Behaviour of Similarity-Based Neuro-Fuzzy Networks and Evolutionary Algorithms in Time Series Model Mining

Valdés, Julio; Barton, Alan; Paul, R.

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC-CNRC

Behaviour of Similarity-Based Neuro-Fuzzy Networks and Evolutionary Algorithms in Time Series Model Mining *

Valdés, J., Barton, A. and Paul, R.
November 2002

* published in International ICONIP'02 Conference. Singapore, 18 November 2002, NRC 44954.

Canada

BEHAVIOR OF SIMILARITY-BASED NEURO-FUZZY NETWORKS AND EVOLUTIONARY ALGORITHMS IN TIME SERIES MODEL MINING

Julio J. Valdés, Alan Barton [*]

National Research Council Canada
Institute for Information Technology
1200 Montreal Road, Ottawa ON K1A 0R6
Canada
julio.valdes@nrc.ca, alan.barton@nrc.ca

Robyn Paul [†]

Dept. of Electrical and
Computer Engineering
University of Waterloo
Canada
rpaul@uwaterloo.ca

ABSTRACT

This paper presents the first in a series of experiments to study the behavior of a hybrid technique for model discovery in multivariate time series using similarity based neuro-fuzzy neural networks and genetic algorithms. This method discovers dependency patterns relating future values of a target series with past values of all examined series, and then constructs a prediction function. It accepts a mixture of numeric and non-numeric variables, fuzzy information, and missing values. Experiments were made changing parameters controlling the algorithm from the point of view of: *i)* the neuro-fuzzy network, *ii)* the genetic algorithm, and *iii)* the parallel implementation. Experimental results show that the method is fast, robust and effectively discovers relevant interdependencies.

1. INTRODUCTION

Multivariate time-varying processes occur in a wide variety of important domains such as medicine, economics, industry, environmental sciences, etc. Processes of this kind involve many variables changing simultaneously with time. These are heterogeneous in nature consisting of numeric and non-numeric quantities typically with missing values. Moreover they are the result of measurements and observations with very different degrees of indetermination and precision (e.g. fuzzy data). One of the most important data mining and knowledge discovery tasks in the study of time dependent information is finding *interesting dependencies* between past and future values of the observed variables (i.e. *dependency patterns* or *models*), and suitable prediction estimators for forecasting purposes. The use of classical methods is limited by different factors. Some factors are related to the underlying assumptions about the data concerning type, volume, homogeneity, complexity, precision, the curse of dimensionality, etc. In many cases these methods are based on assumptions which don't hold or are unpractical to verify. From a soft-computing approach to problems of this kind, a technique for model discovery and prediction in multivariate time series was introduced in [7]. That method accepts heterogeneous and large series with different degrees of imprecision, also possibly with missing data. Very preliminary applications showed interesting behavior with respect to speed, performance and sensitivity to detect internal dependencies. This paper presents the first in a series of systematic experiments to study the method's properties.

[*]Thanks to R.Orchard and F.Famili from the Integrated Reasoning Group, National Research Council Canada, for supporting this research.

[†]The third author performed the work while at the National Research Council Canada

2. METHOD OUTLINE

The objective is to extract plausible *dependency models* in heterogeneous multivariate time varying processes, expressing the relationship between future values of a previously selected time series (the target), and the entire set of series. Heterogeneity means the presence of ratio, interval, ordinal or nominal scales, and fuzzy magnitudes. Moreover, the series may contain missing values. The first step is to set a conceptual class of functional models and in this case a generalized non-linear auto-regressive (AR) model was used (1) (others are also possible),

$$S_T(t) = \mathbf{F}\begin{pmatrix} S_1(t-\tau_{1,1}), \cdots, S_1(t-\tau_{1,p_1}), \\ S_2(t-\tau_{2,1}), \cdots, S_2(t-\tau_{2,p_2}), \\ \cdots \\ S_n(t-\tau_{n,1}), \cdots, S_n(t-\tau_{n,p_n}) \end{pmatrix} \quad (1)$$

where $S_T(t)$ is the target signal at time t, S_i is the i-th time series, n is the total number of signals, p_i is the number of time lag terms from signal i influencing $S_T(t)$, $\tau_{i,k}$ is the k-th lag term corresponding to signal i ($k \in$

$[1, p_i]$), and F is the unknown function describing the process. The next step is the simultaneous determination of: the number of required lags for each series, the particular lags within each one carrying the dependency information, and the prediction function. A natural requirement on function F is the property of minimizing a suitable prediction error. This is approached with a soft computing precedure based on: (a) exploration of a subset of the entire *model space* with a genetic algorithm, and (b) use of a similarity-based neuro-fuzzy system representation for the unknown prediction function.

Evolving neuro-fuzzy networks with genetic algorithms has been done for a long time, but only for training purposes and in the context of a *single* network. The present approach differs by working in the space of *all possible networks* representing models given by (1) above. This involves the construction and evaluation of *thousands* or even *millions* of networks. Thus, the use of conventional feed-forward networks, with their long training times, becomes prohibitive. Other difficulties include finding the number of hidden layers and their composition, using mixed numeric, non-numeric, fuzzy and missing values, etc. All of these difficulties can be addressed by the heterogeneous neuron model [5], [6], [1]. It considers a neuron as a general mapping between heterogeneous multidimensional spaces.

For the h-neuron used here, the aggregation function is given by a similarity, while the activation function is given by the identity. This neuron maps a n-dimensional heterogeneous space onto the [0,1] real interval in such a way that the output expresses the degree of similarity between the input pattern and neuron weights. A hybrid network using heterogeneous neurons in the hidden layer and classical neurons in the output layer is suitable for the purpose of model mining (Fig-1).

Each neuron in the hidden layer computes its similarity with the input vector and the k-best responses are retained (k is a pre-set number of h-neurons to select). They represent the fuzzy memberships of the inputs w.r.t. the classes defined by the hidden layer neurons. Neurons in the output layer compute a normalized linear combination of the expected target values used as neuron weights (W_i), with the k-similarities coming from the hidden layer.

$$output = (1/\Theta) \sum_{i \in \mathcal{K}} h_i W_i, \qquad \Theta = \sum_{i \in \mathcal{K}} h_i \qquad (2)$$

where $\mathcal{K}$ is the set of k-best h-neurons of the hidden layer and h_i is the similarity of the i-best h-neuron w.r.t the input vector, representing a fuzzy estimate for the predicted value.

Assuming that a similarity function $\mathcal{S}$ has been chosen and that the target is a single time series, this *case-based* neuro-fuzzy network is built and trained as follows: Define a similarity threshold $T \in [0, 1]$ and extract the subset $\mathcal{L}$ of the set of input patterns Ω ($\mathcal{L} \subseteq \Omega$) such that for every input

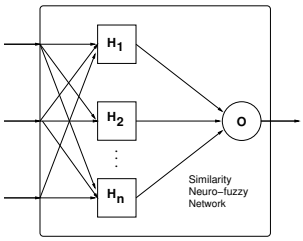

Figure 1: Neuro-fuzzy network composed by h-neurons in the hidden layer and classical neurons in the output layer

pattern $x \in \Omega$, there exist a $l \in \mathcal{L}$ such that $\mathcal{S}(x, l) \geq T$. The hidden layer is constructed by using the elements of $\mathcal{L}$ as h-neurons. While the output layer is built by using the corresponding target outputs as the weights of the neuron(s). This training procedure is *very* fast and allows for the rapid construction and testing of many networks.

A parallel implementation following a master-slave approach was made using LAM/MPI [3] and the GaLib [8]. The slaves construct and evaluate individual neuro-fuzzy networks based on models received from the master, which controls the genetic algorithm process at the population level. System architecture is shown in Fig-2

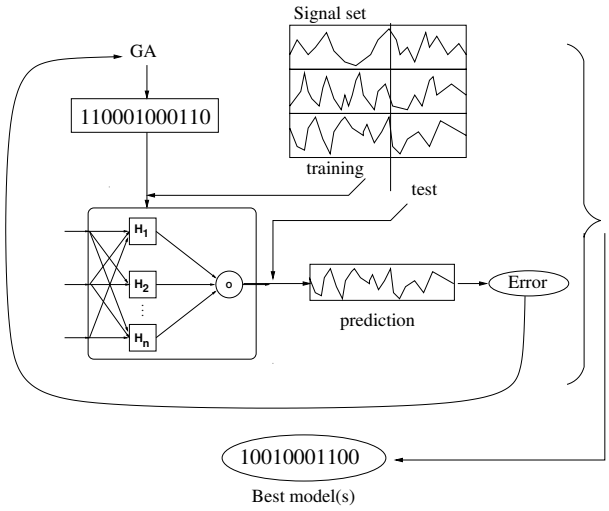

Figure 2: Similarity-based heterogeneous neuro-fuzzy networks are wrapped by a genetic algorithm.

The system's behavior is controlled by three classes of factors related to: *i)* the neuro-fuzzy network, *ii)* the genetic algorithm, and *iii)* the parallel implementation. Related to *(i)* are the specific similarity function modeling the neuron's computation (Sf), the number of responsive neurons in the hidden layer (Rn) representing the number of terms used to compute (2), the similarity threshold (St) determining the

hidden layer composition, the maximun lag depth (Ld), and the relative percentage of the training set vs test set (Rp) when learning the prediction function for a given time series dependency model. In all experiments St was kept fixed and equal to 1.

The process of model search is performed by the genetic algorithm. Binary chromosomes coding model components as given by (1) were used with both single and double point crossover operators and standard bit-reversal mutation. Selection was kept constant (roulette wheel method) and complete population replacement with elitism were used. In *(ii)* the influence of the number of generations (Ng), the population size (Ps), the crossover and mutation probabilities (Cp, Mp) and the type of crossover operator (Co) were investigated. As for *(iii)* the number of physical nodes was fixed at three, whereas the number of slaves (Ns) was varied.

3. EXPERIMENTAL SETUP

In the context of this research, a series of increasingly more complex experiments is planned. Therefore, this being the first in the series, simplicity was chosen over complexity. A data set consisting of one real-valued time series without missing data was chosen, namely the American relative sunspot numbers (mean number of sunspots for the corresponding months in the period $1/1945 - 12/1994$), from AAVSO - Solar Division containing 600 observations [4]. No preprocessing was applied to the time series. This is not the usual way to analyze time series data, but by eliminating additional effects, the properties of the proposed procedure in terms of approximation capacity and robustness are better exposed. The reported error measure is the *mean squared error* (MSE).

The experiments were conducted on a Beowulf cluster consisting of three dual Xeon processor units operating at 2 Ghz frequency with 1Gb RAM each. The cluster operates with 100 Mbit Ethernet connections. The operating system is Red Hat Linux 7.2 running LAM-MPI version 6.5.4/MPI 2, C++/ROMIO.

This first set of experiments contains 180 runs varying the three classes of controlling factors and their corresponding parameters (see Table-1).

We specified an experimental standard as follows: $< Sf = norm.\ euclidean, Rn = 7, Ld = 30, Rp = 50\%, Ng = 100, Ps = 50, Cp = 0.6, Mp = 0.01, Co = single-point, Ns = 15 >$.

Param.Class	Name	Values
(i)	Sf	$(1/(1+d))$
	Rn	$< 1, 3, 5, 7, 13, 20 >$
	Ld	$< 5, 10, 20, 30, 50 >$
	Rp	$< 25\%, 50\%, 75\% >$
(ii)	Ng	$< 2, 10, 100, 1000 >$
	Ps	$< 10, 50, 100 >$
	Cp	$< 0.4, 0.6, 0.8 >$
	Mp	$< 0.005, 0.01, 0, 02 >$
	Co	< single-point, double-point >
(iii)	Ns	$< 6, 15, 30 >$

Table 1: Experimental parameters. In *(i)* d stands for euclidean, clark, or canberra normalized distances (see [2] for definitions).

4. RESULTS

4.1. Influence of neuro-fuzzy network parameters

In a special experiment using the standard settings, 8 replicates were run for each similarity function. Results are shown in Table-2.

Similarity Func.	MSE error	min MSE	max MSE
euclidean	18.70	18.60	18.89
clark	18.70	18.32	19.23
canberra	18.51	18.34	18.74

Table 2: Mean MSE in eight replicate experiments with different similarity functions (MSE = mean squared error).

This result indicates that within the set of selected similarities, the one based on Canberra's distance performs slightly better. However, its range overlaps with that of euclidean. Clark's distance varies in a wide range, almost subsuming the others, although performing like the euclidean.

The behavior of the studied similarity functions taking into account all experiments is shown in Table-3.

Similarity Func.	MSE error	avg time(secs)
euclidean	19.352689	164
clark	19.3223295	214
canberra	19.1620806	281

Table 3: Mean MSE using different similarity functions in all experiments (MSE = mean squared error).

As in the previous case of replicted runs with the standard set of parameters, Canberra's distance again gives better performance errors, but at a higher computational cost, especially when compared with the euclidean distance. However, this is conditioned to the properties of the particular

data set investigated. Considering the wide range of variation of all parameters included in these results, it is clear that, at least for data of this kind, the method is robust.

The behavior of the number of responsive neurons for the standard experimental settings is shown in Fig-3. All exhibit a typical "elbow" shape, clearly indicating the existence of an optimal value. It is interesting to observe that in order to achieve a reasonable fuzzy estimate for the predicted output, comparatively few terms are required in (2).

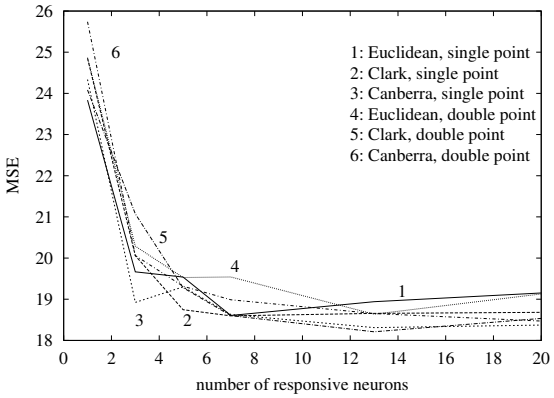

Figure 3: Behavior of the number of responsive neurons w.r.t. the similarity function and the crossover operator.

The behavior of the maximum lag depth for the standard experimental settings is shown in Fig-4, compared with the two crossover operators and the three similarity measures. In all cases there is an optimal maximum lag depth which seems to be controlled by the kind of crossover operator.

4.2. Influence of genetic algorithm parameters

Using the standard settings, the number of generations (Ng) exhibits an approximately negative exponential relationship with MSE. For example, 10 generations ($Ps = 50$) take 16 secs, whereas 1000 takes 1212 secs. On grand average, the complete construction, evaluation and representation of a model (including constructing and training the neuro-fuzzy network) takes only 0.025 secs. Replacing the kind of neuro-fuzzy network used in this algorithm with a classical one (e.g. a standard feed forward network trained with back propagation), would increase the time taken for the model mining process to impractical levels.

In general the differences introduced by the use of different crossover operators are small. In 8 replicate experiments with the standard settings the average observed MSE values were 18.79 and 19.05 for single-point and double-point respectively. Nevertheless, the mean for single-point is slightly smaller and this is reflected in the corresponding ranges $[18.61 - 19.26]$ and $[18.71 - 19.54]$ respectively.

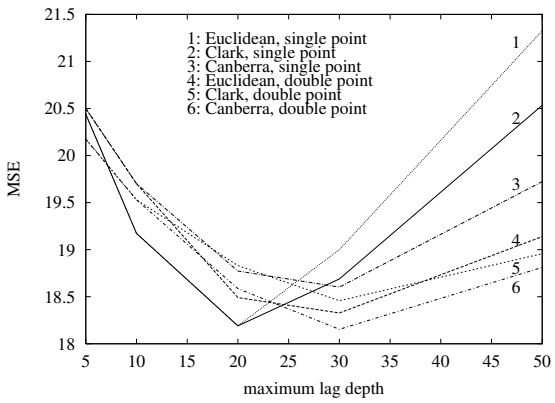

Figure 4: Behavior of the maximum lag depth w.r.t. the similarity function and the crossover operator.

Crossover probability doesn't seem to substantially affect MSE behavior. For the crossover values selected for the experiments with the standard settings, the MSE ranges over $[18.61 - 19.25]$ with a flat minimum containing four values falling at 18.61. However, under the same conditions, the mutation probability (Mp) does seem to have a slight positive effect on MSE, as it varies in a narrow range $[18.39 - 18.88]$.

4.3. Influence of parallel implementation

Since in all 180 experiments the number of units was kept to the maximum value available -3-, the only potentially influencing factor is Ns. The fastest observed time (137 secs) was with the number of slaves equal to the total number of available CPUs -6-. With 30 slaves, the time increased non-linearly to 168 secs, due to management overhead in the cluster (network comminication, etc). This performance is quite acceptable taking into account that it corresponds to the construction and evaluation of 5000 neuro-fuzzy networks and that this technique was conceived for model mining and not for real-time forecasting.

4.4. Examples

Fig-5 shows the approximations given by the best and worse models found for the sunspots data (univariate case) in the 180 experiments described in the previous sections.

A second example data set consisting of 7 time series of average monthly temperatures from different sites in California, recorded during the period 1895-1989 [4], is shown in Fig-6. The Central Coast Drainage (the top series) was chosen as the target for a model mining experiment with the standard parameters except for Rp which was set to 75%. The performance of the best model found for the test set is shown in Fig-7.

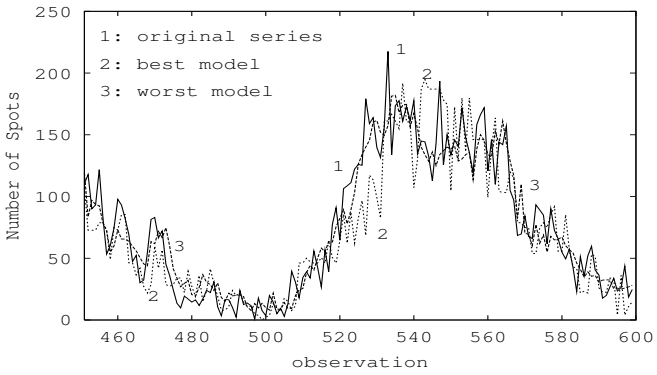

Figure 5: Best and worst models for the sunspots time series data (test set). MSE (best)=17.43, MSE(worst)=25.78.

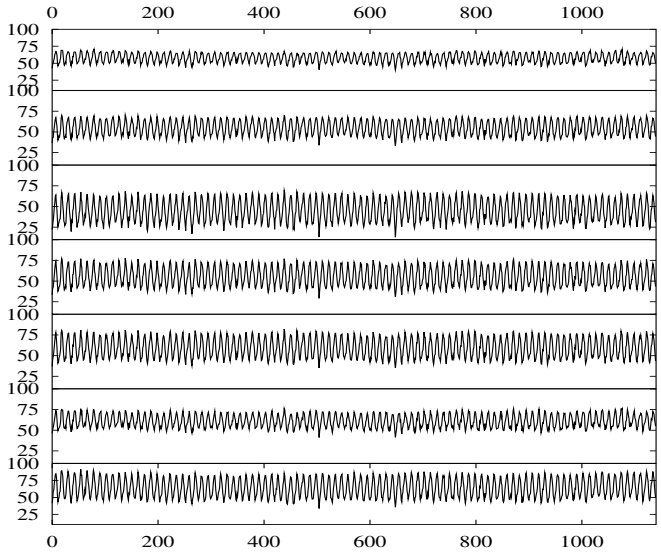

Figure 6: Temperature data from 7 California sites.

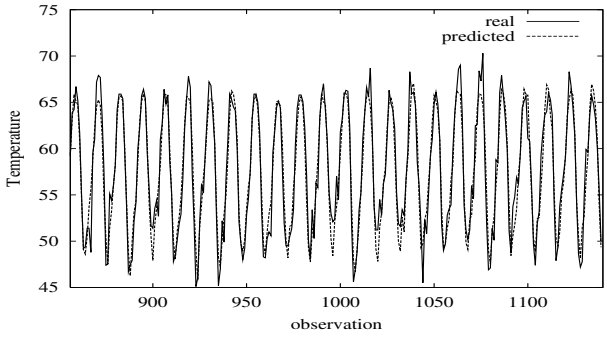

Figure 7: Behavior of the best model found for the target series (Central Coast) in the test set. MSE = 1.811373.

5. CONCLUSIONS

The results are still preliminary and conditioned to the underlying properties of the particular data set used, but they show that the algorithm is fast, robust, and able to discover reasonably good models. Errors are low for most parameter combinations, indicating that for data of this kind first approximation models and predictions can be obtained very quickly. The speed with which models can be generated and explored, makes it a suitable data mining technique for rapid prototyping. Once a set of reasonable models has been found, more sophisticated or accurate techniques can be used in a subsequent stage. Clearly, further experiments are necessary with larger and more complex data sets in order to determine the advantanges and limitations .

6. REFERENCES

[1] Belanche, Ll. : Heterogeneous neural networks: Theory and applications. PhD Thesis, Department of Languages and Informatic Systems, Polytechnic University of Catalonia, Barcelona, Spain, July, (2000)

[2] Chandon, J.L., Pinson, S. : *Analyse Typologique. Thorie et Applications*. Masson, Paris, (1981)

[3] : *MPI Primer/ Developing with LAM*. Ohio Supercomputer Center. The Ohio State University, (1996)

[4] Masters, T. : *Neural, Novel & Hybrid Algorithms for Time Series Prediction*. John Wiley & Sons, (1995)

[5] Valdés, J.J., García, R. : A model for heterogeneous neurons and its use in configuring neural networks for classification problems. *Proc. IWANN'97, Int. Conf. On Artificial and Natural Neural Networks*. Lecture Notes in Computer Science **1240**, Springer Verlag, (1997), 237–246

[6] Valdés, J.J., Belanche, Ll., Alquézar, R. : Fuzzy heterogeneous neurons for imprecise classification problems. *Int. Jour. Of Intelligent Systems*, **15** (3), (2000), 265–276.

[7] Valdés, J.J. : Time Series Models Discovery with Similarity-Based Neuro-Fuzzy Networks and Evolutionary Algorithms. *IEEE World Conference on Computational Intelligence* WCCI'2002, Hawaii, USA, 2002.

[8] Wall, T. : *GaLib: A C++ Library of Genetic Algorith Components*. Mechanical Engineering Dept. MIT (http://lancet.mit.edu/ga/), (1996)