A NEURAL NETWORK BASED TIME SERIES FORECASTING SYSTEM

Paulo Cortez, Miguel Rocha, José Machado and José Neves
Departamento de Informática
Universidade do Minho
Largo do Paço - 4719 Braga Codex - Portugal
{jmac,jneves}@di-ia.uminho.pt

ABSTRACT

The *Neural Network* (*NN*) arena has suffered in the past years a remarkable development as one of the novel fields for *Artificial Intelligence* (*AI*). *Time Series Analysis* (*TSA*) based on models of the variability of observations by postulating trends and cyclic effects, with a view to understand the cause of variation and to improve forecasting, suggests the use of *NNs* that do something like, what is called in statistics, *Principal Component Analysis* (*PCA*). The purpose of this work is to present a logical based *NN* system, along with: (i) *Time Series Forecasting* (*TSF*), with its characteristics of strong noise component and non-linearity in data, showing itself as a field in which the use of *NN's* stuff is particularly advisable; (ii) *PCA* rules, organized in a default hierarchy as logical theories, competing with one another for the right to represent a particular situation or to predict its successors; i.e., assisting in the process of choosing the best network to forecast each series. Some trials will be conducted, and the basic performance measures used as baselines for comparison with other methods.

Keywords: Time Series, Neural Networks, Logic Programming, Prolog.

1. Introduction

One of the basic tenets of science is to make predictions, to unveil relations among world objects where formerly were not apparent. Advances in the theory of Dynamical Systems (*DS*) [13][16][14] have shown that the relation among the *NN* and the Logic Programming (*LP*) paradigms is an important issue both from the theoretical or the practical points of view. Cross-fertilization is the rule, as concepts and methods of one will eventually become available to the other - and this is a source for insight and new perspectives. Also the deep connections between *NN* algorithms and statistics have been obvious from the beginnings of the field. Statistics forms the basis of signal processing, communication theory and *NNs*. Time Series (*TS*), considered as realizations of continuous random processes, where randomness is a result of complex interactions involving many independent and ultimately irreducible degrees of freedom, are alternative approaches to models of the world based on theoretical considerations and using measured data as initial inputs [5]. In the field one of the main concerns related to an effective internal representation of input data, the motivation for what is *Feature Analysis*, aiming at dimensionality reduction. Another is the desire to take a complex problem and describe it understandably, in order to :

- capture the basics of the discriminations that have to be made, and
- capture general "features" intermediate between the input data and the final prediction and make distinctions based on these features, and
- decide what the desirability of feature analysis is and what the intermediate-level features should be.

An obvious *NN* architecture shows out to be that of a multi-layer network, having early layers extracting features that are then used as inputs to higher levels [20]. In this setting, our claim is that the features developed in the hidden layers in a multilayer *NN* may be formally represented by a stratified knowledge base [16][14][10], in terms of the interactions of the different stratums of the knowledge base [12].

The use of *PCA* rules arises of the best way to represent a set of data with the least number of descriptors [23], given as logical theories, and will be used to choose the best network to forecast [6][7][15].

The trials conducted provide some basic performance measures that indicate that the system proposed is better in making short-term predictions than some widely used, like the so-called ARMA or ARIMA methods [2], or those based on non-linear methods, as the ones described in [4][21].

2. Neural Network Topology

The *NNs* used in this work are feedforward, with one hidden layer. The hidden layer has half of the neurons of the input layer (to avoid overparametrically the model) and the output layer is composed of one single neuron. The activation function is sigmoidal in the interval $[0, 1]$, or $[-1, 1]$ if the series being predicted presents negative values. The training is performed using the backpropagation algorithm [9][20][22] with a convergence step of 0.1. To avoid problems with overfitting an automatic stop of the training is set [18]. The training examples are built from the time series by taking $K + 1$ consecutive values (where K stands for the number of inputs to the *NN*). The first K values are the inputs and the last one is the desired output. Examples are created starting with the first value of the series and the process stops when the last value is reached. The *TS*' values are presented to the network in this order and not in a random way. This is assumed to be the best solution due to the temporal nature of the case. The forecasting is made by taking the last K values of the series as inputs to the *NN*, with the *NN*'s output as the forecasted value.

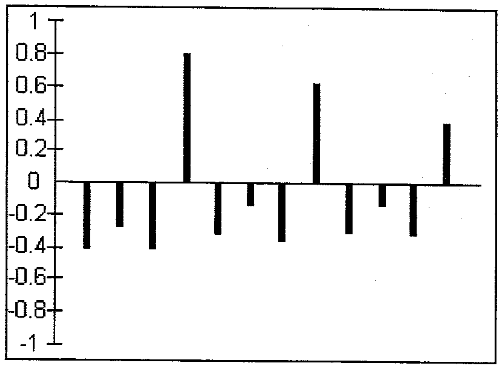

Fig. 1: Autocorrelation coefficients for typical *seasonal* series

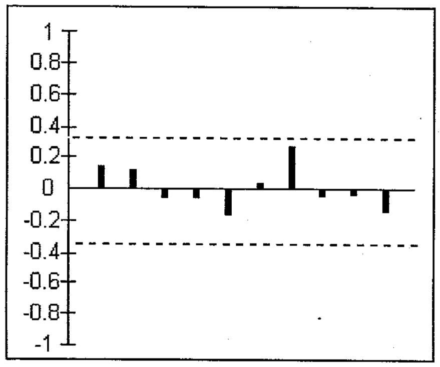

Fig. 2: Autocorrelation coefficients for typical *random* series

3. Building the Forecasting Model

The model used to forecast series relies on a trial-and-error procedure in order to find the optimal number of inputs to the *NN*. Statistical (or *PCA*) rules are used to reduce the number of *NNs* trained during the process. A particular case is the *autocorrelation* coefficient. It gives a measure of the statistical correlation between a series and itself, lagged of K periods. It may be computed as:

$$r_k = \frac{\sum_{t=1}^{n-k}(Y_t-\overline{Y})(Y_{t+k}-\overline{Y})}{\sum_{t=1}^{n}(Y_t-\overline{Y})}$$

Autocorrelations are used in the decomposition of series in its main components, the most significative ones being the *trend* and the *seasonality*. The *trend* is the characteristic of the series that stands for a constant grow (or decline) in the data. It normally arises due to factors like *inflation*, *technological improvements*, etc. The *seasonal* factor is found in series with a periodic behaviour, and is very common in annual series (for example in sales of products dependent on the time of the year).

In order to detect the presence of a *trend* component another *NN* was used. The inputs to this network were the autocorrelation coefficients (in the case for 1-10 lags) and the output was +1 (for series with a trend) and -1 (for stationary series). The training examples came from the autocorrelations coefficients of the series under study.

When the series are non-stationary the detection of its seasonalness is only possible after removing the *trend* from the series. The process usually used to achieve the stationarity in the series is called *differencing*. It consists in replacing the original series by its first differences:

$$X_t' = X_{t+1} - X_t$$

The autocorrelation coefficients are also used in the task of verifying the randomness of the series. Theoretically all autocorrelations of random series should be zero. In practice one may consider the series to be random if, for all lags K, the autocorrelations are in the interval (figure 2):

$$-1.96 * \frac{1}{\sqrt{n}} \leq r_k \leq 1.96 * \frac{1}{\sqrt{n}}$$

This interval is based on a normal distribution with a 95% degree of confidence. The results got shown that seasonal series present their best results with *NNs* with N inputs (where N is the seasonal factor). The non-stationary (with *trend*) series required the use of *NNs* with few inputs. When the two patterns were mixed the best results arose in *NNs* with a number of inputs greater than N. The various hypothesis were compared using the test set referred to in Section 2. It is assumed that the *NN* that best forecasts this set is the optimum and it is used to forecast.

The series' model can be reconstructed at any time. This is particularly useful when the time series have been updated with some new data or some old data has been removed. The user has the option to keep or to rebuild the data model.

4. The System's Architecture

The actual system's architecture reflects, in many aspects, the basic classical model of a blackboard base one. A distributed computing environment that not only supports multi-agents (multi-*NNs* and Knowledge Bases) activities [17], but also conciliates a set of mechanisms that provides any potential user with a flexible and effective manner to do *TSF*. The whole system was developed under Unix, using SICStus Prolog language under X-Windows, and is implemented in a TCP/IP based network of workstations [1][3].

The training of the *NN*, using the backpropagation algorithm was developed in C due to efficiency considerations, allowing for any kind of *NN* feedforward topology. The interplay between SICStus Prolog and C was done by compiling the C module in a executable file and using the **unix(system(...))** SICStus Prolog command. The convergence of the *NN* in the test set is verified every N iterations (N being an user-dependent parameter). A counter is used to keep the number of times a situation of divergence is detected. When this value reaches a limit (user defined) the training stops. The counter is reseted when an optimum value for the test set is achieved. At the end of the training the optimum *NN* for the test set is kept. As an example, consider the series from [11], represented in table 1. The X-Windows interface of the system can be seen in figure 3. The series are clearly non-stationary and non-seasonal. Using the first 35 values of the series as an input to the forecasting system the forecasting model was built. The best *NN* obtained had 5 inputs. Next, the periods 36-40 were forecasted and the results are represented in table 2 (where *SMSE* stands for the *Squared Mean Squared Error*).

5. Conclusions and Future Work

The trials conducted with the forecasting system are summarized in table 3.

- series 3 and 5 are examples of non-stationary and seasonal ones. Their degree of linearity is high. The results obtained from Holt-

Table 1: Example series (source[11])

Period	Value	Period	Value
1	10.2	21	29.2
2	5.4	22	30.9
3	4.0	23	30.2
4	2.5	24	26.3
5	5.1	25	21.9
6	12.7	26	19.8
7	22.6	27	21.5
8	29.5	28	26.2
9	32.2	29	31.3
10	33.8	30	31.5
11	32.3	31	36.8
12	28.4	32	38.8
13	22.6	33	40.3
14	20.2	34	45.3
15	21.8	35	48.7
16	25.5	36	48.9
17	29.2	37	46.9
18	30.9	38	43.6
19	30.1	39	44.0
20	28.8	40	46.5

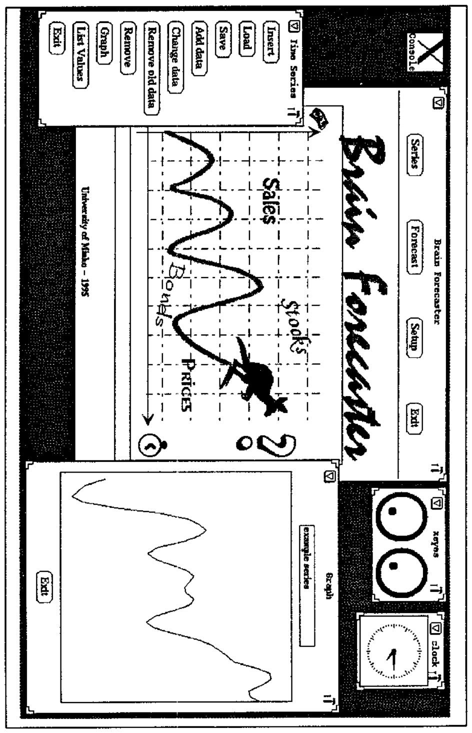

Fig. 3: The series' window for table 1

Table 2: Forecasted values for series from table 1

Period	36	37	38	39	40
Forecast	48.4	46.3	44.4	41.7	45.1

$SMSE = 1.3$

Table 3: Results obtained with the *NN* approach compared with the conventional ones

Series	Inputs	System SMSE	Arima SMSE	H.W. SMSE
1	8	14.08	12.3/16.0	
2	14	850.1		1161.1
3	14	33.74		16.46
4	4	7.74	8.2	10.18
5	14	61.97	50.3	34.5

Winters' (*HW*)[11][8] method are considerably better than those from *NNs*, and

- series 2 is seasonal and non-stationary, but unlike the former its variance is not constant. This increases the non-linearity in data and improves the performance of the *NN* system. Such series use to be forecasted by applying a transformation to the data (eg. logarithmical) which is usually difficult to perform and requires the skills of a trained forecaster. The use of *NNs* is, in this case, advantageous, and
- the performance of the forecast for series 1 is close to ARIMA[2]. This behaviour is observed in all autoregressive series, and
- the best results obtained by the *NN* system were the ones coming from moving average non-stationary series (series 4 in table 3).

The results obtained suggest that the utilization of *NNs* in the field of *TSF* can be very attractive, especially in series with a high degree of non-linearity, but:

- parallelizing the search process of the best *NN* to forecast each group of series, and
- trying some other training algorithms like the RPROP one [19], and
- avoiding the necessity of normalization by using mixed sigmoid-linear *NNs*,

are also tasks to be taken in consideration in future work.

References

[1] J. Almgren, S. Anderson, M. Carlsson, L. Floyd, S. Haridi, C. Frisk, H. Nilsson, and J. Sundberg. *SICStus Prolog Library Manual.* Swedish Institute of Computer Science, Sweden, 1993.

[2] G. Box and G. Jenkins. *Time Series Analysis.* Holden Day, 1970.

[3] I. Brakto. *Prolog, Programming for Artificial Intelligence.* Addison Wesley Publishing Company, New York, USA, 1990.

[4] M. Cosdagli. Non-Linear Prediction of Chaotic Time Series. In *Physica*, volume 35, pages 335–356, 1989.

[5] J. Elsner. Predicting Time Series Using a Neural Network as a Method of Distinguishing Chaos from Noise. In *Proceedings of the 1991 International Conference on Artificial Neural Networks (ICANN-91)*, pages 145–150, Espoo, Finland, 1991.

[6] S. Gallant. *Neural Network Learning and Expert Systems.* MIT Press, Cambridge, USA, 1993.

[7] P. Garrido and J. Neves. Mapping a Prolog Subset to Neural Networks. In *Proceedings of the Euro-Nimes'91*, pages 315–325, Nimes, France, 1991.

[8] J. Hanke and A. Reitsch. *A Business Forecasting.* Allyn and Bancon Publishing Company Inc., Massachussetts, USA, 1989.

[9] A. Lapedes and R. Forber. Non-Linear Signal Processing Using Neural Networks: Prediction and System Modelling. Technical Report LA-UR-87-2662, Los Alamos National Laboratory, 1987.

[10] T. Maibaum. Temporal Reasoning over Deontic Specifications. In J.J. Ch. and R.J. Wieringa, editors, *Deontic Logic in Computer Science*, pages 141–202. John Wiley & Son, ltd, 1993.

[11] S. Makridakis, S. Weelwright, and V. McGee. *Forecasting: Methods and Applications.* John Wiley & Sons, New York, USA, 1978.

[12] J. Neves. A Logic Interpreter to Handle Time and Negation in Logic Data Bases. In *Proceedings of the ACM'84 Anual Conference - The fifth Generation Challenge*, pages 50–54, San Francisco, California, USA, 1984.

[13] J. Neves and P. Garrido. Amalgamating the Neural and Logic Computing Paradigms. In *Proceedings of the 1991 International Conference on Artificial Neural Networks (ICANN-91)*, pages 1469–1472, Espoo, Finland, 1991.

[14] J. Neves and J. Machado. A Many-Sorted First-Order Dynamic Logic. Research Report, Universidade do Minho, Braga, Portugal, 1995.

[15] J. Neves and J. Machado. A Temporal Logic Computational Framework for Presupposition Analysis. Research Report, Universidade do Minho, March 1995.

[16] J. Neves and J. Machado. An Object-Oriented Dynamic Logic Based Modeling and Simulation Environment. To appear in Proceedings of BICSC'95, the Third Beijing International Conference on Simulation Systems and Scientific Computing, Beijing, China, October 1995.

[17] J. Neves, M. Santos, and V. Alves. An Adaptable and Dynamic Architecture for Distributed Problem Solving Based on the Blackboard Paradigm. In *Proceedings of the 7th Australian Joint Conference on Artificial Intelli-*

gence, pages 378–385, Armidale, New South Wales, Australia, 1994.

[18] L. Prechelt. PROBEN1 - A Set of Neural Network Benchmark Problems and Benchmarking Rules. Research Report, Fakultät für Informatik, Universität Karlsruhe, Germany, 1994.

[19] M. Riedmiller and H. Braun. A Direct Adaptative Method for Faster Backpropagation Learning: The RPROP Algorithm. In *Proceeedings of the IEEE International Conference on Neural Networks*, San Francisco, CA, USA, 1993.

[20] D. Rumelhart, G. Hinton, and R. Williams. Learning Internal Representations by Error Propagation. In D. Rulmelhart and J. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, volume 1, pages 318–362, MIT Press, Cambridge MA, 1986.

[21] H. Tong. *Non-Linear Time Series.* Oxford University Press, 1990.

[22] A. Weigend, B. Hubermann, and D. Rumelhart. Predicting the Future: A Connectionist Approach. Technical Report SSL-90-20, Xerox Palo Alto Research Center, 1990.

[23] T. Young and T. Calver. *Classification, Estimation and Pattern Recognition.* Elsevier Publishing Company, New York, USA, 1974.