect it. Some modifications could be added to detect second order motion [12], but for the addressed application, in which we are just interested in translational motions, this is not necessary.

3 Dynamic Filters

The S&H layer is connected to a new neural layer that we have called Collector Layer (CL). The synaptic connection is done through a converging many-to-one excitatory pattern.

CL cells work as dynamic filters to segment the overtaking vehicle. Considering the kind of input information, the CL cells select only the features necessary for the segmentation process. First of all, because of the application addressed is focussed in discriminating between leftward (ego-motion) and rightward (overtaking vehicle) moving points, only the cortical S&H neurons that match these directions (leftward by [-135,215] degrees and rightward by [-45,45] degrees) are connected to the Collector Layer. In the other hand, the configuration of the collector layer neurons embodies different aspects about the character of a moving object that are important for the segmentation task: rigid body motion and scene perspective motion pattern deformations.

All the points in a rigid body move at a similar speed and in the same direction. The presence of detached points, belonging to a rigid body, that move in opposite direction with respect the majority of the points at the rigid body is considered noise. In addition, it is expected all points in a rigid body to be placed in the same neighbourhood of the image.

The motion pattern deformation due to the perspective from the rear-view mirror can be summarized as follows. A moving object (overtaking car) with a constant speed is expected to move slowly when it is localized in the very left side of the image (far away) and its speed will increase as it moves (overtakes) rightwards through the visual field (closer position). Considering this, forward sensitive neurons on the left side of the visual field tuned to higher speeds are less frequent than neurons placed on the right side and vice versa.

The S&H outputs stimulate the CL composed by self-competing collector neurons. Every collector neuron is tuned to a neighbourhood of a characteristic module velocity (by converging connections), in a fix direction. Therefore, every CL neuron integrates the activity of all the outputs from a 5x5 neighbourhood in the S&H layer (upper framed zones with continuous line in Fig. 3) sensitive to the same motion direction and velocity module.

The CL has the configuration of a self-competitive layer, so the collector neuron that receives the maximum stimulus in its spatial influence area (bottom framed zone with continuous line in Fig. 3) inhibits the others and dominates in its local area (Winner Takes All). This helps to detect rigid body motion; in other words, we can find rigid bodies in areas where there are winner collector neurons because they receive the input excitation of a group of MT neurons that are sensitive to the similar velocities (in module and direction) and are positioned in the same spatial area.

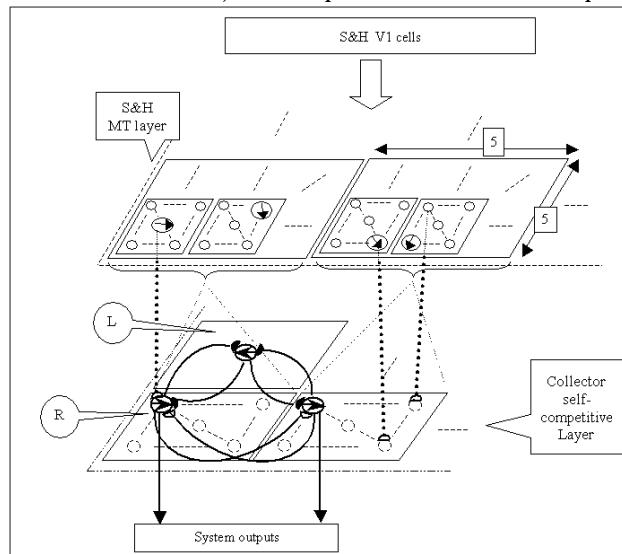


Fig. 3. Every collector makes spatial integration of the velocities (in module and direction) among a population of 5x5 S&H excitatory output neurons. The CL has the structure of a Winner-Takes-All layer, so there is just one collector neuron winner in a neighbourhood. A winner node receives synapses from other winners in an influence area. A group of neurons that detect the same direction support each other (white exciting synapsis), but if there is a neuron detecting an opposite direction it is inhibited by the synapses coming from other nodes corresponding to the same spatial neighbourhood (black inhibiting synapsis). The upper figure shows the synapses among three winner collector neurons. Two neurons detect rightward motion direction (R) and the other detects leftward motion detection (L). The last one is inhibited by the other nodes (local majority).

In the other hand, the winner neurons in an influence area at CL (bottom in Fig. 3) can interact with other winner neurons from other influence areas in their neighbourhood. This interaction has inhibitory or excitatory character, facilitating the domination of large features and inhibiting those winner neurons whose motion

direction is different with respect to the majority of the surrounding winner nodes. In this way, the output response of this filtering neural layer (CL) will be non-zero in areas where there are winner collector neurons non-inhibited by others winner nodes.

The CL neurons are characterized by a time constant that takes into account how the stimulus drives the onset and offset of the elements of this layer. If we choose this time constant to be long, that means that more input frames with a lasting motion pattern are needed to excite a neuron and make it dominate against previous perceived patterns.

The distribution of the specialised collector neurons is non-uniform, i.e. the number of collector cells tuned at low velocities is higher in the left hand side of the layer than the number of collector neurons tuned at high velocities in the same place; and the opposite occurs in the right hand side of the layer. In this way, we facilitate the detection of slow movements in the left side of the visual field and rapid movements on the right side; this is used to correct the perspective deformation.

4 Results

We show the neural segmentation results of the previously described system applied to real overtaking car sequences. Our neural layers are capable of segmenting rigid objects that are moving in opposite horizontal directions.

Fig 4 (a,b), shows overtaking car sequence with dark car in a shining day recorded with a conventional CCD camera. Left column are the original images of the overtaking sequences in two different stages, in the middle column the S&H extracted optical flow is shown. The grey scale indicates the velocity module and arrows show only the motion direction (all arrows have the same length). The right column shows the dynamic filters outputs. The segmented overtaking car is drawn using dark colour (rightward motion) and the background, moving thought the opposite direction, use bright colour. The collector layer receptive fields are sensitive only to synapses belonging to MT neurons tuned in a cone of velocities directions. For example, the bottom car optical flow in Fig 4.b indicate motion out of this allow velocities cone, so the collector layer output discards these components.

Other results are shown in Fig 4 (c,d). An overtaking car sequence in a foggy and rainy day recorded with high dynamic range camera [13]. The weather produces a noise sequence with low contrast, therefore a special camera is necessary for this situation. Consequently, the extracted optical flow is worst than the one of the previous sequence. Other effects that contribute to get worst car segmentations are light reflection on the road and low contrast that produces more filter blurring. Another problem factor is the colour-depth. The high dynamic range camera uses 32 bits precision and we have used only 8 bits depth. This effect produces a noisy pattern that affects mainly the right side of the image and gives us wrong optical flow estimation in this area. But, it is clear the advantage of using dynamic filter outputs to segment correctly the overtaking car.

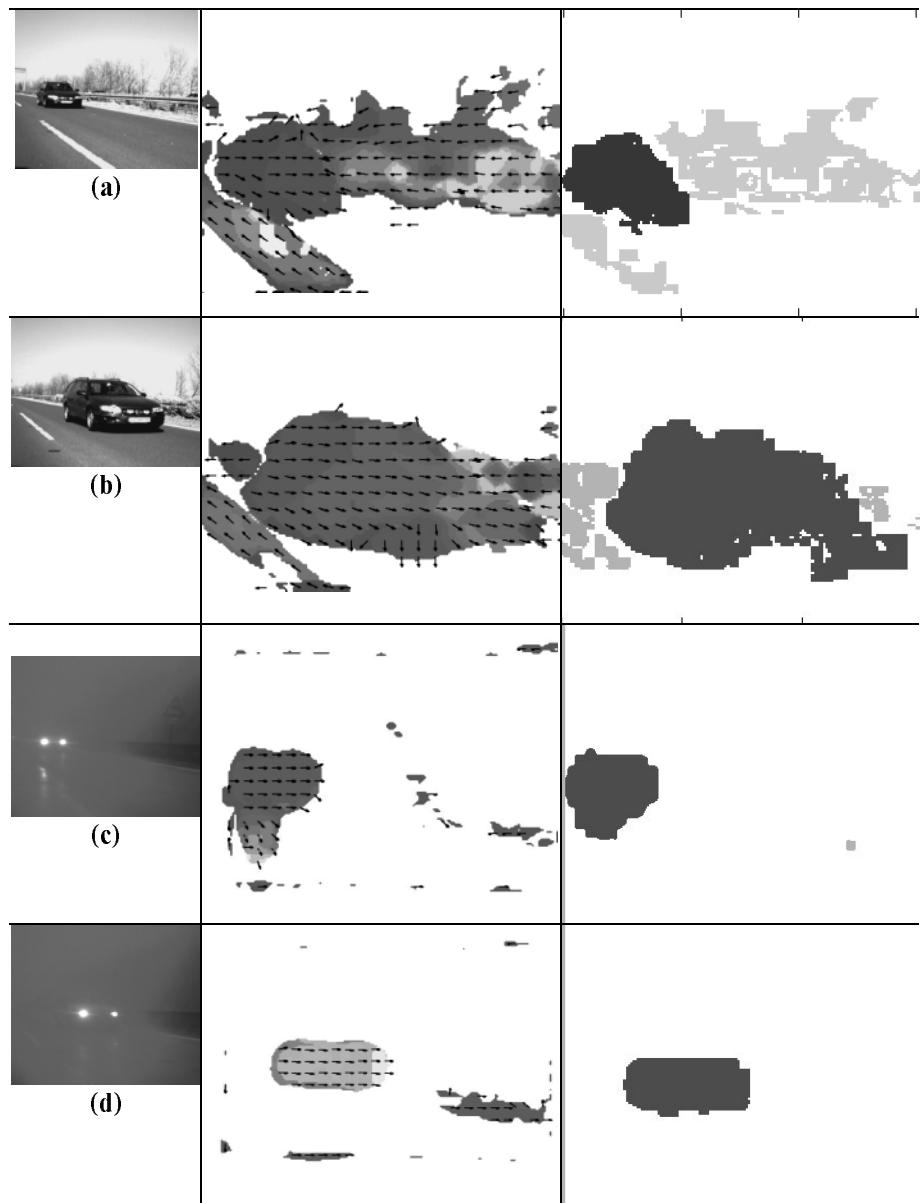


Fig. 4. Overtaking car sequence in a shinning day (a,b) and in a foggy and rainy day (c,d).

5 Conclusions

This paper describes a bio-inspired system viability to segment objects using optical flow extracted from motion information. Motion information from V1 and MT layers is filtered by post processing layer that works as dynamic filters. The correction pattern from S&H MT cells this collector layer can embody aspect that facilitates the segmentation of moving rigid bodies and can also partially correct the deformation of the visual field due to the perspective of the rear view mirror.

The neural system proposed is highly parallel. It is a self-competitive neural mechanism for feature selection. All this enhances the capability of segment the rigid bodies from noisy environments.

Acknowledgements

This work has been supported by the V EU research framework funds through the European Projects ECOVISION (IST-2001-32114) [1] and CORTIVIS (QLK6-CT-2001-00279). We would like also to acknowledge Hella [14] for providing us with the overtaking sequences.

References

1. ECOVISION, <http://www.pspc.dibe.unige.it/~ecovision/>
2. Simoncelli Eero P, Heeger David J. (1998) A model of Neuronal Responses in Visual Area MT. *Vision Research*, 38(5):743-761.
3. Simoncelli, E. P.(1993). Distributed Analysis and Representation of Visual Motion. PhD thesis, Massachusetts Institute of Technology, Dept of E. Eng. Comp. Sci., Cambridge,MA.
4. U. Franke, D. Gavrila, A. Gern, S. Görzig, R. Janssen, F. Paetzold and C. Wöhler. "From door to door- Principles and Application on Computer Vision for driver assistant systems" in Intelligent Vehicle Technologies: Theory and Applications, Arnold 2000.
5. Watson, A.B, Ahumada, A.J. (1983). A look at motion in the frequency domain. In Tsosos, J.K., editor, Motion: Perception and representation, pages1–10. NewYork.
6. Grzywacz, N. M., Yuille, A. L. (1990). A model for the estimate of local image velocity by cells in the visual cortex. Proceeding so the Royal Society of London A,239,129–161.
7. Heeger, D.J. (1987). Model for the extraction of image of Image flow. *Journal of the Optical Society of America A*, 4, 1455–1471.
8. Freeman, W. T, Adelson, E.H. (1991). The design and use of steerable filters. *IEEE Pattern Analysis and Machine Intelligence*,13,891–906.
9. Hubel, D.H. y Wiesel, T.N.(1962). Receptive fields, binocular interactions and functional architecture in the Cat's Visual Cortex. *J. Physiology*, 160, pp. 106-154.
10. Adelson, E. H, Bergen, J.R. (1986).The extraction of spatio-temporal energy in human and machine vision. In Proc. of IEEE Workshop on Motion, pp. 151–156.
11. Heeger , D. J. (1992). Normalization of cell responses in cat striate cortex. *Visual Neuroscience*, 9, 181–198.
12. Sperling G., Chubb, C., Solomon, J.A., and Lu, Z-L (1994). Fullwave and halfwave processes in second order motion and tecture. Higher-order processing in the visual system. Chichester, UK: Wiley, pp. 287-303.
13. IMS-CHIPS: <http://www.ims-chips.de/home.php3?id=e0841>.
14. Dept. of predevelopment EE-11, Hella KG Hueck & Co., Germany, www.hella.de

Background Pixel Classification for Motion Detection in Video Image Sequences

P. Gil-Jiménez, S. Maldonado-Bascón, R. Gil-Pita, and H. Gómez-Moreno

Dpto. de Teoría de la señal y Comunicaciones. Universidad de Alcalá,
28871 Alcalá de Henares (Madrid) Spain

{pedro.gil,saturnino.maldonado,roberto.gil,hilario.gomez}@uh.es

Abstract. The main objective in motion detection algorithms for video surveillance applications is to minimize the false alarm probability while maintaining the probability of detection as high as possible. Many motion detection systems fail when the noise in a specific zone is high, increasing the false detection probability, and so the system can not detect motion in these zones. In this paper we present an alternative scheme that tries to solve the mentioned problem using the classification capacity of a neural network.

1 Introduction

A video surveillance system arises when, in some place, a special need for security or safety is required. The first generation of this kind of systems consists on many video cameras and a video monitor, operated by people. A system working that way brings on two important drawbacks. First, the amount of information a person can process is extremely low and their efficiencies decrease with time. Second, when a video image is processed by humans in a public place, people involved in that situation can feel their privacy violated [1].

In this context, the second generation of video surveillance systems is based on digital signal processing, especially on artificial intelligence and image processing [1]. In these systems, only the important events are stored, saving a big amount of storage capacity that would be wasted otherwise. These systems can improve the work conditions of the video surveillance operators, releasing them from the more monotonous tasks, having more acceptance between general people.

2 Motion detection

When motion detection is needed in a video surveillance system several methods can be implemented. The easiest way to develop such a system is to calculate the absolute value of the difference between the current frame and a reference frame. Then, the difference matrix is thresholded. Those pixels whose value are higher than an arbitrarily chosen threshold are supposed to be part of an object moving into the scene, and those that are smaller belong to the background.

The reference frame can be either, the previous frame [2], or an estimation of the background calculated throughout the last processed frames [3] [4].

$$M(i, j) = |I(i, j) - B(i, j)| > th \quad (1)$$

The critical point of this kind of systems is the value of the threshold to be chosen, and this value must be applied to the whole image. If the value of the mentioned threshold were small, the false alarm probability would increase, due mainly to the noise picked up by the camera, and the noise generated by a non-static background, i. e. a branch tree moving in a windy day. On the other hand, when the value of the threshold is high, the miss probability increases because the absolute difference between the background gray level and the object gray level is likely to be lower than the current threshold. Moreover, in real video images, the noise level and illumination conditions could be different in each part of the frame.

To solve this problem, many schemes implement an statistical module that tries to determine the characteristics of every pixel on the image along the time. To achieve this, the algorithm establishes a training period in which, supposedly, there is no motion at all in the sequence, and all the variance of every pixel is due to the noise. Calculating the pixels statistics is possible to infer the behavior for each part of the image, and so, decide whether the next sample of a particular pixel corresponds again to the background, if its value is close to the expected according to the calculated statistics, or belong to a new object moving across the image. The more common algorithms used in such schemes suppose that the probability density function (p.d.f.) is gaussian [5], and, after calculate the mean (μ) and the typical deviation (σ), the next pixel will be considered as background if its value is within the interval ($[\mu - k\sigma, \mu + k\sigma]$), and motion otherwise, where k is an arbitrary constant that must be chosen by the user.

Although this method works correctly if the pixel has a low level noise, when the background generates a high level noise, due, for example, to a branch tree in a windy day, or to a high density road in which the car motion are not of interest in the system, the variance of this zone would be very high and so, the gray level interval in which the new sample can be considered motion is very small, increasing the miss probability in those zones.

Furthermore, if in a particular moment a sudden illumination change occurs without any other kind of motion in the scene, the described algorithms normally fail, because when the absolute difference or the typical deviation interval check is done, the algorithms usually mark this pixel as a motion pixel when it should have been marked as background. If the illumination change affected the whole scene, almost all the image would be marked as motion.

3 Background pixel classification

The proposed method tries to classify each zone of the image into one of the following groups, according to its behavior along the last processed frames. And furthermore, once the algorithm has succeed in the classification of every zone,

the decision about whether the next sample belong again to the background, or is part of a new moving object has its own rules, depending on the group the zone was classified on.

3.1 Groups description of background behavior

According to the more typical sequences used in video surveillance systems, we have decided to create the following background behavior groups, even though the system is not restricted to this set of groups, and new groups could be created in future work. The groups have been chosen mainly observing the more problematic pixel behavior in typical motion detection scenes.

- **Static background:**

This group corresponds to those pixels whose values are almost constant along the sequence. In real world images, this group could be all those zones that are not affected by background motion or fast illumination changes. The rule of decision of the existence of motion in this group is achieved by computing the mean and typical deviation of the pixel, being the next sample considered background again if its value is within the interval $[\mu - k\sigma, \mu + k\sigma]$, or motion otherwise. Figure 1 shows an example of a static background pixel behavior. To solve the sudden illumination changes, the algorithm classifies each of the pixels belonging to this group into one of these two subgroups:

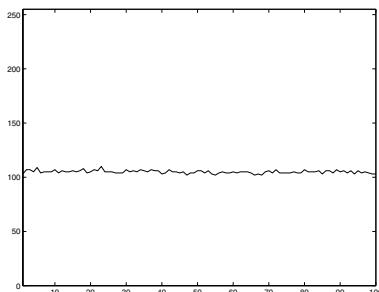


Fig. 1. Static pixel example

- **Static background with high texture information:**

This static background subgroup corresponds to those pixels that, belonging to the static background group, the texture information of its neighborhood is high enough to compute the angle difference between two consecutive vector, each made up from the referred pixel and its neighborhood pixels in the same frame. Computing the angle difference (NCCF), instead of the euclidean difference [6], is more robust in the presence of sudden illumination changes.

$$NCCF = \frac{\sum_i \sum_j (I(i,j) \cdot B(i,j))}{\sqrt{\sum_i \sum_j I^2(i,j)} \sqrt{\sum_i \sum_j B^2(i,j)}} \quad (2)$$

- **Static background without texture information:**

This static background subgroup corresponds to those pixels that, belonging to the static background group, the texture information of its neighborhood is scarce, and so, the illumination changes errors must be detected via high level algorithms.

- **Noisy background:**

This group corresponds to those pixels whose values, and its neighborhood pixel values are absolutely random along the time, and their variance are high enough to compute the motion estimation via thresholding the absolute difference between the current frame and the reference image. This behavior can be obtained for example from the tree leaves or from a water surface. To decide the existence of motion on the next frame, we can suppose that the spatial variance of a new object moving through a zone of this kind is lower than the spatial variance of the background. So, for those pixels belonging to this group, the motion estimation can be achieved computing the spatial variance of the new frame. Figure 2 shows an example of a noisy background pixel behavior.

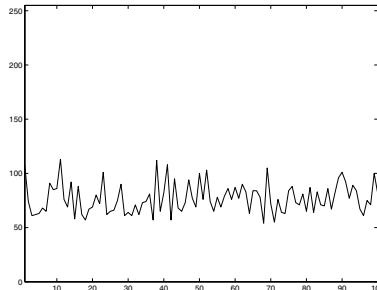


Fig. 2. Noisy pixel example

- **Impulsive background:**

In many other situations the background pixel gray level is practically stationary along the time, but randomly, its value changes suddenly and comes back again to its original value, in an impulsive-like manner. This behavior can correspond, for instance, to a road in which, randomly, a car appears in the scene on a frame, and disappears on the next. Since we are not interested in this kind of events, but in very odd ones, the system can interpret

these zones as impulsive-background zones, so the system will not generate motion alarms in these zones with this kind of events, unless a special event happens, like a car stopping in that road for a very considerable time. The alarm will be triggered if the pixel value remains out of its stationary value during a number of samples. Figure 3 shows an example of an impulsive background pixel behavior.

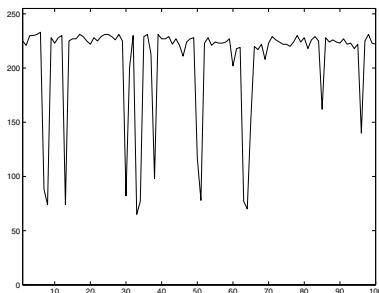


Fig. 3. Impulsive pixel example

4 Experimental results



Fig. 4. First image of the experimental sequence

A sequence of 100 images were chosen for testing the proposed scheme. The sequence was taken with a standard video camera from an exterior scene, with an interval of 0.5 seconds between samples. Figure 4 is the first image of this sequence, and figure 5 is the image number 64. On this image are marked the position of the pixels shown in figures 1 (static pixel [$x=60, y=130$]), 2 (noisy pixel [$x=326, y=87$]) and 3 (impulsive pixel [$x=304, y=196$]). As a training set we took a set of 20 pixels belonging to each group (static group, noisy group and impulsive group) of length 100 (the length of the sequence), and a validation set comprised by another 20 pixels for each group. The pixels for the noisy group were taken mainly from the pixels corresponding to the tree line, the pixels for the impulsive group were taken from the road, and the pixels for the static group were taken from the rest of the image. The test set is composed by the whole image.

To achieve the experiment we have designed a multilayer perceptron [7] [8] with 200 inputs. The inputs to the neural network are the 100 samples of a pixel, and the 100 values of its corresponding DCT to give robustness against temporal shifts. The multilayer perceptron has 5 neurons on its hidden layer and 3 outputs, corresponding with the 3 background pixel groups (static, noisy and impulsive groups). The training has been early stopped using the validation set for generalization purposes.

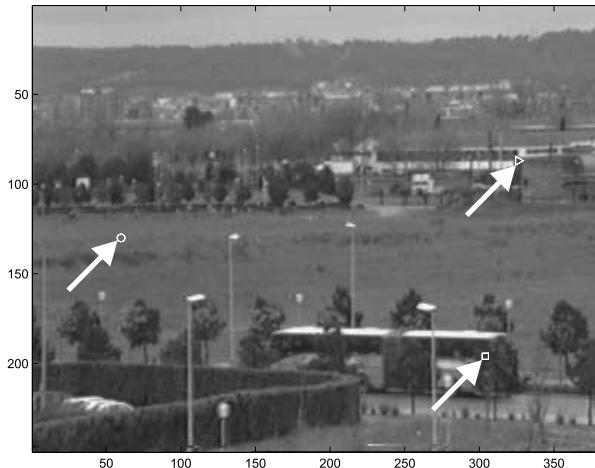


Fig. 5. Image 64 of the experimental sequence

The background pixel classification image has been divided into 2 different images (figures 6 and 7) for clarity. Figure 6 shows the static vs non-static classification results after having repeated 10 times the experiment, taking the best result over the validation set for regularization purposes. Black pixels correspond with the static background group, while white ones correspond with the non-

static background group. As we can see, static background pixels correspond to those that belong mainly to the grass surface, the mountain and the sky, while the pixels belonging to the tree line and the roads have been marked as non-static background pixels.



Fig. 6. Static vs non-static classification

In figure 7 is shown the noisy vs impulsive classification results obtained in the last experiment. White pixels correspond with the noisy background group and black pixels correspond with the impulsive background group. Gray pixels are those pixels belonging to the static background group shown in figure 6. Pixels that correspond with the tree line have been marked by the neural network as noisy pixels, while the pixels from the roads have been marked as impulsive pixels.

5 Conclusion and future work

Motion detection is one of the main tasks of a video surveillance system. Normally, the detection is done by comparing the current frame with a previously stored reference. In this paper we have proposed an alternative method to get the reference, that can improve the results of the system when working in adverse conditions, specially reducing the false alarm probability and the miss probability in such cases. Instead of thresholding the difference between the current frame and the reference, as it is the typical scheme, we first classify each zone of the image depending on its observed behavior, and so, perform the motion detection according with this classification.

The main drawback of the proposed method is the high computational cost of the classification for every pixel, so we have had to work with relatively small

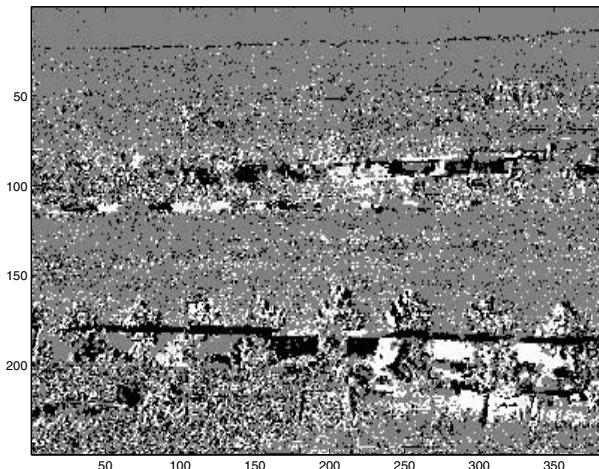


Fig. 7. Noisy vs impulsive classification

images. However, since the background behavior is suppose not to change permanently the classification can be done from time to time, reducing the computational requirements of the system. The directions of our future work aims to reduce the amount of signals that have to be classified using functions of spectral analysis and image and video compression, and so, reduce the computational cost, allowing the system to work in real time, as it is the basic requirement for this kind of systems.

References

1. Foresti, G.L., Mähönen, P., Regazzoni, C.S.: Multimedia Video-Based Surveillance Systems. Kluwer Academic Publishers (2000)
2. di Stefano, L., Neri, G., Viarani, E.: Analysis of pixel-level algorithms for video surveillance applications. In: CIAP01. (2001) 541–546
3. Gao, D., Zhou, J.: Adaptive background estimation for real-time traffic monitoring. In: IEEE. (2000) 330–333
4. Ridder, C., Munkelt, O., Kirchner, H.: Adaptive background estimation and foreground detection using kalman filtering. In: ICAM. (1995) 193–199
5. Ren, Y., Chua, C., Ho, Y.: Motion detection with non-stationary background. In: CIAP01. (2001) 78–83
6. Dawson-Howe, K.: Active surveillance using dynamic background subtraction. In: TR. (1996)
7. Haykin, S.: Neural Networks. A comprehensive foundation. Second edn. Prentice Hall Inc. (1999)
8. Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press Inc. (1995)

COBRA: An Evolved Online Tool for Mammography Interpretation

Carlos-Andrés Peña-Reyes¹, Rosa Villa², Luis Prieto³, and Eduardo Sanchez¹

¹ Swiss Federal Institute of Technology in Lausanne - EPFL, Switzerland
`{Carlos.Pena, Eduardo.Sanchez}@epfl.ch`

² Centro Nacional de Microelectrónica, Barcelona, Spain
³ Duran i Reynals Hospital, Barcelona, Spain.

Abstract. This paper presents an evolved rule-based tool for mammography interpretation denominated “COBRA: Catalonia online breast-cancer risk assessor.” COBRA is designed to aid radiologists in the interpretation of mammography to decide whether to perform a biopsy on a patient or not while providing a human-friendly explanation of the underlying reasoning. From a diagnostic point of view, the tool exhibits high performance measures (i.e., sensitivity, specificity, and positive predictive value). From an interpretability point of view, COBRA’s behavior is explained by only 14 rules containing, in average, 2.73 conditions per rule.

1 Introduction

Mammography remains the principal technique for detecting breast cancer. Despite its undoubtable value in reducing mortality, mammography’s positive predictive value (PPV) is low: only between 15 and 35% of mammographic-detected lesions are cancerous [1, 2].

A good computerized tool for medical decision support should possess two characteristics, which are often in conflict. First, the tool must attain the highest possible *performance*, i.e., detecting as much as possible the malignant cases, while minimizing the number of unnecessary biopsies. Second, it would be highly beneficial for such a system to be human-friendly, exhibiting so-called *interpretability*. This means that the physician is not faced with a black box that simply spouts answers (albeit correct) with no explanation; rather, we would like for the system to provide some insight as to *how* it derives its outputs.

In this paper we propose to use a (fuzzy) rule-based system exhibiting the two aforementioned characteristics, to construct a computerized tool to assist mammographic interpretation. To design this system, we applied Fuzzy CoCo—a cooperative coevolutionary fuzzy modeling technique conceived to provide a good balance between accuracy and interpretability [3, 4]. In the next section we describe the mammography interpretation problem, which is the focus of our interest herein. Section 3 then presents COBRA, the online explanation tool we developed, together with a short description of the evolutionary method used to find the system. In Section 4 we discuss on the results and on the diagnostic performance of the evolved rule-based system implemented. Finally, we conclude in Section 5.

Table 1. Variables corresponding to a patient's clinical data.

v_1	Age	[28-82] years	
v_2	Menstrual history	1	Premenopausal
		2	Postmenopausal
v_3	Family history	1	None
		2	Second familiar
		3	First familiar
		4	Contralateral
		5	Homolateral

2 The Catalonia mammography database

The *Catalonia mammography database* was collected at the Duran y Reynals hospital in Barcelona. It consists of 15 input attributes and a diagnostic result indicating whether or not a carcinoma was detected after a biopsy. The 15 input attributes include three clinical characteristics (Table 1) and two groups of six radiologic features, according to the type of lesion found in the mammography: mass or microcalcifications (Table 2).

A radiologist fills out a reading form for each mammography, assigning values for the clinical characteristics and for one of the groups of radiologic features. Then, the radiologist interprets the case using a five-point scale: (1) benign; (2) probably benign; (3) indeterminate; (4) probably malignant; (5) malignant. According to this interpretation a decision is made on whether to practice a biopsy on the patient or not. The Catalonia database contains data corresponding to 227 cases, all of them sufficiently suspect to justify a biopsy recommendation. For the purpose of this study, each case was further examined by three different readers—for a total of 681 readings—but only diverging readings were kept. The actual number of readings in the database is 516, among which 187 are positive (malignant) cases and 329 are negative (benign) cases.

3 Proposed Solution: Evolving a Rule-Based System

The tool proposed, the so-called COBRA system, is an evolved (fuzzy) rule-based system. It was conceived using Fuzzy CoCo [3] so as to exhibit both good diagnostic performance and human-friendly explanatory capabilities. In the rest of this section we describe briefly the COBRA system and the evolutionary technique used to develop it.

3.1 The COBRA System

The solution scheme we propose is depicted in Figure 1. It is composed of four elements: a user interface, a reading form, a database, and a diagnostic decision unit which consists of a fuzzy system and a threshold unit. Based on the 15 input attributes collected with the reading form, the fuzzy system computes a continuous appraisal value of the malignancy of a case. The threshold unit then

Table 2. Variables corresponding to radiologic features. There are two groups of variables that describe the mammographic existence of microcalcifications and masses.

Microcalcifications		Mass	
<i>v</i> ₄	Disposition	<i>v</i> ₁₀	Morphology
1	Round	1	Oval
2	Indefinite	2	Round
3	Triangular or Trapezoidal	3	Lobulated
4	Linear or Ramified	4	Polilobulated
<i>v</i> ₅	Other signs of group form	<i>v</i> ₁₁	Margins
1	None	1	Well delimited
2	Major axis in direction of nipple	2	Partially well delimited
3	Undulating contour	3	Poorly delimited
4	Both previous	4	Spiculated
<i>v</i> ₆	Maximum diameter of group [3-120] mm	<i>v</i> ₁₂	Density greater than parenchyma
<i>v</i> ₇	Number	1	Not
1	<10	2	Yes
2	10 to 30	<i>v</i> ₁₃	Focal distortion
3	>30	1	Not
<i>v</i> ₈	Morphology	2	Yes
1	Ring shaped	<i>v</i> ₁₄	Focal asymmetry
2	Regular sharp-pointed	1	Not
3	Too small to determine	2	Yes
4	Irregular sharp-pointed	<i>v</i> ₁₅	Maximum diameter
5	Vermicular, ramified	[5-80] mm	
<i>v</i> ₉	Size irregularity		
1	Very regular		
2	Sparingly regular		
3	Very irregular		

outputs a biopsy recommendation according to the fuzzy system's output. The threshold value used in this system is 3, which corresponds to the "indeterminate" diagnostic. Fuzzy CoCo is applied to design the fuzzy system in charge of appraising malignancy [5]. The web-based user interface was developed to provide online access to the system. In the developed tool, the user fills the reading form. In response, COBRA provides, in addition to the final biopsy recommendation, information about the rules involved in the decision. The tool can be also used to train novel radiologists as the reading form can access previously diagnosed cases contained in the database. It is available at the URL: <http://lslwww.epfl.ch/~cobra>.

3.2 Fuzzy CoCo: A Cooperative Coevolutionary Approach to Fuzzy Modeling

Fuzzy logic (which is a superset of conventional, Boolean logic) is a computational paradigm that provides a mathematical tool for representing and manipulating information in a way that resembles human communication and reasoning processes [6]. *Fuzzy modeling*, i.e., the task of identifying the parameters of

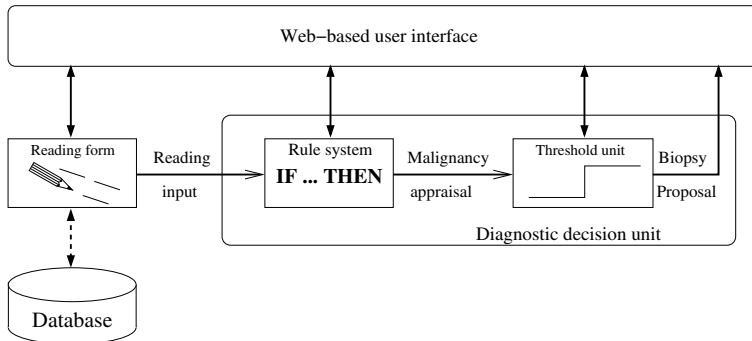


Fig. 1. The COBRA system comprises a user interface, a database containing selected cases, and a diagnostic decision unit which is the core of the system. In the decision unit the rule-based system estimates the malignancy of the case and the threshold unit outputs a biopsy recommendation.

a fuzzy inference system so that a desired behavior is attained, becomes difficult when the available knowledge is incomplete or when the problem space is very large, thus motivating the use of automatic approaches to fuzzy modeling—such as evolutionary algorithms. Fuzzy CoCo is a cooperative coevolutionary approach to fuzzy modeling, wherein two coevolving species are defined: database (membership functions) and rule base [3].

In Fuzzy CoCo, individuals of the first species encode values defining completely the membership functions for the variables of the system. Individuals of the second species define a set of rules of the form:

if (v_1 is A_1) and ... (v_n is A_n) then (output is C),

where the term A_v indicates which one of the linguistic labels of fuzzy variable v is used by the rule. The evolution of the two populations is controlled by two instances of a simple genetic algorithm. They apply fitness-proportionate selection to choose the mating pool, and apply an elitist strategy with an elitism rate Er to allow a given proportion of the best individuals to survive into the next generation. Standard crossover and mutation operators are applied with probabilities P_c and P_m , respectively.

An individual undergoing fitness evaluation establishes cooperations with one or more representatives, or *cooperators*, of the other species, i.e., it is combined with individuals from the other species to construct fuzzy systems. The fitness value assigned to the individual depends on the performance of the fuzzy systems it participated in. Cooperators are selected both fitness-proportionally and randomly from the last generation since they have already been assigned a fitness value. For a more detailed exposition of Fuzzy CoCo see [3].

3.3 Fuzzy-CoCo setup

Fuzzy parameters We used prior knowledge about the Catalonia database to guide our choice of fuzzy parameters. In addition, we took into account semantic

and syntactic criteria, defining constraints, respectively, on the membership-function and on the rule parameters [3, 7].

Genome encodings Fuzzy CoCo searches for four parameters: input membership-function values, relevant input variables, and antecedents and consequents of rules. To encode these parameters into both species' genomes, it is necessary to take into account the heterogeneity of the input variables as explained below.

Species 1: Membership functions. The fifteen input variables are of different types of values: continuous, discrete, and binary. The membership-function genome encodes only the 11 non-binary variables—three continuous and eight discrete—each with two parameters. The six parameters of continuous variables are encoded with 7 bits whereas the 16 discrete-variable parameters encode in 4 bits each. The total genome length is thus 106 bits.

Species 2: Rules. As each database case presents three clinical characteristics and six radiologic features according to the type of lesion found: mass or microcalcifications, the rule-base genome encodes, for each rule, 9 two-bit parameters (i.e., the three antecedents of the clinical-data variables and the six antecedents of one radiological-feature group) and two one-bit parameters (i.e., a bit to indicate whether the rule applies for mass or microcalcifications and the rule consequent). Furthermore, the genome contains an additional parameter corresponding to the consequent of the default rule. The genome contains, thus, $20 \times N_r + 1$ bits, where N_r denotes the number of rules.

Fitness function Table 3 provides expressions for four commonly employed diagnostic criteria which are important for evaluating the performance of our systems: sensitivity, specificity, accuracy and positive predictive values. Besides these criteria, the fitness function provides extra selective pressure based on two syntactic criteria: simplicity and readability.

Our fitness function combines the following five criteria: 1) F_{sens} : sensitivity, computed as the percentage of positive cases correctly classified; 2) F_{spec} : specificity, computed as the percentage of negative cases correctly classified (note that there is usually an important trade-off between sensitivity and specificity which renders difficult the satisfaction of both criteria); 3) F_{acc} : classification performance, computed as the percentage of cases correctly classified; 4) F_r : rule-base size fitness, computed as the percentage of unused rules; and 5) F_v : rule-length fitness, computed as the average number of unused variables per rule.

The fitness function is computed in three steps—basic fitness, accuracy reinforcement, and size reduction—as explained below:

1. Basic fitness. Based on sensitivity and specificity, it is given by

$$F_1 = \frac{F_{sens} + \alpha F_{spec}}{1 + \alpha},$$

where the weight factor $\alpha = 0.3$ reflects the greater importance of sensitivity.

2. Accuracy reinforcement. Given by

$$F_2 = \frac{F_1 + \beta F'_{acc}}{1 + \beta},$$

Table 3. Diagnostic performance measures. The values used to compute the expressions are: True positive (TP): correctly-detected positive cases, true negative (TN): correctly-detected negative cases, false positive (FP): negative cases diagnosed as positive, and false negative (FN): positive cases diagnosed as negative.

Performance criteria	Expression
Sensitivity	$\frac{TP}{TP + FN}$
Specificity	$\frac{TN}{TN + FP}$
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$
Positive predictive value (PPV)	$\frac{TP}{TP + FP}$

Table 4. Diagnostic performance of the average and the top evolved systems. Shown below are sensitivity, specificity, accuracy, and positive predictive value (PPV). In parentheses are the values, expressed in number of cases, leading to such performance measures.

	Average	Best system
Fitness	0.8834	0.9166
Sensitivity	98.78% (184.7/187)	100% (187/187)
Specificity	59.69% (196.4/329)	69.30% (228/329)
Accuracy	73.86% (381.1/516)	80.43% (415/516)
PPV	58.32%	64.93% (187/288)

where $\beta = 0.01$. $F'_{acc} = F_{acc}$ when $F_{acc} > 0.7$; $F'_{acc} = 0$ elsewhere. This step slightly reinforces the fitness of high-accuracy systems.

3. Size reduction. Based on the size of the fuzzy system, it is given by

$$F = \frac{F_2 + \gamma F_{size}}{1 + 2\gamma},$$

where $\gamma = 0.01$. $F_{size} = (F_r + F_v)$ if $F_{acc} > 0.7$ and $F_{sens} > 0.98$; $F_{size} = 0$ elsewhere. This step rewards top systems exhibiting a concise rule set, thus directing evolution toward more interpretable systems.

4 Results

A total of 469 evolutionary runs were performed, all of them searching for systems with up to 20 rules. Considering the best individual per run (i.e., the evolved system with the highest fitness value), all of them found systems whose fitness exceeds 0.83. Table 4 shows the diagnostic performance of both the average and the best run of all evolutionary runs. As mentioned before, the usual positive predictive value (PPV) of mammography ranges between 15 and 35%. As shown in Table 4, Fuzzy CoCo increases this value beyond 58% (58.32% for the average system and 64.93% for the best one).

Table 5. Four examples of possible input readings.

Variable	Case 1	Case 2	Case 3	Case 4
Clinical data				
Age	45	50	41	47
Menopause	pre	post	pre	pre
Antecedents	none	none	none	homolateral
Microcalcifications				
Group shape	undefined	–	–	undefined
Other features	nipple-oriented	–	–	none
Diameter	12 mm	–	–	8 mm
Number	< 10	–	–	< 10
Morphology	Sharp-pointed	–	–	ring-shaped
Regularity	Spar. irregular	–	–	very regular
Mass				
Morphology	–	round	oval	round
Margins	–	poorly delim.	poorly delim.	poorly delim.
Density	–	smaller	smaller	bigger
Distortion	–	no	yes	yes
Asymmetry	–	no	yes	no
Area	–	12 mm	10 mm	14 mm

The best evolved system exhibits a sensitivity of 100% and a specificity of 69.3% (i.e., 228 of the 329 negative cases are correctly detected as benign). Even though evolution was allowed to search for systems with 20 fuzzy rules, the best system found effectively uses 14 rules. In average, these rules contain 2.71 variables (out of 15) which are, furthermore, Boolean. Note that this latter fact does not contradict the use of a fuzzy approach as Boolean systems are a subset of the, more general, set of fuzzy systems. In fact, while fuzzy modeling techniques may find Boolean solutions, the contrary does not hold.

To illustrate how COBRA makes a diagnostic decision and the type of rules involved in it, we present below four interesting example cases. Table 5 shows the input reading values for the four cases. Cases 1 to 3 correspond to usual readings exhibiting either microcalcification or mass as radiologic features. Case 4 contains both types of features.

For each case, the system evaluates the activation of each rule and takes into account the values of the proposed diagnostics—i.e., 1 for Benign and 5 for Malignant. The system's malignancy appraisal is computed as the weighted average of these diagnostics. The final biopsy is given to the user together with the rules participating in the decision. Table 6 shows the rules activated for each case, the diagnostic they propose, the appraisal value, and the final biopsy suggestion using 3 (i.e., indeterminate) as threshold.

The system takes into account all active rules to compute a malignancy appraisal, that is it searches for, and ponders, all possible indicia of benignity and malignancy before making a decision. In particular, cases 3 and 4 illustrate well such behavior.

Table 6. Rule activation, appraisal, and diagnostic suggestion for the four example cases presented in Table 5.

Active rule	Case 1	Case 2	Case 3	Case 4
Rule 1	Benign	—	—	Benign
Rule 2	—	—	—	Benign
Rule 7	Benign	—	—	—
Rule 9	—	—	Benign	—
Rule 13	—	Malignant	—	Malignant
Rule 14	—	Malignant	Malignant	—
Appraisal	1 (Benign)	5 (Malignant)	3 (Indeterminate)	2.33 (Prob. Benign)
Suggestion	No biopsy	Biopsy	Biopsy	No biopsy

5 Conclusion

We presented the coevolutionary design of a (fuzzy) rule-based mammographic-interpretation assessment tool exhibiting both good performance and high interpretability. In this problem, the interpretability-accuracy trade-off is of crucial import, imposing several conditions on the system parameters thus limiting the flexibility of the system.

Developing COBRA, we concentrated on increasing the interpretability of solutions, obtaining excellent results. We note, however, that the consistency of the entire rule base and its compatibility with the specific domain knowledge can only be assessed by further interaction with medical experts (radiologists, oncologists). Besides, the developed tool must be fine-tuned through further tests submitted to the subjective reading of different radiologists.

References

1. L. Porta, R. Villa, E. Andia, and E. Valderrama. Infraclinic breast carcinoma: Application of neural networks techniques for the indication of radioguided biopsias. In J. Mira, R. Moreno-Díaz, and J. Cabestany, editors, *Biological and Artificial Computation: From Neuroscience to Technology*, volume 1240 of *Lecture Notes in Computer Science*, pages 978–985. Springer, 1997.
2. S. G. Orel, N. Kay, C. Reynolds, and D. C. Sullivan. Bi-rads categorization as a predictor of malignancy. *Radiology*, 211(3):845–880, June 1999.
3. C. A. Peña-Reyes. *Coevolutionary Fuzzy Modeling*. PhD thesis, École Polytechnique Fédérale de Lausanne - EPFL, 2002.
4. C. A. Peña-Reyes and M. Sipper. Fuzzy CoCo: A cooperative-coevolutionary approach to fuzzy modeling. *IEEE Transactions on Fuzzy Systems*, 9(5):727–737, October 2001.
5. C. A. Peña-Reyes, M. Sipper, and L. Prieto. Sensitive, specific, and interpretable: Evolving a fuzzy mammographic-interpretation assessment tool. In *2002 IEEE International Fuzzy Systems Conference Proceedings*, pages 837–842. IEEE Neural Network Society, 2002.
6. R. R. Yager and L. A. Zadeh. *Fuzzy Sets, Neural Networks, and Soft Computing*. Van Nostrand Reinhold, New York, 1994.
7. S. Guillaume. Designing fuzzy inference systems from data: An interpretability-oriented review. *IEEE Transactions on Fuzzy Systems*, 9(3):426–443, June 2001.

CBA generated receptive fields implemented in a Facial expression recognition task

Jose M. Jerez¹, Leonardo Franco², and Ignacio Molina³

¹ Escuela Técnica Superior de Ingeniería en Informática
Departamento de Lenguajes y Ciencias de la Computación
Universidad de Málaga

jja@lcc.uma.es

² Center for Computational Neuroscience
Department of Experimental Psychology
University of Oxford

leonardo.franco@psy.ox.ac.uk

³ Escuela Técnica Superior de Ingeniería en Telecomunicación
Departamento de Tecnología Electrónica
Universidad de Málaga
aimc@dte.uma.es

Abstract. Biologically inspired receptive fields are used to process input facial expressions in a modular network architecture. Local receptive fields constructed with a modified Hebbian rule (CBA) are used to reduce the dimensionality of input images while preserve some topological structure. In a second stage, specialized modules trained with backpropagation classify the data into the different expression categories. Thus, the neural net architecture includes 4 layers of neurons, that we train and test with images from the Yale Faces Database. A generalization rate of 82.9% on unseen faces is obtained and the results are compared to values obtained with a PCA learning rule at the initial stage.

1 Introduction

Face perception is a very important component of human cognition. Faces are rich in information about individual identity, but also about mood and mental state, being accessible windows into the mechanisms governing our emotions. Facial expression interactions are relevant in social life, teacher-student interaction, credibility in different contexts, medicine, etc. Face expression recognition is also useful for designing new interactive devices offering the possibility of new ways for humans to interact with computer systems.

From a neurophysiological point of view face recognition appears to be very important. Experiments both in monkeys and humans show the existence of dedicated areas in the brain where neurons respond selectively to faces ([1–3]). Also it has been shown that complex visual processing related to discrimination of faces is a very rapid task that can be completed in approximately 100 msec suggesting the involvement of a mainly feed-forward neural mechanism [4].

In this work we construct a modular neural network including two parts, trained in different ways: first, a hidden layer of neurons having the task of developing receptive fields, with the aim of reducing the dimensionality of the data, to be further classified by the second stage of specialized modules, sometimes called "experts", trained with backpropagation.

Neural-based models have been proposed in the last decades to imitate the perceptual capabilities of simple cells in striate cortex of biological systems [5–8], and some have been tested in natural scenarios [9, 10]. Stimulating a single neuron model with natural images, PCA [11] learning rule was shown to develop receptive fields (RFs) similar to those found in visual cortex in the early experiments of Hubel and Wiesel [12, 13]. The resulting RFs contained excitatory and inhibitory regions arranged in a preferred orientation. The emergence of these regions are related to long term potentiation (LTP) and depression (LTD) of the learning rule. According to the Hebbian postulate, PCA rule incorporates heterosynaptic LTD but not homosynaptic LTD, both described in the nervous system by different authors [14, 15]. The CBA rule proposed in this work to perform the first processing stage incorporates both types of synaptic competition and it is derived from the one proposed in [16] for neural assemblies formation, with the difference that incorporates a decay term.

The use of modular neural network architecture rather than fully connected ones seems to be a simple and effective solution to complicated tasks and also have the advantage of better generalization properties, reduced training times, and adaptability [17, 18]. Modular networks have been used successfully in several tasks such as speaker verification, face identification, time series prediction, etc. [19–21], and are also very useful tools for exploring hypothesis about brain function [22, 23].

Different systems have been constructed to deal with facial expressions ([24] for an interesting review), but few of them use a neural network approach (see for instance [25] and references therein). For example, in [26] a feedforward network with PCA input encoding of some facial features (eyes and mouth) is trained to classify emotions; Lisetti et al. [25] constructed a backpropagation network to classify the degree of expressiveness of faces; and Franco et al. [27] use an unsupervised algorithm to reduce the dimensionality of the input data and a self-organizing process to form the receptive fields. This work continues to explore the potential of neural networks to perform this kind of task, trying to respect some biological constraints implemented by the CBA rule, and using the capabilities of modular systems to obtain better classification rates.

2 The Database of Images

The Yale Face Database [28] contains 165 gray scale images in GIF format of faces from 15 male individuals of different races and containing different features (moustache, glasses, etc.) We use a subset of the database that consists of 14 subjects displaying 4 facial expressions: neutral, happy, sad and surprised faces. The images were cropped to obtain input images 8 pixels width by 24 pixels

height covering a portion of the face located on the left side. (See Figure 1). The images were centered taking the tip of the nose as reference and some light illumination correction was applied to a couple of images; both operations were carried out by a human observer. The resulting images were transformed to pgm 8-bit gray scale format, ready to be fed into the network after a linearly scaled transformation of pixel intensity into the interval [0, 1].

Figure 1 shows a sample of the different expressions displayed by one of the subjects. In the leftmost image the area of the face cropped and used as input is shown.

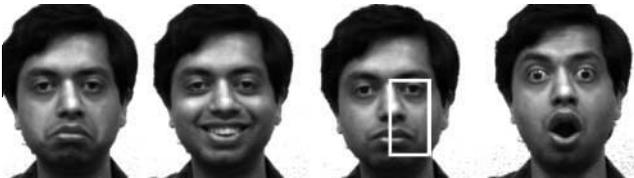


Fig. 1. Sample subject showing the four full face expressions (neutral, happy, sad and surprised). The white rectangle inside the rightmost figure corresponds to the area cropped and used as input for the neural network.

3 Network architecture

The network architecture is similar to that proposed in the work of Franco et al. [27], and it consists of a 4 layer modular neural structure composed of sigmoidal units. The input layer has 192 units corresponding to the 24x8 pixels of the area cropped from the original images. Every input neuron transmits information through a single hebbian weight, projecting to a specific neuron in the first hidden layer, selected according to a self-organized process. Thus, one has at this level a new reduced representation of the images, expressed by the activity of 48 neurons, that preserves some topological aspects of the original input. The whole network architecture is shown in figure 2, where at the top we show a sample input image followed by the structure of the receptive fields corresponding to the first hidden layer neurons.

After this unsupervised compression the network splits into three modules corresponding to the expressions different from the neutral face: happy, sad and surprised. The structure of the modules could depend on the emotion they specialize in; in the case we consider here they have all the same type of architecture: one hidden layer fully connected with one output unit. There is a difference in the number of hidden neurons belonging to each modules since the recognition of happy and surprised faces is much easier than the recognition of sad ones, a fact that was previously known from experiments both with humans and computers [22]. It was necessary to put 4 hidden neurons for sad faces while 3 neurons

were enough for happy and surprised ones. In this way the output of the whole network has 3 neurons that should be all OFF when a neutral face is presented, while when a face displaying an emotion is shown, the corresponding module output unit should be ON.

3.1 Receptive fields formation process

As we mentioned before, the first layer of weights is organized according to the dynamics of both CBA and PCA learning rules, with the aim of obtaining receptive fields of 2×2 pixels from the input images.

The neuron model consists of a vector \mathbf{x} of inputs, a vector \mathbf{w} of synaptic weights, and a scalar output y , given by $y = \mathbf{w} \cdot \mathbf{x}$, that represents the neuron activity.

In a previous work [16] we proposed a new correlational learning rule (BA, for bounded activity) that formed stable neural attractors in a recurrent network. The CBA learning rule is essentially a modification of the BA rule in which an extra term to implement the heterosynaptic LTD has been incorporated. Thus, the resulting synaptic modification equation for the CBA rule is a Hebbian-type learning rule with an specific form of stabilization, defined as

$$\Delta w_i = \alpha x_i y (y - \tau)(\lambda - y) - \beta y w_i, \quad (1)$$

where α is the learning rate, y is the neuron activity and x_i is the input activity from the i -synapsis. The term λ avoids the unbounded weight growing in a plausible way, and can be interpreted as a neuronal parameter representing the maximum level of activity at which the neuron might work. The CBA modification equation also defines a threshold τ that determines whether depression or potentiation occurs depending on pre- and postsynaptic activities. Finally, the parameter β controls the heterosynaptic competition effect, such that the strength of synapses can change even in the absence of presynaptic activity to those synapses. All these parameters adopt positive values. The effect of this adaptation mechanism in the receptive fields formation process is that the graded response elicited after stimulus presentation leads the neural activity either to high or resting levels.

On the other hand, the PCA rule, proposed by Oja [11], is known to perform a principal component analysis of the input vectors, converging to the largest eigenvector, while normalization is ensured. The change in the weight values can be written as

$$\Delta w_i = \alpha y (x_i - y w_i) \quad (2)$$

4 Simulations results

As the amount of data available for training and testing is limited (14 subjects, 56 images), a cross-validation procedure was used [18]. In this procedure 13 out of the 14 available subjects are chosen to train the network and the 4 unseen

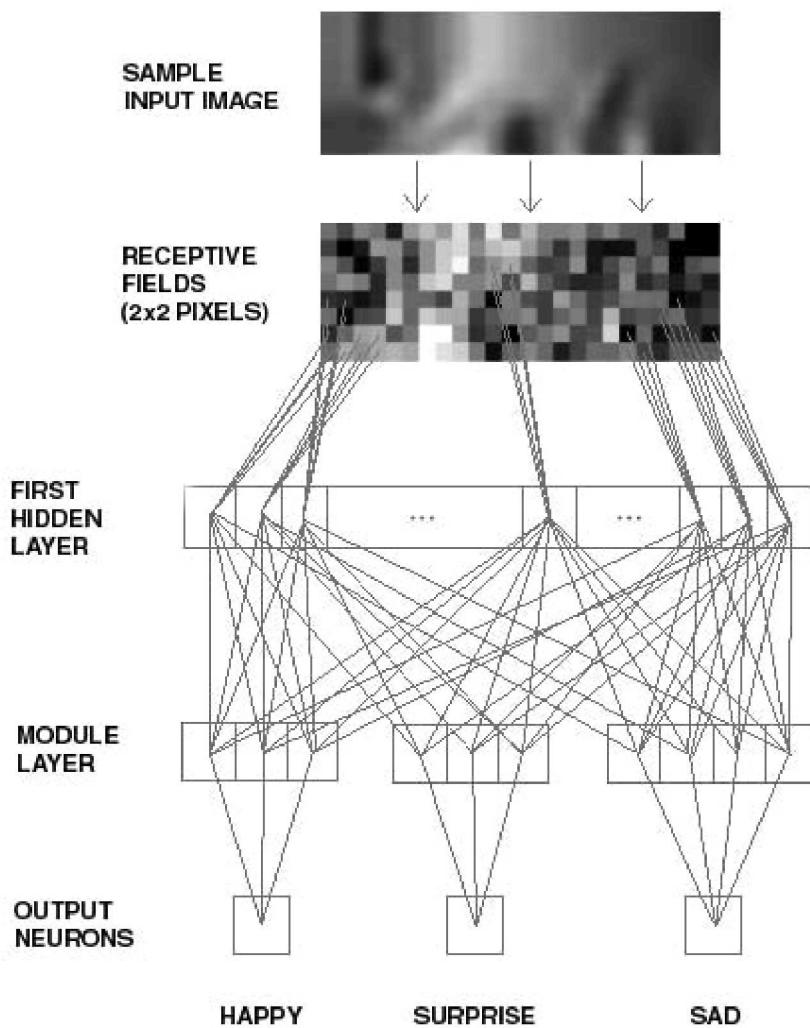


Fig. 2. Schematic structure of the network architecture used to perform facial expression recognition. The network has 192 inputs corresponding to the part of the face considered and being projected via hebbian weights to 48 neurons in the first hidden layer, with self-organized receptive fields. The modules, specialized in the different expressions: happy, sad and surprise, have a one hidden layer structure with an output that should be activated when a face displaying its corresponding expression is presented at the input. At the top, one of the input sample images is shown.

faces of the remaining subject are used to test the generalization ability of the system. The process is repeated 14 times, one for each subject being kept out of the training set, and averages values are taken.

The first layer of 192 weights, one for each input pixel, is trained with the CBA and PCA unsupervised learning rules. The rest of the weights, those belonging to the modules, were trained with the standard backpropagation algorithm (Hertz, Krogh & Palmer, 1993; Haykin, 1995). To prevent overtraining and to obtain a better generalization ability we monitor the training error on each input image, stopping the training on such image when the error was lower than 0.10. Since the backpropagation training is an on-line procedure, at the end of the training phase the average error per example was decreased to 0.02, approximately. All layers of weights were trained at the same time upon the presentation of an input image. Table 1 shows the parameters setting for the proposed neural architecture.

Table 1. Setting of parameters for the neural architecture (CBA, PCA and BP algorithms).

Learning constant for BP algorithm, η	0.01
Learning constant for unsupervised learning, α	0.001
Maximum level of activity, λ	1.0
Threshold level, τ	0.1
Heterosynaptic competition term, β	0.000015
Input values range, x	[0.0, 1.0]
Weights initialization range,	[0.1 ± 0.001]

The generalization error rates produced by the three modules specialized in different expressions for CBA and PCA learning rules are shown in table 2. We can observe that the generalization rate is similar for both rules, although the error is distributed uniformly among the three modules for CBA, whereas the recognition of the sad faces (more difficult to recognize than the others) is worst for PCA rule.

5 Conclusions and future work

We explore the generalization ability of a modular neural system to classify facial expressions. Using a mixed learning scheme composed by unsupervised-supervised training, we obtain a generalization ability on novel faces of 82.9%. The unsupervised processing used the CBA rule that incorporates an homosynaptic term that prevents an excessive growing of the weights and lead to a similar level of recognition rate than the PCA rule. An interesting feature of the results is that a similar performance is obtained for the different modules

Table 2. Generalization error rates for the modules, specialized in happy, sad and surprise faces, and for the whole net using first layer PCA and CBA rules in the formation of receptive fields. The generalization error is measured after the training procedure succeeds, when the training error per example turns out to be around 0.02.

Module	Expression Error (PCA)	Error (CBA)
Happy	0.057	0.057
Sad	0.086	0.057
Surprise	0.027	0.057
Total	0.170	0.171

specialized on different facial expressions, in contrast to the case when the PCA rule is implemented, and it may need further tests to be fully understood.

The advantage of using a modular approach is that it will permit the addition of new modules to recognize different expressions that could be trained separately.

Possible extensions of this work includes testing the system with a more extensive database, using the whole face as input, and studying the effect of changing the sizes of the receptive fields. It would also be desirable that the network itself should be capable of performing the identification of a face in a complex input image, permitting the use of the system in a real environment and potentially to be mounted on a robot.

References

1. Kanwisher, N., McDermott, J., Chun, M.: The fusiform face area: A module in human extrastriate cortex specialized for face perception. *Journal of Neuroscience* **17** (1997) 4302–4311
2. Perret, D., Rolls, E., Caan, W.: Visual neurons responsive to faces in the monkey temporal cortex. *Experimental Brain Research* **47** (1982) 329–342
3. Dosimone, R.: Face selective cells in the temporal cortex of monkey. *Journal of Neuroscience* **3** (1991) 1–8
4. Leiky, S.: Fine discrimination of faces can be performed rapidly. *Journal of Cognitive Neuroscience* **12** (2000) 848–855
5. Von der Malsburg, C.: Self-organization of orientation sensitivity cells in striate cortex. *Kybernetik* **14** (1973) 85–100
6. Bienenstock, E., Cooper, L., Munro, P.: Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. *Journal of Neuroscience* **2** (1982) 32–48
7. Linsker, R.: From basic network principles to neural architecture: Emergence of orientation-selective cells. *Proceedings of the National Academy of Science* **83** (1986) 8390–8394
8. Miller, K.: A model for the development of simple cell receptive fields and the ordered arrangement of orientation columns through activity-dependent competition between on-and off-center inputs. *Journal of Neuroscience* **14** (1994) 409–441

9. Law, C., Cooper, L.: Formation of receptive fields according to the bcm theory of synaptic plasticity in realistic visual environment. *Proceedings of the National Academy of Science* **91** (1994) 7797–7801
10. Shouval, H., Liu, Y.: Principal component neurons in a realistic visual environment. *Network* **7** (1996) 501–515
11. Oja, E.: A simplified neuron model as a principal component analyzer. *Mathematical Biology* **15** (1982) 267–273
12. Hubbel, D., Wiesel, T.: Receptive fields, binocular interaction and functional architecture in the cats visual cortex. *Journal of Physiology* **160** (1962) 106–154
13. Hubbel, D., Wiesel, T.: Receptive fields and functional architecture of monkey striate cortex. *Journal of Physiology* **195** (1968) 215–243
14. Artola, A., Brcher, S., Singer, W.: Different voltage-dependent threshold for inducing long-term depression and long-term potentiation in slices of rat visual cortex. *Nature* **347** (1990) 69–72
15. Abraham, W., Goddard, G.: Asymmetric relationships between homosynaptic long-term potentiation and heterosynaptic long-term depression. *Nature* **305** (1983) 717–719
16. Vico, F., Jerez, J.: Stable neural attractors formation: Learning rules and network dynamics. *Neural Processing Letters* **to appear** (2003)
17. Franco, L., Cannas, S.: Generalization properties of modular networks implementing the parity function. *IEEE Transactions on Neural Networks* **12** (2001) 1306–1313
18. Haykin, S.: *Neural Networks: A comprehensive foundation*. IEEE Press (1994)
19. Bennani, Y.: Multi-expert and hybrid connectionist approach for pattern recognition: speaker identification task. *International Journal of Neural Systems* **5** (1994) 207–216
20. Dailey, M., Cottrell, G.: Organization of face and object recognition in modular neural networks models. *Neural Networks* **12** (1999) 53–1074
21. Petridis, V., Kehagias, A.: Predictive modular neural networks: Applications to time series. Kluwer Academic Publishers, Boston (1998)
22. Dailey, M., G.W., C., Adolphs, R.: A six-unit network is all you need to discover happiness. In: *Proceedings of the 22th Annual Conference of the Cognitive Science Society*. (2000)
23. Jacobs, R.: Nature, nurture, and the development of functional specializations: a computational approach. *Psychonomic Bulletin and Review* **4** (1997) 299–309
24. Fasel, B., Luettin, J.: Automatic facial expression analysis: a survey. *Pattern Recognition* **36** (2003) 259–275
25. Lisetti, C., Rumelhart, D.: Facial expression recognition using a neural network. In: *Proceedings of the 11th International Florida Artificial Intelligence Research Society Conference*, Menlo Park, CA (1998)
26. Padgett, C., Cottrell, G.: A simple neural network models categorical perception of facial expressions. In Mahwah, ed.: *Proceedings of the 20th Annual Cognitive Science Conference*, Madison, WI, Lawrence Erlbaum (1998)
27. Franco, L., Treves, A.: A neural network facial expression recognition system using unsupervised local processing. In Loncaric, S., Babic, H., eds.: ISPA-01 Proceedings of the 2nd IEEE R&EURASIP International Symposium on Image and Signal Processing and Analysis. (2001) 628–632
28. Belhumeur, P., Kriegman, D.: The yale faces database. <http://cvc.yale.edu/projects/yalefaces/yalefaces.html> (1997)

A Hybrid Face Detector based on an Asymmetrical Adaboost Cascade Detector and a Wavelet-Bayesian-Detector

Rodrigo Verschae and Javier Ruiz del Solar

Department of Electrical Engineering, Universidad de Chile.

Email: {rverscha, jruizd}@ing.uchile.cl

Abstract. In this paper is proposed a hybrid face detector that combines the high processing speed of an Asymmetrical Adaboost Cascade Detector with the high detection rate of a Wavelet Bayesian Detector. This integration is achieved by incorporating this last detector in the middle stages of the cascade detector. Results of the application of the proposed detector to a standard face detection database are also presented.

Keywords : Face Detection, Adaboost, Wavelet-based Face Detection.

1. Introduction

Face Analysis (face recognition, face detection, face tracking, facial expression recognition, etc.) is a very lively and expanding research field. The increasing interest in this field is mainly driven by applications like access control to buildings and (computational) systems, identification for law enforcement, borders control, identification and verification for credit cards and ATM, human-machine interfaces, communications, multimedia, and very recently passive recognition of criminals (e.g. terrorists, hooligans) in public places or buildings (airports, train stations, stadiums, etc.).

Face detection is a key step in almost any computational task related with the analysis of faces in digital images. Given an arbitrary image, the goal of a face detection system is to find all contained faces and to determine the exact position and size of the regions containing these faces. When analyzing real-world scenes, face detection is a challenging task, which should be performed robustly and efficiently, regardless variability in scale, location, orientation, pose, illumination and facial expressions, and considering possible object occlusions.

Several approaches have been proposed for the computational detection of faces in digital images. A very comprehensive review can be found in [1][8]. Main approaches can be classified as: (i) feature-based, which uses low-level analysis (color, edges, ...), feature analysis or active shape models, and (ii) image-based, which employs linear subspace methods, neural networks or statistical analysis [1]. Image-based approaches have shown a much better performance than feature based. Within the first category are to mention the seminal works of Sung and Poggio [5], based on a Gaussian modeling of PCA decompositions together with a neural classifier, and the one of Rowley *et al.* [3], which use a neural network composed by retinal-based receptive fields.

According to the current state-of-the-art (for references and exact performance information please see [1] or [8]), the following face detection systems have the higher recognition rates, tested using standard face databases: (i) SNoW (Sparse

Network of Winnows), proposed by Roth *et al.* [2], which uses sparse feature vectors, two linear processing neurons and the Winnow learning rule, (ii) Schneiderman and Kanade detector [4], which uses wavelet analysis together with a Bayesian classifier, and (iii) Yang *et al.* Detector [7], which uses Self-Organizing Map clustering and Fisher linear discriminant analysis.

On the other, the system proposed by Viola and Jones [6] outperforms previous systems in terms of processing speed, by keeping an acceptable recognition rate. This system uses simple, rectangular features (a kind of Haar wavelets), a cascade of filters that discard non-face images, the integral image for fast computation of these filters and asymmetrical Adaboost as a boosting strategy for the training of the detectors.

The aim of this paper is to propose a hybrid face detector that combines the high processing speed of the cascade detector of Viola and Jones with the high detection rate of the detector of Schneidermann and Kanade. This behavior is achieved by incorporating this last detector in the middle stages of the cascade detector. The detection results obtained with both detectors are fused using a new heuristic algorithm that merges overlapping face detection regions.

The paper is structured as follows. In section 2 the proposed hybrid face detector is described. In section 3 some face detection results using the proposed detector are presented. Finally, some conclusions of this work are given in section 4.

2. Proposed Hybrid Face Detector

For simplicity in the explanation of the proposed hybrid detector, in this paper we will call the Viola and Jones detector *cascade Adaboost detector*, and the Schneidermann and Kanade detector *Wavelet-Bayesian detector*.

The proposed hybrid detector takes as starting point the cascade Adaboost detector [6], a very fast face detector which detection rate can be increased by improving the detection of faces which usually present problems, like for example black people faces, low contrast faces, and asymmetrical illuminated faces.

For increasing the detection rate of the hybrid detector, the Wavelet-Bayesian detector is also included [4]. High processing speed is kept by applying this last detector inside the Adaboost cascade, after the first 3 Adaboost filters are applied. Afterwards, the detection results of both detectors are fused.

2.1 System Overview

The block diagram of the proposed detector is presented in figure 1. For detecting faces at different scales a multiresolution analysis is performed by scaling the input image by a factor of 1.2. This scaling is performed until images from about 24x24 pixels are obtained. This operation is carried out in the *Multiresolution Analysis Module*. In the *Window Extraction Module* for each of these scaled versions of the input image, all possible windows of 24x24 pixels are extracted. These windows are then processed by the face detectors.

For detecting the faces, each of the 24x24 windows is processed by the first three filters of the cascade Adaboost detector. When a window is classified as non-face by any of these three filters non-further processing is done. In the opposite case, it goes

further in the cascade. After the application of the third filter, the remaining windows are sent in parallel to the next part of the cascade (filters 4 to 21), which keep on discarding non-face windows, and to the Wavelet-Bayesian detector. This detector makes a histogram equalization of the received windows (*Pre-Proc. Module*) and a Wavelet Decomposition followed by a Bayesian Classification (*Wavelet Face Detection Module*). A switch before the Wavelet-Bayesian detector discards one window every two windows for speeding the processing (see fig. 1). After all windows are processed and classified as faces or non-faces, in the *Overlapping Detection Processing Module* all face windows are processed and fused for determining the size and position of the detected faces.

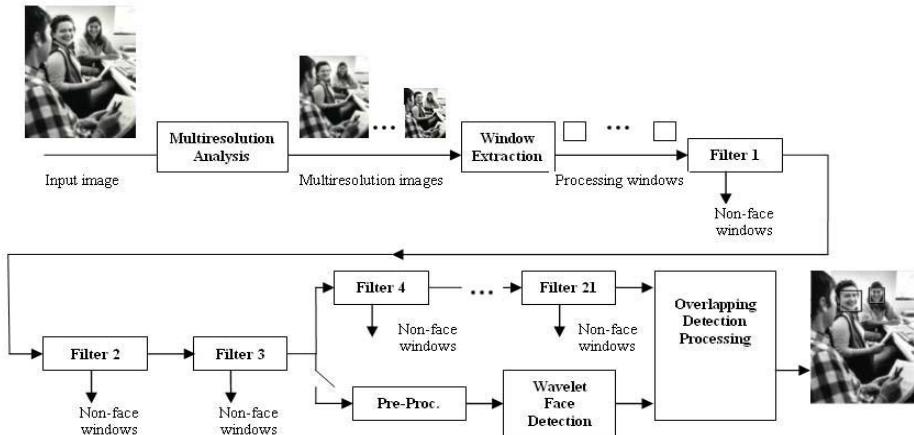


Fig. 1. Block diagram of the proposed hybrid face detector. Filters 1 until 21 correspond to the filters of the cascade Adaboost detector. Pre-Processing and Wavelet Face Detection modules correspond to the Wavelet Bayesian detector.

2.2 Cascade Adaboost Detector

The implemented cascade face detector detects frontal faces with small in-plane rotations and it is based mainly on [6]. This face detector corresponds to a cascade of filters that discard non-faces and let faces to pass to the next stage of the cascade. This architecture seeks to have a fast face detector, considering the fact that only a few faces are to be found in an image, while almost all the image area correspond to non-faces. The fast detection is achieved in two ways: (i) having a small complexity in the first stages of the cascade (filters composed by few detectors, 2 to 5) and greater complexity in the later stages of the cascade (filters composed by many detectors, 100 to 400), and (ii) using simple features called rectangular features (the detectors), which are quickly evaluated using a representation of the image called integral image.

Each of the filters of the cascade is trained using an asymmetric version of Adaboost (see explanation in [6]), which gives more importance to errors occurring during the training process when classifying faces as non-faces than non-faces as faces. Adaboost sequentially trains and selects a small number of rectangular features.

The main problem of this face detector is the large training time, which can extend for weeks or even months.

The final cascade Adaboost detector implemented in this work has 21 layers and was trained in about a month. To train each layer 1338 face images were used, and non-faces images were collected from 8000 images that did not contain any face. All these training images were obtained mainly from Internet, especially from the google image searcher, and from personal images databases. For training the first 2 filters, 4000 and 3000 non-face images were randomly chosen from our dataset. For training the remaining filters of the cascade 1500 non-face images wrongly classified by the already trained cascade were collected (a kind of bootstrapping). For reducing the training time a randomly chosen subset of the set of rectangular features was used in each iteration of the Adaboost algorithm. Each time that a decision rule was trained, only 50% of the training examples (faces and non-faces) were employed. As a result of the training, the final number of detectors used at each of the 21 stages of the cascade was 2, 5, 20, 20, 50, 50, 100, 100, 100, 100, 200, ..., 200 and 400, respectively.

2.3 Wavelet-Bayesian Detector

The Wavelet-Bayesian detector uses a Wavelet decomposition for extracting feature patterns, together with a naive Bayes classifier that first estimates the probability of that a given window corresponds to a face or a non-face, using these patterns, and then it employs the ratio between these two values for determining if the window is a face or a non-face. As in [4], this is carried out using the a posteriori probabilities of different patterns occurring in faces and non-faces. These patterns are extracted from a two level wavelet transform [9].

Groups of 8 coefficients of the wavelet transform of the image were used, and each of the coefficients was quantized to 3 values. The quantization thresholds were chosen from 5 values (see details in [9]). These thresholds were chosen for maximizing the detection rate and diminishing the false positive rate. Six different types of groups of coefficients were used, which represent inter-orientation coefficients (LH-HL level 1 and LH-HL level 2), inter-frequency coefficients (LH level 1 – LH level 2, HL level 1 – HL level 2) and intra-frequency coefficients (LH level 1, LH level 2, HL level 1, HL level 2).

For training the detector 800.000 faces were generated from 1500 faces, using variations in different scales, rotations and displacements. 2.000.000 non-faces were collected from images non-containing faces. All these training images were obtained mainly from Internet, especially from the google image searcher, and from personal images databases. For training this detector much more images were used than in the case of the cascade Adaboost detector, because the training of this last detector is very time consuming. During training 120.000 more non-faces were collected in cases where the detector has wrongly classified them (bootstrapping).

2.4 Overlapping Detection Processing

Face windows are processed and fused for determining the size and position of the detected faces. Overlapping detections were processed for filtering false detections

and for merging correct ones. All detections (detected face region) were separated in disjoint sets using the following heuristic. Considering the inscribed circumference of each square face region, two detections belonged to the same set if the sum of their circumference radius is smaller than 0.4 times the distance between their centers, and if each radius is not larger than twice the other. If a set contained only one element, this detection is discarded. Detections belonging to each set are merged by averaging the coordinates of the corners of all square face regions. Figure 2 shows an example of applying this heuristic on an image which has two close faces.

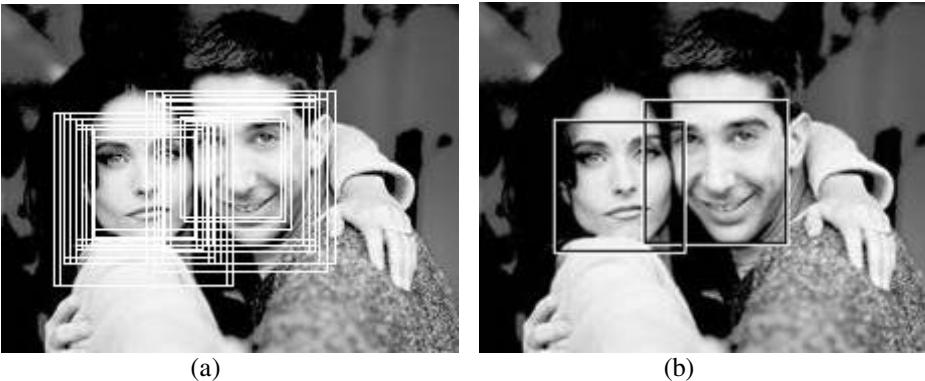


Fig. 2. Results from the use of the heuristic that separates and merges overlapping detections. (a) Overlapping detection, (b) Final detections.

3. Face Detection Results

In order to test the system we used the database used in [3] that correspond to the union of a database of Rowley, Baluja and Kanade and a database of Sung and Poggio. That is available in [10] and is commonly known as MIT+CMU face database. This database consists in 130 images containing 507 faces.

Figure 3 shows a graph comparing the detection results, ROC (Receiver Operating Characteristic) curve, for the hybrid detector, the cascade Adaboost detector and the Wavelet-Bayesian detector. This last detector was used together with three filters from the cascade for speeding the results. Table 1 shows the exact values of this evaluation. For obtaining the different points of the ROC curve for the hybrid detector and the Wavelet-Bayesian detector, different thresholds of the wavelet decomposition were used, while in the cascade Adaboost detector, filters at the end of the cascade were sequentially removed (see details in [9]).

From the presented results it can be notice that the hybrid detector has an increase of 2% to 5% in the detection rate over the cascade detector, improving the detection of black faces and faces with low contrast. This is shown with more details in figure 4. Results for the hybrid detector are presented in figures 4 (a) and (c) and results of the application of the cascade detector on the same images are presented in figures 4 (b) and (d). These results show how the hybrid detector detects black faces and low contrasts that the cascade detector does not detect.

Concerning processing time we can say that following. The time that took to evaluate one point of the ROC curve was about 19 minutes for the hybrid detector

detectors, while it was 10 minutes for the cascade detector. In contrast to that, the time for evaluating the Wavelet-Bayesian detector was more than 6 hours.

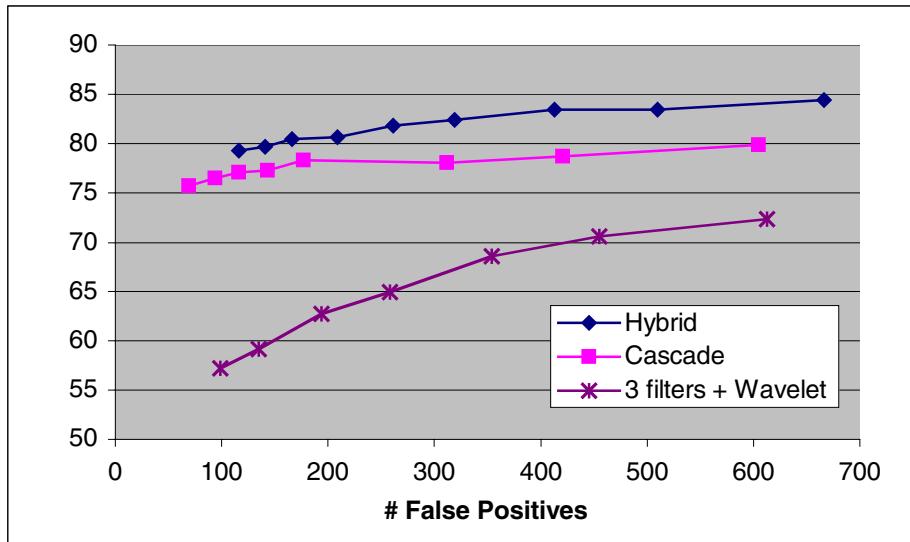


Fig. 3. Comparison ROC curves of Hybrid, Cascade and Wavelet detectors on MIT+CMU test sets.

Table 1. Evaluation of Hybrid, Cascade and Wavelet detectors on MIT+CMU test set.

Hybrid Detector		Adaboost Cascade Detector		3 Filters + Wavelet-Bayesian Detector	
Detection Rate (%)	False Positives	Detection Rate (%)	False Positives	Detection Rate (%)	False Positives
79.29	116	75.73	69	57.19	99
79.68	141	76.52	94	59.17	135
80.47	166	77.12	116	62.72	194
80.67	209	77.31	143	64.89	258
81.85	261	78.3	177	68.63	354
82.44	319	78.1	312	70.61	455
83.43	413	78.69	421	72.38	613
83.43	510	79.88	605		
84.41	666				

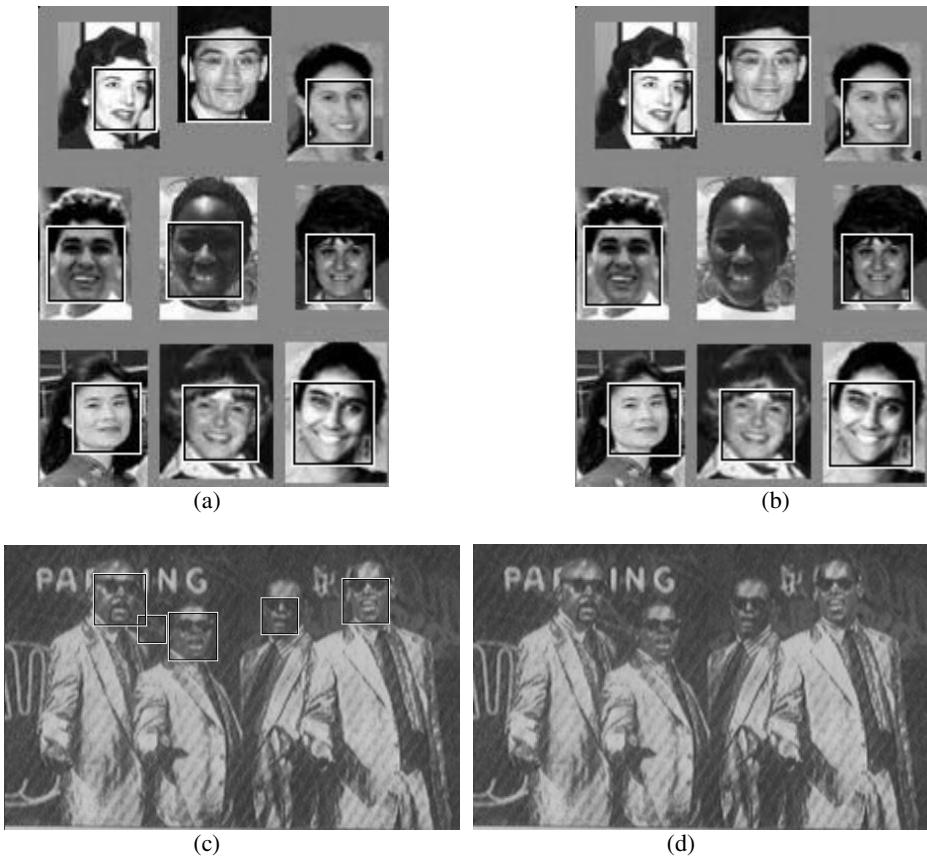


Fig. 4. Detection results of the hybrid detector (a) and (c), and the Adaboost cascade detector (b) and (d).

4. Conclusions

In this article was proposed a hybrid face detector that combines the high processing speed of the cascade Adaboost detector with the high detection rate of the Wavelet-Bayesian detector. This integration was achieved by incorporating this last detector in the middle stages of the cascade detector. Results of the application of the proposed detector to a standard face detection database were presented. These results shown that the obtained detector has higher detection rate than the Adaboost detector, while keeping its high processing speed.

Currently we are working for improving the detection rates of the cascade Adaboost as well as the Wavelet-Bayesian detectors, which should improve the detection rate of the hybrid system. For doing that we are building better face database and optimizing the training of the cascade detectors. The use of methods for compensating the illumination, such as linear function subtracting [3], should also improve the performance of the final detector.

References

1. E. Hjelmås, B. K. Low, "Face detection: A survey", *Computer Vision and Image Understanding* 83, 236-274, 2001.
2. D. Roth, M. Yang, and N. Ahuja, "A SNoW-based Face Detector", *Advances in Neural Information Systems* 12, MIT Press, 2000.
3. H. Rowley, S. Baluja, and T. Kanade, " Neural Network-Based Detection", *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol.20, No. 1, 23-28, 1998.
4. H. Schneidermann and T. Kanade, " A statistical model for 3D object detection applied to faces and cars", *IEEE Conf. on Computer Vision and Pattern Recognition*, Vol. 1, 746 – 751, 2000.
5. K. Sung and T. Poggio, "Example-Based Learning for Viewed-Based Human Face Detection", *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol.20, No. 1, 39-51, 1998.
6. P. Viola and M. Jones, "Fast and Robust Classification using Asymmetric Adaboost and a Detector Cascade", *Advances in Neural Information Processing System* 14, MIT Press, Cambridge, MA, 2002.
7. M. Yang, N. Ahuja, and D. Kriegman, " Mixtures of linear Subspaces for Face Detection", *Fourth IEEE Int. Conf. on Automatic Face and Gesture Recognition*, 70 – 76, 2000.
8. M. Yang, D. Kriegman, N. Ahuja, "Detecting Faces in Images: A Survey", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 24, No 1, pp. 34-58, 2002.
9. R. Verschae, "Detecting Faces in Image Databases", Thesis for the title of Electrical Engineer, Department of Electrical Engineering, Universidad de Chile, 2003 (in Spanish).
10. CMU+MIT face database. Available until March 2, 2003 on http://vasc.ri.cmu.edu/IUS/eyes_usr17/har/har1/usr0/har/faces/test/.

Neural Net Generation of Facial Displays in Talking Heads

Max H. Garzon¹, Kiran Rajaya²,

¹ Computer Science Division
The University of Memphis
Memphis, TN 38152-3240 USA
mgarzon@memphis.edu
<http://www.cs.memphis.edu/~garzonm>

² Department of Biostatistics, St Jude Children's Research Hospital
Memphis, TN 38105 USA
kiran.rajaya@stjude.org

Abstract. Anthropomorphic representations of software agents (talking heads) are being used to facilitate interaction with human users. They are commonly programmed and controlled by ontologies designed according to intuitive and heuristic considerations. Here we describe successful training of a recurrent neural net capable of controlling emotional displays in talking heads, on a continuous scale of negative, neutral, and positive feedback, that are meaningful to users in the context of tutoring sessions on a particular domain (computer literacy). The neurocontrol is a recurrent neural network that autonomously synchronizes the movements of facial features in lips, eyes, eyebrows, and gaze in order to produce facial displays that are valid and meaningful to untrained human users. The control is modular and can be easily integrated with semantic processing modules of larger agents that operate in real-time, such as tutoring and face recognition systems.

1 Introduction

Talking heads are anthropomorphic representations of software agents used to facilitate interaction between avatars/software agents and a human. They are used, in particular, in applications where the bandwidth of the interaction between the agent and the user is very high, so that a text-only interface would either unduly tax the user or would be simply out of the question. They afford new interfaces that have been explored that rely on nonverbal behavior to expand the bandwidth and speed of real-time communication between human and computer [4,5,6]. Many talking heads have been described in the recent literature and their advantages have been discussed in several places. Notable examples are *Ananova* (www.ananova.com), a virtual broadcaster, *Grace*, (www.cmu.edu/cmnnews/020906/020906_grace.html) a robot that recently delivered a speech at the 2002 AAAI conference on Artificial Intelligence, and *Autotutor* (www.autotutor.org) a software agent agent being developed at The

University of Memphis that is capable of tutoring a human user in a restricted domain of expertise such as computer literacy or physics at the level of an untrained human tutor. AutoTutor is an embodied conversational agent consisting of a dialog module that handles the computational intelligence to carry on a conversation with the student, and an interface module embodied by a talking head to convey non-verbal feedback.



Fig. 1. Arto, a talking head for an AutoTutor, an intelligent tutoring system.

Despite their popularity and potential for applications, the design and implementation of talking heads has been primarily a job in heuristics. A notable exception is Massaro's Baldi [9], based on a neurofuzzy model, but the primary emphasis of the model is largely on mouth sequences with a high degree of realism that permits, for example, deaf people to learn to adequately pronounce English language. The usual design method, however, consists in designing some prototypical facial expressions or animations that presumably reflect the desired type of responses, and some heuristic ontology to display them based on pre-programmed action sequences. While the result may suffice in many cases, it is clear that the resulting solutions suffer of a number of problems, such unnaturalness of the expressions, robotic appearance, inappropriate responses, and perhaps worst of all, a continuing programming/scripting effort.

In previous work [1,2], we successfully trained feedforward neural net modules to generate facial features in a talking head to convey nonverbal feedback to tutees in the course of normal conversation to complement speech generated by a synthetic engine. Thus facial features can be classified as *primary* (mouth positions, time durations) and *derived* facial features (to a higher or lesser degree, all the remaining features: brow heights, eye widths, etc.). It was also established that the feedforward neural networks cannot train a network to output the proper signals to control the primitive features (mouth position and time duration) to complete the solution to the original problem.

In this paper, we use these modules to successfully train a recurrent neural network that is capable of generating facial expressions that convey meaningful emotional content to users in the context of tutoring sessions on a particular domain (computer literacy) on a continuous scale of negative, neutral, and positive feedback [1,2,8]. The network autonomously controls and synchronizes the movements of facial features in lips, eyes and eyebrows in order to produce facial animations that are valid and meaningful to untrained human users. The autonomous nature of the resulting network

makes it possible to easily interface it with semantic processing modules of larger agents that operate in real-time, such as AutoTutor.

Moreover, the training procedure requires a novel learning algorithm, *cascade inversion*, that generalizes backpropagation through time. Also, the results obtained appear to reveal dependencies among facial features that may be useful to make identification and recognition of people more efficient, as discussed below in the conclusions.

2 Emotional Facial Displays

Facial behavior covers a wide spectrum in human conversation. All these features are used by human speakers in a highly dynamic and well orchestrated process that has nothing to envy to the complexity and information bandwidth already present in verbal discourse process. For example, facial animations alone are an old and fertile art for communication, as evidenced by long traditions of popular cartoonists and animators. Here, we restrict our attention to a minimal subset of features that allow a simplified but automatic generation of facial displays in a talking head nonverbal for communication from a computer to a human user in a way that is naturalistic, ergonomic, and easily understood by virtually anyone without further training beyond normal human-human interaction.

2.1 Facial Features and Displays

Early work by Ekman provides a full set of 64 facial features in his Facial Action Unit Coding System to describe static human faces [5]. The full set of 64 FACS units as reduced to eight, namely *two eye brows*, *two eye lids*, *two eye directions* -- horizontal and vertical, as well as a *mouth position*, and a *frame duration*. For our purposes, a facial display is an animation consisting of a sequence of various key static frames (ranging from 3 to 6) shown at appropriate speed to be perceived as motion by the human eye. This reduced information still permits a continuous and smooth animation, as Autotutor has the ability to interpolate between facial configurations.

2.2 Data Collection

Facial expressions are universally used by people to enlarge the bandwidth of communication. They are largely unconscious and produced with specific but implicit purposes, which makes them a difficult topic to study. One approach for generating facial expressions might involve motion capture and analysis by expert tutors of facial expressions during tutoring sessions. The resulting data could be used for neural net training directly from video frames. Further considerations show this is not a good approach in the first place due to the richness of human facial expressions. This method would require a conversion from the data captured to the agent's kinematics.

More problematic, the tutors must be facially very expressive. It follows that the performer must be an effective actor in addition to an expert tutor.

We chose a different approach. Based on tutoring scripts from previous evaluations of AutoTutor. We designed and conducted a study where participants are shown a series of interactions between AutoTutor and a student. An *interaction* consists of a question posed by AutoTutor, the student's answer to that question, and AutoTutor's verbal evaluation and facial feedback to a tutee's response. Following the playback of an interaction, the participant designs an appropriate facial expression for AutoTutor's agent, which complements the verbal response. AutoTutor's response helped the study participants critique it and guide expert "drivers" towards producing a more desirable animation. Collecting data this way allows an expert (i.e., expert on designing AutoTutor's facial animations) to craft a fitting animation for a given situation. The expert was able to visualize the agent's expressions, and was able to hone the animation until the participant was satisfied.

We collected 337 animations from 38 students in the computer literacy class at the University of Memphis in Fall 2000. Each animation is determined by a sequence of key frames, where a key frame is a concrete description of the character's facial configuration. Each animation also has a real-valued rating assigned to it for emotional attitude, which corresponds to the *rating* that the participant assigned on the sliding scale. Further details about data collection can be found in [1].

3 Training a Neurocontrol for Facial Displays

The problem at hand is an inverse problem, akin to pole balancing or inverted pendulum [10]. In the forward problem, a given configuration of facial features of the talking head determines an emotional expression that evaluates to a *rating* for pedagogical feedback by AutoTutor (e.g., *confused*, *frustrated*, *negative neutral*, *neutral*, *positive neutral*, and *enthusiastic*) when viewed by a student. Our problem is precisely the inverse problem, in which a rating value is given as input. This is a cosine value coming from an evaluation of the student's contribution by AutoTutor's dialog module comparing against a semantic space built by Latent Semantic Analysis LSA, and corresponds to the kind of emotional response that AutoTutor should give students. The desired solution is to translate that value into a set of facial feature values that will evaluate into the desired rating by a human viewer.

3.1 Primary and Derived Features

Modules that provide partial solutions to the inverse problem for facial gestures were obtained by backpropagation training [7,10] based on input features and the rating of the desired emotional attitude. Feedforward nets with two hidden layers 24-10-10-1 were required for learning to converge. Features that cannot be derived from the rating and the remaining features are called *primary*, while the remaining are *derivable* features. The performance of the trained networks for the each feature can

be seen can be seen in Table 1. Rating and all features, except for the mouth position and time duration, of the frames were shown to be derivable individually.

Table 1. Neural net derivation of emotional attitude rating and facial features

Features	Training	Testing
Rating	89%	61%
Left Eye Brow (LEB)	89%	87.5%
Right Eye Brow (REB)	92%	90%
Left Eye Lid (LEL)	90%	88%
Right Eye Lid (REL)	62%	30%
Horizontal Eye Direction (HED)	68%	37%
Vertical Eye Direction (VED)	44%	37%
Mouth Position (MP)	47%	26%
Time Duration (TD)	48%	20%

Backpropagation was still able to produce networks that jointly derive several features at a time, as seen in Tables 2 and 3. The best module was able to jointly generate four features out of a set of input features consisting of the rating and the remaining facial features, as seen in Table 4.

Table 2. Joint derivation of 2 features at a time, including Left Eye Lid

Additional Feature	Training	Testing
Right Eye Lid (REL)	85%	84%
Vertical Eye Direction (VED)	86%	84%
Horizontal Eye Direction (HED)	79%	77%
Left Eye Brow (LEB)	81%	79%
Right Eye Brow (REB)	81%	78%

Table 3. Joint derivation of 3 features at a time, including Left and Right Eye Lid

Additional Feature	Training	Testing
Horizontal Eye Direction (LED)	79%	79%
Vertical Eye Direction (VED)	84%	83%
Left Eye Brow (LEB)	82%	81%
Right Eye Brow (REB)	82%	78%

Table 4. Joint derivation of four features at a time, including Left and Right Eye Lid, and Vertical Eye Direction

Additional Feature	Training	Testing
Horizontal Eye Direction (HED)	80%	78%
Left Eye Brow (LEB)	81%	80%
Right Eye Brow (REB)	82%	80%

Backpropagation failed to converge in any sort of combination to derive more features. A recurrent network was required. However, attempts to train a network using the data and backpropagation through time was never successful for a large set of facial gestures. A new technique was required.

3.2 Cascade Inversion

In order to complete the solution of the inverse problem, we utilized the modules derived using feedforward networks to derive the remaining primary features through recurrent neural network. We applied a novel method of *cascade inversion*. The central idea can be described as follows. Suppose a feature Y can be derived by network NY using rating and some set of features S which includes a feature X; and vice versa, i.e. suppose that feature X can be derived by network NX from inputs S and feature Y as input. Cascade inversion produces a recurrent network NXY that derives both features X and Y from the rating and input set S alone. Fig. 1 below shows a general cascade inversion training, which was successfully used to obtain a neurocontrol for facial expressions and for hand gestures. Further details about the foundations and applications of this learning algorithm will be given elsewhere.

3.3 Training a Recurrent Network

With some features now jointly derived from only the rating and other facial features, we now obtain the solution for our inverse problem, where given rating and a minimal set of facial features as input, the network produces the remaining features to configure a facial animation that evaluates to the given rating. The architecture was generally a 24-15-10-*n* for first implementation for the recurrent network, where *n* is the number of derived features. Table 5 and Table 6 show the performance of cascade inversion learning.

Table 5. Neural net learning with S={Rating, Mouth Position, Duration}

Cascade Inversion (given 2+2=4 derived features as inputs)	Training	Testing
X=Left Eye Lid & Right Eye Lid	79%	79%
Y=Horizontal & Vertical Eye Direction	90%	90%
Network NXY1	88%	84%
X=Left Eye Lid & Right Eye Lid	98%	90%
Y=Left Eye Brow & Right Eye Brow	24%	16%
Network NXY2 (<i>not used</i>)	61%	53%
X=Left Eye Brow & Right Eye Brow	13%	12%
Y=Horizontal & Vertical Eye Direction	98%	92%
Network NXY3 (<i>not used</i>)	55.5%	52%

Table 6. Neural net learning with S={Rating, Mouth Position, Duration}

Cascade Inversion (given 3+3=6 derived features as inputs)	Training	Testing
X=Left Eye Lid + Right Eye Lid + Left Eye Brow (NX1)	68%	60%
Y=Horizontal + Vertical Eye Direction + Right Eye Brow	56%	84%
Network NXY4=Inverse Problem Solution	62%	72%

Finally, Fig. 2 shows the architecture of the final neurocontrol, which successfully produces the facial expressions from the primary features for facial displays after

single, joint, and cascade inversion learning, namely **rating**, **mouth position**, and **time duration** for each of the three frames.

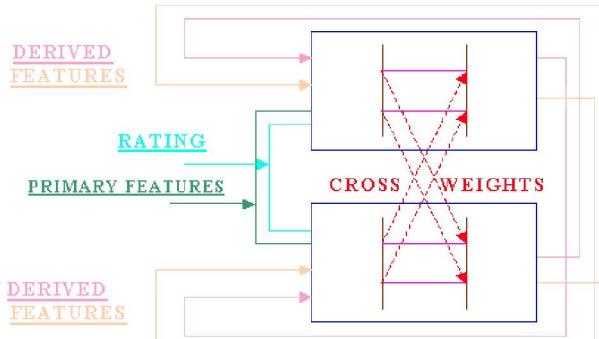


Fig. 2. Cascade Inversion of two neural nets NX (top) deriving and X and NY (bottom) deriving Y, both using a set of features S. A recurrent network NXY is obtained that derives both features X and Y from feature set S alone.

4 Discussion and Conclusion

A neurocontrol for facial expressions was successfully trained using a combination of backpropagation training, backpropagation through time, and cascade inversion, a novel training algorithm to solve inverse problems. Feedforward backpropagation and backpropagation through time, with only primary features (rating, mouth positions, and frame durations) as input, were unable to learn to produce appropriate set of remaining facial features that performed well for the given data. Cascade inversion proved to be an efficient method to achieve our desired goal of solving the inverse problem of a neurocontrol for facial expressions with the given set of features.

We have conducted a preliminary informal evaluation of the quality of facial expressions autonomously generated by the recurrent neurocontrol produced by this method. Except for a bias in the gaze toward the right side of the tutor (which may have been induced by a bias in the data due to the speakers' position relative to the audience), the talking head appears naturalistic enough to be useful, at least for pedagogical purposes. A formal comprehensive evaluation of the overall quality of the facial expressions is required to establish the communication quality of the emotional displays produced by this recurrent network in the context of tutoring sessions and more general computer to human communication.

The results in this paper may have other applications, particularly in related work in facial recognition. The dependencies among facial features place a heavy weight in the information content of the primary facial features for facial displays, namely the dynamics of the mouth (here represented by the mouth positions in the successive frames and duration of each animation) and the emotional state of the individual. The results may have also other implications in the current debate on the nature of

biological networks that explain the amazing ability of human to handle nearly instant recognition of emotional states and identity from snapshots of human faces since very early stages of development, even perhaps prenatally.

Acknowledgments. We are thankful to Evan Drumwright, for organizing and implementing the data collection study with computer literacy students, as well as several students (too many to mention) for manually and painstakingly collecting data as drivers from the participants. We also thank all computer literacy and computer science students who helped provide critical training data required for this. This work has been partially supported by grants from The National Science Foundation NSF/KDI-9720314 and NSF/ROLE-0106965. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

References

1. M.H. Garzon, E. Drumwright, K. Rajaya: Training a Neurocontrol for Talking Heads. Proc. of the IEEE International Joint Conference on Neural Networks (IJCNN-02) Hawaii, World Congress in Computational Intelligence (2002), Computer Society Press, 2449-2453
2. M.H. Garzon, P. Ankaraju, E. Drumwright, R. Kozma: Neurofuzzy Recognition and Generation of Facial Features in Talking Heads. Proc. of NEUROFUZZY-2002, Hawaii, World Congress in Computational Intelligence (2002), 926-931
3. J. Cassell, J. Sullivan, S. Provost, E. Churchill (eds.): Embodied Conversational Agents. The MIT Press (2000)
4. J. Cassell, H. Wilhjalmsson, T. Bickmore (2001) BEAT: the Behavior Expression Animation Toolkit. Proc. SIGGRAPH-2001
5. P. Ekman (1992). An Argument for Basic Emotions. In: N.L. Stein and K. Oatley (eds.) *Basic Emotions*, 169-200
6. M.H. Garzon and The Tutoring Research Group: On Interactive Computation: Intelligent Tutoring Systems. Proc. Theory and Practice of Informatics SOFSEM-1999 (G. Pavelka, G. Tel, M. Bartosek, eds.). Springer-Verlag Lecture Notes in Computer Science **1725**, 261-264
7. B. Hammer, Learning with Recurrent Neural Networks, Springer Verlag Lecture Notes in Control and Information Sciences **254**, 2000
8. J. Lester and B. Stone. Increasing believability in animated pedagogical agents. In Proc. of the First International Conference on Autonomous Agents. W. Lewis Johnson (ed.). ACM Press, February 1997
9. D.W. Massaro, M.M. Cohen, J. Beskow, R.A. Cole (2000). Developing and Evaluating Conversational Agents. In [3], 287-318
10. D.M. Skapura: Building Neural Networks. Addison-Wesley (1994)

Author Index

- Abbott, Derek II-73
Abraham, Ajith I-206, II-512, II-774
Acha, José I. II-273
Affenzeller, Michael I-438
Agís, Rodrigo II-145
Aguirre, Carlos I-78
Albouy, Benoit II-289
Aler, Ricardo II-217
Aleksander, Igor I-86, I-630
Ali, Shawkat I-206
Alique, A. II-758
Alique, J.R. II-758
Al-Khalifa, S. II-798
Allotta, B. II-497
Allende, Héctor I-238, II-441
Alonso-Betanzos, Amparo I-270, II-489
Alvado, Ludovic I-670
Álvarez, Manuel R. ... II-233, II-695
Álvarez Sánchez, José Ramón II-161
Álvarez-Vellisco, Antonio II-249, II-583
Andina, Diego II-249, II-361, II-583
Angulo, Cecilio I-646
Antoranz, J.C. I-366
Aranda Almansa, J. II-369
Arazoza, H. de II-473
Arcay, Bernardino II-639
Atencia, Miguel I-190, I-350, II-449
Aunet, Snorre II-57
Austin, Jim II-663
AuYeung, Andy II-774
Bachiller, M. I-574
Bahamonde, Antonio I-246
Baldassarri, Paola II-201
Baldwin, Jim F. I-286
Banquet, Jean-Paul I-198
Barakova, Emilia I-102, I-110
Barro, Senén I-70
Becerra, J.A. II-169
Beiu, Valeriu II-49
Beligiannis, Grigorios II-409
Bellas, F. I-590
Benatchba, Karima II-393
Benbouzid, Fatima II-782
Benítez-Rochel, R. I-294, I-342
Berg, Yngvar II-57
Bermejo, Sergio II-297, II-711, II-719
Bluff, K. I-614
Bologna, Guido II-544
Bonomini, M.P. II-81
Botella, Guillermo I-710
Botía, Juan A. II-401, II-703
Brégains, Julio II-750
Buldain, David I-334
Burattini, Ernesto II-9
Cabestany, Joan II-719
Calderón-Martínez, José A. ... II-528
Cal Marín, Enrique A., de la II-353
Camacho, Eduardo F. II-337
Campos, Doris I-78
Campoy-Cervera, Pascual ... II-528
Cañas, Antonio II-1, II-89
Canedo, Antonio I-70
Carrillo, Richard R. II-1, II-145
Casañ, Gustavo A. II-766
Castaño, M. Asunción II-766
Castellanos, Juan I-152
Castellanos, Nazareth P. I-32
Castellanos-Moreno, A. I-542
Castillo, Carmen II-489
Castillo, Enrique II-489
Castillo, Jose L. I-366
Castillo Valdivieso, Pedro A. I-358, I-502, I-534, II-655

- Castro, María José I-598
 Català, Andreu I-646
 Celinski, Peter II-73
 Cerdá, Joaquín II-121
 Charbonnier, Sylvie II-599
 Chebira, Abdennasser I-382, I-558, II-647
 Chehbour, Fairouz II-385
 Chillemi, Santi I-24
 Chinellato, Eris II-193
 Cichocki, A. I-158
 Colla, V. II-497
 Combarro, Elías F. I-246, I-326, II-742
 Contreras A., Ricardo II-726
 Corchado, Emilio I-326
 Cortes, P. II-313
 Costa Monteiro, E. II-607
 Cotofana, Sorin D. II-65, II-73
 Cotta, Carlos I-494, II-321
 Cruces, Sergio II-305
 Dafonte, Carlos II-639
 Dalmau, Javier II-711
 Damas, Miguel II-425
 d'Anjou, Alicia II-567
 Dapena, Adriana II-257
 Delgado, Ana E. I-694
 del-Hoyo, Rafael I-334
 Del Moral Hernandez, Emílio I-166
 Deville, Yannick II-241, II-289
 Díaz, Antonio F. II-89
 Díaz, I. I-230, I-262
 Díaz, Javier I-710
 Díaz-Madrid, José Ángel II-25, II-97
 Diego, Javier de II-504
 Doménech-Asensi, Ginés II-25
 Dominguez, David R.C. I-462
 Domínguez-Merino, Enrique I-190, I-406, II-734
 Dorado, Julián I-606, II-750
 Dorronsoro, José R. I-174, II-520
 Dours, Daniel II-393
 Draghici, Sorin II-623
 Drias, Habiba I-414, I-446
 Dunmall, Barry I-630
 Durán, Iván II-305
 Duro, R.J. I-590, II-169
 Eriksson, Jan II-113
 Escudero, Carlos II-257
 Esposito, G. I-1
 Faúndez-Zanuy, Marcos II-671
 Feng, Jianfeng I-62
 Fernández, E. II-81
 Fernández, Fernando I-254, II-217
 Fernández, Javier I-262, II-742
 Fernández, Miguel A. I-694
 Fernandez Lopez, P. I-54
 Fernández-Caballero, Antonio I-694
 Fernández-Redondo, Mercedes I-622, II-129, II-137
 Ferreira, Victor M.G. II-9
 Ferrández, J.M. I-678, II-33, II-81
 Figueiredo, K. I-126
 Fischer, Joern I-302
 Florez-Giraldo, Oscar A. II-185
 Fontenla-Romero, Oscar I-270
 França, Felipe M.G. II-9
 Franco, Leonardo I-734
 François, D. II-105
 Fyfe, Colin I-326
 Gadea, Rafael II-121
 Gail, Annette I-46
 Galindo, Pedro I-374
 Gallardo-Caballero, Ramón II-551
 Gallego, Josune II-567
 Galleske, Ingo I-152
 Galván, I.M. I-278
 Garbo, Angelo Di I-24
 Garcia Arenas, Maribel I-358, I-502, I-534
 García, Beatriz I-478
 García, Jesús II-504
 Garcia Baez, P. I-54
 García Chamizo, Juan Manuel II-591
 García-Bernal, M.A. I-398, I-430
 Garcia-Cerezo, A. II-481

- García-de-Quirós, F.J. II-81
 García-Lagos, F. I-486
 García-Orellana, Carlos J. II-551
 García-Pedrajas, N. I-518
 Garzon, Max H. I-750
 Gasca, Rafael M. II-337
 Gaussier, Philippe I-198
 Gestal, Marcos II-750
 Gheorghe, Marian I-638
 Giannakopoulos, Fotios I-46
 Gil-Jiménez, P. I-718, II-536
 Gil-Pita, R. I-718, II-559, II-790
 Giorno, V. I-1
 Gomez, P. I-678
 Gómez Pulido, Juan A. II-465
 Gómez-Moreno, H. I-718, II-536
 Gómez-Ruiz, José Antonio
 I-318, I-398, I-430
 Gómez-Skarmeta, Antonio F.
 II-401, II-703
 González, Ana I-174
 González, C. I-510
 González, Jesús I-454, I-550
 González Rodríguez, Inés I-286
 González-Careaga, Rafaela .. II-177
 González-Velasco, Horacio ... II-551
 Gonzalo, Isabel I-94
 Górriz, J.M. II-433
 Graña, Manuel II-567
 Gregorio, Massimo De II-9
 Grimaldo, Francisco II-209
 Guerrero, Elisa I-374
 Guerrero-González, Antonio
 II-185
 Guijarro-Berdiñas, Bertha I-270
 Gutiérrez, Germán I-478
 Haber, R.E. II-758
 Hall Barbosa, C. II-607
 Hamami, Latifa II-385
 Hauer, Hans II-25, II-97
 Hauptmann, Christian I-46
 Hernández-Espinosa, Carlos
 I-622, II-129, II-137
 Hernansaez, Juan M. II-703
 Herreras, O. I-9, I-40
 Hervás-Martínez, C. I-518
 Hidalgo-Conde, Manuel II-377
 Hild, Manfred I-144
 Hodge, Vicky II-663
 Hornillo-Mellado, Susana II-273
 Hosseini, Shahram ... II-241, II-599
 Hügli, Heinz I-702
 Ibarz, J.M. I-9, I-40
 Ibri, Sarah I-414
 Isasi, P. I-254, I-278
 Jain, Ravi II-512
 Jarabo-Amores, P. II-559, II-790
 Jerez, Jose M. I-190, I-734
 Jesús, M.J. del I-470
 Joya, Gonzalo .. I-350, I-486, II-449
 Jutten, Christian II-225, II-599
 Kelly, P.M. II-41
 Köppen, Mario I-582
 Koroutchev, Kostadin II-520
 Koudil, Mouloud II-393
 Kwok, Terence I-390
 Ladrón de Guevara-López, I.
 I-398, I-430
 Lang, Elmar W.
 II-265, II-281,
 II-575, II-687, II-695
 Larrañaga, P. I-510
 Larrañeta, J. II-313
 Lassouaoui, Nadia II-385
 Lawry, Jonathan I-286
 Ledezma, Agapito II-217
 Lees, Ken II-663
 Lendasse, A. I-182
 Likothanassis, Spiridon II-409
 Lope, Javier de
 I-214, II-153, II-177
 López Alcantud,
 José-Alejandro II-97
 López, A. I-526
 López, María T. I-694
 López-Aguado, L. I-40
 Lopez-Baldan, M.J. II-481
 López-Ferreras, F.

- López-Rubio, Ezequiel I-318
 Lourens, Tino I-102, II-110
 Lozano, J.A. I-510
 Lozano, Miguel II-209
 Lucas, Simon M. I-222
- Maciá Pérez, Francisco II-591
 Macías-Macías, Miguel II-551
 Madani, Kurosh I-382, I-558, II-647
 Maguire, L.P. II-41
 Makarova, I. I-40
 Maldonado-Bascón, S. I-718, II-536, II-798
 Malvezzi, M. II-497
 Manteiga, Minia II-639
 Maravall, Darío I-214, II-153, II-177
 Marco, Alvaro I-334
 Marín, Francisco J. I-486, II-377
 Mariño, Jorge I-70
 Marrero, A. II-473
 Martinez, J.J. II-33
 Martínez, R. I-574
 Martín-Clemente, Rubén II-273
 Martín-Vide, Carlos I-638
 McGinnity, T.M. II-41
 Medina, Jesús I-654
 Medrano-Marqués, Nicolás .. II-806
 Merelo Guervós, Juan J. I-358, I-502, I-534, II-655
 Mérida-Casermeiro, Enrique I-294, I-342, I-406, I-654, II-734
 Mira, José I-16, I-574, II-161
 Mitrana, Victor I-638
 Moga, Sorin I-198
 Molina, Ignacio I-734
 Molina, José M. I-478, II-504
 Molina-Vilaplana, J. II-185
 Monge-Sanz, Beatriz II-806
 Montañés, Elena I-230, II-742
 Montesanto, Anna II-201
 Moraga, Claudio I-238, II-441
 Morales, Antonio II-193
 Morató, Carmen II-361
- Moreno, Juan Manuel II-113
 Morton, Helen I-86
 Mota, Sonia I-710, II-345
 Mourelle, Luiza de Macedo ... II-17
 Muñoz-Pérez, José I-294, I-318, I-342, I-398, I-406, I-430, II-734
 Muzik, Otto II-623
- Nakadai, Kazuhiro I-118
 Nedjah, Nadia II-17
 Nishimoto, Ryunosuke I-422
 Núñez, Haydemar I-646
- O'Keefe, Simon II-663
 Ojeda-Aciego, Manuel I-654
 Okuno, Hiroshi G. I-118
 Onieva, L. II-313
 Orozco García, José II-615
 Ortega, Julio I-454, I-470, I-550, II-89, II-345, II-425, II-433, II-679
 Ortigosa, Eva M. II-1, II-89, II-145
 Ortiz-Boyer, D. I-518
 Ortiz-Gómez, Mamen I-622, II-129, II-137
 Ouerhani, Nabil I-702
- Pacheco, M. I-126
 Padure, Marius II-65
 Panarese, Alessandro I-24
 Parrilla Sánchez, M. II-369
 Pascual, Pedro I-78
 Pasemann, Frank I-144
- Patricio, Miguel Ángel I-214, II-153
 Paya, Guillermo II-121
 Paz Lopez, Félix, de la II-161
 Pazos, Alejandro I-606
 Pazos, Juan I-662
 Peña-Reyes, Carlos-Andrés I-136, I-726
 Pedreño-Molina, Juan L. II-185
 Pedroso-Rodriguez, L.M. II-473
 Pellegrini, Christian II-544
 Pérez, Olga M. II-377

- Pérez Jiménez, Mario J. I-638
 Pham, Dinh-Tuan II-225
 Pinninghoff, M. Angélica II-726
 Pizarro, Joaquín I-374
 Pobil, Ángel P. del II-193
 Pomares, Héctor I-454, I-550
 Porras, Miguel A. I-94
 Portillo, Javier I. II-504
 Pourabdollah, Siamak II-623
 Prat, Federico I-598
 Prieto, Carlos II-329
 Prieto, Luis I-726
 Puente, Jorge II-329
 Puliti, Paolo II-201
 Puntonet, Carlos G.
 II-233, II-265, II-273, II-425,
 II-433, II-679, II-687, II-695
 Quevedo, José Ramón . I-230, I-246
 Rabuñal, Juan R. I-606
 Rajaya, Kiran I-750
 Ranilla, José I-262, II-742
 Renaud, Sylvie I-670
 Revilla, Ferran II-719
 Reyes García, Carlos A. II-615
 Rincón, M. I-574
 Rivera, A.J. I-470
 Rivero, Daniel I-606
 Ricciardi, L.M. I-1
 Rodellar, V. I-678
 Rodríguez, Alejandra II-639
 Rodríguez, Francisco B. I-32
 Rodríguez, Juan Antonio II-750
 Rodríguez, J.D. I-510
 Rodríguez-Álvarez, Manuel .. II-281
 Rodríguez-Patón, Alfonso I-662
 Rodríguez-Pérez, Daniel I-366
 Rojas, Fernando
 II-233, II-281, II-679
 Rojas, Ignacio
 I-454, I-470, I-550,
 II-233, II-281, II-679
 Romero López, G.
 I-358, I-502, I-534
 Ros, Eduardo
 I-710, II-1, II-145, II-345
 Rosa-Zurera, M. II-559, II-790
 Ruano, António E.B. II-457
 Rybnik, Mariusz
 I-382, I-558, II-647
 Ruiz del Solar, Javier I-742
 Ruiz Fernández, Daniel II-591
 Ruiz Merino, Ramón II-97
 Ruiz-Gomez, J. II-481
 Ruiz-Merino, Ramón II-25
 Rynkiewicz, Joseph I-310
 Sacristán, M.A. I-678
 Saighi, Sylvain I-670
 Salas, Rodrigo I-238, II-441
 Salcedo Lagos, Pedro II-726
 Salmerón, Moisés II-425, II-433
 Sánchez, Eduardo
 I-70, I-136, I-726
 Sánchez, L. I-526
 Sanchez Martin, G. I-54
 Sánchez Pérez, Juan M. II-465
 Sánchez Ramos, Luciano II-353
 Sánchez-Andrés, Juan Vicente
 I-566
 Sánchez-Marcano, Noelia II-489
 Sanchis, Araceli I-478
 Sandoval, Francisco
 I-350, I-486, II-449
 Santos, J. II-169
 Sanz Valero, Pedro II-193
 Sanz-Tapia, E. I-518
 Sarro, Luis M. II-631
 Satpathy, H.P. II-417
 Seijas, Juan II-361
 Serrano, Eduardo I-78, I-174
 Silva, Andrés I-662
 Silva Oliveira, Clayton I-166
 Simon, G. I-182, II-105
 Skarlas, Lambros II-409
 Smith, Kate A. I-390
 Sole-Casals, Jordi II-225
 Soria-Frisch, Aureli I-582
 Soriano Payá, Antonio II-591
 Stadlthanner, K. II-575
 Suárez Araujo, C.P. I-54
 Suárez-Romero, Juan A. I-270

- Taibi, Amine I-446
Tani, Jun I-422
Tascini, Guido II-201
Tatapudi, Suryanarayana II-49
Theis, Fabian J.
..... II-265, II-575, II-687, II-695
Thompson, C.J. II-41
Toledo, F.J. II-33
Tomé, A.M. II-575
Tomas, Jean I-670
Toro, Francisco de II-345
Toro, Miguel II-337
Torrealdea, F. Javier II-567
Torres, Oriol II-113
Torres, Romina I-238, II-441
Torres-Alegre, Santiago II-249
Trelles, Oswaldo II-377
Troć, Maciej I-686
Tsujino, Hiroshi I-102
Unold, Olgierd I-686
Upogui, Andres I-136
Valdés, Mercedes II-401
Valerio, C. I-1
Valle, Carmelo Del II-337
Valls, J.M. I-278
Vannucci, M. II-497
Varela, Ramiro II-329
Varnier, Christophe II-782
Varona, Pablo I-32
Vassiliadis, Stamatis II-65
Vega Rodríguez, Miguel A. II-465
Vega-Corona, Antonio
..... II-249, II-361, II-583
Vela, Camino R. II-329
Vellasco, M. I-126, II-607
Verleysen, M. I-182, II-105
Verschae, Rodrigo I-742
Vicen-Bueno, R. II-559
Vico, Francisco J. I-190
Vieira, Armando II-655
Villa, Alessandro II-113
Villa, Rosa I-726
Villaplana, Javier II-209
Wagner, Stefan I-438
Wallace, J.G. I-614
Weeks, Michael II-663
Wertz, V. II-105
Wolf, A. II-607
Yamaguchi, Yoko I-110
Yáñez, Andrés I-374
Zahedi, Keyan I-144
Zarraonandia, Telmo II-177
Zekour, Sofiane I-446
Zerhouni, Nourredine II-782
Zhang, L.-Q. I-158