Time Series Models Discovery with Similarity-Based Neuro-Fuzzy Networks and Evolutionary Algorithms

Valdés, Julio

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC-CNRC

*Time series models discovery with similarity-based neuro-fuzzy networks and evolutionary algorithms. **

Julio J. Valdés

May 2002

* published in: The 2002 IEEE World Congress on Computational Intelligence (WCCI'02), Proceedings of the 2002 International Joint Conference on Neural Networks, Honolulu, Hawaii, May 12-17, 2002. pp. 2345-2350. NRC 44901.

Canada

Time Series Models Discovery with Similarity-Based Neuro-Fuzzy Networks and Evolutionary Algorithms

Julio J. Valdés
Institute for Information Technology
Integrated Reasoning Group
National Research Council of Canada
1200 Montreal Road, K1A 0R6
Canada
julio.valdes@nrc.ca

Abstract: The discovery of patterns of dependency in heterogeneous multivariate dynamic systems is approached with similarity-based neuro-fuzzy networks and evolutionary algorithms. Search space contains general auto-regressive non-linear models representing the dependency structure of the process. Examples show that the proposed approach gives better results than the classical statistical one.

1 INTRODUCTION

Multivariate time-varying processes, are common in a wide variety of important domains like medicine, economics, industry, communications, environmental sciences, etc. Processes of this kind are usually described by sets of variables, sometimes of heterogeneous nature. Some are numeric representing quantitative magnitudes, and others are non-numeric and describe variation in terms of discrete states. Typical from real world settings is the practical impossibility of recording all variables at all time frames, thereby generating *incomplete information*. On another hand, the degree of accuracy associated with the observed variables is very irregular, resulting in data corpuses with different types and levels of *imprecision*. In recent times, developments in sensor and communication technology enables the simultaneous monitoring and recording of large sets of variables quickly, therefore generating large sets of data.

Most of the classical methods for the study of time dependent data are limited in their applicability by the problems just mentioned. Many of them are based on ideal assumptions which are not fulfilled by real data, or suffer from lack of robustness. Some of them can't be applied in the presence of missing values, or were derived only for single-type data (usually for real valued only). Often they turn too complicated or intractable with the increase in dimensionality, or can't tolerate or even account for imprecision. Finally, the univariate case (or single channel time series) has been the most studied w.r.t the multivariate, for complexity reasons.

Predicting a system's behavior is one of the most important problems investigated, and there is a large set of algorithms and techniques developed from a variety of conceptual approaches [1], [2]. They have very different theoretical foundations, competence, robustness and interpretability. Some of them are based on models concerning the system's structure and composition while others are model-free and rely only on general input-output descriptions. In recent years, model-free approaches based on *soft-computing* techniques [3] are receiving increasing attention. In particular, neural networks, fuzzy systems, evolutionary algorithms and hybrid techniques involving these are excellent function approximators and have been used extensively for time series analysis and prediction [4], [5]. In order to train a neural network or to set a fuzzy prediction system, a model of the time dependencies has to be set forth in advance. In complex, highly multivariate, or poorly known processes, these patterns of internal dependencies are unknown and precisely what is required is to discover them in order to construct suitable prediction operators. The problem of finding these models of internal dependencies has received much smaller attention in comparison with the construction of accurate prediction operators.

This paper presents a soft computing approach to *model discovery*. This is an extremely complex problem and instead of trying to seek a general solution, the present approach presents a strategy for model finding *within* a particular class of dependency patterns and functional relationships. A particular family of models is chosen and the search space of possible particular models is explored with evolutionary algorithms (in this paper with genetic algorithms). Model quality during the evolutionary process is evaluated by constructing a similarity-based neuro-fuzzy network representing the functional relationship and computing its prediction error. This kind of network has the advantage of having extremely short training time, therefore allowing the fast construction and evaluation of many candidate models, as required by the evolutionary algorithm. Once a set of "optimal" models is found, they can be used in a second step for building more accurate prediction operators. These might be feed-forward networks, radial basis functions, fuzzy predictors, or others, using more sophisticated training methods (usually also more time consuming).

2 PROBLEM FORMULATION

The objective is to analyze a multivariate time varying process and to extract plausible dependency models

expressing the relationship between future values of a previously selected time series (the *target*), and the rest of the time series, possibly including itself. From the point of view of the nature of the variables composing the process, some might be numeric (ratio or interval scales), and some qualitative (ordinal or nominal scales), for instance, a Markov chain. Moreover, these series might contain missing values. The set of possible functional models describing the dependency of future values of a target series on the previous values of the others and itself is clearly unlimited. The classical theory of time series has studied extensively the AR, MA, ARMA and ARIMA models (all linear) [1], and others can be considered as well. From the methodological point of view considered here, the main goal is the *pattern* of mutual dependencies. Clearly, the particular choice of the *functional family* will influence the overall result. In this respect, generalized AR or ARMA are intuitively appealing and the simplest one is the generalized non-linear AR model.

As stated previously, the methodology presented here does not rely on a model of a particular kind. However, the generalized AR model expressed in (1) will be used as an example to illustrate the proposed approach, which can be easily extended to other time series models.

Once the model type is chosen, the problem of pattern dependency search can be described as 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.*

$$S_{\text{target}}(t) = \mathrm{F}\begin{pmatrix} S_1(t-\tau_{1,1}), S_1(t-\tau_{1,2}), \ldots, S_1(t-\tau_{1,p_1}), \\ S_2(t-\tau_{2,1}), S_2(t-\tau_{2,2}), \ldots, S_2(t-\tau_{2,p_2}), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, \\ S_n(t-\tau_{n,1}), S_n(t-\tau_{n,2}), \ldots, S_n(t-\tau_{n,p_n}) \end{pmatrix} \quad (1)$$

where :

n : Number of signals.

$S_1, S_2, \cdots, S_n$: Signals.

F: Unknown function.

$p_1, \cdots, p_n$: Number of lags considered for signal S_i.

$\tau_{i,j}$: $j-th$ time lag corresponding to signal i.

The exponential size of the space of possible models (even for only a few series and a limited number of allowed time lags), prevents an approach based on exhaustive search. Also, the lack of assumptions about the prediction function makes the set of candidates unlimited.

2.1 A SOFT COMPUTING MODEL MINING STRATEGY

A direct soft computing approach to the model mining problem can be based on: (a) exploration of a subset of the entire model space with evolutionary algorithms, and (b) use of a neural network of a fuzzy system representation for the unknown prediction function. The use of evolutionary algorithms avoids the search of the entire model space and gives as a result a set of "quasi-optimal" models, in the sense of the model quality criterion defined in advance. The use of a neural network, due to its properties as a general function approximator, allows a flexible, robust and accurate predictor function operator. Feed-forward networks, radial basis functions or other network paradigms are classical choices. However, the use of these classical network paradigms in (b), might turn out to be difficult or even prohibitive, considering that for each candidate model, a function of this kind has to be constructed (i.e. trained) during search space exploration, thus, involving long training times. Issues like the determination of the number of neurons in the hidden layer, the mixing of numeric and non-numeric information (required in real world multivariate problems), and the inclusion of imprecise values add even more complexity. These can be treated in a natural way by using the heterogeneous neuron model [6], [7]. This model considers a neuron as a general mapping from a heterogeneous multidimensional space composed by cartesian products of the so called *extended sets*, to another heterogeneous space. These are formed by the union of real, ordinal, nominal, and fuzzy sets (other kind of sets are possible), with the missing value. Their cartesian product forms the heterogeneous space.

$$\hat{\mathcal{H}}^n \equiv \hat{\mathcal{R}}^{n_r} \times \hat{\mathcal{O}}^{n_o} \times \hat{\mathcal{N}}^{n_n} \times \hat{\mathcal{F}}^{n_f} \quad (2)$$

In this type of neuron (h-neuron), the inputs, as well as the weights, are elements of the n-dimensional heterogeneous input space. Among the many kinds of mappings which can be defined, the one using a *similarity function* [8] as the aggregation function and the identity mapping as the activation function is particularly appealing. 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. Such neuron is shown in Fig-1.

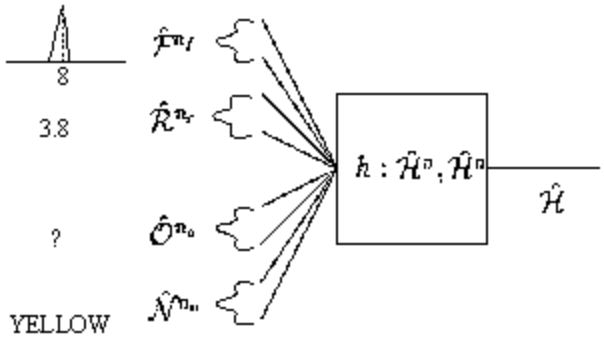

Fig-1. A heterogeneous neuron with fuzzy, real, ordinal and nominal inputs. (? is a missing input value).

This kind of neuron can be used in conjunction with the classical (with dot product as aggregation and the sigmoid function or hyperbolic tangent as activation, respectively), forming hybrid network architectures. Networks constructed in this way exhibit general function approximation properties [9]. The training of such networks is usually done with evolutionary algorithms due to the lack of continuity, typical of heterogeneous spaces, and the presence of missing values which precludes the use of backpropagation-like algorithms.

Since the aggregation is given by a similarity function, there are many possible choices. Moreover, the input data structure can be taken into consideration, resulting in tailored neurons, more sensible to particular data properties.

A type of hybrid network especially appropriate for the task of model mining can be constructed by joining a hidden layer composed by h-neurons with an output layer having classical neurons (for example, with dot-product as aggregation and a linear function as activation). The weights of the output layer neuron are defined as the corresponding values of the target series when the input is a vector identical to the weight vector of a neuron in the hidden layer. In the case of predicting a single real-valued target time series, the architecture is shown in Fig-2.

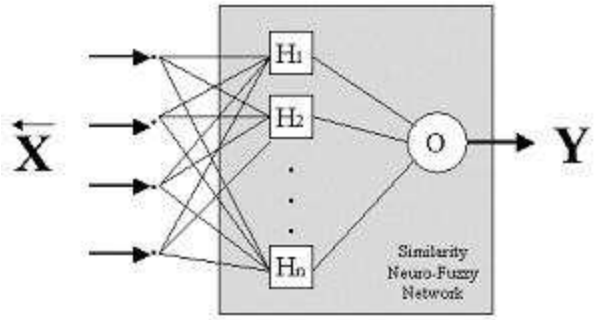

Fig-2. A hybrid network with h-neurons in the hidden layer and one classical neuron (O) in the output layer. Input is a multivariate vector and output is a real value .

The operation of this network can be defined as a casting of a k-best interpolator algorithm: Let each neuron in the hidden layer compute its similarity with the input vector and retain the k-best responses, where k is a pre-set number of selected h-neurons. If there is a hidden neuron identical to the input, then its output (i.e. its similarity) is 1. Then, set the output of the other neurons in the hidden layer to 0 (i.e. disregard them) and go to the output layer. As they are neurons with classical dot-product aggregation, the resulting value will be exactly the weight of the output layer neuron connected with the single responsive h-neuron with similarity 1. Therefore, the network response will be the observed target series value. If there is no h-neuron identical to the input vector, then retain the k-best similarity values, set to 0 the outputs of the rest of the hidden layer neurons and compute the dot-product aggregation of the neurons in the output layer. Using as activation a linear function with a single coefficient, equal to the inverse of the sum of the k-similarities coming from the hidden layer, the output is the estimate given by (3).

$$output = \left(1/\Theta\right)\sum_{k\in K} h_k w_k \ , \qquad \Theta = \sum_{k\in K} h_k \qquad (3)$$

where K is the set of k best h-neurons of the hidden layer and h_k is the similarity value of the k-best h-neuron w.r.t the input vector. Since those similarity values represents the fuzzy memberships of the input vector to the set classes defined by the neurons in the hidden layer, (3) represents a fuzzy estimate for the predicted value.

Assuming that a similarity function S has been chosen and that a single time series is the target, this "case-based" neuro-fuzzy network can be built and trained as follows: Define a similarity threshold T and extract the subset L of the set of input patterns such that for every $l \in L$ and every input pattern x, $S(x,l) \geq T$. Several algorithms for extracting subsets with this property can be constructed in a single cycle through the input pattern set, and are classical in cluster analysis (note that in the particular case of a threshold $T=1$, the hidden layer becomes the whole training set). Now construct the hidden layer by using the elements of L as h-neurons, and use their corresponding outputs as the weights of the output layer neuron O. This training procedure is extremely fast and therefore many hybrid neuro-fuzzy networks of this kind can be constructed and tested. The dimension and composition of input training vectors will depend on the dependency model considered for the set of time series. Different sets of individual lags selected from each time series will define different training sets, and therefore, different hybrid neuro-fuzzy networks. This one-one correspondence between dependency models and neuro-fuzzy networks (defined as above), makes the search in the space of models equivalent to the search in the space of networks. Then, given a model describing the dependencies and a set of time series, a network can be constructed according to the described procedure, and tested for its prediction error on a segment of the target series not used for training (building) the network. Root mean squared error on a test set is a typical measure.

Clearly, for each model there is a quality indicator given by the prediction error on a test set of its equivalent similarity-based neuro-fuzzy network, which is also a representation of the prediction function controlled by the dependencies expressed in the model. In this way, the search for "optimal" models can be made with an evolutionary algorithm minimizing the prediction error measure. Genetic Algorithms and Evolution Strategies are well suited for this task. In the case of genetic algorithms a simple model coding can be used with binary chromosomes of length equal to the sum of the number of lags considered for each of the time series. Clearly, other problem representations are possible. Within each chromosome segment corresponding to a given series,

the non-zero values will indicate which time lags should be included in the model, as shown in Fig-3.

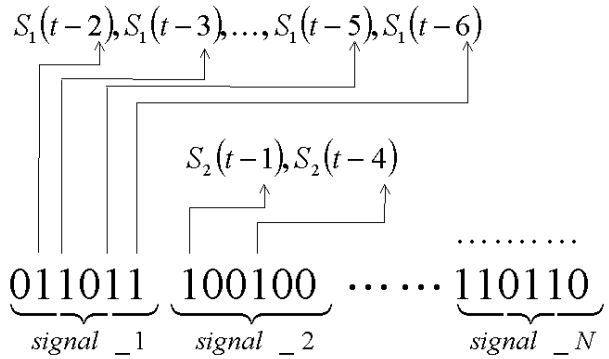

Fig-3. Binary chromosome decodification.

The set of multivariate series is divided into two parts: training and test. Given a binary chromosome, a model is constructed by applying the decodification illustrated in Fig-3. With the model and the multivariate series, a hybrid neuro-fuzzy network is constructed and trained, giving a representation for the prediction function. Then, the network is applied to the test set and a prediction error is obtained. This prediction error is also a measure of network (and model) quality and is used by the genetic algorithm selection and crossover internal operators. Models with smaller prediction error are the fittest. The entire process is illustrated in Fig-4.

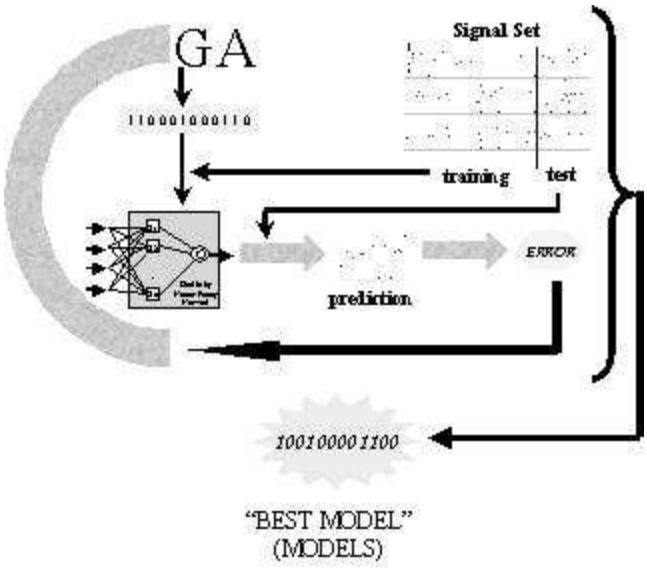

Fig-4. Genetic search in the space of dependency models.

Chromosomes representing models are used for constructing neuro-fuzzy networks which are trained and evaluated, thus rating models according to their prediction error.

At the end of the evolutionary search, the best model (or models) are obtained and if the associated test errors are acceptable, they represents meaningful dependencies within the multivariate process. It is well known that evolutionary algorithms can't guarantee the discovery of the "true" optimum. Thus, the set of models found can be taken only as plausible descriptors of important interrelationships present in the data set. It must be taken into account that models are ranked and evolved according to the prediction error given by the function represented by the particular kind of neuro-fuzzy network used, therefore, maybe other kind of neural networks based on the same model may have better aproximation capabilities. In this sense, the proposed scheme can be seen as giving a *coarse* prediction operator. The advantage is the speed with which many thousands of models can be explored and tested in a systematic way. Once the best of them are found, they can be refined by using more powerful function approximators like other type of neural networks, fuzzy systems, or other techniques, like those of [5], [10] and many others.

3 EXAMPLES

The described technique has been applied to different examples from the literature [11], and a few are presented here for illustrative purposes. Deliberately, no preprocessing of any kind was made to any of the time series used. In other words, raw data were used in all cases. This is not the way in which time series data are analyzed, but by eliminating additional effects the properties of the proposed procedure in terms of approximation capacity and robustness are easier to asses.

For simplicity, the same set of parameters was kept as fixed as possible, with only minor exceptions. In particular, in all cases the similarity threshold for the h-neurons was set to 1. The similarity function used was $S=1/(1+d)$, where d is a normalized euclidean distance, and the number of responsive h-neurons in the hidden layer set to k=7. No attempt to optimize these was made, but the subject will be commented later. The genetic algorithm was applied in its simplest form, with: roulette selection, single point crossover, crossover rate=0.6, mutation rate = 0.01, generation gap=1 (i.e. the entire population is replaced in each generation). All structures were evaluated in each generation and elitism was allowed.

3.1 LYNX DATA

This univariate process describes the annual number of Canadian lynx trapped on the Mackenzie River for the years 1821-1934 [11], and it is an example of a quasi-periodical series with 114 observations. In this case, the first 90 were used as training and the remaining 24 for testing. A maximum time lag of 20 years was set, defining a search space size of 2^(20) models. After 20000 generations with 500 individuals each a stable solution was found starting from generation 14500. The best model found relates future values at time t with the values at lags (t - 1), (t - 2), (t - 10), (t - 14), (t - 15), with a RMS test set prediction error of 549.2.

For comparison, an ARIMA model (20,0,0) was computed under the same conditions, giving a RMS error of 1516.18.

The soft-computing model not only gives better results, but also a simpler dependency structure.

In the soft computing model miner technique only 5 time lags were found to be relevant for the best model found (only 25% of the number of potentially available lags). Actually, the best 5 models resulting from the genetic exploration, have an average of 4.6 lags per model with prediction errors in the range (549.2- 565.7). However, note that in the case of the classical AR model, 20 coefficients are required. When dealing with highly multivariate processes and exploring deeper time windows in terms of maximum lags, big difficulties may arise. Moreover, this model can be used only in the presence of real-valued data.

The behavior of the predicted time series in the test set segment (observations 91-114) for the proposed technique and the AR model of order 20 is shown in Fig-5.

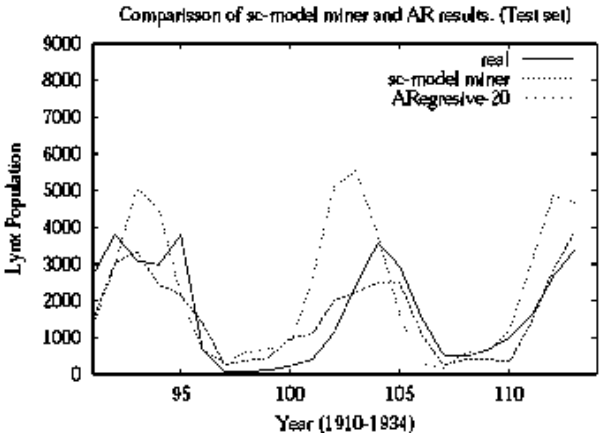

Fig-5. Comparison of the real and predicted lynx population in the test set according to the proposed soft computing technique and a classical autoregressive model of the same order.

It is interesting to observe that the soft-computing technique is not only better in terms of RMS error and model simplicity, , but also in terms of its phase behavior w.r.t the original process. This is not the case of the classical AR model.

3.2 MACKEY-GLASS SERIES.

This is a well known chaotic time series defined as

$$x(t) = 0.9\,x(t-1) + \frac{0.2\,x(t-\tau)}{1 + x^{10}(t-\tau)} \qquad (4)$$

Here $\tau = 18$ was used and a series with 1200 observations was generated. The first 1000 were used for training and the remaining 200 for test. A maximum time lag of 20 time intervals was set, and 1000 generations with 100 individuals each were used in the genetic exploration with the same settings described above. The best model found had a RMS error = 0.00499 and was composed by the following set of lags: lags (t -1), (t - 4), (t - 17), (t -18), (t - 19). The theoretical set of lags is (t-1), (t-18) were retrived, but also a spurious lag at (t-4), as well as two others at (t-17) and (t-19), however, they are symmetrical around the expected (t-18). Only as reference, an ARIMA (20,0,0) model was computed under the same conditions w.r.t the training and test sets, and the results are shown in Fig-6.

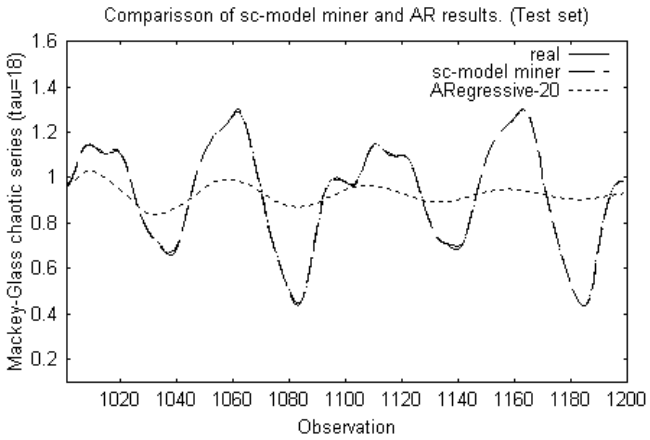

Fig-6. Comparison of the real and predicted Mackey-Glass series in the test set according to the proposed soft computing technique and a classical autoregressive model of the same order.

This example shows that the proposed technique is effectively sensitive to the interdependencies hidden in a process, which is its main purpose. The fact that the approximation matches so closely the theoretical function shows that the influence of the (t-4) is small and that the overall model is a plausible one. Probably other with other experimental parameters could narrow the gap between the real underlying dependency model and the one found by the genetic algorithm.

3.3 GAS FURNACE DATA

In this example, input Gas Rate and Output CO2 in a chemical plant define a bivariate process [11]. The 246 available observations were divided into a training set with the first 249 and a test set with the remaining 47. In this case the model included dependencies from both the input and the output series and a maximum exploration lag of 50 was set for each. The target series was the Output CO2. Only 20 populations were generated, with 50 individuals each. The rest of the genetic algorithm parameters were the same as in the previous experiments.

The best model found was composed by 26 lag terms from the input Gas Rate series, and 25 from the Output CO2, for a total of 51 lags. This is almost one half of the potentially allotted model size. The RMS error obtained for the best model was 2.045 and for the 5 best found the error range was (2.045-2.308), with an average of 48.4 lag-terms/model. Clearly this result indicates the ability of the technique to find dependency models with reasonable size and accuracy.

The predicted Output CO2 for the test set is shown in Fig-7. The prognosed behavior follows the real values reasonably well in both the magnitude and the phase.

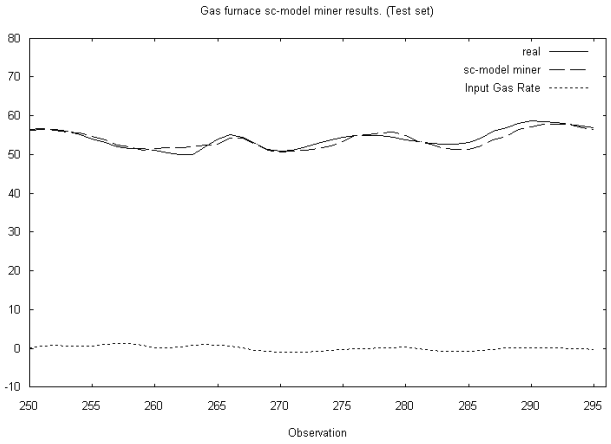

Fig-7. Comparison of the real and predicted Output CO2 in a bivariate process. Only the test set part is shown.

This is confirmed by a regression analysis of the expected and predicted values shown in Fig-8. A highly significant correlation coefficient of 0.908 was obtained and it is also interesting to observe that the slope is almost 1, as should be expected in the case of theoretical coincidence.

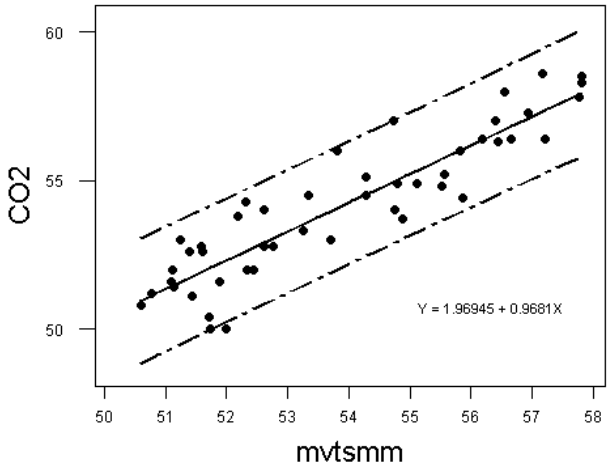

Fig-8. Regression analysis of the real and the multivariate time series model miner prediction for Output CO2 (mvtsmm). Dashed lines shows the 95% confidence band. Correlation coefficient is 0.908.

4. CONCLUSIONS

The proposed soft-computing technique constructed with similarity-based neuro-fuzzy networks and evolutionary algorithms is a reasonable approach to the problem of determining the dependency structure of complex multivariate heterogeneous time-dependent processes. This is shown by the presented examples, some of them using real world data. As with other data exploration techniques, its results can be further refined by using more sophisticated prediction operators, once the dependency structure is known or approximated. The approach exhibit a fair degree of robustness and non-linearity as shown by the examples, in which the raw time series were used without any preprocessing, and the comparison with the classical auto-regressive model.

The technique, like many others, depends on different parameters which must be set forth in advance (some of them are conceptual, like the similarity function used). Their influence on the results must be investigated, as well as the limits for its applicability. For this purpose, meta-evolutionary paradigms are particularly appropriate, as a higher order evolutionary process can explore the space of the parameters. Also, more experiments must be made in order to study the influence of other factors, like the size of the training set required, the tolerance to imprecision and missing information, etc. Nevertheless, this results are only preliminary and further experiments, research and comparison with other approaches are required.

References:

[1] G. Box, G. Jenkins. *Time Series Analysis, Forecasting and Control*, Holden-Day, 1976.

[2] A. Lapedes, R. Farber. *Nonlinear signal processing using neural networks: prediction and system modeling*, Tech. Rep. LA-UR-87-2662, Los Alamos National Laboratory, NM, 1987.

[3] L. Zadeh., *The role of soft computing and fuzzy logic in the conception, design and deployment of intelligent systems*. Proc. Sixth Int IEEE Int. Conf. On Fuzzy Systems, Barcelona, july 1-5, 1997.

[4] G. Klir, *Architecture of Systems Problem Solving*, Plenum Press, 1985.

[5] A. Tucker, S. Swift, X. Liu. *Variable grouping in multivariate time series via correlation*. IEEE Trans. Sys. Man and Cyber. Part B, Vol 31, (2), pp235-245, 2001.

[6] J.J. Valdés, R. García, *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, pp.237-246.

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

[8] J.L.Chandon, S. Pinson, *Analyse Typologique. Théorie et Applications*. Masson, 1981.

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

[10] D. Specht, *Probabilistic Neural Networks*, Neural Networks, Vol. 3, 1990, pp. 109-118.

[11] H.J. Newton, *Timeslab: A Time Series Analysis Laboratory*, Wadsworth & Brooks Publishing House, 1988.