ANNIIP 2008

Kurosh Madani (Ed.)

Artificial Neural Networks and Intelligent Information Processing

Proceedings of the
4th International Workshop on
Artificial Neural Networks and Intelligent Information Processing - ANNIIP 2008

In conjunction with ICINCO 2008
Funchal, Madeira - Portugal, May 2008

INSTICC PRESS

Kurosh Madani (Ed.)

Artificial Neural Networks and Intelligent Information Processing

Proceedings of the
4th International Workshop on
Artificial Neural Networks and
Intelligent Information Processing
ANNIIP 2008

In conjunction with ICINCO 2008
Madeira, Portugal, May 2008

INSTICC PRESS
Portugal

Volume Editor

Kurosh Madani
The University of PARIS XII,
France

4th International Workshop on
Artificial Neural Networks and
Intelligent Information Processing
Madeira, Portugal, May 2008
Kurosh Madani (Ed.)

Printed in Portugal

ISBN: 978-989-8111-33-3
Depósito Legal: 273831/08

Foreword

The nowadays' applicative and technological challenges emanating from industrial, socioeconomic or environment needs open new dilemmas and decisively highlight limitations of conventional computational science. Recent borders' contraction between biological and computational sciences, especially the latest developments in bio-inspired artificial systems over the last decade, may play a central role in designing adequate solutions to these new challenging dilemmas. The fantastic ever-increasing intellectual dynamics created around bio-inspired Artificial Intelligence and related topics (as Artificial Neural Networks, Humanoid Robotics, Ambient Intelligence, etc…), uphold by escalating interest of both confirmed and young researchers on this relatively juvenile science, generates a reach multidisciplinary synergy between a large number of scientific communities making conceivable a forthcoming emergence of viable solutions to real-world dilemmas.

Since 2005, the international workshop on *Artificial Neural Networks and Intelligent Information Processing (ANNIIP)*, within the frame of the prestigious ICINCO International Conference, takes part in the aforementioned appealing dynamics by offering a privileged space to overhaul and exchange the knowledge about further theoretical advances, new experimental discoveries and novel technological improvements in the promising area of the bio-inspired Artificial Intelligence. The present book is the outcome of the fourth edition of this annual event.

Around a deliberately limited number of papers, the objective of this book is to convene a set of relevant recent works focusing bio-inspired Artificial Intelligence related fields and applications. Conformably to our philosophy, the choice of publishing a relatively restricted number of papers has been motivated on the one hand by the premeditated desire to give a large space to exchanges and discussions during the ANNIIP workshop, and on the other hand by the strong principle of the presentation of each accepted article by its authors. If "Bio-inspired Artificial Intelligence" and its real-world applications remain, as in the previous editions of this international workshop, the foremost themes of this 4th ANNIIP edition, a particular interest has been devoted to the equilibrium between theoretical and applicative aspects.

It is important to remind that scientific relevance and technical quality of a collective volume emerge from quality of its contributors: those who contribute by the high quality of their manuscripts and those who take part in reviewing of submitted papers ensuring the distinction of the book by their valuable expertise. I would like to express again my acknowledgements to contributors of all accepted papers: *You are the central reason of the nobles of this tome.*

Also, I would like to reedit my gratitude to Reviewing Board and Program Committee for the valuable work that they accomplished: my heartfelt recognition to those who already were members of ANNIIP Program Committee as well as my sincere thanks to those who kindly accepted to enlarge the Reviewing Board of this new edition of the workshop.

It is also essential to be reminiscent that frequently, creative dynamics is the result of fruitful human contacts within a same scientific field or the consequence of interaction of humans from different scientific communities and since 2004, the date of the its first edition, the ICINCO multi-conference has been an outstanding bench of such creative synergies. For that, again, I would like to express my warm appreciation and my particular gratitude to my friend Prof. Joaquim Filipe, ICINCO 2008 Conference's Chair, for his faith in young science of "Bio-inspired Artificial Intelligence" and for his reliance on devoting once more this privileged space to the ANNIIP workshop within his valuable conference.

Finally, if ICINCO Organizing Committee's professionalism became an obvious skill of the organization of this international event in so accurate way, it should never be forgotten that the organization of a prestigious conference remains a challenging undertaking requiring a reliable and a solid team. I would like acknowledge all of the ICINCO Organizing Committee's members, with a special word for Marina Carvalho from ICINCO Secretariat who, since 2005, is a key person in ANNIIP workshop's organization.

Kurosh Madani
PARIS XII – Val de Marne University
France

Workshop Chair
Kurosh Madani
The University of PARIS XII
France

Program Committee

Veronique Amarger, University of Paris XII - Val de Marne, France

Ezzedine Ben Braiek, ESSTT, University of Tunis, Tunisia

Abdennasser Chebira, University of Paris XII - Val de Marne, France

Amine Chohra, University of Paris XII - Val de Marne, France

Suash Deb, National Inst. of Science & Technology, India

Peter Geczy, National Institute of Advances Industrial Science & Technology, Japan

Vladimir Golovko, Brest Technical University, Belarus

Robert E. Hiromoto, University of Idaho, USA

Jouko Lampinen, Helsinki University of Technology, Finland

Thomas Längle, University of Karlsruhe, Germany

Thorsteinn Rögnvaldsson, Örebro University, Sweden

Dattatraya Kodavade, D.K.T.E.S.'s Textile & Engineering Institute, Ichalkaranji, India

Hichem Maaref, University of Evry - Val d'Essonne, France

Kouider N. M'Sirdi, Paul Césanne University - Marseille, France

Georgy Penev, St-Petersburg Institute Informatics of Russian Academy of Sciences (SPIIRAS), Russia

Anatoly Sachenko, Institute of Intelligent Computer Systems, Ukraine

Rauf Sadikhov, Belarusian State University of Informatics and Radio-electronics, Belarus

Christophe Sabourin, University of Paris XII - Val de Marne, France

Khalid Saeed, Bialystok Technical University, Poland

Table of Contents

Papers

PAPERS

Combining Selective-Presentation and Selective-Learning-Rate Approaches for Neural Network Forecasting of Stock Markets

Kazuhiro Kohara[1]

[1]Department of Electrical, Electronics and Computer Engineering
Chiba Institute of Technology, 2-17-1, Tsudanuma, Narashino, Chiba, 275-0016, Japan
kohara.kazuhiro@it-chiba.ac.jp

Abstract. We have investigated selective learning techniques for improving the ability of back-propagation neural networks to predict large changes. We previously proposed the *selective-presentation* approach, in which the training data corresponding to large changes in the prediction-target time series are presented more often, and *selective-learning-rate* approach, in which the learning rate for training data corresponding to small changes is reduced. This paper proposes combining these two approaches to achieve fine-tuned and step-by-step selective learning of neural networks according to the degree of change. Daily stock prices are predicted as a noisy real-world problem. Combining these two approaches further improved the performance.

1 Introduction

Prediction using back-propagation neural networks has been extensively investigated (e.g., [1-5]), and various attempts have been made to apply neural networks to financial market prediction (e.g., [6-16]), electricity load forecasting (e.g., [17]) and other areas. In the usual approach, all training data are equally presented to a neural network (i.e., presented in each cycle) and the learning rates are equal for all the training data independently of the size of the changes in the prediction-target time series. Also, network learning is usually stopped at the point of minimal mean squared error between the network's outputs and the desired outputs.

Generally, the ability to predict large changes is more important than the ability to predict small changes, as we mentioned in the previous paper [12]. When all training data are presented equally with an equal learning rate, the BPNN will learn the small and large changes equally well, so it cannot learn the large changes more effectively. We have investigated selective learning techniques for improving the ability of neural networks to predict large changes. We previously proposed the *selective-presentation* and *selective-learning-rate* approaches and applied them into stock market prediction [12-14]. In the *selective-presentation* approach, the training data corresponding to large changes in the prediction-target time series are presented more often. In the *selective-learning-rate* approach, the learning rate for training data corresponding to small changes is reduced. The previous paper [12] also investigated another stopping criterion for financial predictions. Network learning is stopped at the point having the

maximum profit through experimental stock-trading.

This paper proposes combining the *selective-presentation* and *selective-learning-rate* approaches. By combining these two approaches, we can easily achieve fine-tuned and step-by-step selective learning of neural networks according to the degree of change. Daily stock prices are predicted as a noisy real-world problem.

2 Combining Selective-Presentation and Selective-Learning-Rate Approaches

To allow neural networks to learn about large changes in prediction-target time series more effectively, we separate the training data into large-change data (L-data) and small-change data (S-data). L-data (S-data) have next-day changes that are larger (smaller) than a preset value.

In the *selective-presentation* approach, the L-data are presented to neural networks more often than S-data. For example, all training data are presented every fifth learning cycle, while the L-data are presented every cycle. In the *selective-learning-rate* approach, all training data are presented in every cycle; however, the learning rate of the back-propagation training algorithm for S-data is reduced compared with that for L-data. These two approaches are outlined as follows.

Selective-Presentation **Approach**

1. Separate the training data into L-data and S-data.
2. Train back-propagation networks with more presentations of L-data than of S-data.
3. Stop network learning at the point satisfying a certain stopping criterion (e.g., stop at the point having the maximum profit).

Selective-Learning-Rate **Approach**

1. Separate the training data into L-data and S-data.
2. Train back-propagation networks with a lower learning rate for the S-data than for the L-data.
3. Stop network learning at the point satisfying a certain stopping criterion (e.g., stop at the point having the maximum profit).

We combine these two approaches to achieve fine-tuned and step-by-step learning of neural networks according to the degree of change. The outline is as follows.

Combining *Selective-Presentation* and *Selective-Learning-Rate* Approaches

1. Separate the training data into L-data and S-data.
2. Separate L-data into two subsets: L1-data and L2-data, where changes in L2- data are larger than those in L1-data.
3. Separate S-data into two subsets: S1-data and S2-data, where changes in S2-data are larger than those in S1-data.
4. Train back-propagation networks with more presentations of L1- and L2-data than of S1- and S2-data, and with a lower learning rate for L1- and S1-data than for L2 and S2-data.
5. Stop network learning at the point satisfying a certain stopping criterion (e.g., stop

at the point having the maximum profit).

In general, we can separate the training data into N subsets ($N \geq 2$): D_1-, D_2-, …, and D_N-data, where changes in D_i-data are larger than those in D_{i-1}-data, and give "*selective intensities*" I (number of presentations times learning rate) to D_1-, D_2-, …, and D_N-data as $I_1 < I_2 < I_3 < \ldots < I_N$.

3 Evaluation through Experimental Stock-Price Prediction

We considered the following types of knowledge for predicting Tokyo stock prices. These types of knowledge involve numerical economic indicators [12-14].

1. If interest rates decrease, stock prices tend to increase, and vice versa.
2. If the dollar-to-yen exchange rate decreases, stock prices tend to decrease, and vice versa.
3. If the price of crude oil increases, stock prices tend to decrease, and vice versa.

We used the following five indicators as inputs to the neural network.

- TOPIX: the chief Tokyo stock exchange price index
- EXCHANGE: the dollar-to-yen exchange rate (yen/dollar)
- INTEREST: an interest rate (3-month CD, new issue, offered rates) (%)
- OIL: the price of crude oil (dollars/barrel)
- NY: New York Dow-Jones average of the closing prices of 30 industrial stocks (dollars)

TOPIX was the prediction target. EXCHANGE, INTEREST and OIL were chosen based on the knowledge of numerical economic indicators. The Dow-Jones average was used because Tokyo stock market prices are often influenced by New York exchange prices. We assume that tomorrow's change in TOPIX is determined by today's changes in the five indicators according to the knowledge. Therefore, the daily changes in these five indicators (e.g. Δ TOPIX(t) = TOPIX(t) - TOPIX(t-1)) were input into neural networks, and the next-day's change in TOPIX was presented to the neural network as the desired output (Figure 1). The back-propagation algorithm was used to train the network. All the data of the daily changes were scaled to the interval [0.1, 0.9]. A 5-5-1 multi-layered neural network was used (five neurons in the input layer, five in the hidden layer, and one in the output layer).

3.1 Experiments

We used data from a total of 409 days (from August 1, 1989 to March 31, 1991): 300 days for training, 30 days for validation (making decisions on stopping the network learning), and 79 days for making predictions. In Experiment 1, all training data were presented in each cycle with an equal learning rate ($\varepsilon = 0.7$). In Experiment 2, L-data were presented five times as often as S-data. Here, the large-change threshold was 14.78 points (about US$ 1.40), which was the median of absolute value of TOPIX

daily changes in the training data. In Experiment 3, the learning rate for the S-data was reduced up to 20% (i.e., $\varepsilon = 0.7$ for the L-data and $\varepsilon = 0.14$ for the S-data). Experimental conditions in Experiments 1, 2, and 3 are shown in Table 1, 2, and 3.

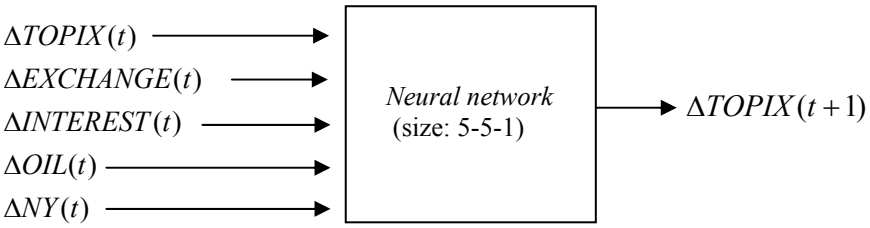


Fig. 1. Neural prediction model.

Table 1. Experimental conditions in Experiment 1: conventional technique.

	S-data	L-data
Range of absolute value of ΔTOPIX(t)	0 to 50%	50 to 100%
Number of data	150	150
Relative number of presentations (P)	1	1
Learning rate (ε)	0.7	0.7
P times ε (relative value)	0.7 (1)	0.7 (1)

Table 2. Experimental conditions in Experiment 2: selective presentation.

	S-data	L-data
Range of absolute value of ΔTOPIX(t)	0 to 50%	50 to 100%
Number of data	150	150
Relative number of presentations (P)	1	5
Learning rate (ε)	0.7	0.7
P times ε (relative value)	0.7 (1)	3.5 (5)

Table 3. Experimental conditions in Experiment 3: selective-learning-rate.

	S-data	L-data
Range of absolute value of ΔTOPIX(t)	0 to 50%	50 to 100%
Number of data	150	150
Relative number of presentations (P)	1	1
Learning rate (ε)	0.14	0.7
P times ε (relative value)	0.14 (1)	0.7 (5)

Experimental conditions in Experiment 4 are shown in Table 4. S-data were separated into S1- and S2-data, where changes in S2-data were larger than those in S1-data. Here, the boundary between S1- and S2-data was at the 25% point. (The 25% point means that 25% of the data is between the minimum data and the 25% point data. The 50% point corresponds to the "median.") L-data were separated into L1- and L2-data, where changes in L2-data were larger than those in L1-data. Here, the boundary between L1- and L2-data was the 75% point. The 25%, 50%, and 75% points were 5.36 (about US$ 0.51), 14.78 (US$ 1.40) and 31.04 points (US$ 2.94), respectively. L1-, L2-, S1-, and S2-data each had 75 data. In Experiment 4, L1- and L2-data were presented five times as often as S1- and S2-data. In Experiment 4, the learning rate for L1- and S1-data was reduced to 50% (i.e., $\varepsilon = 0.7$ for L2- and S2-data, and $\varepsilon = 0.35$ for L1- and S1-data). Relative *selective intensities* (number of presentations times learning rate) for S1-, S2-, L1-, and L2-data were 1, 2, 5, and 10, respectively.

Table 4. Experimental conditions in Experiment 4: the hybrid technique.

	S1-data	S2-data	L1-data	L2-data
Range of absolute value of ΔTOPIX(t)	0 to 25%	25 to 50%	50 to 75%	75 to 100%
Number of data	75	75	75	75
Relative number of presentations (P)	1	1	5	5
Learning rate (ε)	0.35	0.7	0.35	0.7
P times ε (relative value)	0.35 (1)	0.7 (2)	1.75 (5)	3.5 (10)

In each experiment, network learning was stopped at the point having the maximum profit (the learning was stopped at the point having the maximum profit for the validation data during 8000 learning cycles). The prediction error and profit were monitored after every hundred learning cycles.

When a large change in TOPIX was predicted, we tried to calculate "Profit" as follows: when the predicted direction was the same as the actual direction, the daily change in TOPIX was earned, and when it was different, the daily change in TOPIX was lost. This calculation of profit corresponds to the following experimental TOPIX trading system. A buy (sell) order is issued when the predicted next-day's up (down) in TOPIX is larger than a preset value which corresponds to a large change. When a buy (sell) order is issued, the system buys (sells) TOPIX shares at the current price and subsequently sells (buys) them back at the next-day price. Transaction costs on the trades were ignored in calculating the profit. The more accurately a large change is predicted, the larger the profit is.

In each experiment, the momentum parameter α was 0.7. All the weights and biases in the neural network were initialized randomly between -0.3 and 0.3. In each experiment the neural network was run four times for the same training data with different initial weights and the average was taken.

3.2 Results

The experimental results are shown in Table 5. Multiple regression analysis (MR) was also used in the experiments. The "prediction error on large-change test data" is the mean absolute value of the prediction error for the test L-data.

Applying our *selective-presentation* approach (Experiment 2) reduced the prediction error for test L-data and improved profits: the prediction-error on L-data was reduced by 7% (1- (21.3/22.9)) and the network's ability to make profits through experimental TOPIX-trading was improved by 30% (550/422) compared with the results obtained with the usual presentation approach (Experiment 1).

The prediction error and profits in Experiment 3 (*selective-learning-rate* approach) were comparable to those in Experiment 2 (*selective-presentation* approach). Combining *selective-presentation* with *selective-learning-rate* approaches (Experiment 4) further reduced the prediction error for test L-data and improved profits: the prediction-error was reduced by 10% (1- (20.7/22.9)) and the network's ability to make profits was improved by 38% (581/422).

Table 5. Experimental results.

	MR	Exp. 1	Exp. 2	Exp. 3	Exp. 4
Presentation method	equal	equal	*selective*	equal	*selective*
Learning rate		equal	equal	*selective*	*selective*
Prediction error for large-change data (relative value)	24.3 (1.06)	22.9 (1)	21.3 (0.93)	21.3 (0.93)	20.7 (0.90)
Profit on test data (relative value)	265 (0.62)	422 (1)	550 (1.30)	563 (1.33)	581 (1.38)

4 Conclusions

We investigated selective learning techniques for forecasting. In the first approach, training data corresponding to large changes in the prediction-target time series are presented more often, in the second approach, the learning rate for training data corresponding to small changes is reduced, and in the third approach, these two techniques are combined. The results of several experiments on stock-price prediction showed that the performances of *selective-presentation* and *selective-learning-rate* approaches were both better than the usual presentation approach, and combining them further improved the performance. Next, we will apply these techniques today's stock market and other real-world forecasting problems. We also plan to develop a forecasting method that integrates statistical analysis with neural networks.

References

1. Weigend, A., Huberman, B., Rumelhart, D.: Predicting the future: a connectionist approach. International Journal of Neural Systems, Vol. 1, No. 3. (1990) 193-209
2. Vemuri, V., Rogers, R. (eds): Artificial Neural Networks: Forecasting Time Series. IEEE Press, Los Alamitos, CA (1994)
3. Pham, D., Liu, X.: Neural Networks for Identification, Prediction and Control. Springer (1995)
4. Kil, D., Shin, F.: Pattern Recognition and Prediction with Applications to Signal Characterization. American Institute of Physics Press (1996)
5. Mandic, D., Chambers, J.: Recurrent Neural Networks for Prediction. John Wiley & Sons (2001)
6. Azoff, E.: Neural Network Time Series Forecasting of Financial Markets. John Wiley and Sons, West Sussex (1994)
7. Refenes, A., Azema-Barac, M.: Neural network applications in financial asset management. Neural Computing & Applications, Vol. 2, No. 1. Springer-Verlag, London (1994) 13-39
8. White, H.: Economic prediction using neural networks: the case of IBM daily stock return. Proceedings of International Conference on Neural Networks. San Diego, CA (1988) II-451-II-458
9. Baba, N., Kozaki, M.: An intelligent forecasting system of stock price using neural networks. Proceedings of International Conference on Neural Networks. Singapore (1992) I-371-I-377
10. Freisleben, B.: Stock market prediction with backpropagation networks. Lecture Notes in Computer Science, Vol. 604. Springer-Verlag, Heidelberg (1992) 451-460
11. Tang, Z., Almeida, C., Fishwick, P.: Time series forecasting using neural networks vs. Box-Jenkins methodology. Simulation, Vol. 57, No. 5. (1991) 303-310
12. Kohara, K., Fukuhara, Y., Nakamura, Y.: Selective presentation learning for neural network forecasting of stock markets. Neural Computing & Applications, Vol. 4, No. 3. Springer-Verlag, London (1996) 143-148
13. Kohara, K., Fukuhara, Y., Nakamura, Y.: Selectively intensive learning to improve large-change prediction by neural networks. Proceedings of International Conference on Engineering Applications of Neural Networks. London (1996) 463-466
14. Kohara, K.: Selective-learning-rate approach for stock market prediction by simple recurrent neural networks. Lecture Notes in Artificial Intelligence, Vol. 2773. Springer-Verlag, Heidelberg, (2003) 141-147
15. Kohara, K.: Neural networks for economic forecasting problems. In: Cornelius T. Leondes (ed): Expert Systems -The Technology of Knowledge Management and Decision Making for the 21st Century-. Academic Press. San Diego, CA (2002)
16. Kohara, K.: Foreign Exchange Rate Prediction with Selective Learning BPNNs and SOMs. Proceedings of World Multi-Conference on Systemics, Cybernetics and Informatics. Orland, FL (2005) 350-354
17. Park, D., El-Sharkawi, M., Marks II, R., Atlas, L., Damborg, M.: Electric load forecasting using an artificial neural network. IEEE Transactions on Power Systems. Vol. 6, No. 2. (1991) 442-449

Learning and Evolution in Artificial Neural Networks: A Comparison Study

Eva Volna

University of Ostrava, 30ht dubna st. 22, 701 03 Ostrava, Czech Republic
eva.volna@osu.cz

Abstract. This paper aims at learning and evolution in artificial neural networks. Here is presented a system evolving populations of neural nets that are fully connected multilayer feedforward networks with fixed architecture solving given tasks. The system is compared with gradient descent weight training (like backpropagation) and with hybrid neural network adaptation. All neural networks have the same architecture and solve the same problems to be able to be compared mutually. In order to test the efficiency of described algorithms, we applied them to the *Fisher's Iris data set* [1] that is the bench test database from the area of machine learning.

1 Learning in Artificial Neural Networks

Learning in artificial neural networks is typically accomplished using examples. This is also called training in artificial neural networks because the learning is achieved by adjusting the connection weights in artificial neural networks iteratively so that trained (or learned) artificial neural networks can perform certain tasks. The essence of a learning algorithm is the learning rule, i.e. a weight-updating rule, which determines how connection weights are changed. Examples of popular learning rules include the delta rule, the Hebbian rule, the anti-Hebbian rule, and the competitive learning rule, etc. More detailed discussion of artificial neural networks can be found in [2]. Learning in artificial neural networks can roughly be divided into supervised, unsupervised, and reinforcement learning. Without commonness, we are going to target multilayer feedforward neural networks that are adapted with backpropagation algorithm.

Supervised learning is based on direct comparison between the actual output of an artificial neural network and the desired correct output, also known as the target output. It is often formulated as the minimization of an error function such as the total mean square error between the actual output and the desired output summed over all available data. A gradient descent-based optimization algorithm such as backpropagation [2] can then be used to adjust connection weights in the artificial neural network iteratively in order to minimize the error. There have been some successful applications of backpropagation in various areas [3]–[5], but backpropagation has drawbacks due to its use of gradient descent. It often gets trapped in a local minimum of the error function and is incapable of finding a global minimum if the error function is multimodal and/or nondifferentiable.

2 NeuroEvolutionary Learning in Artificial Neural Networks

Evolutionary algorithms refer to a class of population-based stochastic search algorithms that are developed from ideas and principles of natural evolution. They include evolution strategies, evolutionary programming, genetic algorithms etc. [6]. One important feature of all these algorithms is their population based search strategy. Individuals in a population compete and exchange information with each other in order to perform certain tasks. Evolutionary algorithms are particularly useful for dealing with large complex problems, which generate many local optima. They are less likely to be trapped in local minima than traditional gradient-based search algorithms. They do not depend on gradient information and thus are quite suitable for problems where such information is unavailable or very costly to obtain or estimate. They can even deal with problems where no explicit and/or exact objective function is available. These features make them much more robust than many other search algorithms. There is a good introduction to various evolutionary algorithms for optimization in [6].

Evolution has been introduced into artificial neural networks at roughly three different levels [7]: connection weights; architectures; and learning rules. The evolution of connection weights introduces an adaptive and global approach to training, especially in the reinforcement learning and recurrent network-learning paradigm where gradient-based training algorithms often experience great difficulties. The evolution of architectures enables artificial neural networks to adapt their topologies to different tasks without human intervention and thus provides an approach to automatic artificial neural network design as both connection weights and structures can be evolved. The evolution of learning rules can be regarded as a process of learning to learn in artificial neural networks where the adaptation of learning rules is achieved through evolution. It can also be regarded as an adaptive process of automatic discovery of novel learning rules.

The evolutionary approach to weight training in artificial neural networks consists of two major phases. The first phase is to decide the representation of connection weights, i.e., whether in the form of binary strings or real strings. The second one is the evolutionary process simulated by an evolutionary algorithm, in which search operators such as crossover and mutation have to be decided in conjunction with the representation scheme. Different representations and search operators can lead to quite different training performance. In a binary representation scheme, each connection weight is represented by a number of bits with certain length. An artificial neural network is encoded by concatenation of all the connection weights of the network in the chromosome. The advantages of the binary representation lie in its simplicity and generality. It is straightforward to apply classical crossover (such as one-point or uniform crossover) and mutation to binary strings [6]. Real numbers have been proposed to represent connection weights directly, i.e. one real number per connection weight [6]. As connection weights are represented by real numbers, each individual in an evolving population is then a real vector. Traditional binary crossover and mutation can no longer be used directly. Special search operators have to be designed. Montana and Davis [8] defined a large number of tailored genetic operators, which incorporated many heuristics about training artificial neural networks. The idea was to retain useful feature detectors formed around hidden nodes during evolution.

One of the problems faced by evolutionary training of artificial neural networks is the permutation problem [7] also known as the competing convention problem. It is caused by the many-to-one mapping from the representation (genotype) to the actual artificial neural network (phenotype) since two artificial neural networks that order their hidden nodes differently in their chromosomes will still be equivalent functionally. In general, any permutation of the hidden nodes will produce functionally equivalent artificial neural networks with different chromosome representations. The permutation problem makes crossover operator very inefficient and ineffective in producing good offspring. The role of *crossover* has been controversial in neuroevolution as well as among the evolutionary computation community in general. However, there have been successful applications using crossover operations to evolve neural networks [9]. Compared with the *mutation* only system, the performance of the system using crossover operations is in general better and that it also helps to compress the overall size of search space faster.

3 Comparison between Evolutionary Training and Gradient-based Training

The evolutionary training approach is attractive because it can handle the global search problem better in a vast, complex, multimodal, and nondifferentiable surface. It does not depend on gradient information of the error (or fitness) function and thus is particularly appealing when this information is unavailable or very costly to obtain or estimate. For example, the same evolutionary algorithms can be used to train many different networks: recurrent artificial neural networks [10], higher order artificial neural networks [11], and fuzzy artificial neural networks [12]. The general applicability of the evolutionary approach saves a lot of human efforts in developing different training algorithms for different types of artificial neural networks. The evolutionary approach also makes it easier to generate artificial neural networks with some special characteristics. For example, the artificial neural networks complexity can be decreased and its generalization increased by including a complexity (regularization) term in the fitness function. Unlike the case in gradient-based training, this term does not need to be differentiable or even continuous. Weight sharing and weight decay can also be incorporated into the fitness function easily.

However, evolutionary algorithms are generally much less sensitive to initial conditions of training. They always search for a globally optimal solution, while a gradient descent algorithm can only find a local optimum in a neighbourhood of the initial solution.

4 Hybrid Training

Most evolutionary algorithms are rather inefficient in fine-tuned local search although they are good at global search. This is especially true for genetic algorithms. The efficiency of evolutionary training can be improved significantly by incorporating a local search procedure into the evolution, i.e. combining evolutionary algorithm's

global search ability with local search's ability to fine tune. Evolutionary algorithms can be used to locate a good region in the space and then a local search procedure is used to find a near-optimal solution in this region. The local search algorithm could be backpropagation [2] or other random search algorithms. Hybrid training has been used successfully in many application areas: Lee [13] and many others used genetic algorithms to search for a near-optimal set of initial connection weights and then used backpropagation to perform local search from these initial weights. Their results showed that the *hybrid algorithm* approach was more efficient than either the genetic algorithm or backpropagation algorithm used alone. If we consider that backpropagation often has to run several times in practice in order to find good connection weights due to its sensitivity to initial conditions, the hybrid training algorithm will be quite competitive. Similar work on the evolution of initial weights has also been done on competitive learning neural networks [14] and Kohonen networks [15].

It is interesting to consider finding good initial weights as locating a good region in the weight space. Defining that basin of attraction of a local minimum as being composed of all the points, sets of weights in this case, which can converge to the local minimum through a local search algorithm, then a global minimum can easily be found by the local search algorithm if an evolutionary algorithm can locate a point, i.e. a set of initial weights, in the basin of attraction of the global minimum. Fig. 1 illustrates a simple case where there is only one connection weight in the artificial neural networks. If an evolutionary algorithm can find an initial weight such as w_{I2}, it would be easy for a local search algorithm to arrive at the globally optimal weight w_B even though w_{I2} itself is not as good as w_{I1}.

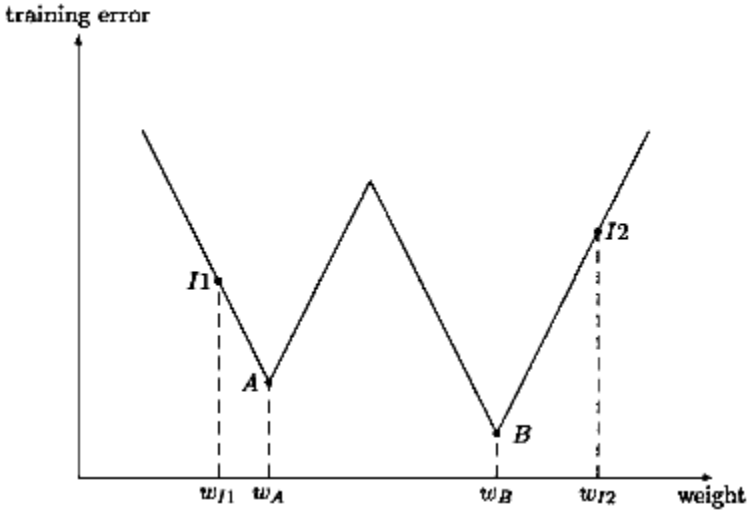


Fig. 1. An illustration of using an evolutionary algorithm to find good initial weights such that a local search algorithm can find the globally optimal weights easily. w_{I2} in this figure is an optimal initial weight because it can lead to the global optimum w_B using a local search algorithm.

5 Experiments

In order to test the efficiency of described algorithms, we applied it to the Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by Sir Ronald Aylmer Fisher as an example of discriminated analysis [1]. It is sometimes called *Anderson's Iris data* set because Edgar Anderson collected the data to quantify the geographic variation of Iris flowers in the Gaspe Peninsula. The dataset consists of 50 samples from each of three species of Iris flowers (*Iris setosa, Iris virginica and Iris versicolor*). Four features were measured from each sample, they are the length and the width of sepal and petal. Based on the combination of the four features, Fisher developed a linear discriminated model to determine which species they are (see Table 1). There are three-layer feedforward neural networks with architecture is 4 - 4 - 3 (e.g. four units in the input layer, four units in the hidden layer, and three units in the output layer) in all experimental works, because the *Fisher's Iris data set* [1] is not linearly separable and therefore we cannot use neural network without hidden units. All nets are fully connected. The input values of the training set from the Table 1 are transformed into interval <0; 1> to be use backpropagation algorithms for adaptation.

Weight Evolution in Artificial Neural Networks: the initial population contains 30 individuals (weight representations of three-layer neural networks). There are connection weights represented by real numbers in each chromosome. We use the genetic algorithm with the following parameters: probability of mutation is 0,01 and probability of crossover is 0,5. Adaptation of each neural network starts with randomly generated weight values in the initial population.

The Gradient Descent adaptation deals through backpropagation with the following parameters: learning rate is 0.4, momentum is 0.

The Hybrid Training combines parameters from genetic algorithms and backpropagation. It makes one backpropagation epoch with probability 0,5 in each generation.

6 Conclusions

History of error functions is shown in the Figure 2. There are shown average values of error functions in the given population. The "*gradient descent adaptation*" represents an adaptation with the backpropagation. There are shown average values of error functions, because the adaptation with backpropagation algorithm was applied 10 times for each calculation.

All networks solve *Fisher's Iris data set* [1] in our experiment. Now we can compare results from all experiments, e.g. weight evolution, gradient descent adaptation, and hybrid training. Other numerical simulations give very similar results. If we can see from Figure 2, the hybrid training shows the best results from all of them. In general, hybrid algorithms tend to perform better than others for a large number of problems, because they combine evolutionary algorithm's global search ability with local search's ability to fine tune

Table 1. The set of patterns (the Fisher's Iris Data training set), where **Se** means *setosa*, **Vi** means *virginica*, and **Ve** means *versicolor.*

Sepal Lengt	Sepal Width	Petal Length	Petal Width	Spe-cies	Sepal Lengt	Sepal Width	Petal Length	Petal Width	Spe-cies	Sepal Lengt	Sepal Width	Petal Length	Petal Width	Spe-cies
5,1	3,5	1,4	0,2	**Se**	6,3	3,3	6	2,5	**Vi**	7	3,2	4,7	1,4	**Ve**
4,9	3	1,4	0,2	**Se**	5,8	2,7	5,1	1,9	**Vi**	6,4	3,2	4,5	1,5	**Ve**
4,7	3,2	1,3	0,2	**Se**	7,1	3	5,9	2,1	**Vi**	6,9	3,1	4,9	1,5	**Ve**
4,6	3,1	1,5	0,2	**Se**	6,3	2,9	5,6	1,8	**Vi**	5,5	2,3	4	1,3	**Ve**
5	3,6	1,4	0,2	**Se**	6,5	3	5,8	2,2	**Vi**	6,5	2,8	4,6	1,5	**Ve**
5,4	3,9	1,7	0,4	**Se**	7,6	3	6,6	2,1	**Vi**	5,7	2,8	4,5	1,3	**Ve**
4,6	3,4	1,4	0,3	**Se**	4,9	2,5	4,5	1,7	**Vi**	6,3	3,3	4,7	1,6	**Ve**
5	3,4	1,5	0,2	**Se**	7,3	2,9	6,3	1,8	**Vi**	4,9	2,4	3,3	1	**Ve**
4,4	2,9	1,4	0,2	**Se**	6,7	2,5	5,8	1,8	**Vi**	6,6	2,9	4,6	1,3	**Ve**
4,9	3,1	1,5	0,1	**Se**	7,2	3,6	6,1	2,5	**Vi**	5,2	2,7	3,9	1,4	**Ve**
5,4	3,7	1,5	0,2	**Se**	6,5	3,2	5,1	2	**Vi**	5	2	3,5	1	**Ve**
4,8	3,4	1,6	0,2	**Se**	6,4	2,7	5,3	1,9	**Vi**	5,9	3	4,2	1,5	**Ve**
4,8	3	1,4	0,1	**Se**	6,8	3	5,5	2,1	**Vi**	6	2,2	4	1	**Ve**
4,3	3	1,1	0,1	**Se**	5,7	2,5	5	2	**Vi**	6,1	2,9	4,7	1,4	**Ve**
5,8	4	1,2	0,2	**Se**	5,8	2,8	5,1	2,4	**Vi**	6,7	3,1	4,4	1,4	**Ve**
5,7	4,4	1,5	0,4	**Se**	6,4	3,2	5,3	2,3	**Vi**	5,6	2,9	3,6	1,3	**Ve**
5,4	3,9	1,3	0,4	**Se**	6,5	3	5,5	1,8	**Vi**	5,6	3	4,5	1,5	**Ve**
5,1	3,5	1,4	0,3	**Se**	7,7	3,8	6,7	2,2	**Vi**	5,8	2,7	4,1	1	**Ve**
5,7	3,8	1,7	0,3	**Se**	7,7	2,6	6,9	2,3	**Vi**	5,6	2,5	3,9	1,1	**Ve**
5,1	3,8	1,5	0,3	**Se**	6	2,2	5	1,5	**Vi**	6,2	2,2	4,5	1,5	**Ve**
5,4	3,4	1,7	0,2	**Se**	6,9	3,2	5,7	2,3	**Vi**	5,9	3,2	4,8	1,8	**Ve**
5,1	3,7	1,5	0,4	**Se**	5,6	2,8	4,9	2	**Vi**	6,1	2,8	4	1,3	**Ve**
4,6	3,6	1	0,2	**Se**	7,7	2,8	6,7	2	**Vi**	6,3	2,5	4,9	1,5	**Ve**
5,1	3,3	1,7	0,5	**Se**	6,3	2,7	4,9	1,8	**Vi**	6,1	2,8	4,7	1,2	**Ve**
4,8	3,4	1,9	0,2	**Se**	6,7	3,3	5,7	2,1	**Vi**	6,4	2,9	4,3	1,3	**Ve**
5	3	1,6	0,2	**Se**	7,2	3,2	6	1,8	**Vi**	6,6	3	4,4	1,4	**Ve**
5	3,4	1,6	0,4	**Se**	6,2	2,8	4,8	1,8	**Vi**	6,8	2,8	4,8	1,4	**Ve**
5,2	3,5	1,5	0,2	**Se**	6,1	3	4,9	1,8	**Vi**	6,7	3	5	1,7	**Ve**
5,2	3,4	1,4	0,2	**Se**	6,4	2,8	5,6	2,1	**Vi**	6	2,9	4,5	1,5	**Ve**
4,7	3,2	1,6	0,2	**Se**	7,2	3	5,8	1,6	**Vi**	5,7	2,6	3,5	1	**Ve**
4,8	3,1	1,6	0,2	**Se**	7,4	2,8	6,1	1,9	**Vi**	5,5	2,4	3,8	1,1	**Ve**
5,4	3,4	1,5	0,4	**Se**	7,9	3,8	6,4	2	**Vi**	5,5	2,4	3,7	1	**Ve**
5,2	4,1	1,5	0,1	**Se**	6,4	2,8	5,6	2,2	**Vi**	5,8	2,7	3,9	1,2	**Ve**
5,5	4,2	1,4	0,2	**Se**	6,3	2,8	5,1	1,5	**Vi**	6	2,7	5,1	1,6	**Ve**
4,9	3,1	1,5	0,2	**Se**	6,1	2,6	5,6	1,4	**Vi**	5,4	3	4,5	1,5	**Ve**
5	3,2	1,2	0,2	**Se**	7,7	3	6,1	2,3	**Vi**	6	3,4	4,5	1,6	**Ve**
5,5	3,5	1,3	0,2	**Se**	6,3	3,4	5,6	2,4	**Vi**	6,7	3,1	4,7	1,5	**Ve**
4,9	3,6	1,4	0,1	**Se**	6,4	3,1	5,5	1,8	**Vi**	6,3	2,3	4,4	1,3	**Ve**
4,4	3	1,3	0,2	**Se**	6	3	4,8	1,8	**Vi**	5,6	3	4,1	1,3	**Ve**
5,1	3,4	1,5	0,2	**Se**	6,9	3,1	5,4	2,1	**Vi**	5,5	2,5	4	1,3	**Ve**
5	3,5	1,3	0,3	**Se**	6,7	3,1	5,6	2,4	**Vi**	5,5	2,6	4,4	1,2	**Ve**
4,5	2,3	1,3	0,3	**Se**	6,9	3,1	5,1	2,3	**Vi**	6,1	3	4,6	1,4	**Ve**
4,4	3,2	1,3	0,2	**Se**	5,8	2,7	5,1	1,9	**Vi**	5,8	2,6	4	1,2	**Ve**
5	3,5	1,6	0,6	**Se**	6,8	3,2	5,9	2,3	**Vi**	5	2,3	3,3	1	**Ve**
5,1	3,8	1,9	0,4	**Se**	6,7	3,3	5,7	2,5	**Vi**	5,6	2,7	4,2	1,3	**Ve**
4,8	3	1,4	0,3	**Se**	6,7	3	5,2	2,3	**Vi**	5,7	3	4,2	1,2	**Ve**
5,1	3,8	1,6	0,2	**Se**	6,3	2,5	5	1,9	**Vi**	5,7	2,9	4,2	1,3	**Ve**
4,6	3,2	1,4	0,2	**Se**	6,5	3	5,2	2	**Vi**	6,2	2,9	4,3	1,3	**Ve**
5,3	3,7	1,5	0,2	**Se**	6,2	3,4	5,4	2,3	**Vi**	5,1	2,5	3	1,1	**Ve**
5	3,3	1,4	0,2	**Se**	5,9	3	5,1	1,8	**Vi**	5,7	2,8	4,1	1,3	**Ve**

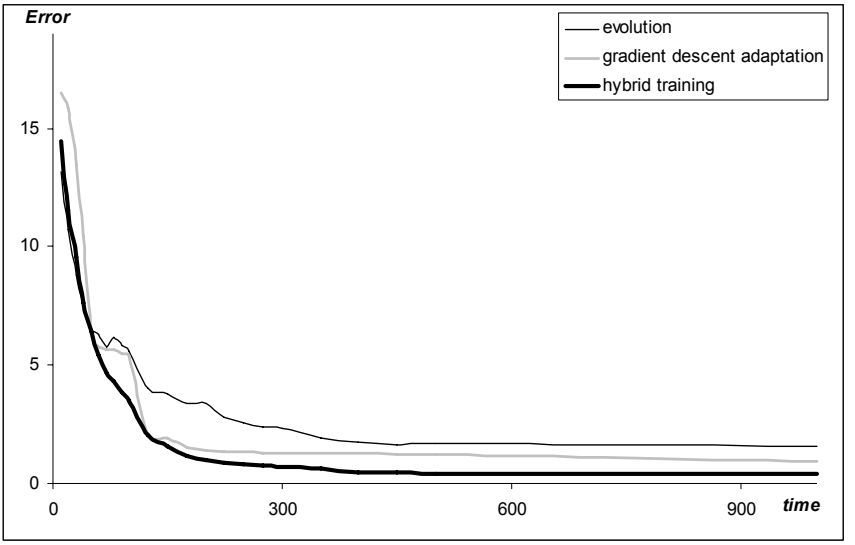


Fig. 2. The error function history.

References

1. http://en.wikipedia.org/wiki/Iris_flower_data_set (from 16/1/2008)
2. Hertz, J., Krogh, A., and Palmer, R. *Introduction to the Theory of Neural Computation.* Reading, MA: Addison-Wesley, 1991.
3. Lang, K. J. Waibel, A. H., and Hinton, G. E."A time-delay neural network architecture for isolated word recognition," Neural Networks, vol. 3, no. 1, pp. 33–43, 1990.
4. Fels S. S. and Hinton, G. E. "Glove-talk: A neural network interface between a data-glove and a speech synthesizer," IEEE Trans. Neural Networks, vol. 4, pp. 2–8, Jan. 1993.
5. Knerr, S., Personnaz, L., and G. Dreyfus, "Handwritten digit recognition by neural networks with single-layer training," IEEE Trans, Neural Networks, vol. 3, pp. 962–968, Nov. 1992, neural networks that uses genetic-algorithm techniques."
6. Bäck, T., Hammel, U., and Schwefel, H.-P. "Evolutionary computation: Comments on the history and current state". *IEEE Trans, Evolutionary Computation*, vol. 1, pp. 3–17, Apr. 1997.
7. Yao, X. "Evolving artificial neural networks", In *Proceedings of the IEEE* 89 (9) 1423-1447, 1999.
8. Montana, D., and Davis, L. "Training feedforward neural networks using genetic algorithms." In *Proceedings 11th Int. Joint Conf. Artificial Intelligence*, pp. 762–767San Mateo, CA: Morgan Kaufmann, 1989.
9. Pujol, J. C. F. and Poli, R. Evolving the topology and the weights of neural networks using a dual representation, Applied Intelligence, 8(1):73–84, 1998.
10. Angeline, P. J., Saunders, G, M., and Pollack, J. B. An evolutionary algorithm that constructs recurrent neural networks, *IEEE Transactions on Neural Networks*, pages 54–65, 1994.
11. Janson D. J. and Frenzel, J. F. "Application of genetic algorithms to the training of higher order neural networks," *J, Syst, Eng.*, vol. 2, pp. 272–276, 1992.

12. Lehotsky, M. Olej, V. and Chmurny, J. “Pattern recognition based on the fuzzy neural networks and their learning by modified genetic algorithms,” *Neural Network World,* vol. 5, no. 1, pp. 91–97, 1995.
13. Lee, S.-W. “Off-line recognition of totally unconstrained handwritten numerals using multilayer cluster neural network,” IEEE Trans. Pattern Anal. Machine Intell., vol. 18, pp. 648–652, June 1996.
14. Merelo, J. J. Paton, M. Canas, A., Prieto, A. and Moran, F. “Optimization of a competitive learning neural network by genetic algorithms,” in *Proc. Int. Workshop Artificial Neural Networks (IWANN’93*), Lecture Notes in Computer Science, vol. 686, Berlin, Germany: Springer-Verlag, 1993, pp. 185–192.
15. Wang, D. D. and Xu, J. “Fault detection based on evolving LVQ neural networks,” in *Proc. 1996 IEEE Int. Conf. Systems, Man and Cybernetics*, vol. 1, pp. 255–260.

ZISC Neuro-Computer for Task Complexity Estimation in T-DTS Framework

Ivan Budnyk, Abdennasser Chebira and Kurosh Madani

Images, Signals and Intelligent Systems Laboratory (LISSI/EA 3956), University Paris 12
Senart-Fontainebleau Institute of Technology, Bat A, Av. Pierre Point, 77127 Lieusaint, France
{Ivan.Budnyk, Chebira, Madani}@Univ-Paris12.fr
http://lissi.univ-paris12.fr

Abstract. This paper deals with T-DTS, a self-organizing information processing system and especially with its complexity estimation mechanism which is based on a ZISC © IBM ® neuro-computer. The above-mentioned mechanism has been compared with a number probabilistic complexity estimation techniques already implemented in T-DTS.

1 Introduction

This work connects closely the Tree-like Divide to Simplify (T-DTS) [1] framework, which is a hybrid multiple Neural Networks software platform constructing a decomposed-tree of neural structures aiming to solve complex classification problems using "divide and conquer" paradigm. In other words, the solution of a complex classification task is approached by dividing the initial complex task into a set of classification sub-problems with reduced complexity. So, the splitting mechanism and associated policy play a key role in tree-like self-organization to obtain multi-neural structure as well as in its classification performances. In T-DTS, the splitting is performed using a complexity estimation mechanism acting as a regulation loop on decomposition process [2]. If several probabilistic (or statistical) complexity estimation perspectives have been investigated in [3], an appealing slant consists of using Artificial Neural Network's (ANN) learning issued mechanisms as indicators for complexity estimation: especially, those modifying the ANN topology. Among various available ANNs, a promising candidate is the Restricted Coulomb Energy (RCE) neural model and the relatively simple learning mechanism of such ANN, modifying directly its topology (number of neurons in hidden layer). Moreover, the standard CMOS based ZISC © IBM ® neuro-computer, implementing this kind of neural model, is an attractive feature offering hardware implementation potentiality of a ZISC based difficulty/complexity tasks estimator [4]. If the expected impact of the complex task decomposition is to increase the classification quality, an additional impact of a ZISC based difficulty/complexity tasks estimator is the decreasing of learning time (generally, processing time consumer).

In this paper we compare the ZISC based complexity estimation indicator with those based on probabilistic (or statistical) measures described in [2] and [3], already

implanted in T-DTS. We compare obtained results in order to show special niche which takes ZISC complexity estimator among the others.
In *Section 2* we describe T-DTS concept and its software platform, highlighting key role of complexity estimating unit and providing its' component analysis. *Section 3* presents ZISC based *complexity* estimating issue and its influence on T-DTS performance. *Section 4* gives results of the above-mentioned comparison obtained on the basis of a simple 2D benchmark and two real-world classification problems (available in UCI Machine Learning depository). *Section 5* includes summary and our further progress.

2 Hybrid Multiple Neural Networks Framework - T-DTS

In essence, T-DTS is a modular structure [5]. The purpose is based on the use of a set of specialized mapping Neural Network, called Processing Units (PU), supervised by a set of Decomposition Units (DU). Decomposition Units are a prototypes based on Neural Networks. Processing Units are modeling algorithms of Artificial Neural Networks origin. Thus, T-DTS paradigm allows us to build a tree structure of models in the following way:

- at the nodes level(s) - the input space is decomposed into a set of optimal sub-spaces of the smaller size;
- at the leaves level(s) - the aim is to learn the relation between inputs and outputs of sub-spaces obtained from splitting.

Following the main paradigm T-DTS acts in two main operational phases:
Learning: recursive decomposition under DU supervision of the database into subsets: tree structure building phase;
Operational: Activation of the tree structure to compute system output (provided by PU at tree leaf's level

General block scheme of the functioning of T-DTS is described on Fig. 1. The proposed schema builds an open software architecture. It can be adapted to specific problem using the appropriate modeling paradigm at PU level: we use mainly Artificial Neural Network computing model in this work. In our case the tree structure construction is based on a complexity estimation module. This module introduces a feedback in the learning process and control the tree building process. The reliability of tree model to sculpt the problem behavior is associated to the complexity estimation module. The whole decomposing process is built on the paradigm "*splitting database into sub-databases - decreasing task complexity*". It means that the decomposition process is activated until a low satisfactory ratio complexity is reached. T-DTS software architecture is depicted on Fig. 2. T-DTS software incorporates three databases: decomposition methods, ANN models and complexity estimation modules databases.
T-DTS software engine is the *Control Unit.* This core-module controls and activates several software packages: normalization of incoming database (if it's required), splitting and building a tree of prototypes using selected decomposition method, sculpting the set of local results and generating global result (learning and generalization rates). T-DTS software can be seen as a *Lego system* of decomposition methods,

processing methods powered by a control engine an accessible by operator thought Graphic User Interface.

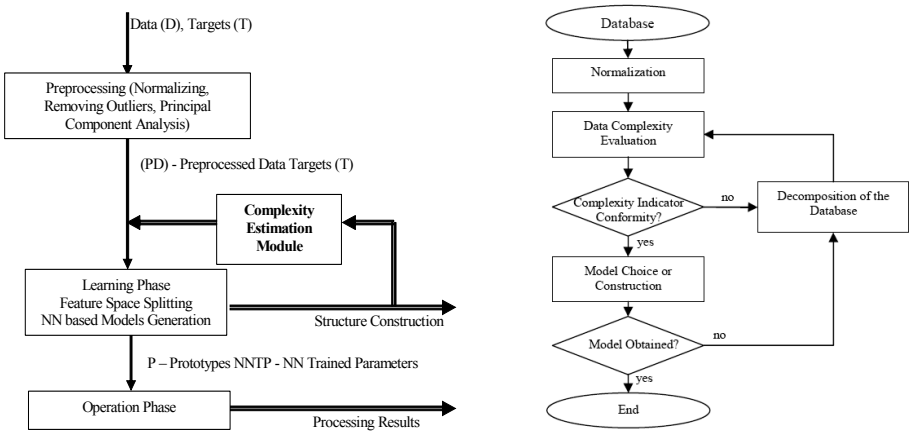


Fig. 1. Bloc scheme of T-DTS: *Left* – Modular concept, *Right* – Algorithmical concept.

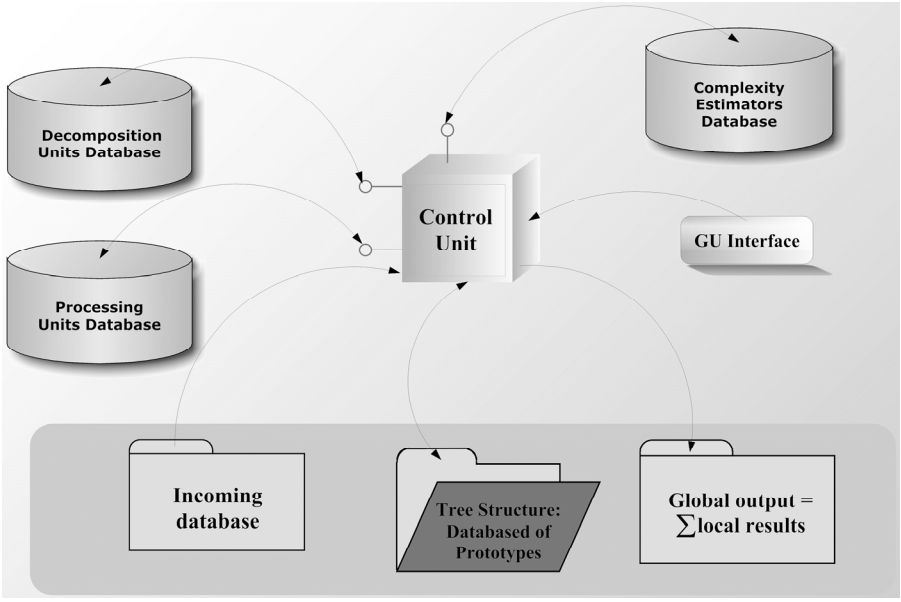


Fig. 2. T-DTS software architecture.

Those three databases can be independently developed out of the main frame and more important - they can be easily incorporated into T-DTS framework.

For example, SOM-LSVMDT [6] algorithm; witch is based on the same idea of decomposition, can be implement by T-DTS by mean of LSVMDT [7] (Linear Support Vector Machine Decision Tree) processing method incorporation into PU database.

The current T-DTS software (version 2.02) includes the following units and methods:

1. Decomposition Units:
   - CN (Competitive Network)
   - SOM (Self Organized Map)
   - LVQ (Learning Vector Quantization)
2. Processing Units:
   - LVQ (Learning Vector Quantization)
   - Perceptrons
   - MLP (Multilayer Perceptron)
   - GRNN (General Regression Neural Network)
   - RBF (Radial basis function network)
   - PNN (Probabilistic Neural Network)
   - LN
3. Complexity estimators are [3] based on the following criteria:
   - MaxStd (Sum of the maximal standard deviations)
   - Fisher measure.
   - Purity algorithm
   - Normalized mean distance
   - Divergence measure
   - Jeffries-Matusita distance
   - Bhattacharyya bound
   - Mahalanobis distance
   - Scattered-matrix method based on inter-intra matrix-criteria [8]
   - ZISC© IBM ® based complexity indicator [4]

3 ZISC Complexity Indicator

3.1 Complexity Estimation in T-DTS Framework

In this subsection we pass ahead of the possible question concerning the word *"complexity"*. Let us highlight that in this part we are not focused on studying statistical complexity as it is used in the areas of physics and informational theory. There are different concepts of *complexity* which are depending on chosen language base, the type of difficulty focused on the type of formulation desired within that language [9]. We have to mention a growing criticism concerning the term *complexity*, because it has been misused without a proper definition [10].
Accordingly to T-DTS concept Fig. 1, complexity estimation module plays a key-role in decomposition process, and so is essential in tree structure construction process. There are two main problems:

- finding of the optimal threshold value for selected complexity estimating indicator,
- lack of the universal complexity estimator (method of complexity estimating) that could be applied for any classification task independently of data nature.

We define a task *complexity* as the amount of neurocomputer computational resource that it takes to solve a classification problem [11].

Thus the *complexity* here is the limited supply of these *resources* (amount of neurons) once *the appropriate program* (classification methods) is supplied. Our primary study interest is in classification complexity in term of computational difficulty of neuro-computer IBM © ZISC-036 ® hardware to obtain satisfactory learning and generalization rates using RBF algorithm and adjusted initial parameters.

3.2 Complexity Estimating based on IBM© ZISC-036® Hardware

This complexity criterion is based on IBM © ZISC-036 ® neuro-computer hardware which is a fully integrated circuit based on neural network paradigm [12]. It is a parallel neural processor based on the K-Nearest Neighbor (KNN) [13] and Reduced Coulomb Energy (RCE) [14] algorithms. The principal idea for extracting information about classification task complexity is linked to the limitation of resources (neurons) of IBM © ZISC-036 ® neuron-computer.
We expect that a more complex problem will involve a more complex ZISC neural network structure. The simplest neural network structure feature is the number n of neurons created during the learning phase. The following indicator is defined, where n is a parameter that reflects complexity:

$$Q = \frac{n}{m} \ , \ m \geq 1, n \geq 0 \tag{1}$$

We suppose that there exists some function $n = g(.)$ that reflects problem complexity. The arguments of this function may be the signal-to-noise ratio, the dimension of the representation space, boundary non-linearity and/or database size. In a first approach, we consider only $g(.)$ function's variations according to m (database size) axis: $g(m)$. We suppose that our database is free of any incorrect or missing information.
On the basis on $g(m)$, a complexity indicator is defined as follow:

$$Q_i(m) = \frac{g_i(m)}{m} \ , m \geq 1, g_i(m) \geq 0 \tag{2}$$

We expect that for the same problem, as we enhance m, the problem seems to be less complex: more information reduces problem ambiguity. On the other hand, for problems of different and increasing complexity, Q_i indicator should have a higher value. In order to estimate a task (sub-task) complexity we approximate Q_i indicator and figure the complexity ration out by the proposed method in [14].

In next section we discuss the results of the tests obtained by ZISC complexity estimator and compare them to the results of the others complexity estimators of T-DTS framework.

4 Results and Discussion

In order to evaluate ZISC complexity estimator performance we have used for the range of validation problems mentioned in the work [3]. There are:

1. Specific two-class 2D benchmark problems:

- 2 stripes simply separated by line $X=0$. Each stripe belongs to different class.
- 10 stripes. Each of the class consists 5 stripes. The borders between classes are lines $X=b_i$ *(i = 1,2, ... 9)*

2. Tic-tac-toe endgame classification problems. The aim is to predict whether each of 958 legal endgame boards for tic-tac-toe is won for `x'. This problem is hard for the covering family algorithm, because of multi-overlapping [15] that has a place.
3. Splice-junction DNA Sequences classification problem from Genbank 64.1 (ftp site: genbank.bio.net) has the following description:
   - Number of Instances: 3190
   - Number of Attributes: 62
   - Missing Attribute Values: none
   - Class Distribution:
     1) EI: 767 (25%)
     2) IE: 768 (25%)
     3) Neither: 1655 (50%)

We used T-DTS in decomposition mode supposing that task must be divided into two sub-tasks at least.

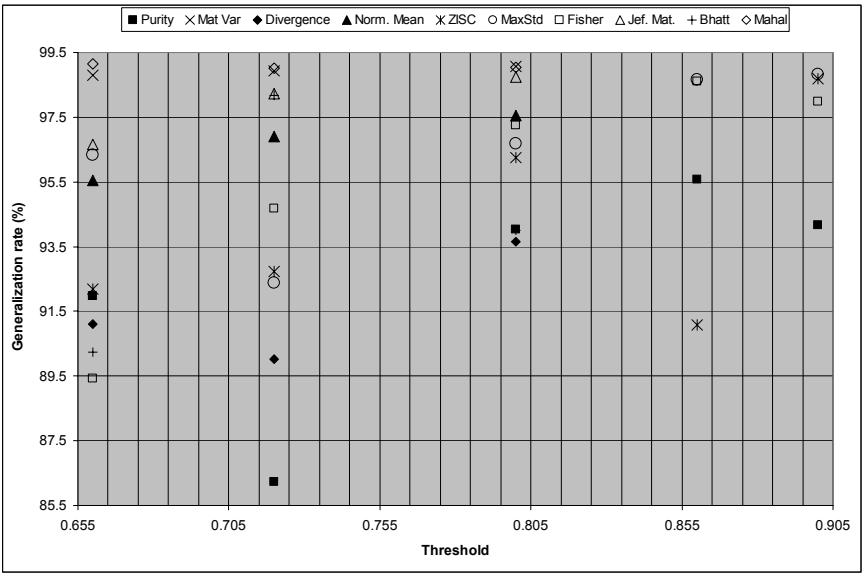


Fig. 3. 2D-benchmark, two classes, 2-stripes. The vectors number - 2000. Learning database 50% (1000 vectors). DU - CN. PU – LVQ.

For each problem and chosen complexity estimation method the optimal decomposition threshold have been adjusted. For ZISC complexity estimator the initializing intra-parameters have been also optimized. On Fig. 3 the X axis represents the decomposition threshold ratio, on Y axis the Generalization rate for the whole range of complexity estimation criteria.

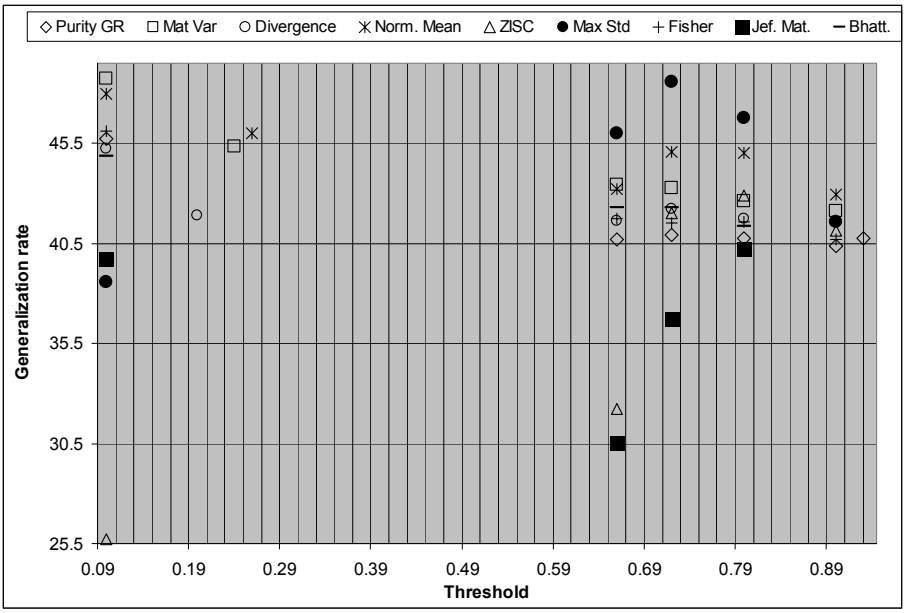


Fig. 4. 2D-benchmark, two classes, 10-stripes. The vectors number - 2000. Learning database 20% (400 vectors). DU - CN. PU – LVQ.

For two class's benchmark within two sub zones, ZISC complexity indicator takes an average position among the other complexity indicators; however for threshold 0.900 the ZISC based complexity estimator can achieve the maximal (e.g. the best) generalization rate attaining approximately 99%. The result obtained for the other case (see Fig. 4) spotlights the same conclusion: ZISC complexity indicator is not the worst method among others. It is relevant to call attention to the fact that if the problem here is already a same 2-D classification dilemma, the complexity has extremely been increased (for 10-stripes variant). Moreover, the learning database's size has been reduced in order to check T-DTS generalization / decomposition ability within the worst-case constraints. In fact, in such worst-case conditions, crop up from conjunction of intrinsic classification complexity and information leakage (emerging from learning database reduced size) it is expectable to face such low generalization rate (around 45%). Finally, the interesting result of Fig.4 highlights the ZISC (and more generally, the "learning") based complexity estimators' main limit related to the requirement of sufficient amount of training data.

For Tic-tac-toe highly overlapping problem (results of Fig. 5), ZISC complexity estimator holds second position. Only Mahalanobis distance based measure of complexity estimation has achieved better generalization rate than ZISC. However, considering the same test with a reduced learning database (including 20% of data), it is interesting to note that, if the leader complexity estimator was Bhattacharyya bound based criterion, again the second rank has been captured by ZISC based complexity estimator.

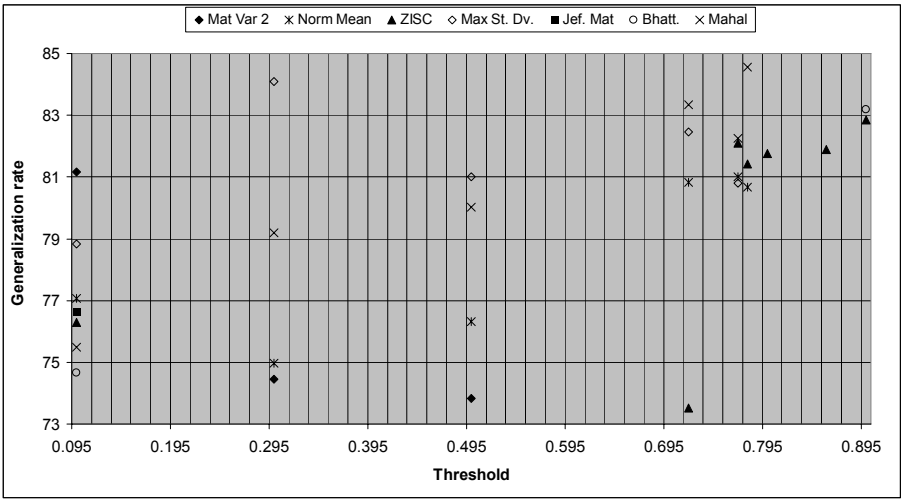


Fig. 5. Tic-tac-toe endgame problem. Number of vectors - 958. Learning database size 50% (479 vectors). Decomposition unit - CN. Processing unit – MLP.

Another important note concerning the Tic-tac-toe end game problem is that in this case, Purity and Divergence based indicators cannot be applied. In fact, based essentially on clusters' borders overlap, these indicators conclude on high complexity of both problem and sub-problems, due to the high overlapping. As result, it is impossible to optimize thresholds for those criteria.

For Splice-junction DNA Sequences classification problem results are given in Fig. 6. One can remark that ZISC based complexity estimator is a leader among the indicators. Inapplicable indicators for this problem are: MaxStd (The sum of maximal standard deviation during decomposition process indicates problem as complex), Normalized mean can not be applied because each vector consists 62 attributes and the complexity ratio which is based on root square deviation indicates problem as a complex. In this regard, there is no surprise why Fisher criterion is the worst one.

Summarizing the comparison of the indicators, we can state that ZISC complexity indicator is matchless among the others. It is so, because ZISC based approach is not sensitive to the number of attributes (of the input vector). For example (see Fig. 6) ZISC complexity estimator for a threshold grater than 0.8 is the only indicator able to provide correct feedback of complexity ratio to T-DTS. However, ZISC requires optimization of the initializing internal parameters and moreover, ZISC estimation complexity procedure is time-costly in comprising to the others.

Concerning the quality side of the obtained results for real-world problems (Tic-tac-toe and DNA classification problem) we can conclude the following:

- for Tic-tac-toe endgame problem, T-DTS can reach 84% of generalization rate for the Mahalanobis and Bhattacharyya criteria's. With ZISC complexity estimator we can achieve maximum of 82% generalization rate. Those results are average in comparison to IBL family algorithms (*Instanced Based Learning*) [15]. However some of IBL algorithms (as the IB3-CI algorithm) lead to better results, they are specially adjusted for this problem.

- for DNA, we have obtained maximal generalization rate of 80% using ZISC estimator and this is the best among all other criteria's. This result corresponds to the results gained in the work [3].

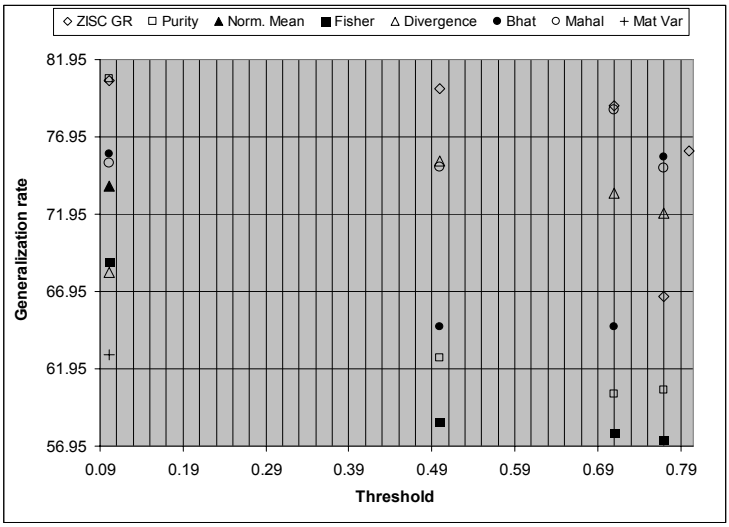


Fig. 6. DNA Sequences problem. M=1900. Learning DB size - 20%. DU - CN. PU – MLP.

Finally, one can remark that the generalization rate of 82%, reached by ZISC based T-DTS for Tic-Tac-Toe end problem, is very close to 84% (obtained by Mahalanobis and Bhattacharyya based T-DTS). So it presents quite high generalization rates. For DNA sequence classification, ZISC based T-DTS, overcome all other complexity based T-DTS. In these last two cases, the representation space has a high dimension. One can conclude that ZISC based T-DST is an appealing candidate for solving high dimension problems. We are conducting other experiments in order to check this hypothesis.

5 Conclusions

Incorporating a new complexity estimator, based on ZISC neuro-computer, the goal was to check the performance of T-DTS regarding other various complexity estimation modules, already implemented in T-DTS framework. We have used the T-DTS framework to solve three classification problems. The proposed complexity estimation criterion has been evaluated using as well benchmark as real-world problems. We have shown that the ZISC based complexity estimator allows the T-DTS based classifier to reach better learning and generalization rates. We have also illustrated that this indicator is matchless for classification tasks relating problems with high dimension feature space. In this case, statistical based complexity indicators fail to work. However, the ZISC based complexity estimator requires sufficient amount of

training data which is a general operation condition (requirement) for all artificial learning based techniques.
The main future perspective of this work is related to T-DTS automatic optimization of and to the extension of database decomposition methods.

References

1. Bouyoucef E., Chebira A., Rybnik M., Madani K.: Multiple Neural Network Model Generator with Complexity Estimation and Self-Organization Abilities: International Scientific Journal of Computing (2005) Vol. 4. Issue 3. 20–29.
2. Rybnik M.: Contribution to the Modeling and the Exploitation of Hybrid Multiple Neural Networks Systems: Application to Intelligent Processing of Information, PhD Thesis, University Paris XII, LISSI (2004)
3. Bouyoucef E.: Contribution à l'étude et la mise en œuvre d'indicateurs quantitatifs et qualitatifs d'estimation de la complexité pour la régulation du processus d'auto organisation d'une structure neuronale modulaire de traitement d'information, PhD Thesis (in French), University Paris XII, LISSI (2006)
4. Budnyk I., Chebira A., Madani K.: Estimating Complexity of Classification Tasks Using Neurocomputers Technology: International Scientific Journal of Computing (2007)
5. Madani M., Rybnik M., Chebira A.: Data Driven Multiple Neural Network Models Generator Based on a Tree-like Scheduler: Lecture Notes in Computer Science edited Mira. J., Prieto A.: Springer Verlag (2003). ISBN 3-540-40210-1, 382-389.
6. Mehmet I. S., Bingul Y., Okan K. E., Classification of Satellite Images by Using Self-organizing Map and Linear Support Vector Machine Decision Tree, 2nd Annual Asian Conference and Exhibition in the field of GIS, (2003).
7. Chi H., Ersoy O.K., Support Vector Machine Decision Trees with Rare Event Detection, International Journal for Smart Engineering System Design (2002), Vol. 4, 225-242.
8. Fukunaga K.: Introduction to statistical pattern recognition, School of Electrical Engineering, Purdue University, Lafayette, Indiana, Academic Press, New York and London, (1972).
9. Edmonds B., What is Complexity? - The philosophy of complexity per se with application to some examples in evolution: Heylighen F. & Aerts D. (Eds. 1999): The Evolution of Complexity, Kluwer, Dordrecht.
10. Feldman D.P., Crutchfield J.P.: Measure of statistical complexity: Why? Phys. Lett.(1998) A 238: 244-252.
11. Budnyk I., Chebira A., Madani K.: ZISC Neural Network Base Indicator for Classification Complexity Estimation, ANNIIP (2008), 38-47.
12. Madani K., Chebira A.: Data Analysis Approach Based on a Neural Networks Data Sets Decomposition and it's Hardware Implementation, PKDD, Lyon, France (2000).
13. Dasarathy B.V. editor (1991) Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques. ISBN 0-8186-8930-7
14. Park. J., Sandberg J. W.: Universal Approximation Using Radial Basis Functions Network, Neural Computation (1991) Volume 3, 246-257.
15. Aha. D. W., Incremental Constructive Induction: An Instance-Based Approach. In Proceedings of the Eight International Workshop on Machine Learning. Morgan Kaufmann (1991), 117-121.

Transient IC Engine Monitoring Under Temperature Changes Using an AANN

Xun Wang[1], George W. Irwin[1], Geoff McCullough[2], Neil McCullough[2] and Uwe Kruger[3]

[1] Intelligent Systems and Control Research Group, Queen's University Belfast BT9 5AH, U.K.
{x.wang, g.irwin}@ee.qub.ac.uk
[2] Internal Combustion Engines Research Group, Queen's University Belfast BT9 5AH, U.K.
{g.mccullough, n.mcdowell}@qub.ac.uk
[3] Department of Electrical Engineering
The Petroleum Institute, PO Box 2533, Abu Dhabi, U.A.E.
ukruger@pi.ac.ae

Abstract. This paper reports on non-linear principal component analysis for fault detection on an internal combustion (IC) engine. An auto-associative neural network (AANN) model is built from transient engine data collected under varying atmospheric conditions. The experimental data used for modelling was collected for two different drive cycles, the Identification Cycle and the New European Drive Cycle. The key issue here is to decide which data should be used for training the neural network to produce good fault detection generalisation under different atmospheric conditions and with a different drive cycle. This is achieved successfully, with the Q monitoring statistic indicating an absence of unwanted false alarms under fault-free operation, along with successful detection of air leaks of varying magnitude in the inlet manifold.

1 Introduction

The specific provisions of more general emissions legislation relating to the detection of faults within an internal combustion engine is commonly known as On-Board Diagnostics (OBD). This details both the component parts of the engine to be tested and at what frequency. This monitoring entails the diagnosis of any fault, which could cause the tailpipe emissions of carbon monoxide (CO), unburned hydrocarbons (HC) and oxides of nitrogen (NO_x) to rise above legislated values.

The automotive industry currently employs a combination of signal and model-based diagnostic techniques for OBD, the latter being based mainly on physical models of the system. However, as the emissions thresholds have reduced in response to increasingly stringent regulation demands, the OBD fault detection thresholds have tightened accordingly, thereby increasing the challenge in engine modelling and monitoring (Stobart, 2003) with over 50% of engine control unit software being currently devoted to OBD. The complexity of physical models will have to increase dramatically if the smaller variations in emissions, constituting a fault under future OBD regulations are to be successfully detected. Further, such models will require extensive on-line validation for each engine and vehicle derivative. Consequently, the cost of developing and validating physical models will increase exponentially. Over

the last decade the mathematical complexity and computational intensity of physical model-based OBD has stimulated research on alternative fault detection and diagnosis approaches suitable for automotive engines (Nyberg, 1999; Grimaldi *et al.*, 2001; Crossman *et al*., 2003; Kimmich *et al.*, 2005.)

Some work has also been done on statistical methods, including principal component analysis (PCA) where a reduced set of statistically independent score variables are generated for process monitoring. Unfortunately, while PCA in its original form is only applicable to linear data, a nonlinear extension in the form of an auto-associative neural network (AANN) (Kramer, 1992) is now available, where the scores are produced in the network bottleneck layer. Because of its conceptual simplicity and close relation to linear PCA, our previous work on diesel engines used an AANN for monitoring during steady-state operation (Antory et al., 2005), including increased sensitivity in detecting minor faults (Wang et al., 2008a) by incorporating additional analysis in the form of the statistical Local Approach.

Nonlinear PCA (NLPCA) has also proved to be effective when applied to experimental data recorded from the air intake system of a gasoline engine during *transient* operation (Wang et al., 2008b). Here the AANN was trained on a modified identification (MI) cycle, specially designed to ensure that the engine speed and throttle position covered the complete operating map at rates similar to those experienced during normal operation. Importantly, the resulting model proved suitable for more general use with the New European Drive Cycle (NEDC), a standardised test for all new model types representing a mixture of urban and motorway driving. In this work also, rather than using the same operational cycle for AANN training and testing, two completely different engine drive cycles will be employed, as detailed later.

The major limitation in all the previous work is the absence of any consideration of the effects of atmospheric changes on the ability of the model-based fault detection to generalise in terms of avoiding false alarms while correctly identifying fault conditions. Thus, the experimental data used for validating the AANN model and the faulty data sets were all recorded under similar atmospheric temperature and pressure conditions to those for the training data. Moreover, none of the measured engine variables previously included in the monitoring model was in fact significantly affected by such atmospheric changes. Any AANN model derived for IC engine monitoring under laboratory conditions, should also be capable of being generalised to a wider range of driving conditions. The aim of the present paper is to address this important deficiency. The practical significance of the results reported later is that the experimental data sets available for neural modelling not only covered two different operational cycles, as required for dynamic monitoring, but were also recorded under different temperature conditions.

Some work on the effect of changes in atmospheric pressure and temperature on the performance of an engine has been reported in the literature, but from the perspective of engine design (Sher, 1984). Here computer code was developed to simulate the engine cycle for the purpose of evaluating an optimal engine design giving the best performance at high altitude conditions. The focus of this paper is data modelling of a modern automotive petrol engine, based on which its fault monitoring under different driving conditions can be achieved. This further significantly extends

the generalisation abilities of the AANN IC engine model described in our previous work.

The paper is organised as follows. An introduction to the experimental petrol IC engine facility, the engine drive cycles and the data collection regime is given next in Section 2. Following a brief review of NLPCA, Section 3 then presents the experimental condition monitoring results. The paper ends with a brief discussion, some conclusions and suggestions for future research.

2 Automotive Engine Tests

This section briefly describes the experimental engine test-bed, followed by detailed explanations of the data collection regime under both normal and faulty operating conditions.

2.1 Engine Test Cell

The target application was a four-cylinder 1.8 litre spark ignition engine, manufactured by Nissan. This engine represents current technology with devices such as variable valve timing, inlet swirl plates, exhaust gas recirculation and a close-coupled catalyst. The engine installation can be seen in .

The engine was installed in a state-of-the-art test facility at Queen's University Belfast. An AC dynamometer with a Ricardo S3000 controller was used to control the engine throughout the simulated transient drive cycles. Sensor signals were recorded using the testcell data acquisition hardware – a Ricardo TaskMaster 500/2000 system, capable of recording up to 32 analogue input channels simultaneously.

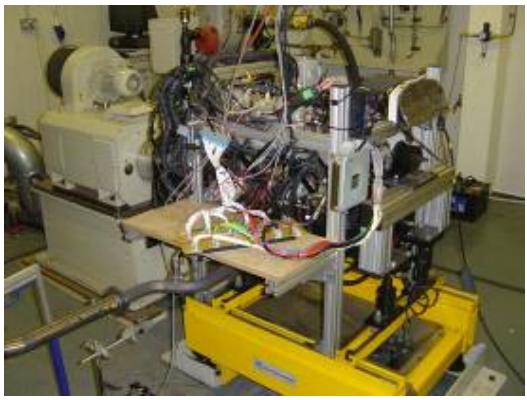


Fig. 1. View of engine test-bed and dynamometer.

The intake subsystem of this engine was investigated. In order to simplify the air intake modelling, the exhaust gas recirculation (EGR) function was disabled. The following five variables were used to analyse this subsystem: crankshaft rotational speed (rev/min), pedal position (%), mass air flow (kg/h), inlet manifold pressure

(bar) and inlet manifold temperature (ºC). Rotational speed and pedal position formed the engine inputs, while the other variables represented the dynamic behaviour of the intake system. Importantly, these variables are all available from current sensors fitted to the standard vehicle and so the modelling and fault detection system for OBD outlined in this paper requires no additional hardware.

In contrast to our previous work on this IC engine where atmospheric conditions were not considered, it will be seen that including the inlet manifold temperature in the data modelling allows atmospheric temperature changes to be incorporated, as shown later in subsection 2.3.

2.2 Atmospheric Changes

The atmospheric temperature and pressure both affect the dynamic performance of an IC engine and so must be accounted for in any model-based OBD scheme. This study is confined to consideration of the variation in atmospheric temperature, since atmospheric pressure control in the engine test cell was not available.

The experimental engine data sets were collected at two different temperature conditions. Normal room temperature (RM) was around 20-25ºC and required no manual intervention. The elevated temperature (ET) condition was controlled at 30-35 ºC by electrical heating of the combustion air. The engine's performance under these two temperature conditions is analysed below, after the two drive cycles have been introduced.

2.3 Drive Cycles

The New European Drive Cycle (NEDC) is used for emissions certification of light duty vehicles in Europe. It is composed of various sections which simulate both urban and motorway driving conditions. The measurement of the resulting exhaust emissions forms one component of the Type Approval test, which is compulsory for any new vehicle model entering the European market. Alternative drive cycles include the FTP 75 used in the USA and the 10-15 Mode Cycle adopted by Japan.

The variation in the engine speed and pedal position inputs produced by the NEDC for a sampling rate of 10Hz are shown in the top two plots of Fig. 2. Note that Fig. 2 shows two experimental data sets, corresponding to the RM (black) and ET (red) temperature conditions respectively. In addition to the five engine variables, the figure also includes the atmospheric temperature. The reason for including the atmospheric temperature is to help visualise the impact of its change on the engine performance. This variable is not to be included in the subsequent engine modelling.

It is clear from Fig. 2 that the NEDC is a highly transient cycle which includes periods of rapidly varying engine speed and pedal position inputs as gear changes are simulated. During vehicle deceleration, when the pedal position is zero, the engine speed is often higher than idle speed. Here the engine is motored through the transmission which contributes to the vehicle's braking requirement. These phases of the NEDC were simulated during testing by supplying a torque input from the motoring dynamometer to maintain the commanded engine speed.

In the example shown in Figure 2, the temperature of the air drawn into the intake manifold in the RM case is increased slightly above the atmospheric temperature due to heat transfer from the warm engine. The average inlet manifold temperature is therefore around 26°C. Conversely, in the ET case, the temperature of the air supplied to the engine is higher than that of surrounding environment and so some heat is lost by convection from the intake manifold, reducing its temperature to around 32°C on average. Assuming the volumetric efficiency of the engine remains the same for a given combination of inputs, the increase in temperature between the RM and ET cases reduces the density of the air, and hence the mass air flow rate, entering the engine by around 2%. The intake manifold pressure was unaffected by the change in atmospheric temperature.

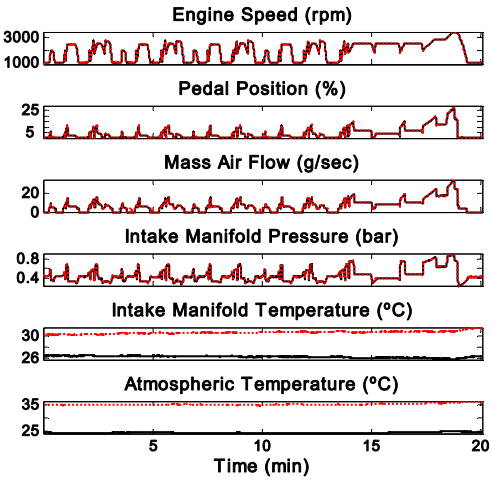


Fig. 2. Engine variables for the NEDC (10Hz sampling) with RM (black) and ET (red) atmospheric temperatures.

While the NEDC is indeed a highly dynamic cycle, and is purported to be representative of typical vehicle use, Fig. 4 shows that the engine control inputs do not cover the whole operational map. For example, engine speed does not exceed 3500rpm while the throttle pedal position is less than 15% for most of the cycle, briefly reaching a maximum of around 28% at 3500rpm during the motorway driving phase. This represents only 55% of the peak torque available at that engine speed. This analysis implies that data from the NEDC would be unsuitable for training an AANN model. Inaccurate predictions would be produced and false alarms generated, when the IC engine is operated outside the regions of the map accessed during training.

An alternative drive cycle, the Kimmich identification cycle (KI cycle), was also considered to provide better coverage of the operating region (Kimmich et al., 2005). However, although this cycle indeed produces a broader range of engine speeds and throttle positions, the *rate* at which these variables change is very significantly lower than that experienced in real driving. The NEDC by contrast is more dynamic in nature and does produce more realistic transients. A modified identification (MI) cycle was therefore designed to generate suitable data for transient modelling. This

was developed by examining the sections of the NEDC where the engine speed was undergoing the greatest transients, which occurs during accelerations in 1st gear. The timescale of the KI cycle was then reduced by a factor of 5.7 such that the rate of change of engine speed matched that of the most transient section of the NEDC. The resulting MI drive cycle therefore combines the benefits of both the KI one and the NEDC as it produces good coverage of the engine operating map, while also simulating realistic dynamics. The engine inputs for the MI cycle and corresponding outputs for the same two cell temperature conditions as before are shown in Figure 3. The same observation can be made as for the NEDC responses in Fig. 2 viz. that only the inlet manifold temperature has been affected by the atmospheric temperature change.

Comparing the operating maps of the MI and NEDC cycles in Fig. 4, it is clear that any model built on engine data from the former cycle will better represent a much wider range of IC engine operation, as required.

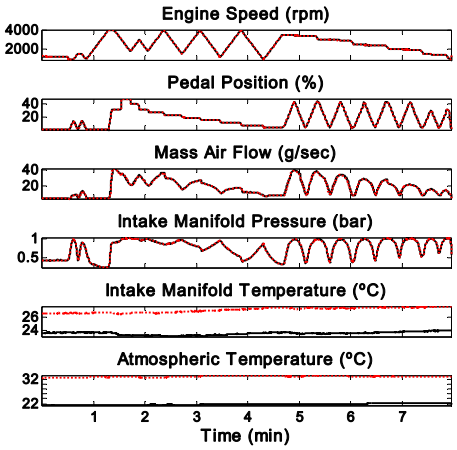


Fig. 3. Engine variables for the modified identification (MI) cycle with RM (black) and ET (red) atmospheric temperatures.

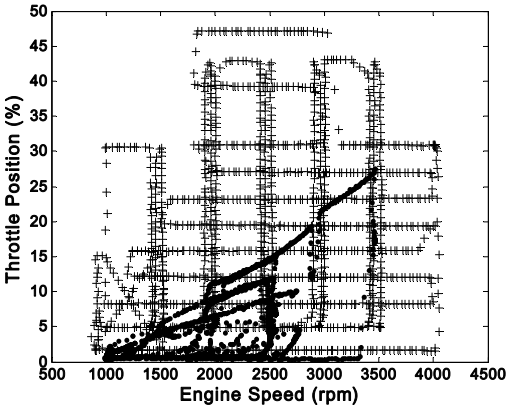


Fig. 4. Comparison of engine operating maps for the NEDC (circles) and MI (crosses) drive cycles.

2.4 Data Collection

The MI cycle last about 8 minutes, providing 4785 data points. This cycle was repeated three times for both RM and ET conditions without introducing any engine fault. This generates two sets of engine data for model building, the third being reserved to validate the model. A further single set of NEDC data was recorded during normal 'fault-free' operation under both RM and ET conditions, in order to assess the generalisation capability of the trained AANN model on unseen data.

2.5 Air Leak Fault

In this investigation, the faulty conditions took the form of air leaks of varying sizes in the intake manifold. This is indicative of a process, rather than sensor fault, and is representative of a leakage past a gasket or fitting between the throttle plate and the intake valve. A minor air leak potentially may not be noticeable to the driver. Nevertheless, when a fault of this type occurs, the driver would adjust the throttle pedal until the desired torque is achieved. Thus, in this fault scenario it is imperative to preserve the values of the engine speed and pedal position between the fault-free and faulty conditions. The fault was introduced by drilling a hole into a bolt which was then screwed into the inlet manifold of the engine downstream of the throttle plate. A total of four such bolts were used: a solid one to produce the fault-free condition, and three others with 2mm, 4mm, and 6mm diameter holes to introduce faults of differing magnitude. Data representing all three faulty conditions were collected for both the MI cycle and the NEDC.

3 Nonlinear PCA

The AANN shown in Fig. 5 is a special neural network architecture with 3 hidden layers, referred to as the mapping, bottle-neck, and demapping layers respectively.

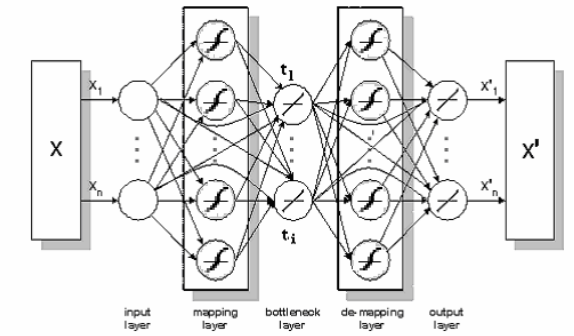


Fig. 5. Auto associative neural network architecture for nonlinear PCA.

The AANN represents an identity mapping for a given set of *n* variables, such that the network inputs and outputs are identical. A hyperbolic tangent function was used

as the activation function in the mapping and de-mapping layers, while the other two layers contained linear activation functions. Note also the presence of direct links from the inputs to the bottleneck layer and from the bottleneck layer to the outputs.

This network is regarded as a nonlinear version of PCA because of its similarity in producing 'scores', normally fewer in number than the original variables from a given process. The scores generated by linear PCA are based on the linear relationships among the physical variables, and can effectively represent the variance in these variables. The scores can then be used for reconstructing the original variables or for monitoring any unknown data from the same process. In the case of NLPCA, the $i < n$ nodes in the bottleneck layer represent the nonlinear scores, $\mathbf{t}_1, \cdots, \mathbf{t}_i$. They are obtained by capturing the nonlinear relationship between the engine variables in the input layer. The scores are then able to reconstruct the original variables by passing them through the de-mapping layer to the output layer. If only linear relationships exist between the engine variables, the scores from such a NLPCA model should in theory be the same as those from a PCA one.

The mathematical description of the identity mapping can be described in two parts. The scores are produced in Figure 5 by the mapping layer that constitutes a nonlinear transformation on the inputs. Thus, for the k^{th} score:

$$\mathbf{t}_k = G_k(\mathbf{x}) \tag{1}$$

The variables at the output layer $\mathbf{x}'$ can then be obtained using:

$$x'_j = H_j\begin{pmatrix} t_1 & t_2 & \cdots & t_i \end{pmatrix}^T \tag{2}$$

The AANN network parameters are trained by minimising the cost function:

$$J = \sum_{j=1}^{n} (x_j - x'_j)^2 \tag{3}$$

When the NLPCA has been trained from fault-free data, a Q statistic can be calculated based on the model prediction error as

$$Q = \mathbf{e}^T\mathbf{e} \tag{4}$$

Here $\mathbf{e}$ refers to the difference between the model prediction vector and its measured value for one sample. Q follows a central χ^2 distribution and appropriate confidence limits can be estimated as discussed in Jackson, 1991. The values of the Q statistic from the training data were used to calculate 95% and 99% confidence limits, which are subsequently used as benchmarks for monitoring unknown data from the engine.

4 Results

Choosing the best training set from the fault-free MI cycle data sets to use for AANN training required careful consideration. Three data sets had been generated for each of the two atmospheric temperature conditions. Moreover, the atmospheric temperature varied by 2 or 3 degrees during the three repetitions of the MI drive cycle. Although minor, this variation will have an impact on the generalisation of any engine model. The principle used in selecting training data here was to choose the fault-free data with the widest possible range of temperature conditions, while leaving adequate fault-free data for validation. Closer inspection showed that data sets one and three, under either RM or ET temperature conditions, provided a much wider range of temperature coverage than any alternative pairing. The training data used in this work therefore consisted of four MI cycles corresponding to data sets one and three under each of RM and ET temperature conditions. The second data sets, collected under both RM and ET temperature conditions were then employed for validation purposes.

Subsection 4.1 provides details of how the model was trained, along with its validation on the MI cycle. The generalisation capability of the resulting engine model is assessed in section 4.2, while its ability to detect air leak faults with the engine operating under both the MI and NEDC cycles is presented in subsections 4.3 and 4.4 respectively.

4.1 Training and Validation

The NLPCA built on the training data had a 5-10-4-10-5 structure, with 10 nodes in the mapping and de-mapping layers and 4 in the bottle-neck layer. Having trained the model it was subsequently validated using a new set of data recorded during a fault-free MI cycle. The performance of the model during this training and validation process is illustrated by the variation in the Q statistic shown in Figure 6.

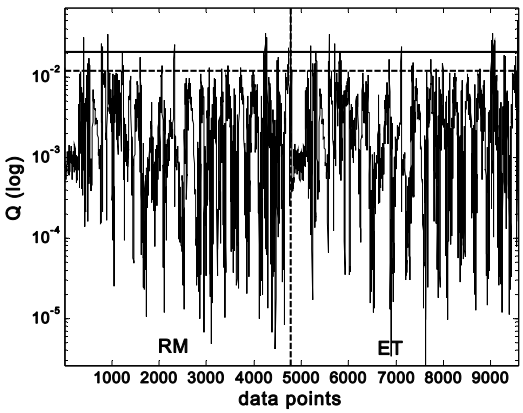


Fig. 6. Q statistic variation on fault-free MI cycle data collected under two atmospheric temperature conditions for model validation.

Here the upper limit represents the 99% confidence level, whereas the lower one is for a 95% threshold, both of which were derived from the Q statistic of the AANN training data. The resulting numbers of violations shown for data recorded under both the RM and ET conditions are statistically acceptable, confirming the modelling validity of the trained AANN.

4.2 Generalisation

When a model is used in practice, the new operational inputs often take a form previously unseen by the model during its training. This is particularly the case with automotive IC engines, which impose a stringent requirement for generalisation if the results are to be of practical significance. This aspect of the NLPCA model was therefore challenged by comparing the measured and predicted values of both mass air flow and manifold air pressure produced by the NEDC with the engine operating under fault-free conditions. The excellent generalisation capability of the trained AANN is supported by Fig. 7, which shows the low numbers of violations of the confidence limits by the Q statistic under both temperature conditions. This reveals that there is little difference between the measured engine data and the corresponding predictions. This is an important finding as the NEDC engine inputs of speed and pedal position vary at rates ranging from steady-state up to the highly transient conditions found during 1st gear accelerations and motored deceleration phases. Moreover, the atmospheric conditions when these data sets were collected also differed from those of the training data.

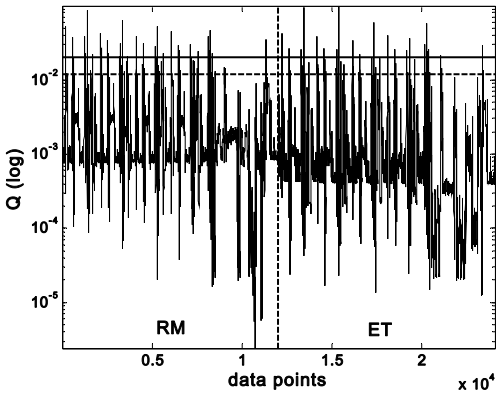


Fig. 7. Q statistic variation on fault-free NEDC and two temperature conditions for model generalisation.

4.3 Fault Detection on MI Data

This section assesses the AANN model's ability to detect a fault produced by running the MI cycle in the presence of a 2mm, 4mm, and 6mm air leak on the inlet manifold. Fig. 8 and Fig. 9 show the variation in the corresponding Q statistics for data recorded under RM and ET conditions respectively. It can be seen that in either case the

number of violations of the confidence limits naturally grows as the magnitude of the fault increases. Since the 2mm air leak does not have a significant impact on the engine performance, its Q statistic does not produce as many obvious violations to the 99% confidence limits as for the larger two air leaks. Even so, this minor fault can still be detected by referencing to the 95% confidence limit.

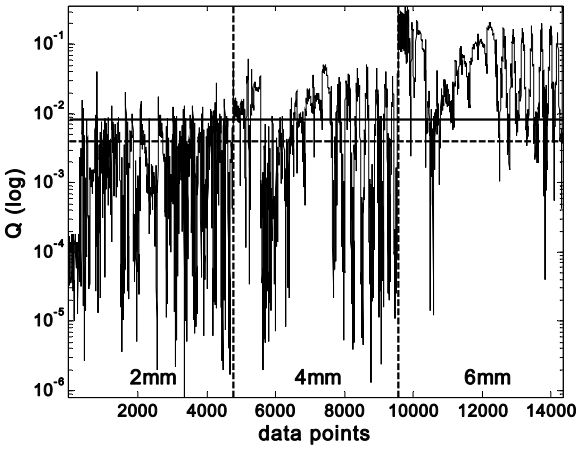


Fig. 8. Monitoring 2mm/4mm/6mm air leak faults for the MI cycle at the RM temperature condition.

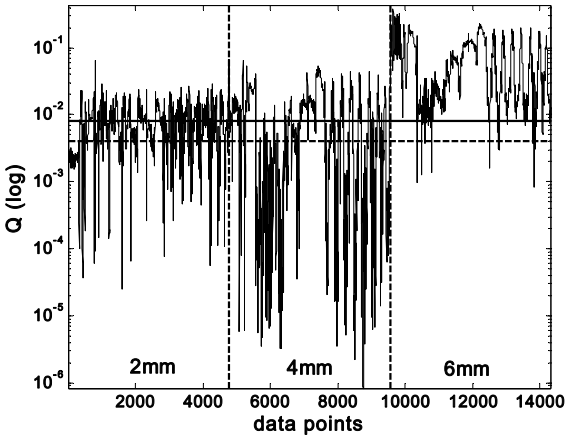


Fig. 9. Monitoring 2mm/4mm/6mm air leak faults for the MI cycle at the ET temperature condition.

It should be noted that there are certain regions in the faulty data set where the impact of the air leak fault may not be apparent in the monitoring statistic. This is expected, as this fault would not affect the engine when it is operating at high throttle openings. Under such circumstances, the manifold pressure is close to, or equal to, the atmospheric pressure. Consequently, the pressure difference across the leakage orifice is small and the flow rate of air through it is therefore negligible.

4.4 Fault Detection on NEDC

The variations in Q statistic for the AANN model shown in Fig. 10 and Fig. 11 cover the three fault conditions for the NEDC under both RM and ET temperature conditions respectively. Pleasingly, these results confirm successful detection of all three air leak faults, despite the model having being trained using the substantially different MI identification cycle.

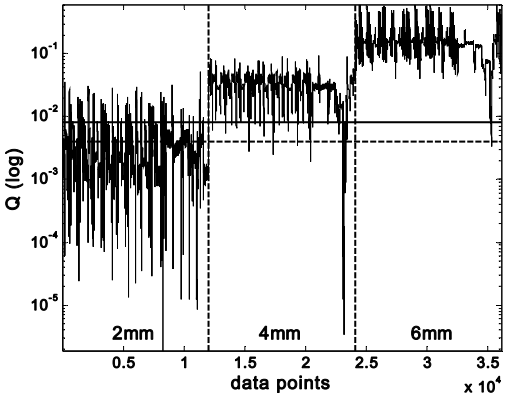


Fig. 10. Monitoring 2mm/4mm/6mm air leak faults for the NEDC cycle at the RM temperature condition.

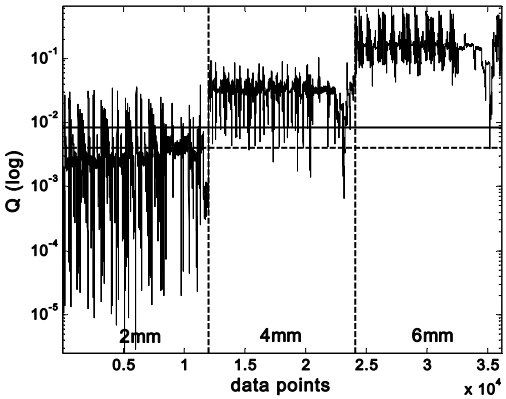


Fig. 11. Monitoring 2mm/4mm/6mm air leak faults for the NEDC cycle at the ET temperature condition.

5 Conclusions

This paper showed the capability of an AANN in both modelling and air-leak fault detection for an automotive gasoline IC engine. The modelling data was derived for two different transient drive cycles under different atmospheric temperature conditions. The model trained using MI cycle data sets that were recorded under all

available atmospheric temperature conditions produced the best generalisation to measurements from the unseen NEDC cycle. There was an absence of unwanted false alarms under fault-free conditions, and successful detection of air leaks of varying magnitude in the inlet manifold.

Acknowledgements

The authors are grateful to the U.K. Engineering and Physical Science Research Council (Grant No. EP/C005457) for their financial support.

References

1. Antory, D., U. Kruger, G.W. Irwin and G. McCullough 2005. Fault diagnosis in internal combustion engines using nonlinear multivariate statistics. *Proc Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 219(4), 243-258.
2. Crossman, J.A., H. Guo, Y.K. Murphey and J. Cardillo 2003. Automotive signal fault diagnostics – part I: signal fault analysis, signal segmentation, feature extraction and quasi-optimal feature selection. *IEEE Transactions on Vehicular Technology* 52(4), 1063-1075.
3. Grimaldi, C.N. and F. Mariani 2001. OBD Engine Fault Detection Using a Neural Approach. *SAE Paper No. 2001-01-0559.*
4. Jackson, J.E. 1991. *A Users Guide to Principal Components.* Wiley Series in Probability and Mathematical Statistics. John Wiley, New York.
5. Kimmich, F., A. Schwarte and R. Isermann 2005. Fault detection for modern diesel engines using signal- and process model-based methods. *Control Engineering Practice* 13, 189-203.
6. Kramer, M.A. 1992. Autoassociative neural networks. *Computers and Chemical Engineering* 16(4), 313-328.
7. Nyberg, M. 1999. Model Based Diagnosis of Both Sensor Faults and Leakage in the Air-Intake System of an SI Engine. *SAE Paper No. 1999-01-0860.*
8. Sher, E. 1984. Effect of atmospheric conditions on the performance of an air-borne two-stroke spark-ignition engine. *Proc Institution of Mechanical Engineers, Part D: Transport Engineering.* 198(15), 239-251.
9. Stobart, R. 2003. Control oriented models for exhaust gas aftertreatment; A Review and Prospects, SAE Paper No. 2003-01-1004.
10. Wang, X., U. Kruger, G.W. Irwin, G. McCullough and N. McDowell 2008a. Nonlinear PCA with the local approach for diesel engine fault detection and diagnosis, *IEEE Trans. Control Systems Technology* 16 (1), 122-129
11. Wang, X., G.W. Irwin, G. McCullough, N. McDowell and U. Kruger 2008b. Application of nonlinear dynamic PCA to automotive engine modelling and fault monitoring, *Control Engineering*

End Milling: A Neural Approach for Defining Cutting Conditions

Orlando Duran, Nibaldo Rodriguez and Luiz Airton Consalter

Pontificia Universidad Catolica de Valparaiso, Av. Brasil 2241, Chile
FEAR Universidade de Passo Fundo, P. Fundo, RS, Brazil
{nibaldo.rodriguez,orlando.duran}@ucv.cl, lac@upf.br

Abstract. The purpose of this paper is to present a new adaptive solution based on a feed forward neural network (FNN) in order to improve the task of selecting cutting conditions for milling operations. From a set of inputs parameters, such as work material, its mechanical properties, and the type of cutting tool, the system suggests feed rate and cutting speed values. The four main issues related to the neural network-based techniques, namely, the selection of a proper topology of the neural network, the input representation, the training method and the output format are discussed. The proposed network was trained using a set of inputs parameters provided by cutting operations manuals and tool manufacturers catalogues. Some tests and results show that adaptative solution proposed yields performance improvements. Finally, future work and potential applications are outlined.

1 Introduction

Process planning is a function that establishes a set of manufacturing operations and their sequence, and specifies the appropriate resources (machines, cutting tools, fixtures, inspection instruments, etc.) and process parameters in order to convert a blank to a finished component expressed by an engineering drawing and other technical information. The use of computer tools to assist the process plan generation was firstly reported by Niebel [2]. From that work, many other researchers have explored the use of computer systems to obtain a coherent process plan in an automated manner. Recently, the term CAPP (computer aided process planning) appears frequently in the technical literature and can be considered as one of the most active fields of research in manufacturing. In computer integrated manufacturing environments, CAPP can be considered as the link between the CAD phase and CAM. Many researchers have developed solutions to close the gap between the design phase and the generation of manufacturing instructions and/or machine control code. A primer reference in CAPP is Chang [1], who classifies CAPP approaches into two categories, i.e. variant and generative. The variant approach usually incorporates the use of group technology paradigm to recover the most suitable process plan that corresponds to the most similar workpiece to that is being planned. This first approach usually lacks in flexibility and does not consider relationship between features of workpiece. On the other hand, generative process planning generates in an automated manner, a brand new process plan, only based on experience and technical knowledge. According to Chang and Chang [3], most generative CAPP

lack in learning ability for environmental changes. Because of that reason, recent researches have being focused on integrating variant and generative CAPP with Expert Systems-based techniques. A number of approaches have been reported in literature to solve the problems occurring in integrating process planning and other computer aided manufacturing applications. Leung [4] published an extensive literature review on CAPP. Other significant number of references was reported by Marri et al. [5]. More recently, Chang and Chang [3] reported artificial intelligence applications for CAPP implementations. In an exploratory examination of the CAPP literature it is easy to conclude that ANN have been intensively used to solve problems such as tool selection, cutting conditions definition, sequencing of operations, etc. Unlike the most published applications of ANN in process planning, where neural networks are used as a means to create optimization models, the proposed approach involves three key differences: neural networks are capable of storing knowledge in a distributed manner, neural networks are capable of learning to recognize relationships between inputs and outputs, while such relationships must be explicitly defined in optimization models, and neural networks are capable of generalizing (i.e., giving a ”closest-fit” answer) when presented with data not used in deriving the relationships learned. This paper presents a proposal of artificial neural network for determining cutting parameters for milling operations. The subsequent content is organized as follows: The second section discusses the process of definition of cutting conditions and the use of Artificial Intelligence techniques for this purpose; the third section presents the suggested approach, specifically it discusses the structure of the developed networks, the training data sets, and the details of the training process. Furthermore, some aspects of the obtained output of the developed networks are presented. Finally, in the last section some conclusions and future works are drawn.

2 Artificial Intelligence and Determination of Cutting Parameters

A workpiece is composed by a set of surfaces or features. These surfaces are obtained by a sequence of machining operations, such as turning, milling, boring and so on. For each one of these features process planners must select adequate tools and optimal cutting parameters. The choice at this stage may be tentative and has to be governed by experience, intuition and based on information gathered in machining handbooks. Mainly, this step of process planning considers the selection of the following parameters:

- The cutting speed (vc) and the rotational speed of the part or of the tool (N).
- The feed rate (fn) or the feed speed in translation of the machine elements (vf).
- The depth of cut (ap) or engagement determining the width of the material to be removed.

Despite the fact that such machining parameters are calculated according to practical values found in handbooks or from experience, they have to be updated or refined to adapt the values to match a specific situation for extracting the best performance of the cutting resources. Influence diagrams have been developed for representing complex decision problems based on incomplete and uncertain information from a variety of

sources. Nestler and Shulz [8] presented a simple example of an influence diagram for machining optimizations and discussed their use in optimization of cutting conditions (Fig1).

Data from machining handbooks were separated into different types of machining process, such as turning, drilling, boring, end milling, etc. and classified according to the workpiece material. As it is well known, workpiece materials are classified in various groups covering a wide range of materials according to their hardness: ferrous, non-ferrous, etc. The machining handbooks provide the machining parameters for different tool-work-piece combinations. A comprehensive review of the information obtained from the literature, and from industry, has indicated that the recommended cutting speeds and feed rates for any machining operation may vary considerably [6]. In addition to the proper selection of cutting speeds and feed rates, the optimum condition depends on variables such as part configuration, condition of the machine and fixturing, tolerances and surface quality. Because the effects of these variables on tool life are not always precisely known, it becomes difficult to recommend optimum conditions for a machining operation. Therefore, the recommendations presented in the machining handbooks are nominal ones and should be adjusted by a certain order of processing approach [7].

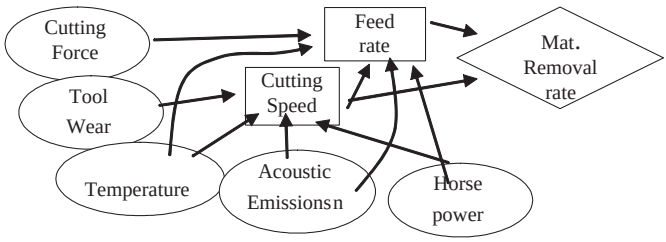


Fig. 1. An influence diagram for determining machining parameters.

Among the applications to determine cutting conditions using an intelligent approach, one must distinguish the systems that perform the task in an off-line mode and the systems that operate in an on-line mode. Off-line calculation of cutting conditions consists in defining all the necessary parameters for execute a workpiece before the process is initiated. According Nestler and Shulz [8], at present, neural networks are especially used to solve sophisticated problems such as determining and modeling correlations between input and output parameters. Hashmi et al. [7] points to other direction to use AI tools for defining cutting conditions, the use of fuzzy logic models for representing knowledge extracted from catalogues or handbooks. According Hashmi et al. [7] fuzzy logic strategy can simulate an operator's 'experience and expertise' in decision-making process to facilitate the operator to select drilling parameters from expert database which can be incorporated in computerized automated systems. On the other hand, the on-line approach corresponds to an automated attempt to adapt and optimize the machining parameters based on sensor information on machining responses in real time. One of the main differences between off-line and on-line determination of cutting conditions is the need of information of temperature and acoustic emissions.

Table 1. Subset of training data.

Operation type	Work Mat Code	Hardness HB	Mill type	Vc m/min	Vf mm/min
1	30,22	90	1	1000	1910,8
2	30,22	90	1	1100	4904,4
3	30,22	90	1	1250	26871,0
4	30,22	90	1	130	4968,1
1	1,10	125	1	155	288,0
2	1,10	125	1	200	891,7
3	1,10	125	1	375	8061,3
4	1,10	125	1	690	26369,4
1	1,10	125	3	145	450,2
2	1,10	125	3	160	1452,2
1	1,20	150	1	135	257,0
1	7,10	150	1	135	257,9

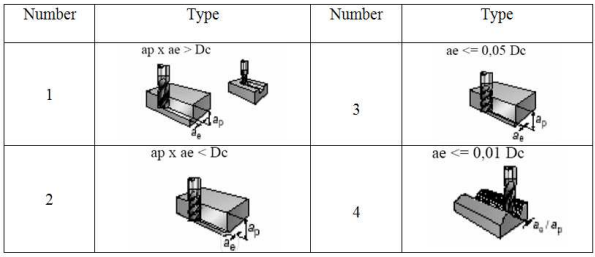


Fig. 2. Types of operations considered in the experiment.

Hence any on-line intelligent system has to be integrated with a sensing device system for extracting in real time conditions of cutting processes. AI techniques allows modeling the information coming from the sensors systems and optimizing machining parameters by learning from machining events such as tool wear, machine breakdowns and other failures.

3 Neural Network Model

Several papers recommend that feed forward multilayer backpropagation nets with one or two hidden layers with 10 to 30 neurons each are appropriate for handling cutting data selection problems [8]. The approach suggested here is to use two neural networks one for the selection of cutting speed and the second one for the selection of the feed speed. Both networks use the same set of inputs. Therefore, the same training and testing sets were used for each one of the nets. There is just one difference between the two training sets: the second network uses the set of training data plus the correspondent cutting speed values. Thus, the propose here is to link both networks to work in an integrated manner to produce cutting parameters from the same set of input data, namely, workpiece material and type of milling operation and cutter. Most of the information used in the preparation of learning and testing data was extracted from a

Table 2. Alternative Configurations tested and their performance indicators.

ID of ANN cfg.	Num.of hidden layers	Num. of layer 1	Num. of layer 2	Num. of layer 3	MAD (mm/min)	GF %
1	3	10	10	10	31,0	100,00
5	2	7	7	0	18,5	90,91
2	3	11	11	11	11,3	85,46
4	3	6	7	8	13,0	83,64
3	2	8	8	0	1,3	83,64
16	2	10	10	0	23,5	80,00
6	2	9	10	0	30,1	78,18
13	2	6	6	0	47,4	74,55
7	2	7	8	0	31,8	74,55
9	3	6	6	6	95,4	72,73
19	2	11	11	0	15,3	72,73
14	3	7	7	7	14,2	72,73
10	3	9	10	11	25,3	69,09
11	3	8	9	10	10,8	69,09
12	3	9	9	9	8,1	69,09
17	2	6	7	0	46,1	67,27
20	2	9	9	0	47,2	65,46
15	2	8	9	0	3,2	63,64
8	3	7	8	9	5,0	60,00
18	3	8	8	8	34,5	58,18

Table 3. Configuration parameters selected for each network.

Network ID	Output parameter	Num. of Inputs	Num. of Hidden Layers	Num. of Neurons Lay.1	Num. of Neurons Lay.2	Num. of Neurons Lay.3
1	Vc	4	3	7 sig	7 sig	7 lin
2	Vf	5	2	8 sig	8 sig	-

Sandvik Coromant Catalog. The collected information is in the form of tables, which show the recommended cutting conditions for different types and geometries of cutters and materials of workpieces depending on factors affecting machinability. As the original information on cutting conditions is given in the form of intervals representations, the midpoints of such intervals were used as the representative values to construct the training and testing data sets. In addition, two machining specialists refined these data. The specialist adapted the data sets assuming a potentially real situation and a given and known machine tool. By doing so, it will be feasible to incorporate empirical knowledge to the training process. It was considered for this experiment four types of operations, as shown in (Fig.2).

The selected input parameters were: milling operation type, workpiece material, workpiece material hardness and type of mill. Table 1 shows a subset of the used training data. As can be noticed in table 2, the desired output data are distributed in a wide interval. This is due to the large number of different types of workpiece materials. This situation may be seen as a rather complex problem to be handled by any neural network. This situation leads the authors to choose the strategy of two networks, the first

Table 4. Percent error according to the operation type.

Operation type	network 1 %	network 2 %
1	1,44	15,11
2	3,27	24,76
3	2,06	1,91
4	0,44	35,34

Table 5. Percent error according to the workpiece material.

Material type	network 1 (%)	network 2 (%)
High Alloy Steel	1,37	8,38
Hardened Steel	3,09	18,26
Titanium alloys	0,91	24,11
Aluminium alloys	0,25	1,92

with a single output to estimate the cutting speed and the second one to estimate the feed speed. The first column represents the operations type, namely, the relation between the depth of cut (ap) and cutting width (ae), as shown in Figure 2. The second and the third columns in table 1 represent workpiece material and its Brinell hardness. Different workpieces materials were represented using the Coromant Material Code. In a similar way, the type of mill was represented as an integer that regards the material and geometrical configuration of the milling tool. In the experiment four types of commercial end mills were considered. As it was commented previously, network designers have to test several configurations, including different numbers of hidden layers and different number of neurons in each one of the hidden layers. There can be any number of hidden layers in a neural network. In common use most neural networks will have only one hidden layer. It is very rare for a neural network to have more than two hidden layers. The number of neurons for the hidden layer(s) depends on the complexity of the problem and should be set empirically. With too few neurons the network may not converge in training, whilst with too many hidden-layer neurons the network starts to lose generalization ability [9]. In this work, we compare the performance of networks with two and three hidden layers and the number of neurons in each layer ranging from 5 to 15. Designer selects the network configuration that presents the minimum error and the fastest rate of convergence. A series of alternative configurations have been tested to determine the proper configuration of both networks. Several tests were conducted varying the number of hidden layers, different numbers of neurons for each layer and different transitions functions. The learning algorithm adopted in all these tests was the backpropagation algorithm with momentum. As it is usual in the approaches using this kind of networks, the set of data was divided into two subsets: the first used as the training data set contains 138 patterns and the second, used as a validation set to evaluate the responses of the net to unseen information, contains 55 patterns. Through the use of these two sets, networks parameters can be adjusted and the generalization ability can be evaluated. To select the most appropriate configuration two traditional performance indicators were used. We refer to the medium absolute deviation (MAD),

which was computed in two phases, during the training process and the testing process. The deviations were calculated as the difference between the actual speed and the estimated speed, both for the training and testing sets. The second performance indicator that was used to evaluate the generalization capacity of the tested configurations is the Generalization Factor (GF), defined by Eq. (1).

$$GF = \frac{k}{n} * 100 \tag{1}$$

Where n is the number of patterns that compose the validation set and k is the number of such patterns estimated with an error less than 2 % (this value having been fixed as a threshold level). Table 2 summarizes the results obtained from an alternative configuration of neural networks for estimating feed speed (network number two). It is quite evident that the configuration number 1 (with three hidden layers) showed better generalization performance since GF is 100%. However, the MAD seemed to be a little high from the operational point of view. If one considers an error of 31 mm/min in the estimated feed speed, this may lead to undesirable or inappropriate time estimations and tool life expectations significantly overestimated. Thus the selected configuration was number 3, in Table 2 (with two hidden layers), which MAD is 1,3 mm/min, considered as acceptable from the operational point of view. Moreover, the generalization factor of near 85% seems to be adequate, if one consider that the training set takes into account a wide variety of types of milling tools and workpiece materials. The same type of analysis was conducted to obtain the architecture of the first ANN that performs the estimation of the cutting speed. Table 3 shows the selected configuration parameters for both networks. Figure 3 (left) shows the net output and expected cutting speed values obtained after the network was entirely trained and a comparison between the original testing data and the parameters estimated by the neural network (Fig.3 right). For the selection of feed speed a single layer network with one hidden layer was trained using the same training data set used for training the network that selects the cutting speed. Figure 4 (left) shows the convergence of the output mean error for the network that was trained for selecting the cutting speed, where it can be noticed the low number of epochs needed for attain the minimum required error. Figure 4 (right) shows the convergence of the output mean error for the network during the training process of the network for selecting the cutting speed. Again, the low number of epochs needed for attain the minimum required error can be noticed. Figure 5 (left) shows the net output and expected feed speed values obtained after the network was entirely trained and on the right side of the Figure 5 a comparison between the original testing data and the parameters estimated by the neural network is shown. To evaluate the performance of the two developed networks, an additional test set was prepared for simulation and comparisons ends. The results of the simulation tests were classified according different criteria. Table 4 shows the performance in terms of percent error of the two both networks according to the operation type (refer Figure 1). Table 5 presents the results of the test grouped according to the material workpiece. Finally, table 6 presents the results obtained by the two networks according the hardness of the workpiece. As can be appreciated, from table 5 and 6, network 1 presented the best results with errors within the 3%. As it can be observed in the tables shown above, the approach presents different performance between network 1 and network 2, i.e. network 1 performs better estima-

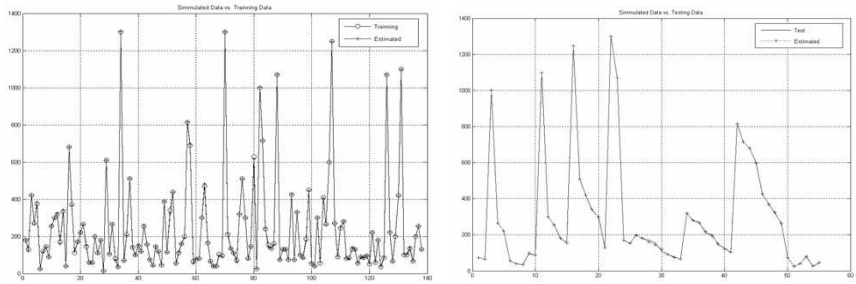


Fig. 3. Comparison between the train data set and the output produced by the trained network(left) and between by the test data set and the output of the trained network (right).

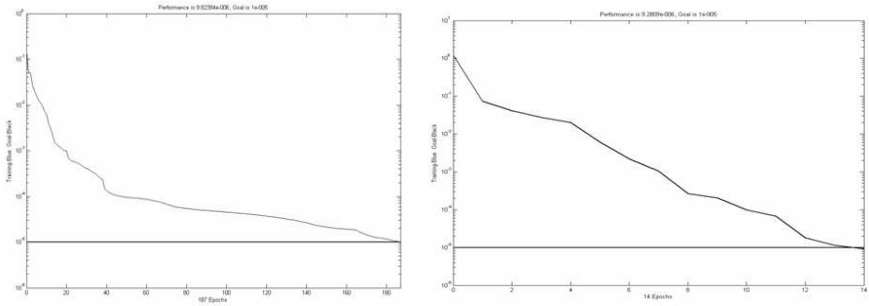


Fig. 4. Convergence during the process of training the first network (left) and during the process of training of the second network (right).

tion of the cutting speed than the estimations performed by the network 2 for the feed speed. This is caused, we believe, by the great variability of the recommended values of feed speed among different materials and operations types. The values used in the test set present a mean value of 6000 mm/min approximately with a standard deviation of 9000 mm/min. However, and it can be observed in tables 5 and 6, there are some applications where the networks presented acceptable results (under 10 % of error), i.e. operation type 3 and milling of High alloy steel and aluminum steel materials.

Table 6. Percent error according to the workpiece material hardness.

Material Hardness	network 1 %	network 2 %
350 HB	0,76	18,02
350 HB	2,37	13,70

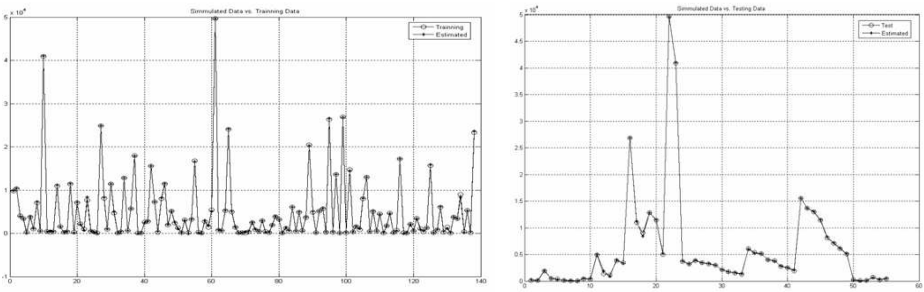


Fig. 5. Comparison between the training data and the data obtained by the trained network for estimating Vf (left) and between test data and the networks output (right side).

4 Conclusions

This paper presented a neural network-based automated approach for cutting conditions selection in milling operations. Two prototype networks were developed. The first network is for selecting cutting speed and the second one for selecting feed speed, both using almost the same set of input parameters. Both neural networks are interrelated, since the output produced by the first network is used as an input in the second one. The developed approach aims at selecting cutting parameters in off-line mode. The main difficulty found in the reported experiments was the fact that the training data set considers a great variety of workpiece materials. That situation leads to a wide interval of cutting conditions affecting convergence mechanism during the training process and the generalization capabilities during the utilization phase. In spite of this fact, the obtained results show that the developed networks have an acceptable performance in simulating cutting conditions selection process, with a generalization performance of about to 85 approximately. Future research points to develop and test new architectures of neural networks to enhance the selection process performance, especially in estimation of feed speed. Also, this work should be extended to other process planning functions such as machine selection, tool and fixture selection and sequence of manufacturing operations.

References

1. Chang T.-C: Expert Process Planning for Manufacturing,Addison Wesley,1990,Massachusetts.
2. Niebel, B.W.: Mechanized Process Selection for Planning new Designs,ASTME Paper 737,1965.
3. Chang P.T., Chang C.: An Integrated Artificial Intelligent Computer-Aided Process Planning System", Int. Journal of Computer Integrated Manufacturing,Vol.13, No.6, pp. 483-497,2000.
4. Leung, H.C.: Annotated Bibliography on Computer Aided Process Planning, International Journal of Advanced Manufacturing Technology, Vol. 12, No. 5, pp. 309-3297, 1996.
5. Marri, H.B., Gunasekaran, A., Grieve, R.J.: Computer Aided Process Planning : a state of art, International Journal of Advanced Manufacturing Technology, Vol. 14, No. 4, pp. 261-268, 1998.

6. Park K.S., Kim, S.H: Artificial Intelligence Approaches to Determination of CNC Machining Parameters in Manufacturing: a Review, Artificial Intelligence in Engineering, Vol. 12, pp. 127-134, 1998.
7. Hashmi, K., L.D.Graham, B. Mills: Fuzzy Logic Based Selection for the drilling Process, Journal of Materials Processing Technology, Vol. 108, pp.55-61, 2000.
8. Nestler A., G.Shulz: Cutting Values Prediction with Neural Networks, Second In- ternational Conference on Industrial Tools ICIT99, 1999.
9. Sukthomya W., J.Tannock: The training of neural networks to model manufacturing processes, Journal of Intelligent Manufacturing, Vol. 6, pp. 39-51, 2005.

NN and Hybrid Strategies for Speech Recognition in Romanian Language

Corneliu-Octavian Dumitru and Inge Gavat

Faculty of Electronics Telecommunications and Information Technology
Politehnica University Bucharest, 313 Splaiul Independetei, Bucharest, Romania
{odumitru, igavat}@lpsv.pub.ro

Abstract. In this paper we present results obtained with learning structures more "human likely" than the very effective and widely used hidden Markov model. Good results were obtained with simple artificial neural networks like the multilayer perceptron or the Kohonen maps. Hybrid structures have proven also their efficiency, the neuro-statistical hybrid applied enhancing the digit recognition rate of the initial HMM. Also fuzzy variants of the MLP and HMM gave good results in the tested tasks of vowel recognition.

1 Introduction

To make a first step on the way to bring near to HSR (Human Speech Recognition) the ASRU (Automatic Speech Recognition and Understanding) performance, it could be important to compare how speech recognition, as the receiving part of the verbal communication, is realised by machines and by humans [8]. Even though the process of verbal communication is a very natural one and thus seems to be a fairly easy for humans, there are several underlying operations that need to be carried out before the communication can be considered successful. The operations designed for ASR try to model what we know about speech and language, or what we assume about it. The models are often much simpler than the reality, and thus are imperfect.

In Fig. 1, a schematic overview of both speech recognition processes, the HSR and ASRU is shown [6]. The first operation in human communication comprises the hearing of the message. We first have to realise that somebody is talking to us and then we have to listen to what he is saying. In ASR, an equivalent process is done, by recording the message with a microphone.

Both systems, human and automatic, need to have some knowledge about the sounds that are used. If one of the talkers uses sounds the other talker does not know, they cannot understand each other. For this we need a vocabulary that is a set of words.

When humans process the message, they extract the meaning out of what was said. They can do so by inferring the meaning from the actual sequence of words that they recognised, because they can directly associate meanings to words and more important to word sequences.

The system however, searches for the word or word sequence that was most likely spoken, given acoustic signal. Even if this was successful, it is still far away from understanding what the meaning of this sequence of words is. Of course, approaches already exist that try to extract the meaning out of recognised word sequences.

Although the procedures in the human and the automatic systems seem to be very likely, the results are very different, the differences being pointed by Lippmann in his well–known study [11]. For complicated tasks, involving sentence recognition, the performance difference is not so surprising and can mainly be explained by the advantage constituted by the natural context use of humans [3]. For simple tasks, like vowel recognition for instance when the context is not advantaging humans, they are indeed better than the machine and that remains surprising [4].

Because of the good human performance in the speech recognition task, it could be interesting to mimic the most important human action in this process, namely the learning and to do it also in a more "human likely" maner by involving neuronal and fuzzy techniques.

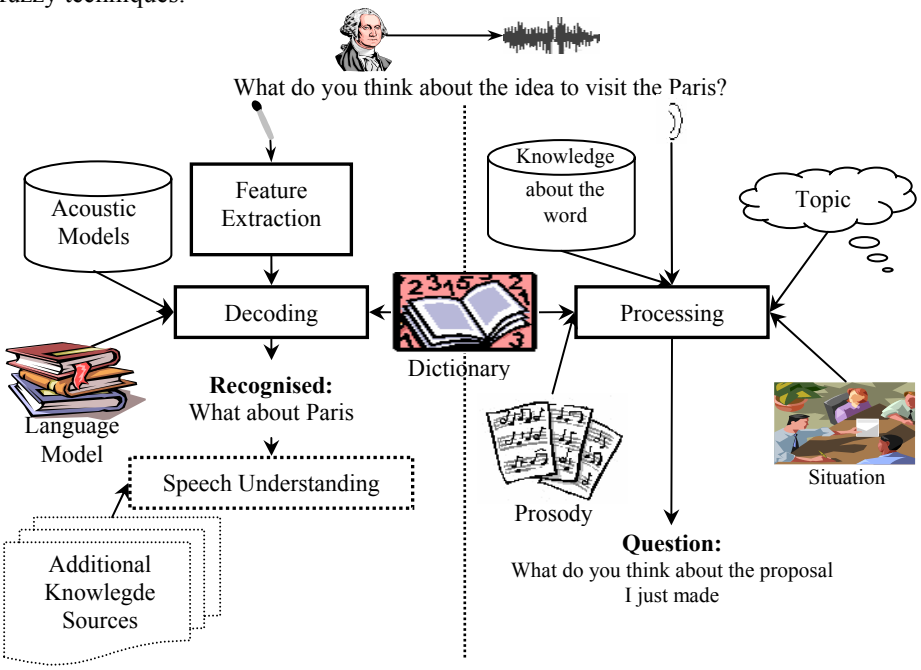


Fig. 1. Overview of a HSR system (right side) and an ASRU system (left side).

The paper has the following structure. Section 2 will describe the basic learning structures applied to build acoustical models. There are investigated classical neural strategies, like the multilayer perceptron (MLP) and the Kohonen maps (KM) but also hybrid strategies like a neuro-statistic model or fuzzy variants of a neural structure, namely the MLP and of a statistical structure, namely the HMM. The Section 3 presents a large variety of conditions into which can be carried out the basic experiments with our Automatic Speech Recognition System for Romanian Language (ASRS_RL) in tasks for vowel recognition and digit recognition. Section 4 concludes the paper.

2 Learning Strategies

Learning is the basic process for humans in acquiring knowledge and was successfully mimicked in technical systems. Artificial Neural Networks generating the neural strategies are a good example. They lead to good recognition performance and can also improve through hybridization the performance of the statistical method based on HMM.

2.1 Neural Strategies

The neural strategies can model very well an important characteristic in human learning, namely associativity: all inputs of the ANN concur to obtain the resulting output, and therefore the recognition performance is high. Due to the fixed number of inputs their flexibility in accommodating time sequences is to low for more complicated recognition tasks, like continuous speech recognition, but appropriate to recognize vowels or digits. Further we will discuss two fundamental artificial networks, namely the MLP and the KM.

Multilayer Perceptron (MLP)

MLP is the most common ANN architecture used for speech recognition. Typically, MLPs have a layered feed–forward architecture, with an input layer, one or more intermediate (hidden) layers, and one output layer. The structure without hidden layer is called Boolean network and is a simple perceptron [13].

Each layer computes a set of linear discriminative functions, followed by a nonlinear function, which is often a sigmoid function.

The numbers of neurons in the hidden layer was experimentally determined, trying to achieve an optimum between the following two opposite requirements: (a) lower computing volume and more rapid process of convergence in the learning period; (b) better performances from the correct classification of input patterns percentage.

In the learning phase are determined the optimum values [14] for weights connecting the pairs of neurons from the adjoint layers in the input–output direction using the Back-Propagation algorithm.

Kohonen Maps

Kohonen maps are competitive neural networks with topological character. The setting up of the winner neurons at output is done with keeping the topological relations between the input vectors.

That is the reason for which this neural network is successfully used in pattern recognition [9]. In the learning phase the structures are trained and the weights of the networks are established in two steps: (a) the determination of the winner neurons; (b) the adaptation of the weights for the winner neurons and for the neurons existing in a certain neighborhood. In this step important are: (a) the neighborhood dimension $r(t)$ decreasing during the learning; (b) the learning rate $\eta(t)$ following in our experiments one of the laws [5]:

$$\eta(t) = t^{-1} \text{ or } \eta(t) = t^{-1/2} \tag{1}$$

2.2 Hybrid Strategies

Hybrid strategies can improve the performance of the emerging ones. In order to make they more "human likely", we have applied a neuro-statistical hybrid, adding to a HMM a MLP as *a posteriori* probability estimator and realizing fuzzy variants of a MLP and a HMM.

Neuro-statistic Hybrid (HMM –MLP)

The HMM-based speech recognition methods make use of a probability estimator, in order to approximate emission probabilities $p(x_n/q_k)$, where x_n represents the observed data feature, and q_k is the hypothesized HMM state. These probabilities are used by the basic HMM equations, and because the HMM is based on a strict formalism, when the HMM is modified, there is a great risk of losing the theoretical foundations or the efficiency of the training and recognition algorithms. Fortunately, a proper use of the MLPs can lead to obtain probabilities that are related with the HMM emission probabilities [10].

In particular, MLPs can be trained to produce the *a posteriori* probability $p(x_n/q_k)$, that is, the *a posteriori* probability of the HMM state given the acoustic data, when each MLP output is associated with a specific HMM state. Many authors have shown that the outputs of an ANN used as described above can be interpreted as estimates of *a posteriori* probabilities of output classes conditioned by the input, so we will not insist on this matter, but we will mention an important condition, useful for finding an acceptable connectionist probability estimator: the system must contain enough parameters to be trained to a good approximation of the mapping function between the input and the output classes [14].

Thus, the *a posteriori* probabilities that are estimated by MLPs can be converted in emission probabilities by applying Bayes' rule (2) to the MLP outputs:

$$\frac{p(x_n / q_k)}{p(x_n)} = \frac{p(q_k / x_n)}{p(q_k)} \tag{2}$$

That is, the emission probabilities are obtained by dividing the *a posteriori* estimations from the MLP outputs by estimations of the frequencies of each class, while the scaling factor $p(x_n)$ is considered a constant for all classes, and will not modify the classification.

This was the idea that leads to hybrid neuro-statistical methods, that is, hybrid MLP-HMM methods, applied for solving the speech recognition problem.

Fuzzy Variants

Human judgement is rarely a binary one, and therefore binary logic even if very simple, is not the best solution to model human acting in speech classification tasks. It seems that fuzzy logic, able to a nuanced, shaded processing is much more suitable and indicated to be used in machine performing such tasks. In this subsection of our paper we will introduce fuzzy logic on two ways: realizing a fuzzification of the input parameters, like in the fuzzy – MLP, or introducing instead the probabilistic similarity measure applied in the usual HMM the fuzzy similarity measure, like in the fuzzy (generalized) HMM.

Fuzzy-MLP

Introducing a fuzzy processing of the input features of the MLP is a solution to improve the MLP performances [16].

First, the input values are described through a combination of 3 membership values in the linguistic property sets: low, medium and high. For doing this the π membership function is used:

$$\pi(r,c,\lambda)=\begin{cases}2(1-\|r-c\|/\lambda)^2 & for\ 0\le\|r-c\|\le\lambda/2\\ 1-2(\|r-c\|/\lambda)^2 & for\ 0\le\|r-c\|\le\lambda/2\\ 0 & otherwise\end{cases} \tag{3}$$

where: $\lambda>0$ is the radius of the π function with c as the central point, $\|\ .\ \|$ denotes the Euclidian norm.

For each component F_{ji} of the input vector F_j the parameters of the π membership function for each linguistic property: low (l), medium (m) and high (h) are computed using the relations:

$$\begin{aligned}\lambda_{m(Fij)}&=(F_{ji\max}-F_{ji\min})/2\\ c_{m(Fij)}&=F_{ji\min}+\lambda_{m(Fij)}\\ \lambda_{l(Fji)}&=(c_{m(Fji)}-F_{ji\min})/f_{dn}\\ c_{l(Fji)}&=c_{m(Fji)}-0.5\lambda_{l(Fji)}\\ \lambda_{h(Fji)}&=(F_{ji\max}-c_{m(Fji)})/f_{dn}\\ c_{h(Fji)}&=c_{m(Fji)}+0.5\lambda_{h(Fji)}\end{aligned} \tag{4}$$

where $F_{ji\max}$, $F_{ji\min}$ denote the upper and lower bounds of the observed range of feature and F_{ji} and f_{dd} is a parameter controlling the extent of overlapping.

After this, the structure of the fuzzy neural network, like the classical one, is composed from a hidden layer and an output layer.

The output vector is defined as the fuzzy class membership values. The membership value of the training $F_i=(F_{i1}\ F_{i2}\ ...F_{in})^t$ to class C_k is computed using:

$$\mu_k(F_i)=1/(1+z_{ik}/f_d))^{f_c} \tag{5}$$

where: f_d, f_c are constants controlling the amount of fuzziness in the class-membership set, z_{ik} is the weighted distance between the input vector F_i and the mean $O_k=(O_{k1}\ O_{k1}...O_{k1})^t$ of the *k*-th class, defined as:

$$z_{ik}=\sqrt{\sum_{j=1}[(F_{ji}-O_{kj})/v_{kj}]^2} \tag{6}$$

where: v_{kj} is the standard deviation of the *j*-th vectors' component from the C_k class.

In the training stage, the Back-Propagation algorithm is used to determine the weights which minimized the mean square error (*mse*) between the real output d_j and

the desired one y_j :

$$mse = \sum_{\substack{j=1 \\ F \in train}}^{n} (\sum (d_j - y_j)^2) \tag{7}$$

During training, the learning rate is gradually decreased in discrete steps {1, 0.5, 0.3, 0.1, 0.05, 0.03, 0.01}, until the network converges to a minimum error solution.

Fuzzy-HMM

The generalized model $\overline{\lambda} = (\overline{A}, \overline{B}, \overline{\pi})$ can be characterized by the same parameters [7] like the classical, well known model. The major difference in the fuzzy variant, is the interpretation of the probability densities for the classical HMM, as fuzzy densities. On this way the probabilistic similarity measure applied in the classical HMM is replaced by a more suitable fuzzy similarity measure.

The succession of parameter vectors, called the observation sequence, *O*, produces the state sequence *S* of the model, and, visiting for example at the moment *t+1* the state $q_{t+1} = S_j$, the symbol b_j is generated. The corresponding symbol fuzzy density $b_j(O_t)$ measures the grade of certainty of the statement that we observed O_t given that we are visiting state S_j .To perform classification tasks, the fuzzy similarity measure must be calculated. Based on the fuzzy forward and backward variables, a fuzzy Viterbi algorithm is proposed in [12] for the case of the Choquet integral with respect to a fuzzy measure and multiplication as intersection operator.

The fuzzy formulation of the forward variable α, bring an important relaxation in the assumption of statistical independence.

The joint measure $\overline{\alpha}_{\Omega_y}\left(\{O_1,...,O_t\} \times \{y_j\}\right)$ can be written as a combination of two measures defined on $O_1, O_2, ..., O_t$ and on the states respectively, no assumption about the decomposition of this measure being necessary, where $Y = \{y_1, y_2, ..., y_N\}$ represent the states at time *t+1* (Ω is the space of observation vectors).

For the standard HMM, the joint measure $P(O_1, O_2, ..., O_t, q_{t+1} = S_j)$ can be written as the product $P(O_1, O_2, ..., O_t) \cdot P(q_{t+1} = S_j)$, so that two assumptions of statistical independence must be made: the observation at time *t+1*, O_{t+1}, is independent of the previous observations $O_1, O_2, ..., O_t$ and the states at time *t+1* are independent of the same observations, $O_1, O_2, ..., O_t$.

These conditions find a poor match in case of speech signals and therefore we hope in improvements due to the relaxation permitted by the fuzzy measure.

Training of the generalized model can be performed with the re-estimation formulas also done in [12] for the Choquet integral. For each model we have trained with the reestimation formulas the corresponding generalized models, GHMMs, with 3-5 states, analog to the classical case.

After the training, we have calculated the fuzzy measure $\overline{P}\left(O / \overline{\lambda}\right)$, with the fuzzy Viterby algorithm and made the decisions for recognition in the same manner like for

the classical HMM: the correct decision corresponds to the model for which the calculated measure has a maximum.

3 Experimental Results

The experiment results are made by ASRS_RL, with multiple options for a large variety of speech recognition experiments [1].

In the next two sub-sections are applied the learning strategies presented in Section 2 for vowel and digit recognition and the obtained performance is evaluated.

The used databases (for vowel and digit) are sampled by 16 kHz, quantified with 16 bits, and recorded in a laboratory environment.

Vowel Recognition

The learning strategies applied in our recognition experiments are: the Kohonen maps, the MLP, the fuzzy-MLP and the fuzzy-HMM.

In the First Experiment, the vowel recognition rate using the VDRL database (Vowel Database for Romanian Language) and the MFCC coefficients (in form of 12 mel-frequency cepstral coefficients) was determined; the results obtained with MLP and HMM as learning strategies are comparatively presented in Table 1.

Table 1. Vowel recognition rate in the case of training with MS and testing with MS and FS.

Vowel	MLP		HMM	
	MS	FS	MS	FS
a	100.00%	80.71%	100.00%	82.28%
e	85.81%	43.25%	94.82%	50.67%
i	85.71%	85.71%	95.15%	92.41%
o	90.90%	51.33%	97.00%	52.78%
u	88.88%	71.42%	94.83%	77.85%
Mean	*91.26%*	*66.48%*	*96.36%*	*71.20%*

The database VDRL contained speech data from 19 speakers (9 males and 10 females) each reading the same 5 vowels (*a, e, i, o, u*). The database was organized as follows: one database for male speakers (MS), one database for female speakers (FS). In booth cases one male speaker (MS) and one female speaker (FS) was excluded from the training database and used their data for the testing.

The experimented MLP is a two-layer perceptron trained with Back-Propagation algorithm, having in the output layer 5 nodes corresponding to the 5 vowels to be classified and 100 nodes in the hidden layer (experimentally chosen).

The number of the input nodes is equal to the number of features (12 MFCC).

The HMMs chosen for comparison are Bakis (or left-right) structures with five states and for each vowel one model is created [15].

In Table 1, are displayed only the results in the case of training MS and testing with MS and FS. Similarly results were obtained for the training with FS [2].

In the Second Experiment, the vowels were described by three formant frequencies and the error rates obtained with different learning strategies are given in Table 2.

The database for formants contains 500 formant vectors, 100 for each vowel for the training and 250 formant vectors, 50 for each vowel for the testing.

The learning structures applied [5], [14] for these investigations are:

(1) KM with the input layer with 3 neurons, corresponding to the three formant frequencies and three variants for the output layer: unidimensional with 25 neurons, bidimensional with 5×5 neurons, and toroidal with 25 neurons.

(2) MLP with 3 layers organized as it follows: (a) the input layer with 3 neurons, corresponding to the three formant frequencies; (b) the hidden layer with 0 (Boolean network) or 4 neurons, (c) the output layer with 5 neurons corresponding each to a processed class (in our case the vowels *a, e, i, o, u*).

(3) Fuzzy-MLP.

Table 2. Error rates (%) in for different learning strategies for the case of format.

Vowel	KM 1-dim	KM 2-dim	KM toroidal	Boolean	MLP	Fuzzy MLP
a	2.60%	1.20%	2.00%	4.00%	0.00%	1.50%
e	3.20%	2.40%	2.40%	6.00%	2.50%	1.00%
i	2.20%	1.60%	1.20%	4.00%	1.00%	0.50%
o	1.80%	1.20%	1.20%	10.00%	1.00%	1.00%
u	2.20%	2.10%	1.80%	10.00%	1.00%	0.00%
Mean	*2.40%*	*1.70%*	*1.72%*	*6.80%*	*1.10%*	*0.80%*

In the Third Experiment the parameterization is realized with the mel-cepstral coefficients and the first and second order differences of these coefficients deduced from homomorfic filtering. The error rates obtained in the vowel recognition tests are given in Table 3, comparatively for the generalized HMM (fuzzy-HMM) and the classical HMM.

Table 3. Error rates (%) for generalized and for classical HMMs.

Vowel	Fuzzy - HMM	Classical HMM
a	5.10%	6.90%
e	2.40%	4.80%
i	3.80%	7.30%
o	2.50%	5.90%
u	0.70%	3.90%
Mean	*2.90%*	*5.78%*

Digit Recognition

In the second sub-section the performances obtained in digit recognition are evaluate with the hybrid strategies (HMM– MLP) for unenrolled and enrolled speaker.

The DDRL database (Digit Database for Romanian Language) contained speech data from 9 speakers (6 males and 3 females) each speakers reading 9 digits (*unu, doi, trei, patru, cinci, şase, şapte, opt, nouă*). We excluded two MS and one FS from the database and used them for the testing.

The digit parameters were extracted by cepstral analysis, in form of 12 mel-frequency cepstral coefficients (MFCC). The hybrid system (HMM-MLP) consists of 9 hybrid models corresponding to 9 digits. Each hybrid model is made of 5 states, each state being associated with one output node of the MLP. The MLP has one hidden layer (100 nodes experimentally chossen), and the input layer consisting of 12 nodes.

We compare the two kinds of tests (using DDRL database): first, with enrolled speakers, which mean that the speakers were involved both in training and testing, and second, with unenrolled speakers were the testing speakers are not involved in the training.

The results obtained for hybrid strategies are compared with other learning strategies (hidden Markov models, Support Vector Machine) and the performance being appreciated by their recognition rate and by their generalization capacity [5]. The word recognition rate (WRR) is reported in Table 4.

Table 4. The WRR (%) for different learning strategies.

Learning strategies	Enrolled speakers	Unrolled speakers
HMM-MLP	98.50%	98.30%
HMM	98.00%	97.50%
SVM	97.70%	91.70%

4 Conclusions

This paper reports a study focussed on the learning strategies applied in speech recognition for vowel and digit recognition.

(1) For vowel recognition in Romanian language the following conclusion can be reported:

a) The recognition rates in the case of MLP are higher than in the case of HMM. A possible explanation can be the fact that the model training is discriminative, while in the case of HMM the training is not discriminative, which represents a disadvantage of HMM utilization.

b) The KM 2-dimensional structures and the toroidal have the same performance, weaker is the performance of the 1-dimensional structure. The best balanced situation corresponds to the 2-dimensional map 5x5, in which all neurons are associated to a vowel to be recognized.

The performance obtained in the case of the Boolean network is unacceptable, but the MLP acts well.

Using fuzzy-MLP structure it is an improvement with a mean value of 0.30% comparative with the non-fuzzy structure.

c) A mean decreasing of nearly 3% is realized in the error rate by adopting the fuzzy-HMM instead of the probabilistic one.

(2) For digit recognition in Romanian language using our database (DDRL) the following observation can be reported:

a) Chosen this approach, which combine the HMM with MLP into a hybrid system is a very goad solution because the results are higher the results obtained for HMM and SVM.
b) It is to seen that SVM performances are slightly after that of the HMM, but is really promising, taking into account that the HMM has the benefit of a so long refinement time.

References

1. Dumitru, C.O.: Modele neurale si statistice pentru recunoasterea vorbirii. Ph.D. thesis, Bucharest (2006).
2. Dumitru, C.O., Gavat, I.: Vowel, Digit and Continuous Speech Recognition Based on Statistical, Neural and Hybrid Modelling by Using ASRS_RL. EUROCON 2007, Warsaw, Poland (2007), 856-863.
3. Gavat, I., & all: Elemente de sinteza si recunoasterea vorbirii. Ed. Printech, Bucharest (2000).
4. Gavat, I., Dumitru, C.O., Costache, G.: Speech Signal Variance Reduction by Means of Learning Systems. MEDINF 2003, Craiova-Romania (2003), 68-69.
5. Gavat, I, Dumitru, O., Iancu, C., Costache, G.: Learning Strategies in Speech Recognition. The 47th International Symposium ELMAR 2005, Zadar-Croatia (2005), 237-240.
6. Goronzy, S.: Robust Adaptation to Non-Native Accents in Automatic Speech Recognition. Springer - Verlag Berlin Heidelberg, Germany (2002).
7. Huang, X., Acero, A., Hon, H.W.: Spoken Language Processing – A Guide to Theory, Algorithm, and System Development. Prentice Hall (2001).
8. Juanhg, B.H., Furui, S.: Automatic Recognition and Understanding of Spoken Language–A First Step Toward Natural Human–Machine Communication. Proc. IEEE, Vol.88, No.8, (2000), 1142-1165.
9. Kohonen T.: Adaptive, Associative and Self-Organizing Function in Neural Computing. Artificial Neural Networks, IEEE Press, Piscataway- NJ (1992), 42-51.
10. Lippmann, R. and Singer, E.: Hybrid neural network/HMM approaches to word spotting. Proc. ICASSP ’93. Minneapolis (1993), 565-568.
11. Lippmann, R.P.: Human and Machine Performance in Speech Recognition Tasks. Speech Communications, Vol.22, No.1 (1997), 1-15.
12. Mahomed, M. and Gader, P.: Generalized hidden Markov models. IEEE Transactions on Fuzzy Systems. (2000), 67-93.
13. Morgan, D.P., Scotfield, C.L.: Neural Networks. Prentice Hall, New York (1992).
14. Valsan, Z., Gavat, I., Sabac, B., Cula, O., Grigore, O., Militaru, D., Dumitru, C.O.: Statistical and Hybrid Methods for Speech Recognition in Romanian. International Journal of Speech Technology, Vol.5, No.3 (2002), 259-268.
15. Young, S.J.: The general use of tying in phoneme-based HMM speech recognizers. Proc. ICASSP’92, Vol.1, San Francisco (1992), 569-572.
16. Wang, Z. and Klirr, G.: Fuzzy Measure Theory. New York: Plenum (1992).

Forecasting Internet Traffic by Neural Networks Under Univariate and Multivariate Strategies

Paulo Cortez[1], Miguel Rio[2], Pedro Sousa[3] and Miguel Rocha[3]

[1] Department of Information Systems/R&D Algoritmi Centre
University of Minho, 4800-058 Guimarães, Portugal
pcortez@dsi.uminho.pt
http://www.dsi.uminho.pt/~pcortez
[2] Department of Electronic and Electrical Engineering, University College London
Torrington Place, WC1E 7JE, London, U.K.
m.rio@ee.ucl.ac.uk
[3] Department of Informatics, University of Minho, 4710-059 Braga, Portugal
{pns, mrocha}@di.uminho.pt

Abstract. By improving Internet traffic forecasting, more efficient TCP/IP traffic control and anomaly detection tools can be developed, leading to economic gains due to better resource management. In this paper, Neural Networks (NNs) are used to predict TCP/IP traffic for 39 links of the UK education and research network, under univariate and multivariate strategies. The former uses only past values of the forecasted link, while the latter also uses the traffic from neighbor links of the network topology. Several experiments were held by considering hourly real-world data. The Holt-Winters method was also tested in the comparison. Overall, the univariate NN approach produces the best forecasts for the backbone links, while a Dijkstra based NN multivariate strategy is the best option for the core to subnetwork links.

1 Introduction

Internet traffic prediction is a key issue for understanding communication networks and optimizing resources (e.g. adaptive congestion control and proactive network management), allowing a better quality of service [1–3]. Moreover, traffic forecasting can help to detect anomalies (e.g. security attacks, viruses or an irregular amount of SPAM) by comparing the real traffic with the forecasts [4, 5].

TCP/IP traffic prediction is often done intuitively by network administrators, with the help of marketing information (e.g. future number of costumers) [1]. Yet, this may not be suited for serious day-to-day network administration and the alternative is to use Operational Research and Computer Science methods. In particular, the field of Time Series Forecasting (TSF), deals with the prediction of a chronologically ordered variable, where the goal is to model a complex system as a black-box, predicting its behavior based in historical data [6]. The TSF approaches can be divided into univariate and multivariate, depending if one or more variables are used. Multivariate methods are likely to produce better results, provided that the variables are correlated [7].

Several TSF methods have been proposed, such as the Holt-Winters [6] and Neural Networks (NN) [8, 3]. Holt-Winters was developed for series with trended and seasonal factors and more recently a double seasonal version has been proposed [9]. In contrast with the conventional TSF methods (e.g. Holt-Winters), NNs can predict nonlinear series. In the past, several studies have proved the predictability of network traffic by using similar methods. For instance, the Holt-Winters was used in [4, 10] and NNs have also been proposed [11, 5, 3]. However, these studies only considered univariate (or single link) data, thus not making use of the topology network. By using data from more than one link, there is a potential for better predictions.

This study will use recent hourly data from the United Kingdom Education and Research Network (UKERNA) network. The network includes a backbone made up of 8 core routers that transport data through 21 regional subnetworks. In this paper, we will explore NNs and two multivariate approaches for data selection: using all direct neighbor links and selecting the most probable neighbor that is expected to influence the predicted link. The latter strategy is based in a novel heuristic that uses the Open Shortest Path First (OSPF) [12] protocol and Dijkstra algorithm. These approaches will be compared with the NN univariate case and also the classic Holt-Winters method. Furthermore, we will predict all UKERNA core to core and core to subnetwork links, in a total of 39 connections.

2 Internet Traffic Data

This work will analyze traffic data (in Mbit/s) from the UK academic network backbone (UKERNA)[4], which includes eight core routers and 21 subregional networks. Figure 1 plots the respective direct graph, where **b**dd, **s**dd and **c**dd denote the links within the **b**ackbone core routers, core to **s**ubnetwork and subnetwork to **c**ore, respectively (d is a digit number). The data collection was based in the Simple Network Management Protocol (SNMP), which quantifies the traffic passing through every network interface with reasonable accuracy [13]. SNMP is widely deployed by every Internet Service Provider/network and the collection of this data does not induce any extra traffic on the network. In this work, we will adopt an hourly scale, denoting a short-term forecasting that is often used to for optimal control or detection of abnormal situations [14]. The data was recorded from 12 AM of 14th June 2006 to 12 AM of 23th July 2006. In total, there are 936 hourly observations for each link.

The OSPF is the the most commonly used intra-domain routing protocol [12]. Under this protocol, every link contains a weight that is assigned by the network administrator. The Dijkstra algorithm is used to find the shortest paths between any two nodes of the network and these paths are then used by the routers to direct traffic. Most of the UKERNA OSPF weights are set to 10 and the few exceptions are listed in Figure 1. For instance, the OSPF weight between the core routers of Glasgow and Edinburgh is 100 (links b09 and b18); and the shortest path between Warrington and Edinburgh includes the links b07 and b18.

[4] http://www.ja.net

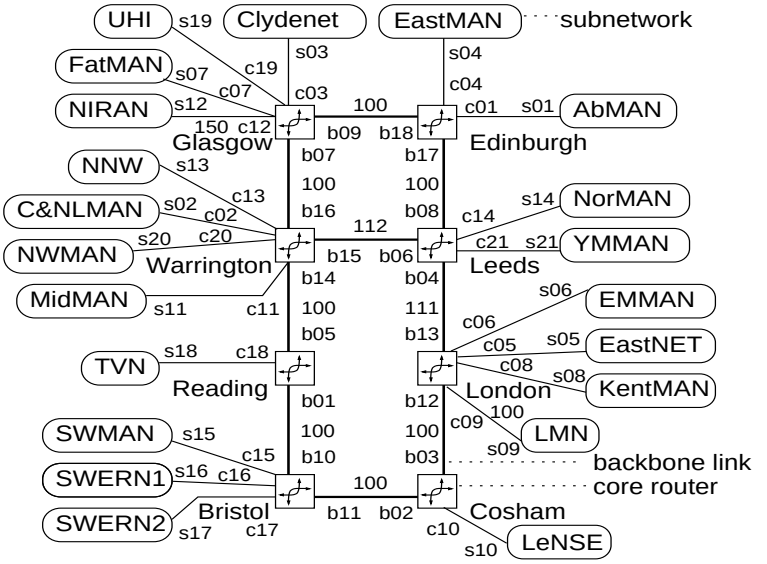


Fig. 1. The schematic of the UK academic Internet network.

As an example, the traffic of two neighbor links, Warrington-Glasgow (b07) and Glasgow-Clydenet (s05), is plotted in Figure 2. In both graphs, there are influences of two seasonal components due to the the intraday and intraweek cycles.

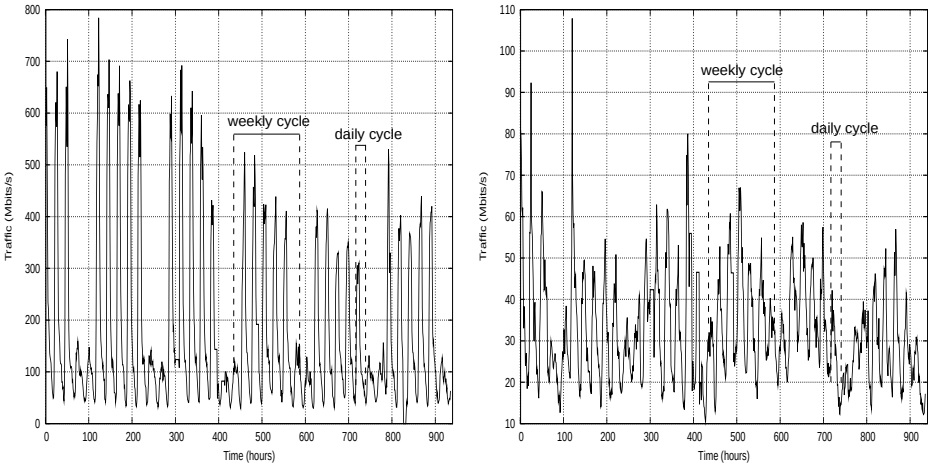


Fig. 2. The IP traffic for b07 (Warrington-Glasgow, left) and s03 (Glasgow-Clydenet, right) links.

3 Forecasting Methods

A Time Series Forecasting (TSF) model assumes that past patterns will occur in the future. Let $y_t = (y_{1t}, \ldots, y_{kt})$ denote a multivariate series, where y_{ij} is the jth chronological observation on variable i and k is the number of distinct time variables ($k = 1$ when a univariate setting is used). Then [7]:

$$\begin{aligned} \widehat{y}_{pt} &= F(y_{1t-1}, \ldots, y_{1t-n}, \ldots, y_{kt-1}, \ldots, y_{kt-n}) \\ e_{pt} &= y_{pt} - \widehat{y}_{pt} \end{aligned} \tag{1}$$

where $\widehat{y}_{pt}$ denotes the estimated value for the pth variable and time t; F the underlying function of the forecasting model; and e_{pt} is the error (or residual).

The overall performance of a model is evaluated by a global accuracy measure, namely the Root Mean Squared Error (RMSE) and Relative RMSE (RRMSE), given in the form [15]:

$$\begin{aligned} RMSE_p &= \sqrt{\textstyle\sum_{i=P+1}^{P+N} e_{pi}^2 / N} \\ RRMSE_p &= RMSE_p / RMSE_{\overline{y}_{pt}} \times 100\,(\%) \end{aligned} \tag{2}$$

where P is the present time; N is the number of forecasts; and $RMSE_{\overline{y}_{pt}}$ is the $RMSE$ given by the simple mean prediction. The last metric ($RRMSE$) will be adopted in this work, since it has the advantage of being scale independent, where 100% denotes an error similar to the mean predictor ($\overline{y}_{pt}$).

Due to the temporal nature of this domain, a sequential holdout will be adopted for the forecasting evaluation. Hence, the first $TR = 2/3$ of the series will be used to fit (train) the forecasting models and the remaining last $1/3$ to evaluate (test) the forecasting accuracies. Also, an internal holdout procedure will be used for model selection, where the training data will be further divided into training ($2/3$ of TR) and validation sets (1/3 of TR). The former will be used to fit the candidate models, while the latter will be used to select the models with the lowest error ($RMSE$). After this selection phase, the final model is readjusted using all training data.

3.1 Neural Networks

Neural Networks (NNs) are innate candidates for forecasting due to their nonlinear and noise tolerance capabilities. Indeed, the use of NNs for TSF began in the late eighties with encouraging results and the field has been growing since [8, 14, 11, 3].

The multilayer perceptron is the most popular NN used within the forecasting domain [8, 11]. When adopting this architecture, TSF is achieved by using a sliding time window[5]. A sliding window is defined by the set of time lags used to build a forecast. For instance, given the univariate time series 1,2,3,4,5,6 and sliding window $\{1, 2, 4\}$, the following training examples can be built: $1, 3, 4 \rightarrow 5$ and $2, 4, 5 \rightarrow 6$. In a multivariate setting, k sliding windows are used: $\{L_{11}, \ldots, L_{1W_1}\}, \ldots, \{L_{k1}, \ldots, L_{kW_k}\}$, where L_{ij} denotes a time lag for the ith variable.

In this work, a fully connected multilayer network with one hidden layer of H hidden nodes and bias connections will be adopted (Figure 3). The logistic activation

[5] This combination is also named Time Lagged Feedforward Network (TLFN) in the literature.

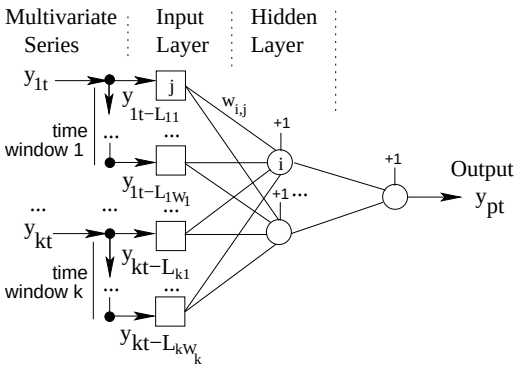


Fig. 3. The multilayer perceptron architecture for multivariate time series forecasting.

function is applied on the hidden nodes and the output node uses a linear function [16]. In past work [3], this architecture outperformed conventional univariate methods such as Holt-Winters and ARMA models. The overall model is given in the form:

$$\widehat{y}_{pt} = w_{o,0} + \sum_{i=I+1}^{I+H} f(\sum_{s=1}^{k} \sum_{r=1}^{W_s} y_{st-L_{sr}} w_{i,j}) \quad (3)$$

where $w_{d,s}$ is the weight from node s to d; (if $d = 0$ then it is a bias connection); $j \in \{1, \ldots, I\}$ is an input node; o is the output node; and f the logistic function ($\frac{1}{1+e^{-x}}$).

Before training, all variables are scaled with a zero mean and one standard deviation. Then, the initial NN weights are randomly set within $[-0.7, +0.7]$ [17]. Next, the training algorithm is applied and stopped when the error slope approaches zero or after a maximum of E epochs. Since the NN cost function is nonconvex (with multiple minima), NR runs are applied to each neural setup, being selected the NN with the lowest mean error [16]. After training, the NN outputs are rescaled to the original domain.

Under this setting, the NN performance will depend on the number of hidden nodes (H), the selection of the k variables used in the multivariate model and the time window used for each variable. All these parameters can have a crucial effect in the forecasting performance. Feeding a NN with uncorrelated variables or time lags may affect the learning process due to the increase of noise. A NN with 0 hidden neurons can only learn linear relationships and it is equivalent to the classic Auto-Regressive (AR) model. By increasing the number of hidden neurons, more complex nonlinear functions can be learned but also it increases the probability of overfitting to the data and thus loosing the generalization capability. Since the search space for these parameters is high, heuristic procedures will be used during the model selection step.

Three strategies are proposed for the variable selection:

- **Single-Link NN (SLNN)**, the simple univariate model where the predictions are based on the past values of the current link (p);
- **All Direct Neighbor Link NN (ADNN)**, based on p plus the previous traffic observed in all direct neighbor links that influence p; and
- **Dijkstra-Assisted NN (DANN)**, based on p plus the neighbor that is expected to influence more the predicted link under the OSPF protocol. First, the Dijkstra algo-

rithm is used to compute the shortest OSPF paths between all nodes of the network. Then, the subset with all paths that include p as an internal or end link is selected. Finally, the heuristic selects the most common[6] direct preceeding neighbor of p in the subset.

Regarding the multivariate methods, DANN selects only $k = 2$ variables, while ADNN uses a higher number of links (from 3 to 7). For instance, when forecasting the Reading-TVN (p=s18) traffic, the ADNN variable set is {s18,b05,b01}[7] (Figure 1). There are 16 OSPF paths ending at TVN that include b05 and only 11 paths that go through b01. Hence, DANN will select the former (i.e. {s18, b05}).

Based on previous univariate IP traffic forecasting work [3], a small range of hidden nodes will be tested, with $H \in \{0, 2, 4, 6\}$. Also, three sliding windows, based on the daily ($K_1 = 24$) and weekly ($K_2 = 168$) cycles, will be considered: $w_1 = \{1, 24, 25\}$, $w_2 = \{1, 168, 169\}$ and $w_3 = \{1, 24, 25, 168, 169\}$. In [3], this sliding window setup obtained high quality results. When a multivariate model is used, then the same window is applied to all links.

3.2 Holt-Winters Methods

The Holt-Winters (HW) [6] is a popular univariate forecasting technique from the family of Exponential Smoothing methods. The predictive model is based on some underlying patterns such as a trend or a seasonal cycle (K_1), which are distinguished from random noise by averaging the historical values. Its popularity is due to advantages such as the simplicity of use, the reduced computational demand and the accuracy of the forecasts, specially with seasonal series.

The general model is defined by:

$$\begin{array}{ll} \text{Level} & S_t = \alpha \frac{y_t}{D_{t-K_1}} + (1-\alpha)(S_{t-1} + T_{t-1}) \\ \text{Trend} & T_t = \beta(S_t - S_{t-1}) + (1-\beta)T_{t-1} \\ \text{Seasonality} & D_t = \gamma \frac{y_t}{S_t} + (1-\gamma)D_{t-K_1} \\ & \widehat{y}_{pt} = (S_{t-1} + T_{t-1}) \times D_{t-K_1} \end{array} \quad (4)$$

where S_t, T_t and D_t stand for the level, trend and seasonal estimates, K_1 for the seasonal period, and α, β and γ for the model parameters. When there is no seasonal component, the γ is discarded and the D_{t-K_1} factor in the last equation is replaced by the unity. More recently, this method has been extended to encompass two seasonal cycles (K_1 and K_2) [9]. In this work, four HW variants will be tested in the model selection phase: n – non seasonal ($K_1 = 1$); d – daily seasonal ($K_1 = 24$); w – weekly seasonal ($K_1 = 168$); and D – double seasonal ($K_1 = 24$ and $K_2 = 168$).

4 Experiments and Results

The experiments were conducted off-line (i.e. after the data was collected) using the **RMiner** [18], an open source library for the **R** statistical environment [19]. In particular,

[6] In case of a draw (which rarely occurs), the heuristic simply selects one of the contenders.

[7] The link c18 is not considered, since its origin (TVN) matches the link destination.

the **RMiner** uses the **nnet** package [17] to implement the NNs. The NNs were trained with $E = 100$ epochs of the BFGS algorithm [20], from the family of quasi-Newton methods and the number of runs was set to $NR = 10$. The HW initial values (e.g. level estimate) were set by averaging the early observations [9] and the internal parameters (e.g. α) were optimized using a 0.05 grid search for the best training error ($RMSE$).

Since the intention is to compared univariate and multivariate approaches, only the links with preceding neighbors will be predicted, i.e., all **s**dd and **b**dd, in a total of 39 connections. The selected forecasting models for each method are shown in Table 1. For the HW, the weekly cycle is the most common model (w) and the double seasonal variant is never used. The weekly effect (w_2) is also the most common case for strategy ADNN. In contrast, the majority of the SLNN and DANN methods use the double seasonal model (w_3). Regarding the NN architectures, in general only linear models are selected. The 15 nonlinear exceptions ($H > 0$) are listed in the table. These results confirm the notion that short term IP traffic can be modeled by small networks.

The forecasts with the selected models were performed on the test sets (with 312 elements) for all links shown in Table 1. Thirty runs were applied for the NNs and the results are shown as the mean $RRMSE$ with the respective 95% t-student confidence intervals. The range of the best $RRMSE$ values is high, showing that some links are much harder to predict than others (e.g. 14.1% for b10 versus 82.5% for s02). Overall, the HW is the worst strategy, since it is the best method for only 2 links (b05 and b14). Regarding the backbone links, the univariate approach (SLNN) is the best NN choice in 10 cases, followed by the ADNN (best in 7 links), while the DANN outperforms the other methods for only one link (b17). This scenario changes when considering the core to subnetwork links (**s**dd), where the DANN is the best method (with 12 wins), while both SLNN and ADNN achieve statistically significant lowest errors in 4 cases.

For demonstrative purposes, the left of Figure 4 presents the average DANN traffic forecasts for the first 60 hours of the s03 series. In this case, a high quality fit is achieved, since the two curves are close. The observed (x-axis) versus the predicted values for a given run (y-axis) is also shown. In the figure, the forecasts (points) are near the diagonal line, which denotes the perfect forecast. Another relevant issue is related with the computational complexity. The proposed solution is very fast and can be used in real-time. For the example, with a Pentium Dual Core 3GHz processor, the DANN model selection phase took 12 seconds, while the 30 runs of the final NN training and testing required only 2.2 seconds.

5 Conclusions

This work analyses the efficiency of several Neural Network (NN) approaches when applied to predict hourly TCP/IP traffic, collected from the United Kingdom Education and Research Network (UKERNA). In particular, three strategies were tested: SLNN – univariate approach based on past patterns from the current link; ADNN – which also includes the past values from all direct neighbors; and DANN – a novel approach that includes only one link neighbor, whose selection is based on the Dijkstra algorithm and OSPF protocol. Also, a comparison was made with the Holt-Winters (HW) method, which is popular for seasonal series.

Table 1. The forecasting $RRMSE$ errors and selected models (in brackets).

Link	SLNN	ADNN	DANN	HW
b01	24.8±0.0 (w_3)	**23.9**±0.0 (w_2)	25.0±0.0 (w_3)	25.2 (w)
b02	**63.2**±0.0 (w_1)	89.5±0.0 (w_2)	63.4±0.0 (w_1)	68.7 (n)
b03	**22.1**±0.0 (w_3)	22.1±0.0 (w_2)	22.2±0.0 (w_3)	27.8 (d)
b04	**21.3**±0.0 (w_3)	21.5±0.0 (w_2)	22.2±0.0 (w_2)	25.2 (w)
b05	**34.1**±0.0 (w_3)	34.7±0.0 (w_2)	35.7±0.0 (w_2)	34.0 (w)
b06	86.6±2.7 (w_1, H=4)	**58.1**±0.0 (w_1)	58.4±0.0 (w_3)	69.0 (w)
b07	**19.7**±0.0 (w_3)	30.6±0.0 (w_3)	20.3±0.0 (w_3)	25.1 (w)
b08	40.7±0.0 (w_2)	**40.7**±0.0 (w_2)	41.2±0.0 (w_2)	44.3 (w)
b09	**56.5**±0.0 (w_2)	57.2±0.0 (w_2)	57.5±0.0 (w_2)	67.5 (w)
b10	**14.1**±0.0 (w_3)	17.7±0.0 (w_2)	15.4±0.0 (w_3)	15.0 (w)
b11	**54.1**±0.0 (w_3)	57.3±0.9 (w_3, H=2)	54.3±0.0 (w_2)	58.0 (n)
b12	62.7±5.9 (w_2, H=2)	**36.1**±0.0 (w_1)	74.6±0.0 (w_3)	45.4 (w)
b13	**30.5**±0.0 (w_3)	31.2±0.0 (w_2)	30.6±0.0 (w_3)	31.7 (w)
b14	19.5±0.0 (w_3)	**19.4**±0.0 (w_3)	19.5±0.0 (w_3)	19.0 (w)
b15	79.9±0.0 (w_3)	**78.7**±0.0 (w_2)	80.4±0.0 (w_3)	87.0 (n)
b16	48.0±1.0 (w_2, H=4)	**37.5**±0.0 (w_2)	38.7±0.0 (w_3)	39.4 (w)
b17	31.5±0.8 (w_3, H=2)	57.5±3.1 (w_1)	**28.3**±0.0 (w_3)	30.1 (w)
b18	**57.3**±0.0 (w_3)	59.2±0.0 (w_2, H=6)	58.4±0.0 (w_3)	80.8 (w)
s01	42.3±0.4 (w_2, H=2)	47.5±2.7 (w_2, H=2)	**41.8**±0.0 (w_2)	45.1 (w)
s02	82.8±0.0 (w_3)	85.2±0.0 (w_2)	**82.5**±0.0 (w_3)	91.9 (w)
s03	33.6±0.0 (w_3)	34.6±0.0 (w_3)	**32.4**±0.0 (w_3)	37.3 (d)
s04	41.3±0.1 (w_2, H=2)	41.4±0.0 (w_2)	41.8±0.0 (w_2)	48.0 (w)
s05	**41.4**±0.0 (w_3)	41.6±0.0 (w_1)	42.0±0.0 (w_3)	47.5 (w)
s06	39.6±0.0 (w_3)	38.3±0.0 (w_2)	**38.2**±0.0 (w_3)	44.1 (w)
s07	45.1±0.0 (w_2)	**42.8**±0.0 (w_2)	42.9±0.0 (w_2)	51.7 (w)
s08	**27.5**±0.0 (w_3)	28.9±0.0 (w_2)	28.3±0.0 (w_3)	34.9 (w)
s09	28.6±0.0 (w_3)	**27.3**±0.0 (w_2)	28.3±0.0 (w_3)	36.5 (w)
s10	35.9±0.5 (w_2, H=6)	**32.8**±0.0 (w_3)	38.6±0.0 (w_3)	33.2 (w)
s11	**68.4**±0.0 (w_3)	69.6±0.0 (w_1)	71.2±1.2 (w_2, H=2)	74.5 (n)
s12	71.8±8.3 (w_3, H=4)	69.0±0.0 (w_2)	**56.7**±0.0 (w_3)	65.4 (w)
s13	48.1±0.0 (w_2)	**44.7**±0.0 (w_2)	44.7±0.0 (w_2)	47.7 (w)
s14	36.8±0.0 (w_3)	43.8±0.0 (w_1)	**34.0**±0.0 (w_2)	37.9 (w)
s15	26.5±0.0 (w_3)	24.9±0.0 (w_2)	**23.6**±0.0 (w_3)	27.2 (w)
s16	33.4±0.0 (w_3)	33.8±0.0 (w_2)	**32.3**±0.0 (w_3)	36.0 (d)
s17	28.3±0.5 (w_2, H=2)	26.7±0.0 (w_3)	**25.5**±0.0 (w_3)	32.2 (n)
s18	54.2±0.0 (w_2)	53.2±0.0 (w_3)	**51.9**±0.0 (w_3)	54.7 (n)
s19	**39.8**±0.0 (w_2)	41.1±0.0 (w_3)	40.1±0.0 (w_2)	39.9 (w)
s20	64.7±0.2 (w_3, H=2)	65.4±0.0 (w_2)	**61.3**±0.0 (w_3)	70.7 (d)
s21	34.2±0.3 (w_3, H=2)	40.6±0.0 (w_3)	**32.6**±0.0 (w_3)	32.9 (w)

bold – statistical significance under a pairwise comparison with other NN methods.
underline – best model.

A large number of experiments was conducted, with a total of 39 forecasted links. Overall, the NN results are quite competitive, outperforming the HW model in all except two cases. Regarding the univariate versus multivariate comparison, the results

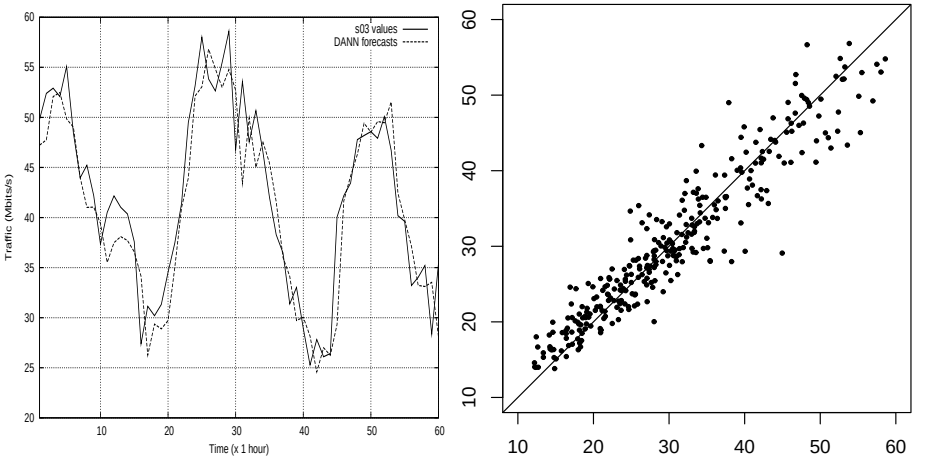


Fig. 4. Example of the forecasts (left) and observed versus predicted values scatter plot (right).

differ according to the link characteristics. Within the backbone links, the SLNN is the best option in 10 of the 18 series, while DANN only excels other strategies for one connection (b17 of Figure 1). However, for the core to subnetwork links, the multivariate DANN strategy provides the best forecasts in 12 of 21 cases, while SLNN achieves the best performance in only 4 links. These results may be explained by the nature of the network topology. The core to subnetwork links are peripheral funnels, thus they are more likely to be influenced by one single neighbor. In contrast, the core routers are large carriers, i.e., they direct traffic from/to a larger number of nodes.

Since small networks were selected, the NNs are very fast and can be applied in real-time. Thus, the proposed approach opens room for producing better traffic engineering tools and methods to detect anomalies in the traffic patterns. This can be achieved without producing any extra traffic in the network and with minimal use of computation resources, since this work was designed assuming a passive monitoring system.

In future work, the comparison will be extended to other forecasting techniques (e.g. ARMA models [21]). Moreover, the proposed approach will be applied to traffic demands of specific Internet applications, such as Voice over Internet Protocol (VoIP). Another promising direction is to explore incomplete information scenarios. For instance, to see if it is possible to forecast the backbone link traffic using only the subnetwork to core connections, i.e., without knowing the past values of the predicted links.

Acknowledgements

This work is supported by the FCT project PTDC/EIA/64541/2006. We would also like to thank Steve Williams from UKERNA for providing us with part of the data used in this work.

References

1. K. Papagiannaki, N. Taft, Z. Zhang, and C. Diot. Long-Term Forecasting of Internet Backbone Traffic. *IEEE Trans. on Neural Networks*, 16(5):1110–1124, September 2005.
2. V. Alarcon-Aquino and J. Barria. Multiresolution FIR Neural-Network-Based Learning Algorithm Applied to Network Traffic Prediction. *IEEE Trans. on Systems, Man and Cybernetics - Part C*, 36(2):208–220, 2006.
3. P. Cortez, M. Rio, M. Rocha, and P. Sousa. Internet Traffic Forecasting using Neural Networks. In *Proceedings of the IEEE 2006 International Joint Conference on Neural Networks*, pages 4942–4949, Vancouver, Canada, 2006.
4. B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based Change Detection: Methods, Evaluation, and Applications. In *Proc. of Internet Measurment Conference (IMC'03)*, Miami, USA, October 2003. ACM.
5. J. Jiang and S. Papavassiliou. Detecting Network Attacks in the Internet via Statistical Network Traffic Normality Prediction. *Journal of Network and Systems Management*, 12:51–72, 2004.
6. S. Makridakis, S. Weelwright, and R. Hyndman. *Forecasting: Methods and Applications*. John Wiley & Sons, New York, USA, 1998.
7. G. Reinsel. *Elements of Multivariate Time Series Analysis*. Springer, San Francisco, CA, second edition, 2003.
8. A. Lapedes and R. Farber. Non-Linear Signal Processing Using Neural Networks: Prediction and System Modelling. Tech. Rep. LA-UR-87-2662, Los Alamos National Laboratory, USA, 1987.
9. J. Taylor, L. Menezes, and P. McSharry. A Comparison of Univariate Methods for Forecasting Electricity Demand Up to a Day Ahead. *Int. Journal of Forecasting*, 21(1):1–16, 2006.
10. Q. He, C. Dovrolis, and M. Ammar. On the Predictability of Large Transfer TCP Throughput. In *Proc. of SIGCOMM'05*, Philadelphia, USA, August 2005. ACM.
11. H. Tong, C. Li, and J. He. Boosting Feed-Forward Neural Network for Internet Traffic Prediction. In *Proc. of the IEEE 3rd Int. Conf. on Machine Learning and Cybernetics*, pages 3129–3134, Shanghai, China, August 2004.
12. T.M. ThomasII. *OSPF Network Design Solutions*. Cisco Press, 1998.
13. W. Stallings. *SNMP, SNMPv2, SNMPv3 and RMON 1 and 2*. Addison Wesley, 1999.
14. X. Ding, S. Canu, and T. Denoeux. Neural network based models for forecasting. In *Proc. of Applied Decision Technologies Conf. (ADT'95)*, pages 243–252, Uxbridge, UK, 1995.
15. I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco, CA, second edition, 2005.
16. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, NY, USA, 2001.
17. W. Venables and B. Ripley. *Modern Applied Statistics with S*. Springer, 4th edition, 2003.
18. P. Cortez. RMiner: Data Mining with Neural Networks and Support Vector Machines using R. In R. Rajesh (Ed.), *Introduction to Advanced Scientific Softwares and Toolboxes*, IAEng publishers, Singapore, In Press.
19. R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2007. ISBN 3-900051-00-3.
20. M. Moller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533, 1993.
21. G. Box and G. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden Day, USA, 1976.

Adaptive Electrical Signal Post-Processing in Optical Communication Systems

Yi Sun[1], Alex Shafarenko[1], Rod Adams[1], Neil Davey[1]
Brendan Slater[2], Ranjeet Bhamber[2], Sonia Boscolo[2] and Sergei K. Turitsyn[2]

[1] Department of Computer Science, University of Hertfordshire, College Lane
Hatfield, AL10 9AB, U.K.
{y.2.sun, a.Shafarenko, r.g.adams, n.davey}@herts.ac.uk
[2] Photonics Research Group, School of Engineering and Applied Science, Aston University
Birmingham B4 7ET, U.K.
{slaterbm, s.a.boscolo, s.k.turitsyn}@aston.ac.uk

Abstract. Improving bit error rates in optical communication systems is a difficult and important problem. The error correction must take place at high speed and be extremely accurate. We show the feasibility of using hardware implementable machine learning techniques. This may enable some error correction at the speed required.

1 Introduction

Performance of a fibre-optic communication link is typically affected by a complex combination of random processes (such as amplified spontaneous emission noise, polarization mode dispersion and so on) and deterministic or quasi-deterministic effects (e.g. nonlinear inter- and intra-channel signal interactions, dispersive signal broadening, various cross-talks and so on) that result from particular system design and operational regimes. Any installed fibre link has its specific transmission impairments, its own signature of how the transmitted signal is corrupted and distorted. Therefore, there is a great potential in the application of an adaptive signal post-processing that can undo some of the signal distortions, or to separate line-specific distortions from non-recoverable errors. Signal post-processing in optical data communication can offer new margins in system performance in addition to other enabling techniques. A variety of post-processing techniques have been already used to improve overall system performance, e.g. tunable dispersion compensation, electronic equalization and others (see e.g. [1-4] and references therein). Note that post-processing can be applied both in the optical and electrical domain (after conversion of the optical field into electrical current). Application of electronic signal processing for compensation of transmission impairments is an attractive technique that became quite popular thanks to recent advances in high-speed electronics.

In this work we apply standard machine learning techniques to adaptive signal post-processing in optical communication systems. To the best of our knowledge this is the first time that such techniques have been applied in this area. One key feature of this problem domain is that the trainable classifier must perform at an extreme speed,

optical communication systems typically operate at bit rates of around 40 GHz. We demonstrate a feasibility of bit-error-rate improvement by adaptive post-processing of received electrical signal.

2 Background

At the receiver (typically after filtering) the optical signal is converted by a photodiode into the electrical current. Detection of the digital signal requires discrimination of the logical 1s and 0s using some threshold decision. This can be done in different ways (e.g. by considering currents at a certain optimized sample point within the bit time slots or by analyzing current integrated over some time interval) and is determined by a specific design of the receiver. Here without loss of generality we assume that discrimination is made using current integrated over the whole time slot. Note that the approach proposed in this paper and described in detail below is very generic and can easily be adapted to any particular receiver design. To improve system performance and minimize the bit-error-rate, we propose here to use a method to adjust the receiver by sending test patterns to transmission impairments specific for a given line. This is achieved by applying learning algorithms based on analysis of sampled currents within bit time slots and adaptive correction of the decisions taking into account accumulated information gained from analysis of the signal waveforms.

3 Description of the Data

The data represents the received signal taken in the electrical domain after conversion of the optical signal into an electrical current. The data consists of a large number of received bits with the waveforms represented by 32 real numbers corresponding to values of electrical current at each of 32 equally spaced sample points within a bit time slot. A sequence of 5 consecutive bits is shown in Figure 1. As already explained the pulse can be classified according to the current integrated over the width of a single bit. For each of time slots in our data we have the original bit that it represents. Therefore the data consists of 32-ary vectors each with a corresponding binary label.

In all we have a stream of 65536 bits to classify. As already explained categorising the vast majority of these bits is straightforward. In fact with an optimally set electrical current integrated over the whole time slot (*energy threshold*) we can correctly classify all but 1842 bits correctly. We can therefore correctly classify 97.19% of the data, an error rate of 2.81%. This is however an error rate significantly too high. The target error rate is less than one bit in a thousand, or 0.1%. Figure 2 (a) gives an example of a misclassification. The middle bit of the sequence is a 0 but is identified from its energy as a 1. This is due to the presence of two $1's$ on either side and to distortion of the transmitted signal. It would be difficult for any classifier to rectify this error.

However other cases can be readily identified by the human eye and therefore could be amenable to automatic identification. Figure 2 (b) shows an example where the bit pattern is obvious to the eye but where a misclassification actually occurs. The central bit is a 1 but is misclassified as a 0 from its energy alone.

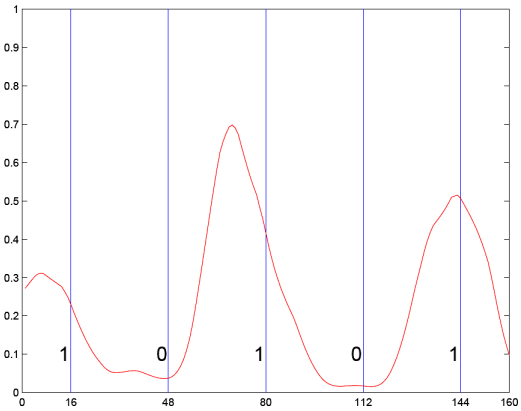


Fig. 1. An example of the electrical signal for a stream of 5 bits - 1 0 1 0 1.

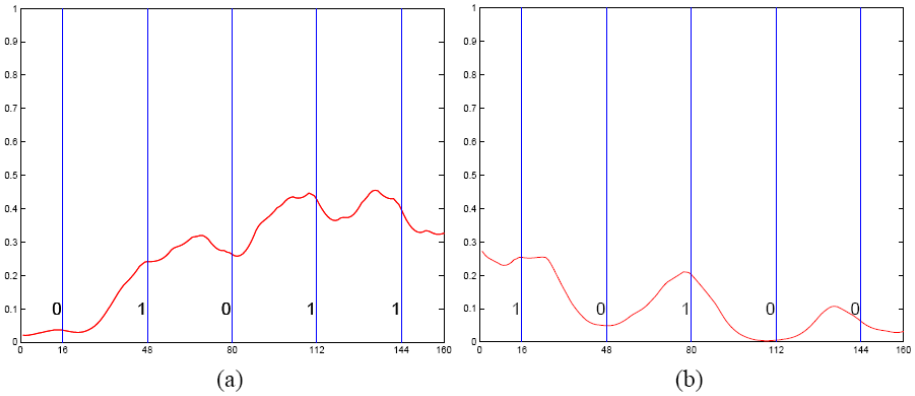


Fig. 2. (a) An example of a difficult error to identify. The middle bit is meant to be a 0, but jitter has rendered it very hard to see; (b) The central bit has been dragged down by the two 0s surrounding it and is classified as a 0 from its energy. However to the human eye the presence of a 1 is obvious.

3.1 Representation of the Data

Different datasets can be produced depending on how the original electrical current is represented. As well as representing a single bit as a 32-ary vector (called the *Waveform*-1 dataset), it can also be represented as a single energy value (the sum of the 32 values, *Energy*-1).

As described above of the 65536 bits, all but 1842 are correctly identified by an energy threshold. There are 32 distinct streams of 5 bits and 9 of these are represented in the misclassified class with a high frequency from 5.86% to 21.17% among the 1842 misclassified cases. These nine sequences are shown in Table 1.

Table 1. Nine sequences for which difficulties are most likely to occur.

0 0 1 0 0
0 0 1 0 1
0 1 0 1 0
0 1 0 1 1
1 0 0 1 1
1 0 1 0 0
1 0 1 0 1
1 1 0 1 0
1 1 0 1 1

As can be seen the majority of these involve a $1\ 0\ 1$ or $0\ 1\ 0$ sequence around the middle bit, and these are the patterns for which difficulties are most likely to occur. Therefore we may also want to take advantage of any information that may be present in adjacent bits. To this end we can form windowed inputs, in which the 3 vectors representing 3 contiguous bits are concatenated together with the label of the central bit being the target output (*Waveform*-3). It is also possible that using adjacent bit information by simply taking 3 energy values instead of the full waveform (*Energy-3*), or using information from a window of 3 bits, with 1 either side of the target bit (*Energy-Waveform-Energy*). Table 2 gives a summary of all the different datasets.

Table 2. The different datasets used in the first experiment.

Name	Arity	Description
Energy-1	1	The energy of the target bit
Energy-3	3	The energy of the target bit and one bit either side
Waveform-1	32	The waveform of the target bit
Waveform-3	96	The waveform of the target bit and the waveforms of the bits on either side
Energy-Waveform-Energy (E-W-E)	34	The waveform of the target bit and the energy of one bit either side of the target bit

4 Approaches Used

4.1 Easy and Hard Cases

One difficulty for the trainable classifier is that in this dataset the vast majority of examples are straightforward to classify. The hard cases are very sparsely represented, so that, in an unusual sense, the data is imbalanced. Figure 3 is a diagram of error rates of 0 and 1 as a function of the energy threshold. It shows that if the energy threshold is set to roughly 2.5, then those bits with energy less than this threshold are correctly classified into the 0 class; on the other hand, if the energy threshold is set to about 11, then those bits with energy greater than this are correctly classified into the 1 class. The optimal energy threshold to separate two classes is 5.01, in which case, 1842 of 65532 are incorrectly classified - a bit error rate of 2.81%. Using this threshold we divide the

data into easy and hard cases, that is, those classified correctly by the method are easy ones, otherwise they are hard cases.

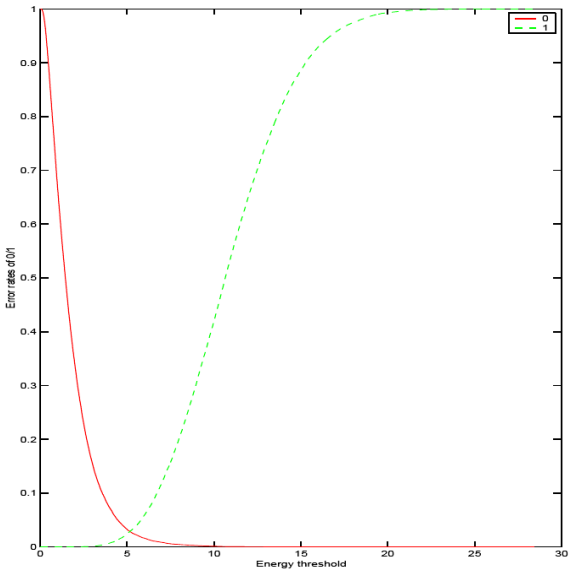


Fig. 3. A diagram of error rates of 0 and 1 as a function of an energy threshold.

4.2 Visualisation using PCA

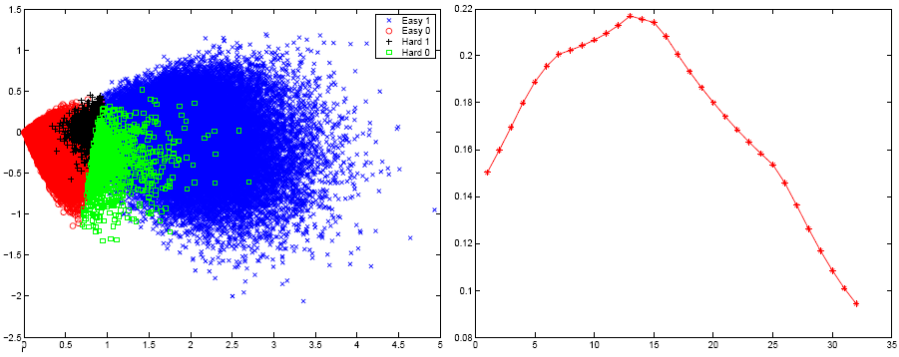
Before classifying bits into two classes, we first look at the underlying data distribution by means of Classical principal component analysis PCA [5], which linearly projects data into a two-dimensional space, where it can be visualised.

We visualise the easy cases using PCA, then project the hard cases into the same PCA projection space. The result is shown in Figure 4 (a). It shows that unsurprisingly the easy 0 and 1 classes are linearly separable. Interestingly, the hard 0 and 1 classes are for the most part also linearly separable. However, the hard 1s have almost complete overlap with the easy 0s, and the hard 0s have almost complete overlap with the easy 1s.

Figure 4 (b) is the eigenwave of the first component in the PCA analysis, which accounts for 86.5% of the total variance.

4.3 Single Layer Neural Network

As already described the classifiers need to be operationally very fast. Therefore the main classifier we use is a simple single layer neural network (SLN) [5]. Once trained (this is done off-line in advance) an SLN can be built in hardware and function with great speed. For comparison purposes a classifier that uses just an optimal energy threshold is implemented, where the threshold is the one giving the maximum accuracy rate (97.19%).



(a) (b)

Fig. 4. (a)Projection of the easy set using PCA, where the hard patterns are also projected into the easy ones' first two principal components space; (b) Eigenwave of the first principal component.

4.4 Identifying Difficult Cases using the Energy

As mentioned in Section 4.2, once the hard cases have been separated from the easy cases, it is possible to linearly separate the 1s and 0s for even the hard cases. Therefore, the question is how to distinguish between easy and hard cases. One way to tackle the problem is to use an energy threshold band. So we can set two thresholds, E_{min} and E_{max}, with $E_{min} \leq E_{max}$, such that if the energy of a bit is less than E_{min} then that bit is definitely a 0, and if is greater than E_{max} then it is a 1. For those bits whose energy lies between E_{min} and E_{max}, they can be considered as difficult cases. Experiment 2 describes how an energy threshold band can be used to select difficult cases which are subsequently used as training data to an SLN.

4.5 Gaussian Mixture Model (GMM)

Another approach we applied in this work is to use a Gaussian mixture model [5].

Distinguish between easy and hard cases can be performed by modelling the class-conditional probability $p(\mathbf{x}|c_i)$ for each easy and hard class first, then by calculating corresponding posterior probabilities using Bayes' theorem.

Since it is usually insufficient to model the conditional density by a single Gaussian distribution, we apply a Gaussian mixture model for each class-conditional probability density. In a Gaussian mixture model, the probability density function of each class is independently modelled as a linear combination of Gaussian basis functions. The number of basis functions, their position and variance and their mixing coefficients are all parameters of the model.

In a Gaussian mixture model, $p(\mathbf{x}|c_i)$ is of a linear combination of component densities $p(\mathbf{x}|j, c_i)$, and be written as follows:

$$p(\mathbf{x}|c_i) = \sum_{j}^{M} p(\mathbf{x}|j, c_i)P(j) \ , \tag{1}$$

where for each component j, we have a Gaussian distribution function

$$p(\mathbf{x}|j, c_i) = \frac{1}{\sqrt{(2\pi)^D |\mathbf{\Sigma}_j|}} \times \exp\left\{ -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_j)^T \mathbf{\Sigma}_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j) \right\} , \tag{2}$$

in which case $\boldsymbol{\mu}_j$ and $\mathbf{\Sigma}_j$ are mean and covariance matrix of each component j respectively. $P(j)$ in equation (1) satisfies

$$\sum_{j=1}^{M} P(j) = 1, \qquad 0 \leq P(j) \leq 1, \tag{3}$$

which guarantee that $p(x|c_i)$ is a valid density function.

The error function is defined as the negative log-likelihood for the dataset given by

$$E = -\ln \mathcal{L} = -\sum_{n=1}^{N} \ln p(\mathbf{x}^n|c_i) = -\sum_{n=1}^{N} \ln \left\{ \sum_{j}^{M} p(\mathbf{x}^n|j, c_i) P(j) \right\}. \tag{4}$$

The *expectation-maximisation* (EM) algorithm [6] is used to estimate parameters $P(j)$, $\boldsymbol{\mu}_j$, and $\mathbf{\Sigma}_j$ of a mixture model for an optimal fit to the training data.

In our experiment, we first estimated parameters of each class-condition density from the training dataset, where the data has been divided into an easy and hard subset using the optimal energy threshold. Then we reassign the class membership for each case using Bayes' theorem, that is

$$P(c_i|x) = \frac{P(c_i)p(x|c_i)}{\sum_{i=1}^{C} P(c_i)p(x|c_i)}. \tag{5}$$

Finally two SLNs were trained on these two reassigned classes. For a new test case, it is given either an *easy* or *difficult* class label using eq.(5), then it is forwarded to the corresponding trained SLN network based on its class membership to give the final discrimination.

5 Experiments

5.1 The First Experiment

We segment the data into 10-fold cross-training/validation sets and one independent test set. Each distinct segment has 5096 easy cases and 148 hard ones. Therefore, each training set includes 47196 cases and each validation set has 5244 cases in total; the independent test set has 12730 easy ones and 362 hard ones. The results reported here are therefore evaluations on the independent test set and averages over the 10 different validation sets. The main results are given in Table 3.

The classifiers do give an improvement over the optimal energy threshold method (*Energy*-1), with the SLN using the Waveform-3 dataset giving the best result. Interestingly the very simple classifier of the SLN/Energy-3 combination nearly decreased the error rate 42% on both validation and test sets when compared to the optimal threshold method. This classifier is simply a single unit with 3 weighted inputs.

To examine more closely how a threshold band performs on the waveform-3 dataset Experiment 2 was undertaken.

Table 3. The results of classifying the different validation and test sets for the different data representations. We also give the standard deviation for the validation sets.

Dataset	Validation Sets			The independent test set		
	mean errors		mean accuracy	errors		accuracy
	easy set	hard set		easy set	hard set	
Energy-1	0	148	97.180	0	362	97.23
Energy-3	22 ± 5	64 ± 7	98.366 ± 0.164	59	148	98.419
Waveform-1	22 ± 5	51 ± 5	98.608 ± 0.136	45	122	98.724
Waveform-3	22 ± 5	50 ± 8	98.633 ± 0.139	45	116	98.770
E-W-E	21 ± 6	50 ± 6	98.642 ± 0.140	49	123	98.686

5.2 The Second Experiment

Three different energy threshold bands are used to filter out the easy cases as discussed in Section 4.4. An SLN is then applied to each of the resultant difficult sets of size approximately 16700, 20600 and 24400, respectively. For the test set, above and below the threshold band, are classified appropriately, and the rest, the difficult set, are classified using the SLN. The results are given in Table 4.

Table 4. The results of classifying the different validation and test sets for each of the threshold bands. We also give the standard deviation for the validation sets.

Band	Validation Sets			The independent test set		
	mean errors		mean accuracy	errors		accuracy
	easy set	hard set		easy set	hard set	
$2.5 \leq x \leq 10.5$	23 ± 6	47 ± 5	98.659 ± 0.117	47	113	98.778
$2.0 \leq x \leq 11.0$	23 ± 5	48 ± 5	98.658 ± 0.122	42	114	98.808
$2.0 \leq x \leq 12.5$	22 ± 5	48 ± 5	98.661 ± 0.132	43	115	98.793

It can be seen that in general there is a classification improvement in the hard set on both validation and test sets when compared with results in Table 3. Using a band range from 2.0 to 11.0 give the best result on the independent test set over all our experimental results.

5.3 The Third Experiment

In this experiment, we divide the dataset into easy and hard subsets using Gaussian mixture models as discussed in section 4.5. For each class (easy/hard), we used a two-gaussian mixture with diagonal covariance matrices. The results are shown in Table 5.

It gives the best mean accuracy on the validation sets in all experiments, but not in the independent test set.

Table 5. The results of classifying the different validation and test sets. We also give the standard deviation for validation sets.

Model	Validation Sets			The independent test set		
	mean errors		mean accuracy	errors		accuracy
	easy set	hard set		easy set	hard set	
2 *gmms 'diag'*	21 ± 4	48 ± 5	98.675 ± 0.126	49	119	98.717

6 Discussion

The fast decoding of a stream of data represented as pulses of light is a commercially important and challenging problem. Computationally the challenge is in the speed of the classifier and the need for simple processing. We have therefore restricted our classifier to be, for the most part, an SLN and the data is either a sampled version of the light waveform or just the energy of the pulse. Experiment 1 showed that an SLN trained with the 96-ary representation of the waveform gave the best performance, reducing the bit error rate from 2.8% to 1.23%. This figure is still quite high and we hypothesised that the explanation was the fact that despite the data set being very large, 65532 items, the number of difficult examples (those misclassified by the threshold method) was very small and dominated by the number of straightforward examples. To see if we could correctly identify a significant number of these infrequent but difficult examples we undertook experiments 2 and 3 .

Although there was a small improvement obtained by using energy threshold band, further work is needed to determine if there is an optimal band size. Similar results were obtained by experiment 3.

This is early work and much of interest is still to be investigated, such as representational issues of the waveform, threshold band sizes, and other methods to identify difficult cases.

References

1. Bulow, H. (2002) Electronic equalization of transmission impairments, in. OFC, Anaheim, CA, Paper TuE4.
2. Haunstein, H.F. & Urbansky, R. (2004) Application of Electronic Equalization and Error Correction in Lightwave Systems. In *Proceedings of the 30th European Conference on Optical Communications (ECOC)*, Stockholm, Sweden.
3. Rosenkranz, W. & Xia, C. (2007) Electrical equalization for advanced optical communication systems. *AEU - International Journal of Electronics and Communications* 61(3):153-157.
4. Watts, P.M., Mikhailov, V., Savory, S. Bayvel, P., Glick, M., Lobel, M., Christensen, B., Kirkpatrick, P., Shang, S., & Killey, R.I. (2005) Performance of single-mode fiber links using electronic feed-forward and decision feedback equalizers. *IEEE Photon. Technol. Lett* 17 (10):2206 - 2208.
5. Bishop, C.M. (1995) *Neural Networks for Pattern Recognition.* New York: Oxford University Press.
6. Dempster, A.P., Laird, N.M, & Rubin, D.B. (1977) Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Stat. Soc. B, 39 :1–38.*

An Authentication Protocol for Wireless Ad Hoc Networks with Embedded Certificates

Robert E. Hiromoto[1] and J. Hope Forsmann[2]

[1] Department of Computer Science, University of Idaho, Moscow, Idaho 83844-1010, U.S.A.
hiromoto@uidaho.edu
[2] Idaho National Laboratory, 2525 N. Fremont, Idaho Falls, Idaho 83415, U.S.A.
Hope.Forsmann@inl.gov

Abstract. Wireless ad hoc networks may be configured as a fixed topology of sensors or allowed to migrate as mobile nodes. The flexibility of these networks, therefore, provides opportunities for their deployment in real-time and in adverse situations as encountered in civil and military applications. These advantages are, unfortunately, curtailed by the unconstrained nature of these networks in providing a trusted level of connectivity. The establishment of secret keys and the authentication of trusted ad hoc group nodes are essential elements for a secure network. In this paper, we develop an authentication protocol for wireless ad hoc networks that is derived from a canonical splitting of time- and frequency-space (channel) over which information propagates under the constraint of a collision-avoidance protocol.

1 Introduction

The rapid deployment of wireless ad hoc networks even under the most severe conditions, has elevated their prominence in all aspects of homeland security and military communications. However, these infrastructure-less networks are prone to require explicit network cooperation, route maintenance in the event of link failure, and information losses resulting from data packet collisions. These routing protocols for dynamic networks have been discussed in [11] and [19]. Furthermore, without data encryption and trusted node authentication, these networks are vulnerable to malicious attacks that may be categorized by the following techniques: eavesdropping; man-in-the-middle; replay; impersonation; session hijacking; reflection; and interleaving attacks.

Intelligent information processing provides network security approaches that has prompted researchers to propose numerous security protocols for the establishment of secret keys; and in particular, the authentication of wireless ad hoc network nodes. A taxonomy and classification of services that rely on authentication are studied by [17, 10]. The use of local time-stamp authentication protocols [14], a hop-by-hop authentication [26], recommendation and reference protocol that is inspired by human behavior [24], location-limited channels with pre-authentication [20, 1], an end-to-end data authentication scheme that relies on mutual trust between nodes [23], a threshold secret sharing using an identity-based cryptosystem to provide end-to-end authentication [6], under loosely time synchronized nodes with one-way chain as a cryptographic key where each such value is associated with a time interval [25].

In [3], a self-organized public-key management scheme is proposed that maintains no centralized services, yet, allows each node to generate private-public key pairs; to issue public key certificates for its neighboring nodes; and to perform authentication via a chain of public-key certificates regardless of the network partition. Several refinements to this proposed scheme are presented in [2] where each certificate is issued with a limited validity time period that contains its issuing and expiration time. After expiration of the valid time period, new certificate are issued to its neighboring nodes. In addition, nodes in alliance with their neighboring nodes form trust groups that communicate using the group's private and public key. The authenticity between nodes is performed by validating their cached public-key certificates chain. The public keys and certificates are modeled as a directed graph for which transitive properties can identify nodes belonging to a trust group.

A major difficulty with the solution specifications for certificate-based authentication, as described above, is the amorphous structure of wireless ad hoc networks in both static and dynamic systems. The unconstrained nature of malicious attacks in such networks are, therefore, difficult if not impossible to protect against. Geometrically, the use of public/private-key certificates is a one-dimensional parameterization of the problem domain. Whether in the form of a certificate-chain or a cluster-key shared with multiple neighboring nodes, the various approaches are limited by their reliance on artificial solutions that do not embrace the dynamical properties or behavior of these systems. With this in mind, we argue for a "canonical" reformulation of the authentication problem in wireless ad hoc networks and derive an alternative solution technique that is described in a two-dimensional setting.

If security is deemed critical then as in all practical engineering considerations, tradeoffs must be identified and enforced. To this end, we propose a different approach to authentication in wireless ad hoc networks that imposes a collision-avoidance policy on data transmission that has the property of untangling the authentication process such that certificates are no longer exchanged explicitly.

In the remainder of the paper, we outline the proposed approach, and provide possible means of implementations.

2 An Authentication Protocol with Embedded Certificates (APEC)

The proposed authentication protocol introduces a sequence of unique, non-overlapping communication time-slots that are assigned to each authenticated node of the network. Time-slots are used as an implicit certificate to ensure trust. In addition, we impose on top of each time-slot a pseudo-randomly corollated frequency channel F_i over which a data packet must be sent in time-slot T_i. This dependence between time-slots and frequency channels introduces a two-dimensional description of network communication and thus allows for a clear decomposition of the problem domain.

The protocol is described in terms of a cluster with a single cluster head or administrator node as follows:

1. Assume that an initial authentication phase has verified the trust of nodes that have joined the cluster.

2. The cluster head sends a public key to all nodes in the cluster.
3. The public key is used by every node to select the appropriate addem, multipliers, etc to construct a common pseudo-random number generator (PRNG) using a hash-table. Two such PRNGs are constructed RandT() and RandF, that produces the sequences for T_i and F_i; respectively.
4. From the public key, a random seed is produce and used to seed RandT(). Each T_i that is generated is in turn used as a seed to generate a corresponding F_i from the pseudo-random number generator RandF().
5. After each complete communication time period, the order of the sequence of time-slots are permuted.

At the end of these steps, a sequence of T_m time-slots with their randomly correlated frequency channel, F_m, are created, and forms m-coordinate, 2-tuples (T_i, F_i). In addition, the details of the PRNG construction are known to each entrusted node in the network; and therefore, allows a deterministic recreation of all present and future m-coordinate pairs. This becomes important when a node cannot send data packets to the cluster head in one hop but instead requires a multi-hop link to reach its destination.

3 A Space-Time Coordinate Basis with Collision-Avoidance

Communication can be parameterized as a space-time, 2-tuple coordinate basis. Time (time-slot), T_i, is taken as the moment (over the time-slot duration) that a message is sent or received. Space is the physical channel over which the message is sent or received. For a wireless network this channel is associated with a particular radio frequency, F_i, over which a message is sent out or received. We characterize this 2-tuple by (T_i, F_i).

In order to avoid ambiguities, it is important that a collision-avoidance protocol be adopted for a wireless network so that no two data packets arrive at a given destination during the same time-slot. This property is reflected in the following definition:

Definition: Two valid communication coordinates (T_i, F_j) and (T_r, F_s) within a wireless ad hoc network must necessarily satisfy the condition that $T_i \neq T_r$; however, it is not sufficient since some time-slots may be forbidden. (collision-avoidance assumption).

The collision-avoidance assumption restricts a clustered network of wireless nodes to communicate only over predetermined, non-overlapping send or receive time-slots. As a consequence, certain types of external attacks can be detected if two or more distinct data packets arrive during the same time-slot.

The collision-avoidance assumption is examined by Forsmann et al., [8]. The perspective of their study is the QoS of unmanned aerial vehicles for autonomous formation flight. For the sake of this paper, we consider only the cluster formation with a single cluster head node, and assume that all communication is directed between a cluster head and each individual node within the cluster. It is, however, possible that the mobility of each vehicle (node) may require the use of intermediate nodes to form a multi-hop message-forwarding link if the sending node drifts out of radio range with the cluster

head. Multi-hop network routing links are addressed using a modified version of the AODV route-discovery protocol as described by Forsmann et al.

Under the assumptions of a collision-avoidance protocol and the two-dimensional representation of communication events within a wireless mobile ad hoc network, we introduce a cryptographic *confusion* algorithm that replaces the traditional certificate-based authentication procedures that have been attempted for these infrastructure-less networks. The following desired set of properties provide the foundation for our approach:

Property 1: A unique time-slot and frequency channel pair is assigned to only one node within the network.

Property 2: It is desirable to *conceal* the selection of the radio frequency channel F_i in each round of a node's communication time-slot sequence.

Property 3: For each communication period a new sequence order for a given node's time-slots T_i are assigned.

Property 4: The length of a time-slot duration depends upon the time-skewing experience by each node's clock. It will be assume that each node is equipped with a GPS device for time synchronization.

The process for *concealing* the selection of T_i and F_i is performed using a pair of *orthogonal* PRNGs as outline in Frederickson et al., [9]. Their work examined the issue of reproducibility of Monte Carlo random walk algorithms for parallel execution. The use of a single PRNG in a parallel processing environment has the effect of reordering the sequence of pseudo-random numbers that are generated. Although this additional randomness may seem advantageous, the final result cannot be compared for correctness with the results of the sequential execution that uses the same PRNG. Without this verification, it is not clear whether or not the parallel program has been implemented correctly.

Analogously, a wireless ad hoc network represents a collection of parallel processing nodes that can be assured a unique $\{T_i, F_i\}$ pair for each entrusted node without incurring inter-node co-channel interference. To ensure this property, each node must be given a different random number sequence in a deterministic fashion. Figure 1 illustrates the *orthogonal* structure for a pair of PRNG that result in a reproducible random number scheme.

Below, we summarize some typical PRNGs in use and techniques used in creating parallel pseudo-random number generators (PPRNG).

4 Pseudo-Random Number Generation

4.1 Types of Generators

The following list are commonly used pseudo-random number generators:

- Additive and subtractive Lagged Fibonacci (LF)
- Generalized Shift Register (GSR)

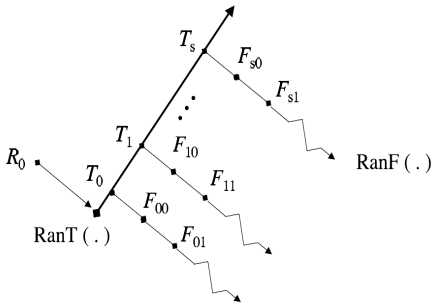


Fig. 1. $RanT$ and $RanF$.

- Multiplicative Linear Congruential (MLC) and
- Combination Generators (CG)

A uniform (double precision) floating-point random number sequence x_i in the interval $[0, 1)$ or $(0, 1)$ are produced by the following recursions:

MLC $(a, m = 2^l)$:

$$X_i = aX_{i-1} \text{mod m};\ x_i = X_i/m, (i = 1, 2, ...)$$

GSR $(r, s, \oplus_l)$:

$$X_i = X_{i-r} \oplus_l X_{i-s};\ x_i = X_i/2^l, (i = r, r+1, ...) \tag{1}$$

LF $(r, s, \pm m = 2^l)$:

$$X_i = X_{i-r} \pm X_{i-s}\ mod\ 2^l;\ x_i = X_i/2^l, (i = r, r+1, ...)$$

CG:

$$Z_i = X_i \odot Y_i; (i = r, r+1, ...)$$

where $\odot$ is either the exclusive-or-operator or addition modulo some integer m, and X and Y are sequences from two independent generators. It is best if the cycle length of the two generators is relatively prime, for this implies that the cycle length of Z will be the product of that of the basic generators. One can show that the statistical properties of Z are no worse than those of X or Y [15]. Good combined generators have been developed by L'Ecuyer [13], based on the addition of Linear Congruential sequences.

4.2 Parallel Pseudo-Random Number Generation

Historically, the introduction of PPRNG was an attempt to address various efficiency issues inherent in the parallel execution of scientific applications such as lattice gauge and Ising model calculations. Within this context, the quality of PPRNGs have been studied by a number of researchers [4], [5], [16], [18],[21]. More recently, their use in games and graphics have drawn interest [22], [12].

As a brief overview, we list below several approaches that have been studied and applied in the design of PPRNGs. The descriptions are only meant to provide a high level, structural description of several different possible parallel techniques.

- Leapfrog – The pseudo-random sequence is partitioned in turn among nodes so that node i gets the sequence $r_i, r_{i+m}, \ldots$, where m is the total number of nodes or time-slots.
- Sequence splitting – The sequence is partitioned by splitting it into non-overlapping contiguous sections. In particular, if it is known that the cycle length of a pseudo-random sequence is l_c then node $i+1$ gets the sequence $r_{[il_c/m]+1}, r_{[il_c/m]+2}, \ldots$ $r_{(i+1)l_c/m}$.
- Independent sequences – For some generators, the initial seeds can be chosen in such a way as to produce long period independent subsequences on each processor.
- Cycle parameterization – For PRNGs' that have more than one distinct cycle, it is possible to select a seed that begins in one cycle and a different seed that begins in a different cycle resulting in two sequences that do not overlap. The Lagged Fibonacci Generator is an example of a generator with this property. By associating each seed with a cycle number i, parameter i determines the cycle from which the sequence is drawn.

4.3 Reproducibility and Cycle Lengths

In order to ensure reproducibility, each node must be given a different random number sequence in a deterministic fashion. This can be accomplished by creating a random seed unique to each node (e.g., the time-slot) that in turn is used to seed a different (*orthogonal*) random number generator that produces a second "independent " pseudo-random (frequency channel) sequence. This deterministic approach is reproducible and requires no inter-node communication, whose costs are typically high. More importantly, the absence of inter-node communication is extremely desirable from a security standpoint.

A pseudo-random number generator defines at most 2^b different configurations (states) where b is the number of bits that represent the number of possible states. The sequence, after generating 2^b different configurations, must repeat .

It is desirable, therefore, to use PPRNGs with the longest cycle length to guarantee the minimal biasing of the results. This, however, does not eliminate the advantages of a small cycle length as long as the coordinates that are produced are not exactly the same. Hence even if the random number sequence repeats, if the $\{T_i, F_i\}$ pair remain unique, the bias remains low.

As a consequence, it is assumed that the correlation between pairs of pseudo-random number generators has little effect on network authentication unless the bias results in (T_i, F_j) cycles.

4.4 Security

Pseudo-random number generation is a process that either provides security or is vulnerable to attacks that can compromise the security of a system. The PRNG process is provocatively attractive to attackers because it is typically a single isolated, hardware/software component whose deterministic output is disguised as random. If an attacker can substitute pseudo-random bits generated in a way that can be predicted, security is totally compromised.

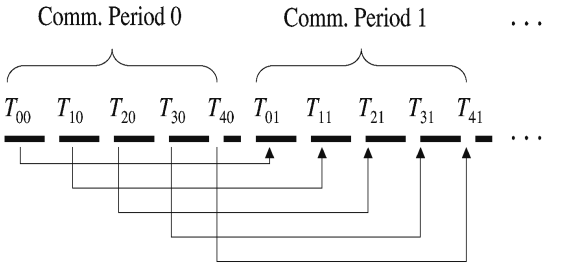


Fig. 2. Direct Mapping Constant Increment $\delta t_i = \tau$.

5 Time-Slot Selection

The sequence of time-slots for one complete communication period

$$\{T_0, T_1, \ldots, T_n, \ldots, T_{m-1}, \}$$

A communication period, τ, is defined as the sum over all m time-slots ($m \geq n$) in the closed interval $[T_0, T_{m-1}]$,

$$\tau = \sum_{i=0}^{m-1} T_i.$$

The requirement ($m \geq n$) for an n node wireless network is imposed for three reasons. First, m may be viewed as a cryptographic parameter that introduces a level of *confusion*, as is shown later. Second, the choice of m introduces idle time-slots over which no data packets should be sent or received. Third, m provides a slight window (delay) between the end of one time-slot and the beginning of the next.

After each communication period, a new sequence of unique non-overlapping time-slots are assigned to each node. This assignment is defined by the following equation:

$$T_{j,i+1} = T_{j,i} + \delta t_i, \tag{2}$$

for $i, j = 0, 1, \ldots, m$. In this notation index j is the node number and i is the iteration or communication period number.

There are several ways of selecting δt_i. One simple approach is to maintain the order of time-slots for each node from one iteration to the next. Figure 2 illustrates the case for $\delta t_i = \tau$.

A second and possibly more secure approach is to apply a permutation on the order of the previous time-slot sequence. The permutation of the sequence is shown in Fig. 3.

For a permutation π, we associate a permutation matrix P_π and a distance matrix D_π whose values represent the number of column swaps from their original position to their final position in the permutation. For example, if $\pi = \{0, 3, 2, 1\}$, then

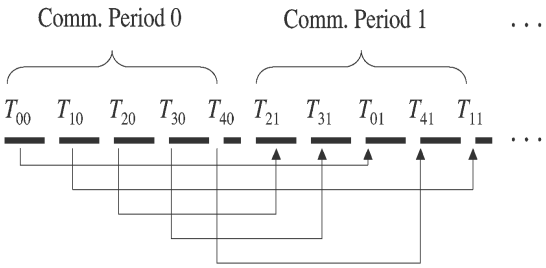


Fig. 3. Permutation Mapping Variable Increment δt_i.

$$P_\pi = \begin{vmatrix} 1\,0\,0\,0 \\ 0\,0\,0\,1 \\ 0\,0\,1\,0 \\ 0\,1\,0\,0 \end{vmatrix},$$

and

$$D_\pi = \begin{vmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2 \\ 0 & 0 & -1 & 0 \\ 0 & +2 & 0 & 0 \end{vmatrix}.$$

The distance matrix can then be used to define a time-slot increment vector $\boldsymbol{\delta t_i}$ given by

$$\boldsymbol{\delta t_i} = \tau\theta^T + \delta t D_\pi \cdot \theta, \tag{3}$$

where

$$\theta = \begin{vmatrix} 1 \\ 1 \\ . \\ . \\ . \\ 1 \end{vmatrix}$$

of length m, and

$$\delta t = \tau/m.$$

The components of Eqn. 3 are then applied to Eqn. 2.

5.1 Multi-Hop Authentication

The requirement for pseudo-random reproducibility is most important in the instant that a network topology has migrated beyond a single-hop radio range to a multi-hop environment. In such a situation, a routing protocol is employed to initiate route discovery. As part of the data transfer process, the destination node may have need of the source nodes identity and the chosen route path. To obtain this information any entrusted node intercepting the packets can recompute the node's time-slot seed using the original public-key and compute the appropriate frequency channel for the current communication period. With this information, the source node and route path can be identified. The implications of this scenario is that only trusted nodes can determine the communication route, destination and source nodes in a calculable time frame.

5.2 Security Complexity

APEC is a cryptographic system protocol based on a source of random bits, whose output is used in creating unique time-slot and frequency channel pairs. APEC employs two "*orthogonal*" pseudo-random number generators that cooperate in parallel to assign node-specific send/receive, time-slots that are in turn used to seed a pseudo-random assignment of time-slot, correlated frequency channels. In this construction, the sequence of frequency channels can be systematically applied in turn to the same time-slot but on different periods of the communication cycle.

The order of time-slots assigned to each node can remain in the same order or be reordered to increase cryptographic confusion. In this paper, a reordering of the time-slot sequence is presented using a simple permutation π.

The security complexity is stated by the following theorem.

Theorem: The brute force complexity for guessing the correct sequence of coordinate basis pair for a b-bit random sequence and a permutation of the sequence order for m time-slots is $O(2^{2b} \times m!)$.

Proof: A pseudo random number generator is a finite state machine with at most 2^b different states where b is the number of bits that represent the state. The brute force complexity of guessing the time-slots is O(2^b). However, since the frequency channels are coupled to the seed of a unique time-slot the complexity of the coupled random number states requires a total of O(2^{2b}). If we also apply a permutation to the previous time-slots sequence for every communication period the brute force complexity for m time-slots is O($m!$). The final complexity is the product of the time-slot, frequency channel and the permutation operations. □

6 Conclusions

Geometrically, the use of public/private keys as certificates for authentication in wireless ad hoc networks is a one-dimensional solution in a two- or higher-dimensional problem domain. The novel abstraction taken here is to design an authentication protocol that embodies the dynamic processes and captures the essential behavior of the

system in a "self-certifiable" manner. In this context, we achieve a canonical formulation of the authentication problem in wireless ad hoc networks and as such derive an alternative solution technique devoid of explicit certificate passing.

The contributions of this paper is the design of a new authentication protocol that relies on a collision-avoidance protocol to guarantee the detection of unauthorized attempts to compromise a wireless ad hoc network of trusted nodes. The protocol creates a random, yet deterministic sequence of coupled time-slots and associated frequency channels that are unique to each node in the network. For each communication period, the order of the sequence of time-slots may remain unaltered or the order can be rearranged deterministically without coordination via inter-node communication. If an analogy can be made between frequency channels to colors in the visual range, the communication in an APEC system would appear to be a coordinated light show.

References

1. D. Balfanz, D. K. Smetters, P. Stewart and H. Chi. Wong, "Talking to Strangers: Authentication in Ad-Hoc Wireless Networks," Symposium on Network and Distributed Systems Security (NDSS '02).
2. Chih-Peng Chang, Jen-Chiun Lin, Feipei Lai, "Trust-group-based authentication services for mobile ad-hoc networks," 1st International Symposium on Wireless Pervasive Computing, 16-18 Jan. 2006,
3. Srdjan Capkun, Levente Buttyan, Jean-Pierre Hubaux, "Self-Organized Public-Key Management for Mobile Ad-hoc Networks," in ACM International Workshop on Wireless Security, WiSe 2002.
4. Chiu, T. W. "Shift-register sequence random number generators on the hypercube concurrent computers," in G. C. Fox, editor, The Third Conference on Hypercube Concurrent Computers and Applications, Volume 2, pages 1421-1429 (1988).
5. Paul D. Coddington and Sung-Hoon Ko, "Techniques for empirical testing of parallel random number generators," Proceedings of the 12th international conference on Supercomputing, pp. 282 - 288 (1998).
6. H. Deng, A. Mukherjee, D. P. Agrawal, "Threshold and Identity-Based Key Management and Authentication for Wireless Ad Hoc Networks," International Conference on Information Technology: Coding and Computing (ITCC'04) (2004).
7. W. Diffie, P.C. van Oorschot, and M.J. Wiener, "Authentication and authenticated key exchanges," Designs, Codes and Cryptography 2 (1992).
8. J. Hope Forsmann, Robert E. Hiromoto, and John Svoboda, "A Time-Slotted On-Demand Routing Protocol for Mobile Ad Hoc Unmanned Vehicle Systems," SPIE 2007, Orlando Florida, April 9-12 2007.
9. P. Frederickson, R. Hiromoto, T. Jordan, B. Smith, and T. Warnock, "Pseudo-Random Trees in Monte Carlo," Parallel Computing, Vol. 1, No. 2, pp. 175-180, (December 1984).
10. S. Hahm, Y. Jung, S. Yi, Y. Song, I. Chong, and K. Lim, "Self-organized Authentication Architecture in Mobile Ad-hoc Networks," International Conference on Information Networking (ICOIN) (2005).
11. D.B. Johnson and D.A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," Mobile Computing, vol. 353, Kluwer Academic Publishers (1996).
12. Sylvain Lefebvre and Hugues Hoppe, "Perfect Spatial Hashing," ACM Trans. Graph. 25, 3, 579588 (2006).

13. P. L'Ecuyer, "Efficient and portable combined random number generators," Comm. of the ACM, 31:742774, 1988.
14. H. Lou, P. Zerfos, J. Kong, S. Lu, and L. Zhang, "Self-Securing Ad Hoc Wireless Networks," Seventh IEEE Symposium on Computers and Communications (ISCC'02).
15. G. Marsaglia, "A current view of random number generators," In Computing Science and Statistics: Proceedings of the XVIth Symposium on the Interface, pages 3-10, 1985.
16. M. Mascagni S. A. Cuccaro and D. V. Pryor, "Techniques for testing the quality of parallel seudorandom number generators", In Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing, pp. 279-284, Philadelphia, Pennsylvania, 1995. SIAM.
17. D. Park, C. Boyed, E. Dawson "Classification of Authentication Protocols: A Practical Appraoch", Proceedings of the Third International Workshop on Information Security.
18. O. E. Percus and M. H. Kalos, "Random number generators for MIMD parallel processors," J. of Par. Distr. Comput., 6:477-497, 1989.
19. C. Perkins, "Ad Hoc On Demand Distance Vector (AODV) Routing", Internet Draft, draft-ietfmanet-aodv-00.txt, November 1997.
20. F. Stajano and R. Anderson, "The Resurrecting Duckling: Security Issues in Ad-Hoc Wireless Networks", M. Roe B. Christianson, B. Crispo, editor, Security Protocols, 7th International Workshop Proceedings, LectureNotes in Computer Science. Springer Verlag, 1999.
21. A. Srinivasan, D. M. Ceperley and M. Mascagni, "Random Number Generators for Parallel Applications," in Monte Carlo Methods in Chemical Physics, D. Ferguson, J. I. Siepmann, and D. G. Truhlar, editors, Advances in Chemical Physics series, Volume 105, John Wiley and Sons, New York (1998).
22. Stanley Tzeng and Li-Yi Wei, "Parallel White Noise Generation on a GPU via Cryptographic Hash," Proceedings of the 2008 symposium on Interactive 3D graphics and games, (2008).
23. L. Venkatraman and D. Agrawal, "A Novel Authentication Scheme for Ad Hoc Networks," IEEE Wireless Communications and Networking Conference (WCNC 2000), vol. 3, pp. 1268-1273.
24. A. Weimerskirch and G. Thonet, "A Distributed Light-Weight Authentication Model for Adhoc Networks," Proc. of 4th International Conference on Information Security and Cryptology (ICISC 2001).
25. S. Zhu, S. Setia and S. Jajodia, "LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks," 10th ACM Conference on Computer and Communications Security (CCS '03) (2003).
26. S. Zhu, S. Xu, S. Setia and S. Jajodia, "LHAP: A lightweight hop-by-hop authentication protocol for ad-hoc networks," Proc. of ICDCS 2003 International Workshop on Mobile and Wireless Network (MWN 2003), May 2003.

Codesign Strategy based upon Takagi Sugeno Control Design and Real-Time Computing Integration

Héctor Benítez-Pérez

Departamento de Ingeniería de Sistemas Computacionales y Automatización
Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas
Universidad Nacional Autónoma de Mexico
Apdo. Postal 20-726, Del. A. Obregón, México D. F., CP. 01000, Mexico
Fax: ++52 55 5616 01 76, Tel: (*)++52 55 5622 36 23
hector@uxdea4.iimas.unam.mx

Abstract. Nowadays the idea of network control systems design considering the restriction results from schedulling analysis becomes a challenge based upon the persepctive of codesign point view since both analytic tools are pursued. A clear strategy is to define in cascade mode the scheduling analysis and afterwards the stability analysis of the respective control strategy. However, any modification in both structures has an integrated impact which is necessary to measure. In that respect the use of time delay impact is a suitable strategy to be followed and is explored in this paper. The use of codesign is to pursuit as a two objective strategy the definition of a valid metric that represents the effects in both, following the idea that stability analysis is affected according to the schedullability analysis. In both analysis a relaxation at the local conditions is feasible but it will have a global cost giving a non valuable approximation.

1 Introduction

Nowadays, the use of multiple tools for complex systems design like Real-Time distributed systems need any background in terms of design, for instance, the relation between analysis constrains expresed in different metrics like reliability, schedullability, safety, stability and so on. The need to relate these measures can be pursued in terms of a hollistic design or codesign sctrategy [4]. In order to define this kind of strategy it is necessary to determine the effects of each technique over the rest. Several approaches can be persued like decision trees, common metrics definitions, stochastic processes and others. However, this problem remains open in terms of a standard approximation amongst the complexity of the goal. One interesting approximation is based upon the codesign way of thinking, by choosing one specific aspect from each technique. Therefore, the individual achievemnt of every technique considering its effects over the rest should be pursuit. The objective of this paper is to review this approximation over a Real-Time Distributed System considering its effects over a specific application such as dynamic system. Specifically, this is studied by the use of schedulling and control design, where schedulability and strability analysis are reviewed to guarantee the feasability of this strategy.

Although this is dependant on the specific strategy to be followed, global characteristics such as the respective analysis can be pursued.

Further on, the union of different techniques allows a holistic view of the problem, although, the result can be restricted to specific algorithms and the inherent restriction of the case study.

Section 2 presents a review of structural codesign based upon the schedulling approximation. Section 3 presents control codesign where fuzzy logic control law is used in order to incorporate scheduler information onto stability analysis. Section 4 presents some concluding remarks of thi approximation.

2 Structural Codesign

The codesign proposal is based upon the iteration between schedulability and stability analysis following online approximation as shown in Fig. 1.

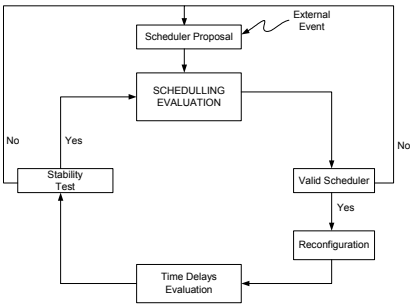


Fig. 1. Dynamic Reconfiguration in terms of Codesign strategy.

Any classical scheduller [1] bounds the time behaviour of the tasks in terms of their own priority where certain modification amongst them produces important differences. For instance, consumption time from tasks named as sensors (t_{sj}), actuators (t_{aj}) and controllers (t_{cj}) can be seen as follows:

$$\begin{aligned} t_{sj} &= t_{sj}^{*} + \Delta t_{sj}^{*} \\ t_{aj} &= t_{aj}^{*} + \Delta t_{aj}^{*} \\ t_{cj} &= t_{cj}^{*} + \Delta t_{cj}^{*} \end{aligned} \tag{1}$$

Where the total time spent by the tasks is equal to t_t^j

$$t_t^j = \sum_{i=1}^{N} t_{cj}^i + \sum_{i=1}^{M} t_{sj}^i + \sum_{i=1}^{P} t_{aj}^i \tag{2}$$

Index *i* is the number of tasks involved per structural elements like sensors (*M*), actuators (*P*) and controllers (*N*). Index j is the current scenario defined by the

scheduller. These time delyas that are the result of priority modification on the peripheral elements as individual manner should change the design parameters at the control law. At least these time delays provide enough information to perform an adecuate control law design. Fig. 2 shows how t_t^j should be bounded to Control period of time.

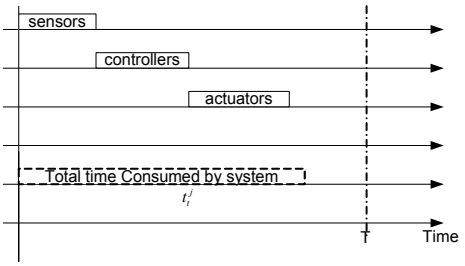


Fig. 2. Bounded Time Inherent to Control Period of Time.

Where T is a long enough time window where t_t^j should take place. Δt is the variation presented per element [12] during element actuations according to the pursued scheduler algorithm such as EDF, RM or FTT [1] [2] [3] [15].

To guarantee schedulability is necessary an effective performance from the control law [4]. This can only be pursued if only if the time delays exist within the bounded time delays used to design a suitable control law as a classical gain schedulling strategy. When task schedulling is performed, it implies a variation Δt giving a modification to the control law. Therefore the classical scehdulability analysis [1] can be modified in order to incorporate this kind of uncertainty giving the following result

$$U = \sum_{i=1}^{N} \frac{c_i + \Delta c_i}{P_i} \leq 1 \tag{3}$$

Where c_i represents the consumption time of each task, Δc_i is the related uncertainty, P_i is the related period, N is the number of tasks and U is the total relation between cosumptions and periods. This last value should be less than one in order to guarantee schedulability. It is important to deploy that any classical schedulling algorithm should fit into this condition as longs as the tasks are periodicals (which is the case herein) and the inherent uncertaintes should be fit into the same condition.

In fact, these time delays can be seen like a phase modification within the communication period from the involved processes. This scenario presents a complete phase modification at the entire system.

The communication network plays a key role in order to define the behaviour of the dynamic system in terms of time variance giving a nonlinear behaviour. In order to understand such a nonlinear behaviour, time delays are incorporated by the use of real-time system theory that allows time delays to be bounded even in the case of causal modifications due to external effects. In order to model this behaviour a

reconfigurable real time schedulling algorithm is proposed, named Structural reconfiguration algorithm (SRA).

This algorithm bounds Time delays through a real-time scheduling algorithm within communication network. According to Figure 1, structural reconfiguration takes place as a result of Earliest Deadline First (EDF) Schedulling algorithm and the associated user request. This reconfiguration causes a control law modification [1] which is the actual control law reconfiguration.

Schedulling approach pontentially modifies frequency execution and communication of tasks [5] in order to give certain priority to some of them during a bounded time as shown in Fig. 3. Furthermore, in this kind of strategy Tasks modifies their priority, it does not imply that neither the period nor the consumtion times are modified. Therefore the tasks would have a bounded delay within the sampling time wich is reflected as changing on the phase.

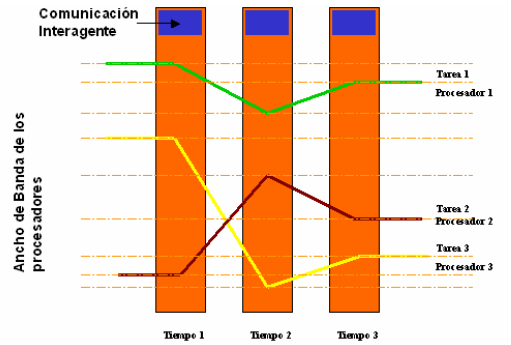


Fig. 3. Task Frequency modification as result of Scheduller.

Potential modifications onto schedulling approach deploy change in the priorities that affects time delays and the respective control law. The delays are measured as Δt [14] and bounded into the inherent control period of time [6] [7].

Now by taking partial results from schedulling algorithm like t_{sj} and the related Δt, the actual time delays are used at the control law for parameters design as shown in following section. The involved time delays are depicted as τ_j^i and come from this scheduling design. Other delays like actuators and control delays are not used in the design of the control law, although play an important role.

Therefore schedulling and control analysis merge together when time delays are complete bounded even in the case of time variance. The main restriction is in terms of predictable time delays.

The approach followed at the control reconfiguration does not take into account scheuller decision in a direct manner. It takes the time delyas as bounded values already defined and used to design a suitabe control law. Therefore, acording to current state plant values, the related fuzzy rule is selected.

3 Control Reconfiguration Approach

The control law is defined as a group of Fuzzy TKS [8] [9] [10] control law related to each local linear system. At the beginning the general structure of each fuzzy rule is:

$$r_i \text{ if } x_1 \text{ is } A_{1i}^c \text{ and } x_2 \text{ is } A_{2i}^c \text{ and } \ldots x_1 \text{ is } A_{li}^c \text{ then } f(k) = Q_i x(t) \qquad (4)$$

where $i = \{1,....,N\}$, N is the number of fuzzy rules, $\{x_1,....,x_1\}$ are current states of the plant, A_{ij}^c are the gaussians membership functions like:

$$A_{ij}^c = \exp \quad \frac{(x_i - c_{ij}^c)^2}{\sigma_{ij}^{c\,2}} \qquad (5)$$

where: c_{ij}^c and σ_{ij}^c are constants to be tuned.

Similar to fuzzy system plant [9], fuzzy control representation is integrated as:

$$w_i = \prod_{j=1}^{l} A_{ij}^c(x_j) \qquad (6)$$

And

$$u(k) = \frac{\sum_{i=1}^{N} w_i (Q_i x(t))}{\sum_{i=1}^{N} w_i} \qquad (7)$$

Wher Q_i is the related i'th control gain. The configuration of the fuzzy logic control (FLC) integrated to the plant, epxressed as well in terms of Fuzzy Takagi Sugeno approach is represented in Fig. 4 [11]. The closed loop system is pursued in terms of local plant and related control gain per rule. In order to pursue this strategy, plant model is shown in terms of its state space representation.

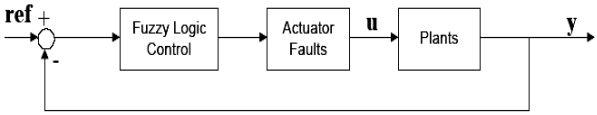


Fig. 4. Plant configuration using FLC control.

Using the proposed dynamic plant based on state space representation, see [11]:

$$\begin{aligned} x(k+1) &= A^P x(k) + B^P u(k) \\ y &= c^P x(k) \end{aligned} \qquad (8)$$

Specially B^P is stated as

$$B^p = \sum_{i=1}^{N} \rho_i B_i \sum_{j=1}^{M} \int_{t_j}^{t_{j+1}} e^{A_i^p (t-\tau)} d\tau \qquad (9)$$

where $\rho_i = 1$ and $\sum_{i=1}^{N} \rho_i = 1$ taking into account that *N* are the total number of possible faults and *M* are the involved time delays from each fault. Current communication time delays are expressed as τ_{j-1}^{i} and τ_{j}^{i} remember that $\sum_{j=1}^{M} \tau_j^i \leq T$ (total period of time inherent from control law design) and B_i in general terms is integrated as

$$B_i = \begin{bmatrix} b_1 \\ b_2 \\ 0_i \\ \vdots \end{bmatrix} \rightarrow, \text{ i fault element}$$

where $b_1 \rightarrow b_N$ are the elements conformed at the input of the plant (such as actuators) and 0_i is the lost element due to local sensor fault where B_i^P represents only one scenario. Remember that the only considered faults are sensor faults. Therefore one input signal is measured. This can lose its confidence but not current value [10]. Since this approximation current B_i^p considers local sensor faults and related time delays of

$$B_i^p = B_i \sum_{j=1}^{M} \int_{\tau_j}^{\tau_j^i} e^{A_i^p (t-\tau)} d\tau \tag{10}$$

Remember that the related time delays are the result of structural reconfiguration (SRA) explained before are calculated according to eqns. 4 and 5.

Back to the controller definition where *N* is the number of possible scenarios, therefore, number of rules.

$$Q_z = \sum_{j=1}^{N} \left[Q_j h_j (x(t)) \right] \tag{11}$$

$$h_j (x(t)) = \frac{w_i}{\sum_{i=1}^{N} w_i} \tag{12}$$

as for the plant integrated to the controller in closed loop, this is expressed as:

$$A_z^p x + B_z^p u \Rightarrow A_z^p x - B_z^p Q_z x \tag{13}$$

$$\Rightarrow A_z^p x - x \left\{ B_i \sum_{j=1}^{N} \int_{j-1}^{i} e^{\left[A_i^p (t-\tau) \right]} d\tau \right\} \left\{ \sum_{j=1}^{N} \left[Q_j h_j (x(t)) \right] \right\}$$

$$\Rightarrow \left[A_z^p - \left\{ \sum_{i=1}^{N} h_i B_i \sum_{j=1}^{N} \int_{j-1}^{i} e^{A_i^p (t-\tau)} d\tau \right\} \left\{ \sum_{j=1}^{N} \left[Q_j h_j (x(t)) \right] \right\} \right] x(t)$$

$$A_z^p = \sum_{i=1}^{N} A_i^p \{h_i(x(t))\}$$

Then the proposed Lyapunov function is:

$$v(x(t)) = x^T(t) P_z x(t) \tag{14}$$

And its derivative is expressed in eqn 14 as a necessary condition for stability

$$\dot{v}(x(t)) = \dot{x}^T(t) P_z x(t) + x^T(t) P_z \dot{x}(t) \le 0 \tag{15}$$

In terms of the plant integrated to the control law this is expressed as follows:

$$\dot{v} = x^T \left(A_z^p - B_z^p Q_z\right)^T P_z x + x^T P_z \left(A_z^p - B_z^p Q_z\right) x + x^T \dot{P}_z x \tag{16}$$

remember that

$$B_z^p = \sum_{i=1}^{N} B_i \sum_{j=1}^{M} \int_{j-1}^{t_j} e^{A_i(t-\tau)} d\tau \; h_i$$

where M is the number of time delays per scenario within the control law inherent period.

$$Q_z = \sum_{j=1}^{N} \{Q_j h_j(x(t))\}$$

Therefore the core of lyapunov function is given as :

$$\dot{x} = \left(A_z^p - B_z^p Q_z\right) x \tag{17}$$

Therefore the derivative of the energy as expressed in 15 can be expressed as:

$$= x^T \left\{ \left(A_z^p - B_z^p Q_z\right)^T P_z + P_z^T \left(A_z^p - B_z^p Q_z\right) + \dot{P}_z \right\} x \tag{18}$$

$$P_z = \sum_{i=1}^{N} \left(g_i(x(t)) P_i\right)$$

$$\dot{P}_z = \sum_{i=1}^{N} \left(g_i(x(t)) P_i\right)$$

and

$$g_z = A_z^p - B_z^p Q_z \tag{19}$$

Now by expressing the same energy function in terms of an inequality relation in a relaxed manner, considering all the possible P_z matrices equals in terms of the same matrix ($g_z = A_z^p - B_z^p Q_z$) for any condition considered along the N fuzzy rules, energy function can be expressed as:

$$\dot{v} \le \sum_{i=1}^{N} \left(\left(h_i^2(x) x^T \left(\left(A_i^P - B_i^P Q_i\right)^T P_z + P_z \left(A_i^P - B_i^P Q_i\right) \right) x \right) \right) + \tag{20}$$

$$\sum_{i=1}^{N} \sum_{i<j} 2h_i(x) h_j(x) x^T \left(\frac{A_i^P \quad B_i^P Q_j + A_j^P \quad B_j^P Q_i}{2} \right)^T P_z + P_z \left(\frac{A_i^P \quad B_i^P Q_j + A_j^P \quad B_j^P Q_i}{2} \right) x$$

Therefore, the controller design can be expressed as:

$$- P_z A^{PT}_i - A_i{}^P P_z + P_z Q_i^T B^{PT}_i + B^P{}_i Q_i P_z > 0 \tag{21}$$

Remember that *i* has a value between 1 to *N*. Therefore for every given plant and the respective controller by decomposing this equation, the P_z matrix should be bounded as:

$$-P_z A^{PT}_i - A^P{}_i P_z - P_z A^{PT}_j - A^P{}_i P_z + P_z Q_j^T B^{PT}_i + B^P{}_i Q_j P_z + P_z Q_i^T B^{PT}_j + B^P{}_j Q_i P_z > 0 \tag{22}$$

Remember that that also *j* has a value between 1 to *N* related to the number of rules. Therefore in terms of Linear Matrix Inequality [9] is given by

$$\begin{matrix} P_z & P_z A_i^T - Q_i^T P_z{}^T B_j^T \\ A_i P_z - B_i Q_j P_z & P_z \end{matrix} > 0 \tag{23}$$

This condition is given for every single time delay and local fault appearance. Furthermore the stability and the convergence of states should be assured by the adequate selection of matrices P_z and the related parameters from both fuzzy systems. In this case a recommendable procedure to follow is multi-objective optimization in order to define those suitable values [12].

The whole system considering this codesign strategy, has been implemented in several environments such as simulation based [12] using True-Time [16] and real-life using CANBUS [13]. Although this approximation is out of the scope of the paper, these implementations have given enough information in terms of the practical experience for current approach. Moreover, related strategies for codesign control theory have been reviewed with preliminar succesful results in [6].

4 Conclusions

The use of codesign as a suitable strategy for networked control and scheduling analysis is a real possibility as explored in this paper. Although it is restricted to the fesability of both techniques, this can be approximated as an iteractive procedure where both techniques need to achieve an aggrement.

In this case time delays are approximated and bounded through a suitable scheulling policy which affects the results of current selected controller. The exploration followed in here is based upon classical schedulling algorithms and fuzzy takagi sugeno approach. The key characteristic of last approach is design of local control law considering bounded time delays per valid scenario from schedulling results.

Several results need to be highlighted such as the convergence of variable time delays due to the use of schedulling approximation and the restricted and known modification onto control law design. Furthermore, bounded time delays as long as they are from the same source, like sensor delays, they modifiy similar control paramters, therefore, control structure does not need to be modified on a large scale.

Future work need to be focus onto strucutral modification from the control law, as well as a deeper study from time delays source. For instace, the complex computing relationship stablished through the operating system, middleware transactions, interprocees communications, communication network protocols, and others.

Acknowledgements

The author would like to thank the financial support of DISCA-IIMAS-UNAM, and UNAM-PAPIIT (IN105303 and IN101307) Mexico in connection with this work. The present paper is part of an ongoing effort of the High-Performance Computing project, within the `MacroproyectoTecnologías para la Universidad de la Informacíon y la Computacion" of the National Autonomous University of Mexico (UNAM).

References

1. Liu, J.; “Real-Time Systems”; Prentice Hall, 2000.
2. Buttazo G., “Hard Real-Time Computing Systems”; Springer, 2004.
3. Lian F. Moyne J. and Tilbury D. ; “Network Design Consideration for Distributed Control Systems”; IEEE Transactions on Control Systems Technology, Vol. 10, No. 2, pp. 297-307, March 2002.
4. Behnam M., and Damir I.; “Real-Time Control and Scheduling Co-Design for Efficient Jitter Handling”; IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2007).
5. Ramírez-Gonzalez T., Quiñones-Reyes P., Benítez-Pérez H., Laureano-Cruces A., and García-Nocetti, F.; “Reconfigurable Fuzzy Takagi Sugeno Networked Control Using Cooperative Agents and Xpc Target”; IEEE International Symposioum on Robotics and Automation; México, 2006.
6. Hristu-Varsakelis D., and Levine W.; “Handbook of Networked and Embedded Control Systems”; Birkahuser, 2005.
7. Zhong Q.; “Robust Control of Time-Delay Systems”; Springer, 2006.
8. Blanke M., Kinnaert M., Lunze J., Staroswiecki M.; “Diagnosis and Fault-Tolerant Control”; Springer, 2003.
9. Tanaka K., Wang H.; “Fuzzy Control Systems Design and Analysis”;Wiley Inter-science, 2001.
10. Abonyi J.; “Fuzzy Model Identification for Control”; Birkhäuser, 2003.
11. Benítez-Pérez H., García-Nocetti F. and Thompson H.; “Fault Classification Based upon Self Organizing Feature Maps and Principal Component Analysis for Inertial Sensor Drift”; International Journal of Innovative Computing, Information and Control, Vol. 3, Issue 2, Abril 2007.
12. Benítez-Pérez, H., and García-Nocetti, F.; “Reconfigurable Distributed Control”; Springer Verlag, 2005.
13. Quiñones-Reyes P., Benítez-Pérez H., Cardenas-Flores F. and García-Nocetti, F.; “Control Reconfigurable Difuso Takagi-Sugeno en Red usando Planificación EDF en XPC Target”; Revista IEEE América Latina, Volume 5, issue 2, pp. 110-115, 2007.
14. Dario Bauso, Laura Giarre, Raffaele Pesenti, “*Distributed Consensus for Switched Networks with Unknown But Bounded Disturbance*s”, IFAC 3rd International Workshop in Networked Control Systems, 2007.
15. Almeida L, Pedreiras P, Fonseca JAG. ***The FTT-CAN Protocol: Why and How.*** IEEE Transactions on Industrial Electronics, vol.49, no.6, Dec. 2002, pp. 1189-201.
16. Cervin, A., Henriksson, D., Lincoln, B., Eker, J., and Arzén, K.; “How Does Control Timing Affect Performance?”; IEEE Control Systems Magazine, Vol. 23, pp. 16-30, 2003.

Receptor Response and Soma Leakiness in a Simulated Spiking Neural Controller for a Robot

David Bowes, Rod Adams, Lola Cañamero, Volker Steuber and Neil Davey

Science and Technology Research Institute, University of Hertfordshire
College Lane, Hatfield, Hertfordhsire, U.K.
{D.H.Bowes, R.G.Adams, L.Canamero, V.Steuber, N.Davey}@herts.ac.uk

Abstract. This paper investigates different models of leakiness for the soma of a simulated spiking neural controller for a robot exhibiting negative photo-taxis. It also investigates two models of receptor response to stimulus levels. The results show that exponential decay of ions across the soma and of a receptor response function where intensity is proportional to intensity is the best combination for dark seeking behavior.

1 Introduction

In real neural systems it is known that leakiness in individual neurons can be caused by a variety of physical processes which can in turn lead to a variety of temporal profiles. Moreover the response of receptors to intensity of stimulus could be either linear or non linear [7]. Although different mechanisms of leakiness are possible, some are easier to implement in a robot than others. It is important to identify which mechanisms will require the least computational effort while producing the desired behaviour.

In this paper we investigate the effect of the specific function representing leakiness and receptor response, to the ability of an artificial neural system to respond appropriately to stimulus gradients in order to perform negative photo-taxis.

Photo-taxic robots [6], [4], [5] based on Braitenberg vehicles [3], controlled by artificial neural networks, have produced both varying results and success [6]. We have developed a robot controller using a simplified artificial spiking neuron model, implemented in an event driven simulation [8], [2]. The artificial neuron has a leaky soma which links to an axon hillock. The axon hillock initiates spikes in one or more collaterals attached to the axon hillock. See Figure 1 for an illustration of the artificial spiking neuron.

Using a modified Braitenberg fear vehicle [3], we simulated the behaviour of negative photo-taxis. In the original Braitenberg fear vehicle the output of the left light sensor is used to set the power output on the left motor, and a corresponding arrangement is made on the right side of the vehicle. In our modified arrangement we add sensory neurons, connected to interneurons, which then connect to the motor. This gives a potentially wider range of behaviours. However this arrangement was only satisfactory when the light intensity difference between the two light sensors was

large. When the differences were not large, the robot was unable to correctly orientate itself towards darkness.

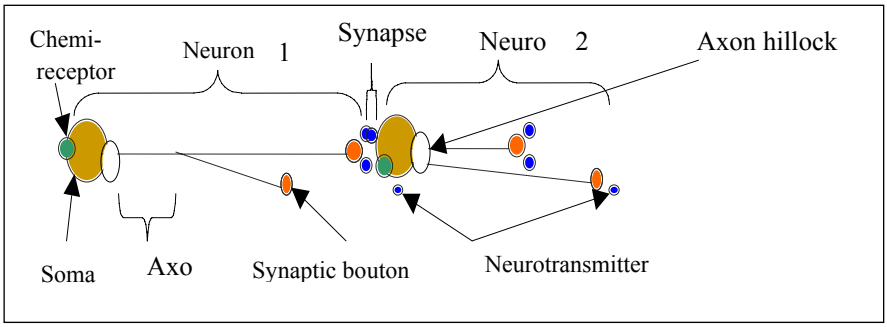


Fig. 1. The model of a simplified spiking neuron showing soma, axons, chemical receptor, synapse and neurotransmitter (note that the dendritic tree is not modeled).

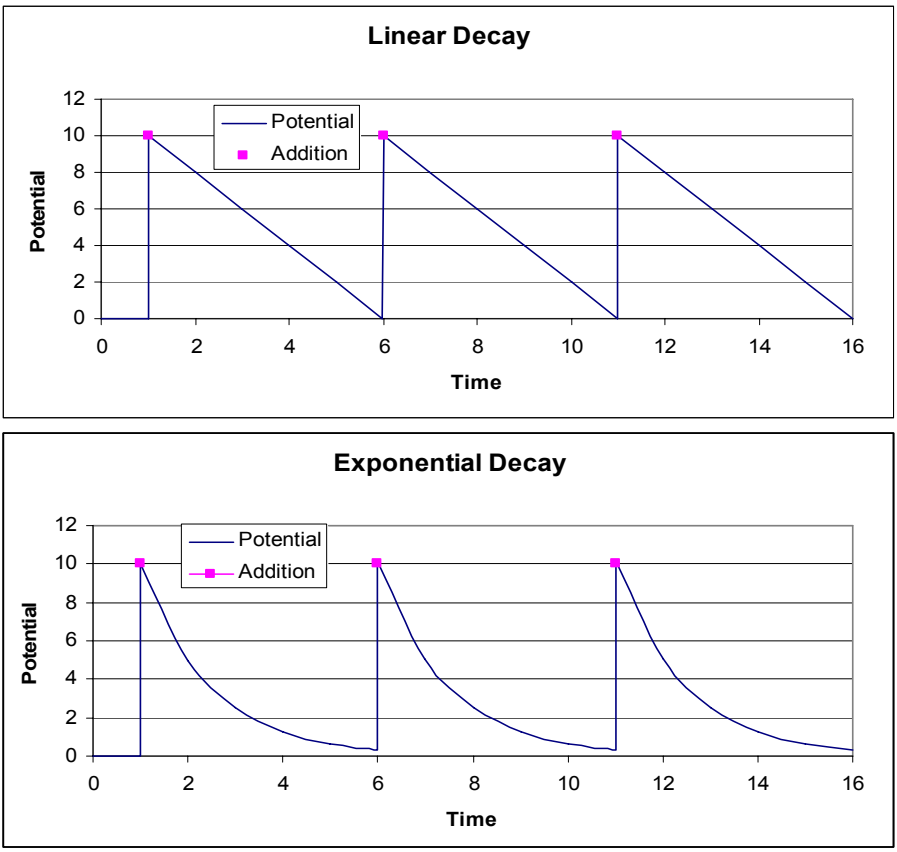


Fig. 2. Membrane potential with different decay functions with a constant rate of voltage additions generated by the arrival of spike events (designated by Addition points in the graph).

Two aspects of the model that could account for this lack of sensitivity were thought to be: the nature of the leakiness of the soma and the relationship between light intensity and firing rate of the light receptors.

In real neurons leakiness is controlled by a variety of mechanisms including voltage dependent and independent ion channels, active transport though pumps, exchanges and axial currents [7].

Subsequently we studied two models for leakiness: linear decay in voltage, which simple, and an exponential decay which is more plausible and could be caused by ions passing through pores along a concentration gradient, see Figure 2.

Both of these are biologically possible, but the latter system, which does not involve the direct use of energy, is more likely in a biological organism where parsimony of design is often found.

The relationship between light intensity and input current to a receptor neuron is complex and may lead to a variety of mappings between light intensity and the firing rate of a receptor.

To investigate this, two models for firing rate of a receptor cell, under different light intensities, were studied: inter spike time being inversely proportional to light intensity, or, inter spike time being proportional to the difference between the maximum possible intensity and actual intensity. The first of these produces a linear relationship between light intensity and firing rate and the second produces a hyperbolic relationship. Both of these are biologically plausible [7]. The first model would result in firing rate being proportional to light intensity and the second method would produce a non-linear relationship as shown in Figure 3.

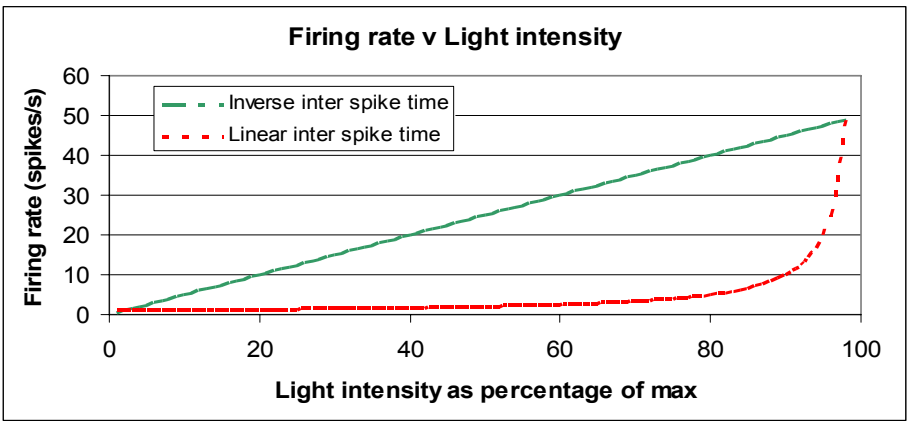


Fig. 3. Firing rates of the sensory neuron using a linear and hyperbolic relation. Note the poor discrimination of the hyperbolic function at low light intensities.).

2 Methodology

The experiment to test which function produced the greatest rotation was carried out on a robot, simulated in software. A schematic of the robot is shown in Figure 4.

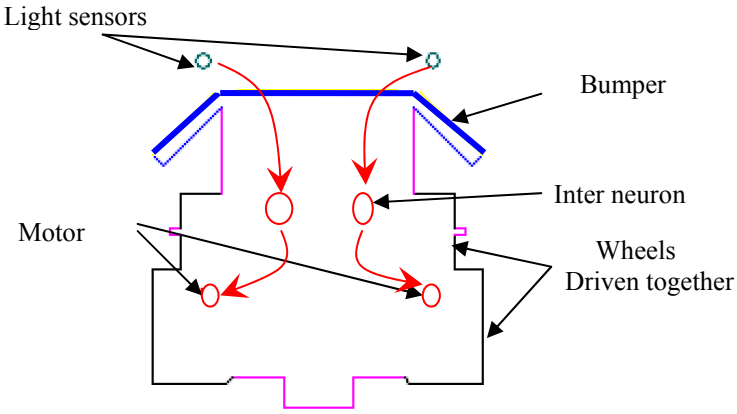
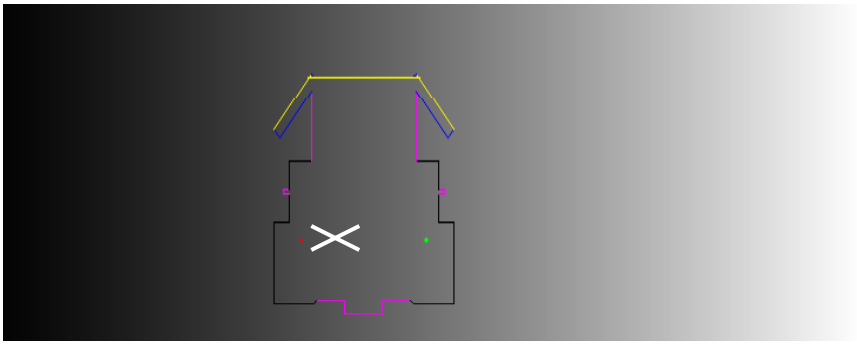


Fig. 4. A schematic of the robot indicating the position of the two light sensors and two interneurons which attach to the motors. Note the connections are excitatory which causes the vehicle to perform negative photo-taxis.

Simulated receptor neurons were attached to light sensors. These sensory neurons then connected to interneurons which subsequently attached to the motors with excitatory ipsilateral connections (left receptor connects to left motor). At the start of each experiment the robot was placed perpendicular to a linear light gradient, see Figure 5.



0 Distance from black : Proportional to light intensity : Arbitrary units 10000

Fig. 5. The Robot on the linear light gradient. The robot is placed orthogonally to the gradient and is "pinned" in position so that it can only rotate (X marks the vertical axis of rotation).

The simulation prevented forward motion of the robot, but allowed rotation so that the robot would spin on the spot (similar to an insect being tested for pheromone orientation in a wind tunnel). The simulation was run for 10 seconds and the total

angle of rotation was recorded. The robot was placed at incremental distances from the left side of the gradient, providing 9 data samples. Each permutation of decay model, receptor firing model and position was repeated 7 times, resulting in a total of 252 experiments.

The two models for potential decay in the soma of the interneuron are:

- Linear decay: which is very simple to calculate
$$c_{(t+1)} = c_t - \alpha$$
- Exponential decay: biologically more plausible
$$c_{(t+1)} = c_t \beta$$

Where c is the synaptic concentration with constants α and β, where $\beta < 1$.

Two models for determining the time between firings of light receptor neurons were:

- Inverse $T \propto \frac{1}{i} + T_{\min}$
so $F \propto i$
producing a linear relationship between light intensity and neuron firing rate.
- Linear $T \propto (k - i) + T_{\min}$
so $F \propto \frac{1}{k - i}$
producing a hyperbolic relationship between light intensity and firing rate.

Where k is the maximum stimulus level achievable, i is the intensity of stimulation F is the firing rate and $T_{\min}$ is a constant which prevents the period being less than 0.01s (below this value the amount of processing increases significantly).

3 Results

The full results for all four experimental conditions are given in Figure 6. This shows that significant rotation only occurs under the condition of exponential voltage decay and inverse inter spike time and then only in the dark region.

A study of the motor logs for all conditions indicated that at high light levels, both motors had been turned on to the maximum level due to very high rates of input firings. This had occurred because the soma of the interneuron had saturated and had prevented the voltage dropping below the threshold value for spiking.

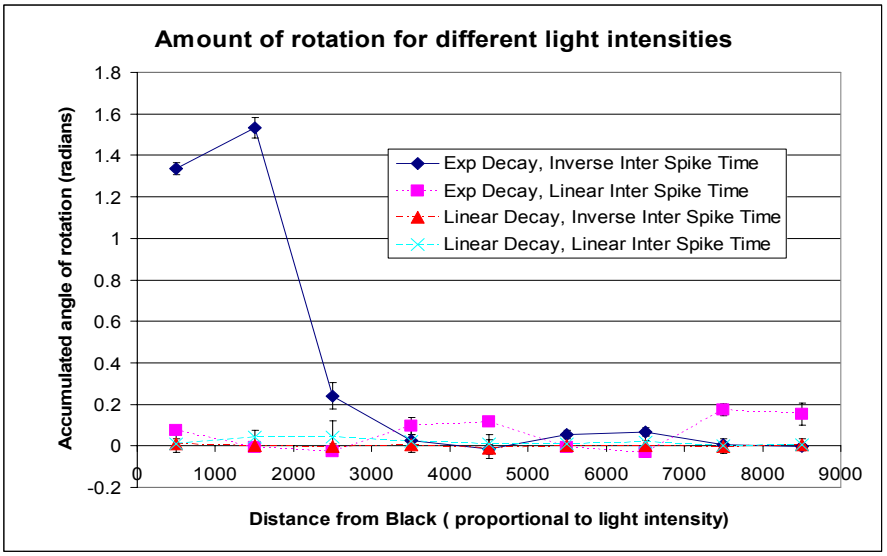


Fig. 6. A graph of the amount of rotation achieved. This shows the results for the four conditions: exponential voltage decay and inverse inter spike time, exponential voltage decay and linear inter spike time, linear voltage decay and inverse inter spike time, and linear decay and linear inter spike time. The exponential voltage decay and inverse inter spike time is the only configuration where significant rotation occurred.

Linear decay of potential performed particularly poorly. In high light intensity the decay of voltage in the interneuron was not sufficient to allow the voltage to fall below the firing threshold – the rate of decay in this model is constant and is therefore not related to the actual voltage. Moreover at low light intensities the voltage decay was too rapid to allow the voltage to ever exceed the firing threshold. Exponential decay of potential was much better: in this case the rate of decay is proportional to voltage, producing much greater sensitivity at the interneuron.

As already noted a linear relationship between inter spike times and light intensity at the light receptor neurons gives rise to a hyperbolic relationship between intensity and firing rate. This gives poor discrimination at low light levels and the results demonstrate the inadequacy of this encoding.

4 Discussion

In real neural systems it is known that leakiness in individual neurons can be caused by a variety of physical processes, which can lead to a variety of temporal voltage profiles. Moreover the response of receptors to intensity of stimulus could be either linear or non linear [7].

Our results show that for the example of a negatively photo-taxic robot, an exponential decay of membrane potential and an inverse relationship between light inten-

sity and inter spike time (resulting in a linear relationship between light intensity and firing rate) produces the greatest rotation at low light levels.

At high light levels the rotation is limited due to both motors being stimulated at the maximum rate because the interneurons receive levels of stimulus which prevents the membrane potential from dropping below threshold.

Interestingly, the most effective neural model is also the most biologically plausible. Diffusion of ions through pores would naturally give rise to an exponential decay whereas linearization of the decay would require additional voltage dependant mechanisms.

The modified Braitenberg wiring using interneurons is biologically more realistic than the original vehicle because it involves some level of processing by the interneurons, but it is not sophisticated enough to cope with the full range of light levels. We are currently investigating models of dynamic normalisation to achieve greater sensitivity to small differences in light intensity.

References

1. Hodgkin, A. L. and Huxley, A. F: A quantitative description of ion currents and its applications to conduction and excitation in nerve membranes. J. Physiol. (Lond.), 117:500-544. (1952).
2. Bowes,D.H; Adams,R.G; Cañamero,L; Davey,N: “A Simplified Spiking Neural Network for a Robot” Technical report, Science and Technology Research Institute, University of Hertfordshire (2007)
3. Braitenberg V: Vehicles: experiments in synthetic psychology. Elliot Sober (1984)
4. Floreano D: Evolutionary Robotics:A Short Tutorial (2005)
5. French, R.L.B.; Cañamero, L., "Introducing Neuromodulation to a Braitenberg Vehicle," Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on , vol., no., pp. 4188-4193, 18-22 April (2005)
6. Bisset, D. L.; Vandenbergh, R. C: The Dynamics of Photo- Taxis: Applying the Agent Environment Interaction System to a Simple Braitenberg Robot European Conference on Artificial Life Vol 4, 327-336 (1997)
7. Kandel ER, Schwartz JH, Jessell TM: Principles of Neural Science, 4th ed. McGraw-Hill, New York (2000)
8. Rochel, O; Martinez, D: An event-driven framework for the simulation of networks of spiking neurons, European Symposium in Artificial Neural Networks, pp 295-300 (2003)
9. GM Sheperd: The Synaptic Organization of the Brain, 5thRev, Oxford UP ,New York.(2004)

Comparison of Defuzzification Methods: Automatic Control of Temperature and Flow in Heat Exchanger

Carlos A. Cosenza[1], Alvaro J. Rey Amaya[2], Omar Lengerke[3], Max Suell Dutra[3] and Magda J. M. Tavera[2]

[1]Production Engineering, COPPE, Federal University of Rio de Janeiro – UFRJ
Cidade Universitária - CT, Bloco F, sala 108 - Ilha do Fundão
Rio de Janeiro, RJ – Postal Box: 68507, Brazil
cosenza@pep.ufrj.br
[2]Energy and Automation Laboratory, Autonomy University of Bucaramanga – UNAB
Calle 48 Nr. 39 -234, Bucaramanga - Colombia
ajrey@unab.edu.co
[3]Robotic and Automation Laboratory – COPPE/UFRJ
Federal University of Rio de Janeiro – UFRJ
Postal Box 68.503 – CEP 21.945-970 – Rio de Janeiro, RJ, Brazil
{olengerke, max, mmorales}@mecanica.coppe.ufrj.br

Abstract. The objective of this work is to analyze the behavior of the traditional control and the fuzzy control, applying them in the flow and temperature control to the load of current of a heat exchanger, as well as the analysis of different methods of defuzzification, utilized just as itself this carrying out the fuzzy control. Acting on the structure of the fuzzy controller some changes of form are carried out such that this tune in to be able to obtain the answer but optimum. In the same way proceeds on the traditional controller, and in this way comparisons on these two types of controls are established. Inside the changes that are carried out on the fuzzy controller this form of defuzzification the information, that is to say the methods are exchanged defuzzification in order then to realize comparisons on the behavior of each one of these.

1 Introduction

In many of the sectors of the industry where include itself thermal processes, is important the presence of a heat exchanger [1] [2]. Said processes do part of the everyday life of an engineer that has as field of action the control, therefore is considered interesting to realize a control to this type of tools. This work in its content studies two large aspects: A comparison between the traditional control and the fuzzy control, and an analysis between some of the different methods of defuzzification that are utilized in the fuzzy logic [3], doing an analysis of each one of they taking in consideration contribute them that other authors have done and leaving always in clear, that the alone results obtained will be applicable al moment of doing control on an exchanger of heat [4]. The system this composed one for two exchangers of heat [5], one of concentric pipes and the other of hull and pipes, to which implemented them an automatic control of the temperature and the flow to the load of the current of heating (Fig. 1).

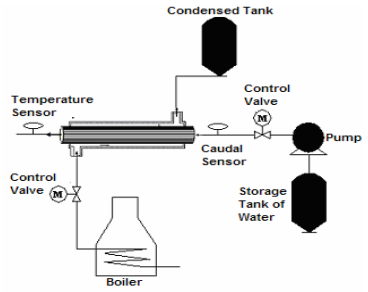


Fig. 1. Assembly of the system.

This control is realized through two proportional valves, one to the input of the water, that is the responsible for maintaining the value of order of the water and the other installed in the line of input of the vapor (source of heat), that is responsible of maintaining the quantity of necessary vapor to obtain the target temperature. The measurement of the flow [6] is realized by means of a sensor of rotary palette and the measurement of the temperature by means of thermocouples [7] [8] [9]. The signals supplied by these sensors are acquired by means of the FieldPoint systems of National Instruments [10], the same one that takes charge of sending the signal to the valves of control, after to be processed the data by the controller [11].

2 Software of Fuzzy and Classic Control

The software of control designed, is formed of two sections, the program of fuzzy control, and the program of the PID (Proportional/Integral/ Derivative) control. These controllers are elaborate in environment Labwindows/CVI, software of the company National Instruments, which permits to realize the pertinent operations with the data captured through the modules of FieldPoint, and that, are utilized in the control of the system. The fuzzy control interface, is the responsible for taking the data of the sensors, so much of temperature, for the case of the control of temperature, as of the sensor of flow, for the control of the same one, to process, and according to an order established, to determine an response, which is sent to the actuators. Basically, this program is responsible of: to schematize the fuzzy sets, according to established by the user, defuzzification of the inputs, to realize the inference of these inputs in the rules, to realize the aggregation in the outputs sets, and to execute the process of defuzzification, to determine the response that help to the system to obtain the stable state. The program of the classic control PID, is the responsible for taking the data of the sensors, so much of temperature, for the case of the control of temperature, as of the sensor of flow, for the control of the same one, to process, and according to an order established, to determine an response, which is sent to the actuators. Basically, this program is entrusted of to execute the three actions of control, proportional, derivative and integral to determine the responses that help to the system to obtain its target state. The PID control system general is represented in figure 2, where R(s), is the signal target or set point, U(s), is the output of the PID controller that goes toward

the plant G(s), and Y(s), is the value taken of the variable to control, which reduces to the reference and the error is determined (controller input).

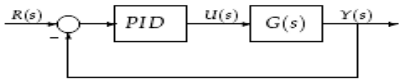


Fig. 2. System of Control PID.

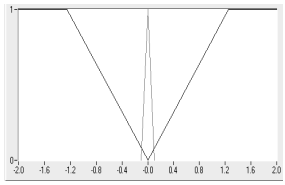
The control of Temperature, have as an objective to obtain that the water that the exchanger of heat leaves achieve the value of the target temperature and then to keep it in the value even with external disruptions. That is to say, should operate on the valve of control who is the one that supplies the quantity of vapor that heats the water. The input to this system of control will be the error of temperature, obtained since the thermocouple placed on the exit of the exchanger, and the exit will control the quantity of necessary current to open or to close the proportional valve (Plant). This control is realized through a PID controller. The flow control has as an objective to obtain that the mass flow of water that enters to the exchanger of heat, achieve the target value, and can to keep it during its operation, and even with disruptions. This means that should operate on the valve of control, who is the one that strangles the quantity of water that enters to the system. The input of the system will be the error obtained through the sensor of flow installed in the input of the system, and the PID controller will control the quantity of necessary current to manipulate the proportional valve. Both processes begin, calculating the difference between the measured temperature and the temperature target or of the flow measured and the flow desired. In this form, know the error. Then, the values of the parameters of control are taken, and the output is calculated that goes toward the plant. This output, will obtain values since 0 to 20 mA, they will represent angles of opening of the proportional valve.

3 System of Fuzzy Control

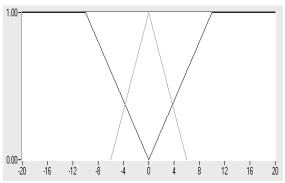
The input to this system of control will be the error of temperature and the gradient, obtained since the sensor placed on the way out of the exchanger, and the exit will control the quantity of necessary current to open the proportional valve. The rules of the system and function of membership are obtained in the table 1 and figure 3, respectively.

Table 1. Temperature control rules assembly.

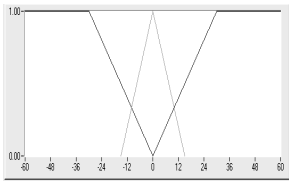
Error / **Δ Error**	**Negative**	**Zero**	**Positive**
Negative	Open	Open	Not operation
Zero	Open	Not operation	Close
Positive	Not operation	Close	Close



Functions of membership for the current at the outset



Functions of membership for the variable derived from the Error



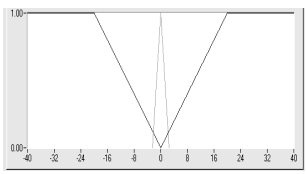
Functions of membership or the variable Error

Fig. 3. Functions of membership temperature control.

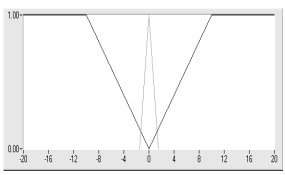
The control of flow has as objective to obtain that the mass flow of water that enters to the exchanger of heat, achieve the value of order, and can to keep it during its operation, and even before disruptions. This means that should act on the valve of control who is the one that strangles the quantity of water that enters to the system. The input of the system, they will be the error obtained through the sensor of flow installed to the entrance of the system, and the change of the error in the time, and the output will control the quantity of necessary current to manipulate the proportional valve. Both processes begin, calculating the difference between the measured temperature and the temperature desired, or of the flow measured and the flow desired. In this form know the Error. Then, calculate the gradient, reducing the new error of the previous one. Once known these variables, that constitute the inputs of the fuzzy logic controller, proceeds to realize the fuzzification, the inference and the defuzzification to obtain the output of the controller. This output, will obtain values since 0 to 20 mA, who will represent angles of opening of the proportional valve. The rules of the system and function of membership are obtained in the table 2 and figure 4, respectively.

Table 2. Flow control rules assembly.

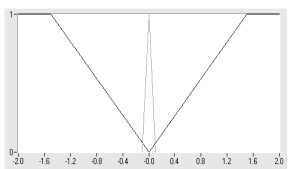
Error / Δ Error	Negative	Zero	Positive
Negative	Close	Close	Not operation
Zero	Close	Not operation	Open
Positive	Not operation	Open	Open



Functions of membership for the variable Error



Functions of membership for the variable derived from the Error



Functions of membership for the current at the outset

Fig. 4. Functions of membership flow control.

3.1 Comparative Results between the Methods of Defuzzification Implemented

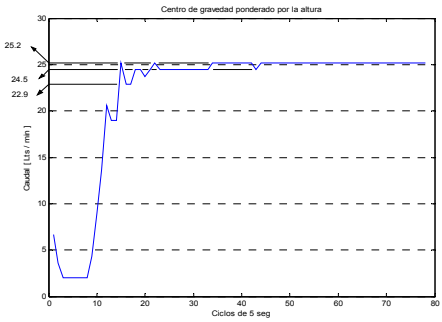
The methods of defuzzification chosen were five; these are know in the area of the control by the names, central gravity weighted by the height, central gravity weighted by the area, average of centers, points of maximum criterion weighted by the height and points of maximum criterion weighted by the area. When refers to the control of a system, the main term on which refers is to the stability that this can offer, for this is necessary to take into consideration, the time that delayed the system in being stabilized, the margin of error between the value desired, (Vc), and the values of stabilization, (Ve) of the system and the influence of the inertia of the system. For the tests of temperature and flow control, is defined a set point of 25 [Lts / min] and 40 [ºC]. The parameters and equations used are observed in the table 3 and the table 4 the different responses are shown in each one of the methods, according to the parameters established in the table 3.

Table 3. Methods and models of defuzzification.

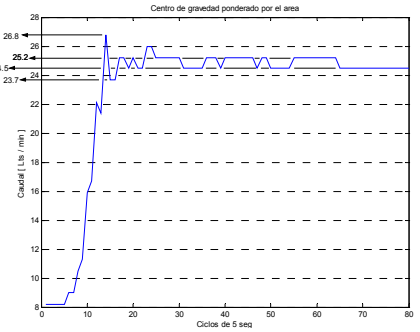
METHODS	EQUATIONS
1. Central gravity weighted by the height	$\bar{x} = \frac{\sum_{i=1}^{n} h_i * w_i}{\sum_{i=1}^{n} h_i}$ Where, w is the center gravity of the resultant assembly after realized the fuzzy operation chosen, and h is the height of the same assembly.
2. Central gravity weighted by the area.	$\bar{x} = \frac{\sum_{i=1}^{n} s_i * w_i}{\sum_{i=1}^{n} s_i}$ Where, w is the center gravity of the resultant assembly after realized the fuzzy operation chosen and s is the area of the same assembly.
5. Points of maximum criterion weighted by the area.	$\bar{x} = \frac{\sum_{i=1}^{n} s_i * G_i}{\sum_{i=1}^{n} s_i}$ Where, G is the point of maximum criterion of the resultant set after to realize the fuzzy4 operation chosen and s is the area of the same set.
3. Points of maximum criterion weighted by the height.	$\bar{x} = \frac{\sum_{i=1}^{n} h_i * G_i}{\sum_{i=1}^{n} h_i}$ Where, G is the point of maximum criterion of the resultant set after to realize the fuzzy operation chosen and h is height of the same set.
4. Average of centers	$y = \frac{\sum_{l=1}^{M} y^{-l}(\mu_{B'}(y^{-l}))}{\sum_{l=1}^{M} \left(\mu_{B'}(y^{-l})\right)}$ Where y^{-l} represents the center of the fuzzy set G^l (defined as the point V in which $\mu_G{}^l(y)$ reaches its value maximum), and $\mu_B(y)$ this defined for the degrees of membership resultant of the fuzzy inference.

Table 4. Response in each one of the methods established.

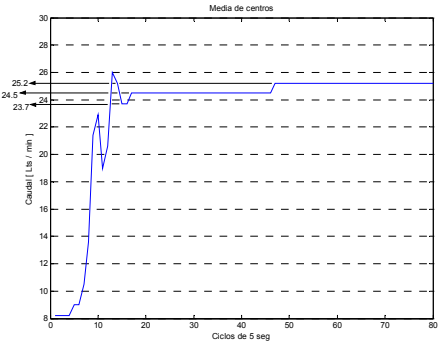
FLOW CONTROL



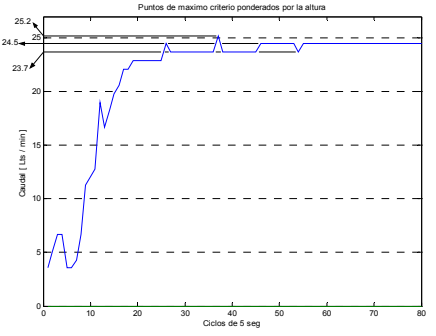
Central gravity weighted by the height



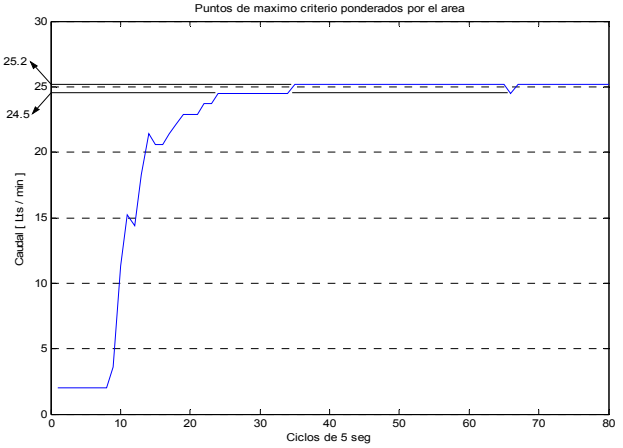
Central gravity weighted by the area



Average of centers



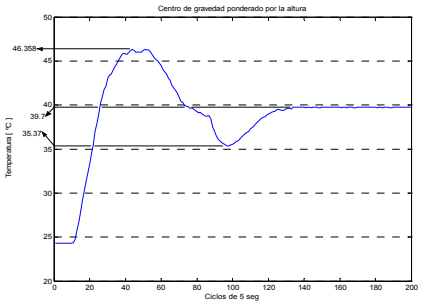
Points of maximum criterion weighted by the height



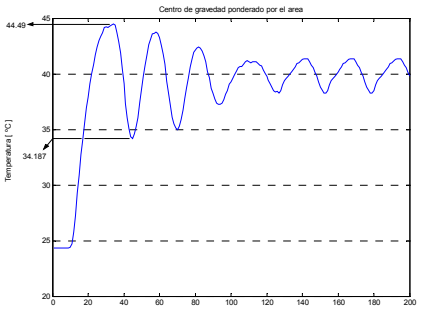
Points of maximum criterion weighted by the area.

Table 4. Response in each one of the methods established (continued).

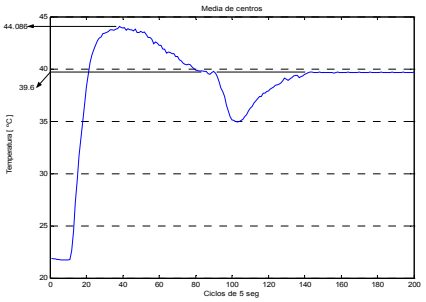
TEMPERATURE CONTROL



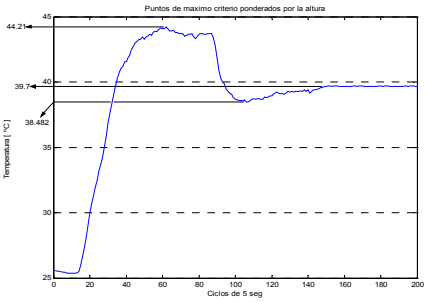
Central gravity weighted by the height



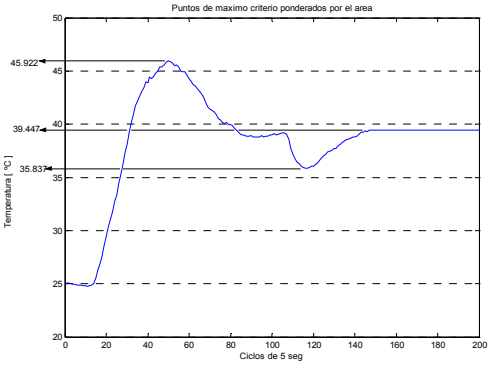
Central gravity weighted by the area



Average of centers



Points of maximum criterion weighted by the height



Points of maximum criterion weighted by the area

A summary of the results obtained in the different methods is shown in the table 5 for the control of flow and the table 6 for the control of temperature.

Table 5. Response of the methods of defuzzification in the control of flow.

Defuzzification Method	Time of stability [sec]	Margin of error (Vc - Ve)	Influence of the inertial of system
Central gravity weighted by the height	105	0.8% above of the set point 2% underneath of the set point	0.8% above of the set point 8.4% underneath of the set point
Central gravity weighted by the area	125	0.8% above of the set point 2% underneath of the set point	7.2% above of the set point 5.2% underneath of the set point
Average of centers	85	0.8% above of the set point 2% underneath of the set point	4% above of the set point 5.2% underneath of the set point
Points of maximum criterion weighted by the height	230	2% underneath of the set point	0.8% above of the set point 5.2% underneath of the set point
Points of maximum criterion weighted by the area	120	0.8% above of the set point 2% underneath of the set point	0.8% above of the set point 2% underneath of the set point

Table 6. Response of the methods of defuzzification in the control of temperature.

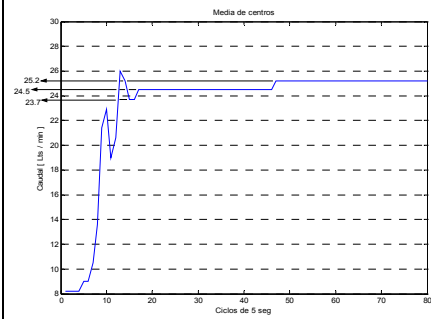
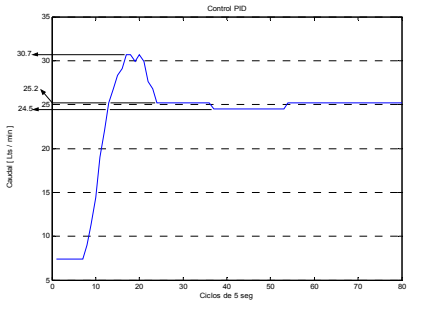
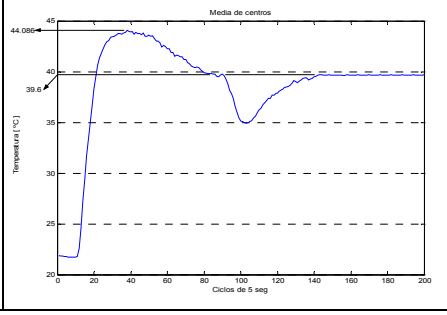
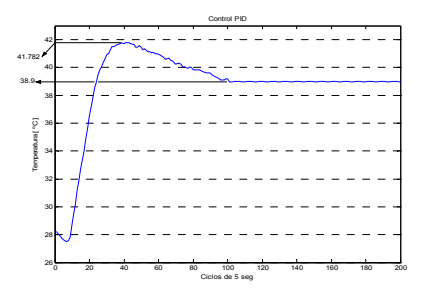
Defuzzification Method	Time of stability [sec]	Margin of error (Vc - Ve)	Influence of the inertial of system
Central gravity weighted by the height	670	0.75% underneath of the set point	40.895% above of the set point 11.575% underneath of the set point
Central gravity weighted by the area	Not stabilized	Not stabilized	11.25% above of the set point 14.5325% underneath of the set point
Average of centers	710	1% underneath of the set point	10.215% above of the set point 12.5% underneath of the set point
Points of maximum criterion weighted by the height	745	0.75% underneath of the set point	10.525% above of the set point 3.795% underneath of the set point
Points of maximum criterion weighted by the area	735	1.38% underneath of the set point	14.805% above of the set point 10.4075% underneath of the set point

3.2 Comparative Analysis between the Classic Controller and Fuzzy Controller

To be able to realize this analysis should make use of concepts that be fundamental at the moment of to evaluate the efficiency of a controller. The concepts to take into consideration in this case are, the time that delayed the system in being stabilized, margin of error between the value of order, (Vc), and the values of stabilization (Ve),

of the system and the Influence of the inertia of the system. For the comparative analysis between the fuzzy controller and the PID controller in the control of flow made use of the tests realized to each one of these controllers, with a set point of 25 [Lts / min] and 40 [ºC]. The results obtained are shown in the table 7.

Table 7. Response of the controllers.

	FUZZY CONTROLLER	PID CONTROLLER
FLOW CONTROL		
TEMPERATURE CONTROL		

A summary of the results obtained in the different methods is shown in the table 8 for the control of flow and the table 9 for the control of temperature.

Table 8. Response of the methods of defuzzification in the control of flow.

Controller	Time of stability [sec]	Margin of error (V_c - V_e)	Influence of the inertial of system
FUZZY CONTROL	85	0.8% underneath of the set point 2% above of the set point	4% underneath of the set point - 5.2% above of the set point
PID CONTROL	115	0.8% underneath of the set point 2% above of the set point	22.8% underneath of the set point 2% above of the set point

Table 9. Response of the methods of defuzzification in the control of temperature.

Controller	Time of stability [sec]	Margin of error (V_c - V_e)	Influence of the inertial of system
FUZZY CONTROL	710	1% underneath of the set point	10.215% above of the set point 12.5% underneath of the set point
PID CONTROL	505	2.75% underneath of the set point	4.455% above of the set point 2.75% underneath of the set point

4 Conclusions

The results obtained in this work show the technical viability of the utilization of the fuzzy logic in the control of flow and temperature to the warming-up current input of an exchanger of heat. With respect to the control of flow and temperature implementing fuzzy logic, can tell that possesses the advantages of need not a mathematical model of precision of the control system. As disadvantage can tell itself, that the design should be realized generally with the method of test and error. Is possible to control through fuzzy techniques industrial process, with the greater facility and with the minimum of errors, suffices with knowing its general behavior to structure a series of fuzzy sets and its respective rules. The tuning of the fuzzy controller, besides depending on the rules matrix, also, depends on the size of the sets of the variable, already itself of input or output. This depends on the same behavior of the system. For the implementation of a fuzzy control, is necessary, the establishment of the methods and alternatives utilized in each one of the blocks that conform it. In this form, can be obtained best results, at the moment of the tuning of the system. The answer of the fuzzy controller does not depend on the method of defuzzification utilized, if not of the adequate utilization of the functions of membership, and of the numbers of linguistic variables utilized for each one of the variables of input and output of the system. Also, depends on the type and of the size of the sets utilized.

References

1. Ruan, D.: Fuzzy Logic Foundations and Industrial Applications. International Series in Intelligent Technologies. Kluwer Academic Publishers, Boston (1996).
2. Harris, J.: An introduction to fuzzy logic applications. Kluwer Acad. Pub., Boston (2000).
3. Ruan D. and Huang C.,: Fuzzy sets and fuzzy information-granulation theory: key selected papers by Lotfi A. Zadeh. International Series on Advances in Fuzzy Mathematics and Engineering. Beijing Normal University Press, Beijing (2000).
4. Alotaibi, M., Sen, B., Goodwine and Yang, K.T.: Controllability of cross-flow heat exchanger. Int. J. Heat Mass Transfer 47, (2004) 913–924.
5. Xia, L., De Abreu-Garcia, J.A. and Hartley, T.T.: Modeling and simulation of a heat exchanger. IEEE International Conference on Systems Engineering. (1991) 453–456.
6. McMillan, G.K.; Considine, D.M.: Process / Industrial Instruments and Controls Handbook. 5th Edition, McGraw-Hill, New York, USA (1999).
7. Youden, W. J.: Experimentation and Measurement. Dover Publications. Mineola, New York. (1998).
8. Morris, A. S.: Measurement and Instrumentation Principles, Third Edition, Butterworth-Heinemann, Great Britain (2001),
9. Michalski, L., Eckersdorf, K., Kucharski, J., and McGhee, J.: Temperature Measurement. 2nd edition, John Wiley, England (2001).
10. Archila, J.F., Dutra, M.S., Lengerke, P. O. and Vega, T.J.: Design and implementation of a control system in closed-loop using the communication system FIELD POINT for a hydrostatic transmission. Published in RESET: Journal Specialized in informatics and electronics systems of Telecommunications Systems. Vol.1, (2006) 48–53.
11. Park, J. and Mackay S.: Practical Data Acquisition for Instrumentation and Control Systems. Newnes Elsevier. Great Britain (2003).

Author Index

Proceedings of the 4th International Workshop on
Artificial Neural Networks and Intelligent Information
Processing - ANNIIP 2008
ISBN: 978-989-8111-33-3
http://www.icinco.org