Mining Multivariate Heterogeneous Time Series Models with Computational Intelligence Techniques

Valdés, Julio; Barton, Alan

Proceedings of the IASTED International Joint Conference on Artifical Intelligence and Applications, 2003

National Research Council Canada
Institute for Information Technology

Conseil national de recherches Canada
Institut de technologie de l'information

*Mining Multivariate Heterogeneous Time Series Models with Computational Intelligence Techniques **

Valdés, J., and Barton, A.
September 2003

* published in Proceedings of the IASTED International Joint Conference on Artifical Intelligence and Applications. September 8-10, 2003, Benalmadena, Spain. ISBN: 0-88986-390-3, ISSN: 1482-7913, ACTA Press, Anaheim, USA, pp. 240-245. NRC 46513.

Mining Multivariate Heterogeneous Time Series Models with Computational Intelligence Techniques.

Julio J. Valdés
National Research Council of Canada,
Institute for Information Technology,
1200 Montreal Road,
Ottawa ON K1A 0R6, Canada
email: julio.valdes@nrc-cnrc.gc.ca

Alan J. Barton
National Research Council of Canada,
Institute for Information Technology,
1200 Montreal Road,
Ottawa ON K1A 0R6, Canada
email: alan.barton@nrc-cnrc.gc.ca

ABSTRACT
This paper presents experimental results of a computational intelligence algorithm for model discovery and data mining in heterogeneous, multivariate time series, possibly with missing values. It uses a hybrid neuro-fuzzy network with two different types of neurons trained with a non-traditional procedure. Models describing the multivariate time dependencies represent dependency patterns within the signals and are encoded as binary strings representing neural networks (evolved using genetic algorithms). The present paper studies its properties from an experimental point of view focussing on: *i)* the influence of missing values, *ii)* the factors controlling the model search process, and *iii)* the effectiveness of the time series prediction results. Results confirm that the algorithm *i)* possesses high tolerance to missing data, *ii)* has an error distribution skewed towards lower error values, *iii)* is capable of learning good models within large signal sets.

KEY WORDS
Neuro-fuzzy-genetic Architectures, Parallel Data Mining, Signal Processing, Neural Networks, Genetic Algorithms, Forecasting and Prediction

1 Introduction

Multivariate time series modelling and prediction is a very important subject because of the recent developments in sensor, communication and computer technologies, which allow the monitoring of complex processes in many domains (industry, environment, medicine, economics, etc.). It is especially important (in poorly known or altogether unknown complex processes) to discover patterns of delayed cause-effect dependencies. That is, to attempt to discover dependencies of one given target signal on past values of all signals (including itself) for particular time lags. This situation becomes more complex when the multivariate time dependent process is composed of heterogeneous variables (numeric, non-numeric, fuzzy quantities and others), and when missing information is present. Finding models in this situation is a formidable task, given the large size of the search space and the complexity of the operations involved. A soft-computing algorithm oriented to this kind of problem was developed elsewhere [9], as well as a parallel implementation [10], using both a special kind of hybrid neuro-fuzzy network and a genetic algorithm. This paper studies its properties in more depth from an experimental point of view, focussing on the influence of a broad range of missing values, the factors controlling the parallel computation, and on the effectiveness of the time series prediction results.

2 The Algorithm

The purpose is to discover *dependency models* in heterogeneous, multivariate, time varying, complex processes from which observations with different degrees of completeness are obtained. A model expresses the relationship between values of a previously selected time series (the target), and a subset of the past values of the entire set of series. Heterogeneity means both *i)* the presence of ratio, interval, ordinal or nominal scales, fuzzy and other magnitudes, and *ii)* the presence of different *physical* magnitudes contained within the signal set (e.g. temperatures, pressures, radiation levels, etc). The class of functional models considered by this algorithm is a generalized non-linear auto-regressive (AR) model (eq-1) (other functional models are also possible),

$$S_T(t) = \mathrm{F}\left(\begin{array}{c} S_1(t-\tau_{1,1}), \cdots, S_1(t-\tau_{1,p_1}), \\ S_2(t-\tau_{2,1}), \cdots, S_2(t-\tau_{2,p_2}), \\ \cdots \\ S_n(t-\tau_{n,1}), \cdots, S_n(t-\tau_{n,p_n}) \end{array}\right) \quad (1)$$

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. This approach requires the simultaneous determination of: *i)* the number of required lags for each series, *ii)* the particular lags within each series carrying the dependency information, and *iii)* the prediction function. A requirement on function F is to minimize a suitable prediction error. This is approached with a soft computing procedure based on: (a) exploration of a subset of the *model space* with a genetic algorithm, and (b) use of a similarity-based neuro-fuzzy system representation for the unknown prediction function. Statistical or other classical approaches either have difficulties handling these kinds of situations or can not handle them at all [2] [7].

Evolving neuro-fuzzy networks with genetic algorithms has been explained in the literature for training *single* networks [6], [3], [5]. Thus, the use of conventional architectures and training procedures becomes prohibitive (due to the long waiting time for computations to complete). Other difficulties with classical approaches include *i)* trying to find the number and composition of the network hidden layers, *ii)* using mixed data types (numeric, non-numeric, fuzzy, etc), and *iii)* properly treating missing values. However, here the situation involves the construction and evaluation of *thousands* or *millions* of networks (i.e. because the network space is equivalent to the model space).

The search for models of the form given by eq-1 is performed with a genetic algorithm in conjunction with a neuro-fuzzy network. The dependency structure of the model is given by the collection of lags (delays) $\tau_{i,k}$ and they are encoded as a binary string, which can be used as the chromosome representation by the genetic algorithm. Assuming that the dependencies are bounded by a maximum lag value W for all N signals, a binary string of length $N \cdot W$ decoded as shown in Fig-1, accounts for a dependency model search space of size $2^{N \cdot W}$. Whereas, the functional component (the function F in eq-1), needs to be found, inside a search space of infinite size. This is approached by using a neuro-fuzzy network which has general function approximation properties. The particular kind of network described below (based on heterogeneous neurons), fulfills this requirement and also has the advantage of having almost instantaneous training time in comparison with classical neural networks. The present approach is based on the heterogeneous neuron model [8], [1], [11], which considers a neuron as a general mapping between heterogeneous multidimensional spaces $h : \hat{\mathcal{H}} \times \hat{\mathcal{H}} \to \mathcal{Y}$, where $\mathcal{Y}$ is an abstract set. If $\overleftarrow{x}, \overleftarrow{w} \in \hat{\mathcal{H}}$ (the input and the neuron weights respectively) and $y \in \mathcal{Y}$, then $y = h(\overleftarrow{x}, \overleftarrow{w})$.

In the *similarity-based* h-neuron model, the aggregation function is given by a *similarity function* $s(x, w)$ between the input and the neuron weights (vectors from a heterogeneous space), whereas the activation is a non-linear function. This neuron maps a n-dimensional heterogeneous space onto the extended [0,1] real interval. The output expresses the degree of similarity between the input pattern and the neuron weights $s : (\hat{\mathcal{H}} \times \hat{\mathcal{H}}) \to [0,1] \cup \{\mathtt{X}\}$, where $\mathtt{X}$ is the symbol denoting the missing value (Fig-2). A hybrid network layout using heterogeneous neurons in the hidden layer and classical neurons in the output layer is suitable for the purpose of model mining. For multivariate heterogeneous time series, where a single time series is targeted for prediction, the network architecture is shown in (Fig-3).

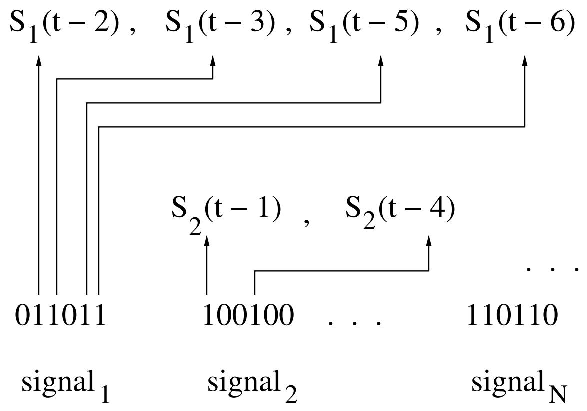


Figure 1. Chromosome decoding for constructing a model based on eq-1 for the given set of signals.

During network operation each hidden layer neuron 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 their weights. 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. The network output is a fuzzy estimate of the predicted value.

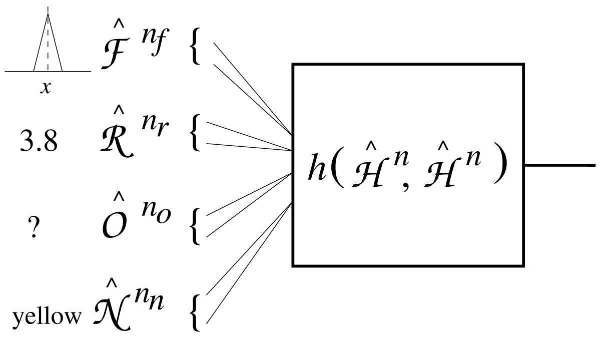

x
$\hat{\mathcal{F}}^{n_f}$
3.8
$\hat{\mathcal{R}}^{n_r}$
?
$\hat{O}^{n_o}$
yellow
$\hat{\mathcal{N}}^{n_n}$
$h(\hat{\mathcal{H}}^n, \hat{\mathcal{H}}^n)$

Figure 2. A heterogeneous neuron with fuzzy, real (crisp), ordinal and nominal inputs.

Given a similarity function $\mathcal{S}$ and a target series the network is built and trained as follows: Set a similarity threshold $T \in [0, 1]$ and extract the subset $\mathcal{L}$ of input patterns Ω ($\mathcal{L} \subseteq \Omega$) such that for every $x \in \Omega$, there exist a $l \in \mathcal{L}$ such that $\mathcal{S}(x, l) \geq T$. The elements of $\mathcal{L}$ will be the hidden layer h-neurons, while the output layer is built

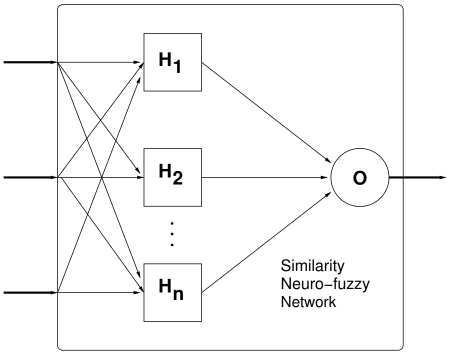


Figure 3. A hybrid neuro-fuzzy network.

by using the corresponding target outputs as the weights of the neuron(s). This procedure is *very* fast and allows for the rapid construction and training of many networks. A distributed parallel implementation following a master-worker (processor farming) approach was made. The system architecture of the parallel implementation of the algorithm is shown in Fig-4.

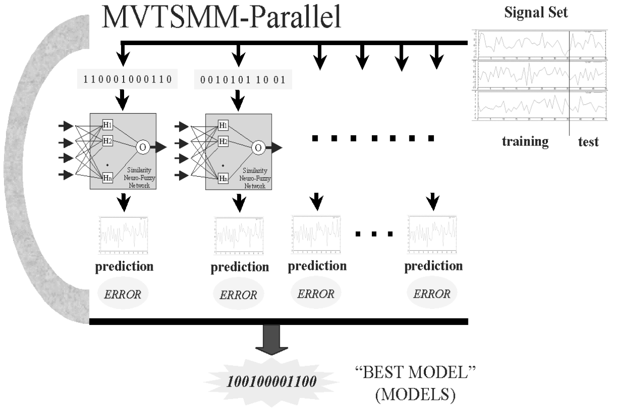


Figure 4. Multivariate Time Series Model Miner System Architecture (MVTSMM). The arc is the parallel genetic algorithm evolving populations of similarity-based networks. They represent different dependency patterns which are generated by the master and evaluated by the workers during the search process.

3 Experimental setup

A multivariate time series data set consisting of 20 series with 1140 observations of average monthly precipitation and temperatures from different sites in the Washington State (USA) was chosen. They were recorded during the period 1895-1989 [4], and compiled by the National Oceanic and Atmospheric Administration (USA). Originally, this data had no missing values and is shown in Fig-5. The precipitation signal for the West Olympic Coastal drainage region (the top series) was chosen as the target for prediction. Contrary to the standard practice in time series analysis, no preprocessing was applied to the time series, in order to test the approximation capacity and robustness of the algorithm in the possibly worst conditions.

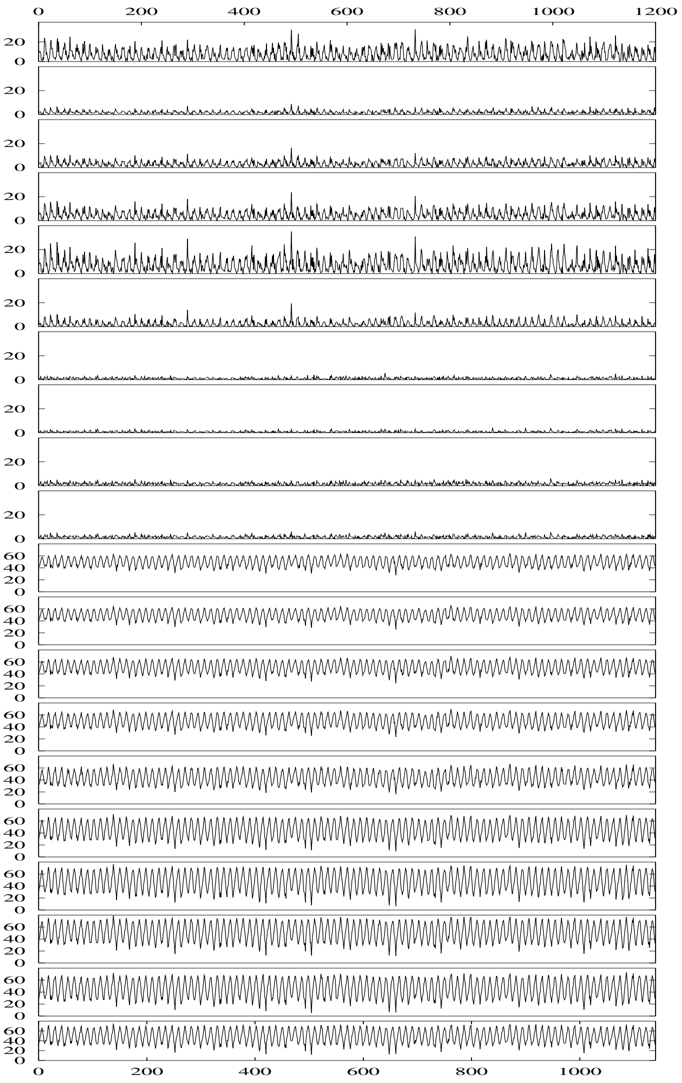


Figure 5. Original data from Washington State observation sites. (Upper 10 are Precipitation series and lower 10, Temperature series).

In addition to the original data, nine sets of time series were constructed by introducing into all 20 original series, 10%, 20%, $\cdots$, up to 90% of uniformly distributed missing values. Missing values were introduced in a *signal-wise* manner. Each series was divided evenly into a training set and a test set. The training set for each signal contains the same percentage of introduced missing values, while the test sets were left intact. In this way, all signals contain exactly the same amount of missing values, as defined by the corresponding preset percentage.

The similarity function used is the non linear transformation ($s = 1/(1+d)$, where s is a similarity and d a distance) of a modified Euclidean distance, accepting missing values. Given two vectors $\overleftarrow{x} =< x_1, \cdots, x_n >, \overleftarrow{y} =< y_1, \cdots, y_n > \in \mathbb{R}^n$, defined by a set of variables (i.e. attributes) $A = \{A_1, \cdots, A_n\}$, let $A_c \subseteq A$ be the subset of attributes s.t. $x_i \neq \mathtt{X}$ and $y_i \neq \mathtt{X}$. The modified distance function is $d_e = (1/card(A_c)) \sum_{A_c} (x_i - y_i)^2$, which is a normalized distance and therefore, independent of the number of attributes. Consequently, *no imputation* of missing values to the data set is performed. That is, they are not replaced by any kind of estimate, but kept as gaps within the corresponding signals. The similarity threshold was set to 1, the number of responsive neurons in the hidden layer was varied in the range [3,11] by 2. The maximum lag depth was varied in the range [5,40] by 5, and the relative percentage of training/test fixed at 50%. For the genetic algorithm, the number of generations was fixed at 100 and the population size to 50. Binary chromosomes encoding model components as given by (equation 1 and Fig-1) were used with a double point crossover operator and standard bit-reversal mutation. Selection was kept constant (roulette wheel method) and complete population replacement with elitism were used. Crossover and mutation probabilities were 0.6 and 0.01 respectively.

Experiments were conceived to observe how Root Mean Squared Error (RMS error) was affected by several factors. Namely, *i)* the size of the maximum time lag from within which the dependencies will be searched (W), *ii)* the number of best responsive neurons used in computing the predicted output estimate, and *iii)* the percentage of missing values contained in the set (from 0 to 90%). A total of 400 experiments were made and from each, the 10 best models were retrieved, for a total of 4000 models investigated.

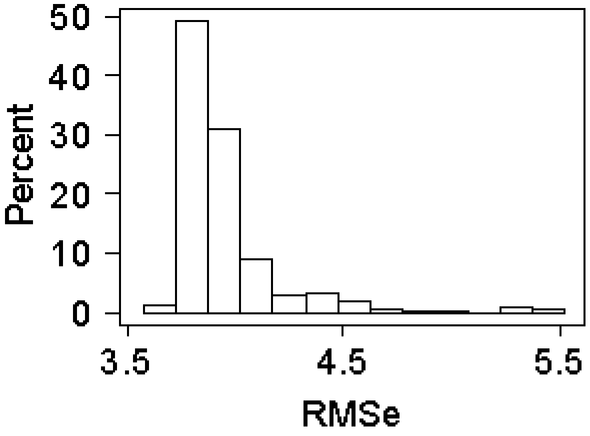


Figure 6. RMS Error distribution for all 4000 experiments

The algorithm was run on a 17 unit cluster at the Institute of Biodiagnostics (National Research Council of Canada), composed by 4 dual Athlon processor units operating at 1.666Ghz frequency, each with 2Gb RAM, 5 Xeon processor units operating at 1.6Ghz with 1Gb RAM, 2 Pentium III units at 1Ghz with 256Mb RAM, 1 Pentium III unit at 1Ghz with 512Mb RAM, 1 Pentium III unit at 930Mhz with 256Mb RAM, and 4 Pentium III units at 860Mhz with 512Mb RAM. This cluster uses Red Hat Linux 7.3, LAM/MPI 6.5.8 and has a shared network drive.

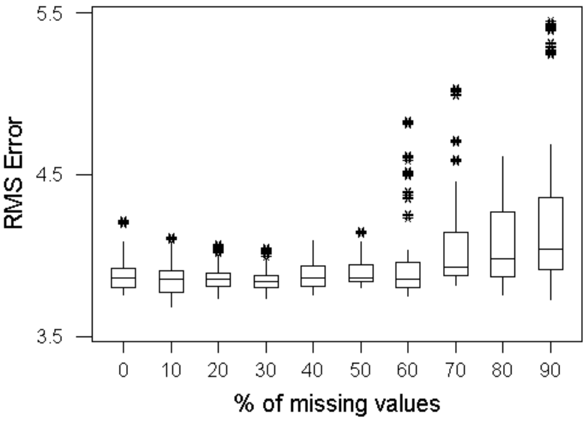


Figure 7. Boxplots showing the main distribution features of the RMS Error for different percentages of missing values, for all 4000 experiments.

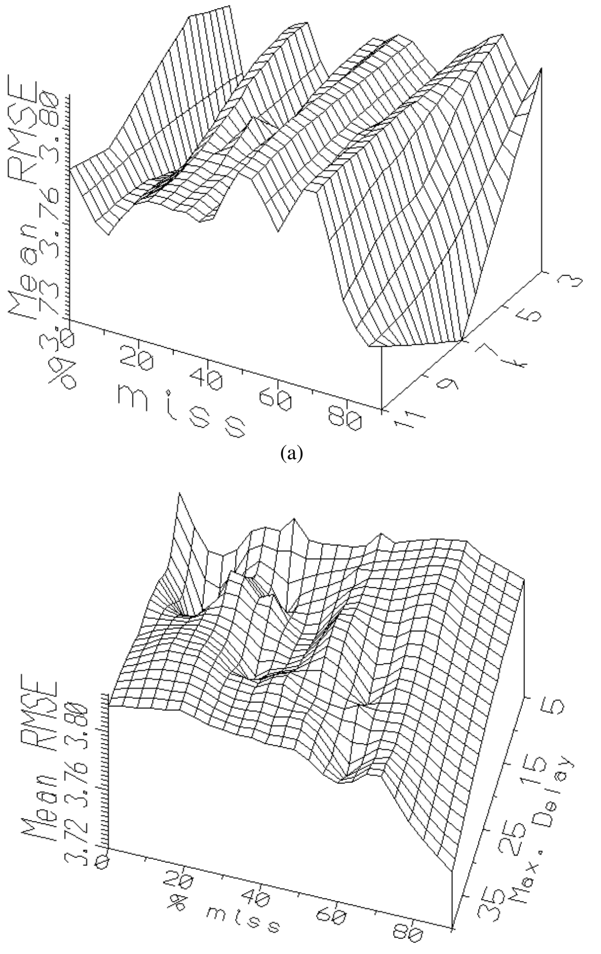
4 Results

The distribution of the 4000 RMS error values (Fig-6) is strongly skewed towards the lower end (Shapiro-Wilkes test rejected normality at 95%), indicating that the algorithm is less prone to give large prediction errors. The RMS error mean was 3.947535 and the bias and standard deviation of its sampling distribution were estimated by bootstrapping to be 0.000136 and 0.004022 respectively. In practical terms, this means that the RMS error is an unbiased estimator.

The relations between the combination of percentage of missing values and the RMS Error are shown in Fig-7 as boxplots of their corresponding distributions.

The different percentages of missing values don't appear to exert a large influence on the RMS error for most of the missing value variants, with the exception of those greater than 70%. Overall, errors for all cases except these last ones, are within a relatively narrow band at the lower end for the Q1-Q3 interval. In the range $[0\% - 60\%]$ there are no significantly different pairwise differences between error distributions. It is remarkable that even when 60% of the data is missing the ability of the algorithm in terms of discovering models capable of producing accurate predictions is not hindered. The algorithm isn't affected until a large number of missing values are introduced into the data set, and even under such bad conditions, the rate of the degrading effect is very low. Notice that a ninefold increase in percentage of missing data barely increases the RMS Error, as indicated by the comparison, for example, between the 90% and 10% cases. All of these features are indicative of very robust behavior.

This skewed distribution is also an indication of the algorithm's robustness, since in 90% of the experiments the time series were incomplete to some degree. The first quartile ($Q1 = 3.8178$) was chosen as an indicator of good error performance. The dependency of mean RMS errors with the number of responsive neurons (*k*) (controlling the fuzzy interpolation of equation-1), and the percentage of missing values is shown in (Fig-8(a)). In general, for a fixed amount of missing values the influence of this parameter is not large. However, for complete or almost complete information, lower values of *k* (fewer terms in 1) have some advantage, whereas for high data shortages, larger values are required (i.e. fuzzy estimates with more terms) as could be expected.



(a)

(b)

Figure 8. (a) effect of % missing and *k* on RMSE, (b) effect of % missing and *W* on RMSE

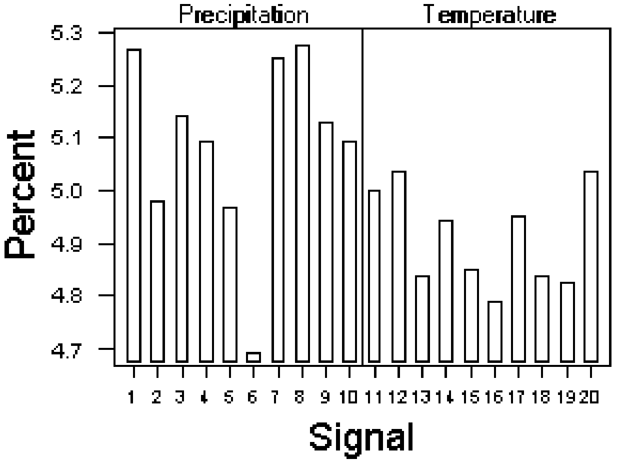


Figure 9. % of number of terms used in each signal for all lower range RMSE experiments

In the case of 90% missing values, the behavior of the mean RMSE with the values of k differs substantially. For $k > 5$ the errors drop, with a minimum at $k = 7$. Considering that this case corresponds to the largest percent of missing values, the situation is contradictory. However, the error estimates are based on a very small number of samples (due to the extreme data dilution), hence their variances should be much higher, probably making the observed error differences non-significant (Fig-7). This situation should be further investigated; nevertheless, the very fact that the algorithm is capable of providing useful results under such severe conditions indicates its high robustness and tolerance to ill-conditioned data.

In the case of the maximum lag depth (*W*) (Fig-8(b)), there is a decreasing trend for the errors as *W* increases for the whole range of missing values. Clearly, as data dilution increases, the algorithm is required to search deeper into the past in order to find meaningful dependencies from the point of view of effectively predicting the future values. It is interesting to note the local minima on the RMS error surface are in almost perfect lineation, showing that for each percentage of missing values there is an optimum *W* (i.e. a defined search space), s.t. dependencies far beyond this lag, are probably meaningless or even noisy.

From the point of view of the composition of the 1001 discovered models in the lower error range, a rough picture is given by the relative contribution of each signal to the model, measured by the number of terms (time lags) from the corresponding signal occurring in the model (Fig-9). It is interesting to see that the distribution is clearly multimodal, with some signals tending to contribute more frequently than others. Moreover, although the class of precipitation signals (the first 10 in Fig-5) contributes the most (as could be expected, since the target is a precipitation signal), there is an important contribution by the temperature signals, showing the physical relationship between precipitation and temperature. Sensitivity analysis are considered for future experiments.

5 Conclusions

The robustness of the algorithm has been shown by its discovery of models that have skewness towards lower error values, despite the large number of signals, their physical heterogeneity and the (abnormally) large quantity of missing values. The algorithm was also capable of using the 20 signals in a non-uniform way when predicting the target signal. This shows the ability of this computational intelligence-based technique for handling high dimensional, time-varying data in an effective manner, without data preprocessing, and without imputation or assumptions regarding the missing information. The method is capable of intelligently assessing which are the most relevant signals w.r.t the target, as well as discovering the most interesting models and learning their composition in terms of the number and distribution of the significant lags. The experiments also revealed the most effective set of parameter values, for data mining applications for signal processing. These results are not generalizable until further experiments in other domains are conducted on larger, more complex data sets.

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] Box, G., Jenkins, G. : Time Series Analysis, Forecasting and Control. Holden-Day. (1976)

[3] Chen S.,Wu, Y., Luk, L. : Combined Genetic Algorithm Optimization and Regularized Orthogonal Least Squares Learning for Radial Basis Function Networks. IEEE-NN Trans. on Neural Networks, Vol 10, No. 5, pp-1239-1243, (1999)

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

[5] Maniezzo V. : Genetic evolution of the topology and weight distribution of neural networks. IEEE Trans. on Neural Networks, Vol 5, pp-39-53, (1999)

[6] Montana D. Davis, L.: Training feedforward neural networks using genetic algorithms. Proceedings of the 11th International Joint Conference on Artificial Intelligence. Morgan Kaufman, California, pp. 762-767, 1989.

[7] Pole,A., West M., Harrison J. : Applied Bayesian Forecasting and Time Series Analysis. Chapman & Hall, (1994).

[8] 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), pp.237-246..

[9] Valdés,J.J. : Similarity-based Neuro-Fuzzy Networks and Genetic Algorithms in Time Series Models Discovery. NRC/ERB-1093, 9 pp. NRC 44919.(2002).

[10] Valdés,J.J., Mateescu, G. : Time Series Model Mining with Similarity-Based Neuro-Fuzzy Networks and Genetic Algorithms: A Parallel Implementation. Third. Int. Conf. on Rough Sets and Current Trends in Computing RSCTC 2002. Malvern, PA, USA, Oct 14-17. Alpigini, Peters, Skowron, Zhong (Eds.) Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence Series) LNCS 2475, pp. 279-288. Springer-Verlag , 2002.

[11] Valdés, J.J. : Similarity-based heterogeneous neurons in the context of general observational models. *Neural Network World*, **12** (5), (2002), 499–508.