ESANN'1997 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 16-17-18 April 1997, D-Facto public., ISBN 2-9600049-7-3, pp. 127-132

δ-NARMA Neural Networks: A Connectionist Extension of ARARMA Models*

Denis Bonnet[1,2], Véronique Perrault[2] and Alain Grumbach[1]

[1] ENST, Département Informatique, Paris, France
[2] SNCF, Direction de la Recherche, Département Prospective, Paris, France

Abstract. Despite their theoretical limitations, ARIMA models are widely used in real-life forecasting tasks. Parzen has proposed an extension of ARIMA models: ARARMA models. ARARMA models consist of an AR model followed by an ARMA model. Following Parzen approach, δ-NARMA neural network are MLP, the units of which are simple non-linear ARMA-based models (ϵ-NARMA units). They are a non-linear extension of ARARMA models. To apply Back-Propagation Through Time algorithm to such a network, we introduce the concept of virtual error. Virtual errors can be seen as the error on hidden layer units. Such networks face the problem of non-stationary time series prediction. Experience shows that δ-NARMA networks outperform classical statistical and connectionist models on three different real-life prediction tasks. It also brings a better understanding of δ-NARMA behavior.

1. Introduction

Statistical models on which Box and Jenkins methodology [1] is based — such as AR, MA, ARMA and ARIMA models – are widely used in industrial applications since they are simple and efficient. Parzen has proposed an extension of ARIMA models called ARARMA models [2]. Given a time series $x(t)$, an ARARMA model consist of an AR model followed by an ARMA model.

Most neural networks only deal with recurrent connections of unit output and thus do not fit the ARMA process framework. Moreover, since they only use one delay on their recurrent connection, Lin, Horne, Tiño and Giles have shown that they are experimentally unable to model long term time dependencies, despite theoretical background [3]. To fill this gap, NARX networks introduce multiple delay-lines and have been shown to better deal with this

*This work is supported by the SNCF (French rail) under grant 0300-4-41235. We would like to acknowledge Dr. Daniel Gauyacq for supporting this work. We would also like to thank Pr. Bernard Burtschy and Dr. Frédéric Chateau for helpful discussions about ARARMA models, the limitations of statistical models and statistical validation.

issue. From the forecasting point of view, the most useful NARX network is the NARMA model [4] which is based on recurrent connections of unit error.

A new neural network architecture for time series prediction should be able to predict non-stationary time series. Therefore, it should be inspired by the modular behavior of ARARMA models and by NARX networks to deal with long-term dependencies.

2. The ϵ-NARMA Model

The ϵ-NARMA model is not only the simplest NARMA model but also it is an ARMA model with a non-linear activation function. ϵ-NARMA model is the basic cell of our connectionist modular approach. An ϵ-NARMA(p,q) predictor $\hat{x}$ is expressed

$$\hat{x}(t) = f\left[\beta + \sum_{i=1}^{p} \phi_i x(t-i) + \sum_{j=1}^{q} \theta_j \delta(t-i)\right]$$

where f is a non-linear function. Obviously, the ϵ-NARMA predictor is unable to compute $\hat{x}(0)$,..., $\hat{x}(p)$. Moreover, to compute $\hat{x}(p)$,..., $\hat{x}(p+q)$, it needs initialization values for errors $\delta(p-q)$,..., $\delta(p)$. To adjust the model to a particular time series, several learning algorithms may be used, such as the least square method or the stochastic gradient descent. The latter leads to the following formulae:

$$[a(t)]_{n-1} = (\beta)_{n-1} + \sum_{i=1}^{p} (\phi_i)_{n-1}\, x(t-i) + \sum_{j=1}^{q} (\theta_j)_{n-1}\, [\delta(t-i)]_{n-1}$$

$$t \in T_1 \begin{cases} (\Delta\phi_i)_n = \nu\ [\delta(t)]_{n-1} f'\left\{[a(t)]_{n-1}\right\}\ x(t-i) \\ (\Delta\theta_j)_n = \nu\ [\delta(t)]_{n-1} f'\left\{[a(t)]_{n-1}\right\}\ [\delta(t-i)]_{n-1} \\ (\Delta\beta)_n = \nu\ [\delta(t)]_{n-1} f'\left\{[a(t)]_{n-1}\right\} \end{cases}$$

where ν is the learning rate, the subscript n denotes the n^{th} iteration of the learning algorithm and $a(t)$ is called activation. Moreover $n \cong t\ [T]$ if $t \in [0;T[$. In the remainder of this article and to avoid confusion when dealing with learning, n is written $mT+t$ where m is the epoch number and T is the overall size of the time series. T_1 is the subset of signal observations on which learning is performed (learning set) and T_2 is the set on which performances are measured (test set).

As seen before, the initialization of this model is a crucial point. Using identical values for $\delta(p-q), \ldots, \delta(p)$ at each epoch implies that the learning algorithm would not be able to tell these initialized error values from truly computed error values. This behavior is not suitable since the mechanism of the error process is different for initialized and computed errors. To avoid this phenomenon, new randomly initialized error values are chosen at each epoch.

3. Principles of δ-NARMA Networks

Frasconi, Gori and Soda Local Feedback Network[1] (LFN) is based on a particular type of unit, for which the output is computed using the value of the inputs and the previous value of the output [5]. From the time series prediction point of view, the propagation formula for a particular LFN having only one input layer and a single output unit is $\hat{x}(t) = f[\beta + \sum_{i=1}^{p} \phi_i x(t-i) + \theta\, \hat{x}(t-1)]$ where f is a sigmoid function. We call this network Local Feedback Unit (LFU). Using not only the last value of the unit output as input but the q last values leads to an obvious extension of LFU: Generalized LFU (GLFU). The propagation formula of a LFU becomes then for a GLFU $\hat{x}(t) = f[\beta + \sum_{i=1}^{p} \phi_i x(t-i) + \sum_{j=1}^{q} \theta_j\, \hat{x}(t-j)]$. As LFU are LFN units, GLFU may be the units of a LFN generalization called Generalized LFN (GLFN). Comparing this equation to ϵ-NARMA propagation formula leads to the seminal idea of δ-NARMA networks. As sigmoid units are combined to obtain MLP and as GLFU are combined to obtain GLFN, ϵ-NARMA models may be combined to obtain a larger network which takes advantage of the ϵ-NARMA forecasting ability. We call this new class of connectionist networks δ-NARMA networks. A δ-NARMA network is therefore a MLP in which sigmoid units have been replaced by ϵ-NARMA networks.

4. Virtual Error

To implement δ-NARMA networks, output error should be computed for every network unit, including hidden units. For an N layer MLP one-step predictor $\hat{x}$ applied to time series x, assuming layer 1 is the input layer and layer N is the output layer, propagation formulae are

$$\left[y_j^1(t)\right]_n = x(t-j)$$

$$1 < k \leq N \begin{cases} \left[y_j^k(t)\right]_n = f\left\{\left[a_j^k(t)\right]_n\right\} \\ \left[a_j^k(t)\right]_n = \left(\beta_j^k\right)_n + \sum_{i=1}^{\Omega^{k-1}} \left(\phi_{ij}^k\right)_n \left[y_i^{k-1}(t)\right]_n \end{cases}$$

Subscript n denotes values related to the n^{th} iteration of the learning process. $[y_j^k(t)]_n$ is the output[2] of the j^{th} unit of the k^{th} layer, $[a_j^k(t)]_n$ its associated activation and $[\delta_j^k(t)]_n$ its associated error. Ω^k is the number of units[3] of the k^{th} layer. $(\phi_{ij}^k)_n$ is the weight of the connection from the i^{th} unit of layer $k-1$ to the j^{th} layer of unit k (static parameter). $(\beta_j^k)_n$ is the bias of the j^{th} unit of layer k. f is the activation function of ϵ-NARMA units.

[1]LFN is obviously a particular case of Elman networks where only local connections are taken into account.

[2]$[y_1^N(t)]_n$ is obviously the network output $[\hat{x}(t)]_n$.

[3]For a one step univariate predictor, the number of units on the last layer Ω^N is 1.

ESANN'1997 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 16-17-18 April 1997, D-Facto public., ISBN 2-9600049-7-3, pp. 127-132

The computed output of an N layer MLP when forecasting $x(t)$ after n learning iterations is denoted $\hat{x}^n(t)$. MLP back-propagation formulae are

$$t \in T_1 \left\{ \begin{array}{ll} \left(\Delta\phi_{ij}^k\right)_{mT+t} &= \nu \ \left[\delta_j^k(t)\right]_{mT+t-1} f'\left\{\left[a_i^{k-1}(t)\right]_{mT+t-1}\right\} \ \left[y_i^{k-1}(t)\right]_{mT+t-1} \\ \left(\Delta\beta_j^k\right)_{mT+t} &= \nu \ \left[\delta_j^k(t)\right]_{mT+t-1} f'\left\{\left[a_i^{k-1}(t)\right]_{mT+t-1}\right\} \end{array} \right.$$

$$\left[\delta_j^k(t)\right]_{mT+t-1} = \left\{ \begin{array}{ll} x(t) - \left[y_j^k(t)\right]_{mT+t-1} & k = N \\ \sum_{p=1}^{\Omega^{k+1}} \left(\phi_{jp}^{k+1}\right)_{mT+t-1} \left[\delta_p^{k+1}(t)\right]_{mT+t-1} & 1 < k < N \end{array} \right.$$

Subscript $mT+t$ denotes values related to the $(mT+t)^{th}$ iteration of the learning process where m is the epoch and T the number of observations. $x(t)$ is the target value of the neural network output $[\hat{x}(t)]_{mT+t} = [y_1^N(t)]_{mT+t}$. T_1 is the learning set.

The $[\delta_j^N(t)]_{mT+t}$ term is obviously an error term. Moreover, for all $k < N$, the $[\delta_j^k(t)]_{mT+t}$ and $[\delta_j^N(t)]_{mT+t}$ terms are homogeneous. Therefore, the $[\delta_j^k(t)]_{mT+t}$ term should also be seen as an error [6]: the virtual error. The output error of any MLP unit may thus be computed using back-propagation formulae. Consequently, sigmoid units of a MLP may be replaced by ϵ-NARMA models. As δ is the usual notation for this term which is – from our point of view – a virtual error, we call these neural networks δ-NARMA.

5. δ-NARMA Learning Algorithm

Assuming layers are numbered from 1 (inputs) to N (output), the propagation formula of a δ-NARMA network are deduced from MLP propagation formula:

$$\left[y_j^1(t)\right]_n = x(t-j)$$

$$1 < k \leq N \left\{ \begin{array}{l} \left[y_j^k(t)\right]_n = f\left\{\left[a_j^k(t)\right]_n\right\} \\ \left[a_j^k(t)\right]_n = \left(\beta_j^k\right)_n + \sum_{i=1}^{\Omega^{k-1}} \left(\phi_{ij}^k\right)_n \left[y_i^{k-1}(t)\right]_n + \sum_{i=1}^{\Upsilon^k} \left(\theta_{ij}^k\right)_n \left[\delta_j^k(t-i)\right]_n \end{array} \right.$$

Υ_j^k is the number of error delays for unit j of layer k, $(\theta_{ij}^k)_n$ is the weight of the recurrent error connection of delay i related to the j^{th} unit of layer k (dynamical parameter).

Using the back-propagation Through Time principle [7] and δ-NARMA propagation equation, the δ-NARMA back-propagation formulae come out as

Type of Predictor	Daily Traffic	Monthly Traffic	Monthly Expenditure
MLP	1.09	1.06	1.04
Elman	1.02	1.03	1.12
GLFN	1.01	0.99	1.04
ARIMA	1.00	1.00	1.00
NARMA	0.95	0.95	1.01
ARARMA	0.89	0.93	0.95
δ-NARMA	0.75	0.88	0.92

Table 1: ARV on validation set scaled to 1 for the ARIMA model

follow:

$$t \in T_1 \begin{cases} \left(\Delta\phi_{ij}^k\right)_{mT+t} = \nu \ \left[\delta_j^k(t)\right]_{mT+t-1} f'\left\{\left[a_i^{k-1}(t)\right]_{mT+t-1}\right\} \ \left[y_i^{k-1}(t)\right]_{mT+t-1} \\ \left(\Delta\theta_{ij}^k\right)_{mT+t} = \nu \ \left[\delta_j^k(t)\right]_{mT+t-1} f'\left\{\left[a_i^{k-1}(t)\right]_{mT+t-1}\right\} \ \left[\delta_j^k(t-i)\right]_{mT+t-1} \\ \left(\Delta\beta_j^k\right)_{mT+t} = \nu \ \left[\delta_j^k(t)\right]_{mT+t-1} f'\left\{\left[a_i^{k-1}(t)\right]_{mT+t-1}\right\} \end{cases}$$

$$\left[\delta_j^k(t)\right]_{mT+t-1} = \begin{cases} x(t) - \left[y_j^k(t)\right]_{mT+t-1} & k = N \\ \sum_{p=1}^{\Omega^{k+1}} \left(\phi_{jp}^{k+1}\right)_{mT+t-1} \left[\delta_p^{k+1}(t)\right]_{mT+t-1} & 1 < k < N \end{cases}$$

Therefore, we use δ-NARMA back-propagation formulae to compute the virtual errors $\delta_j^k(t-i)$ and to compute parameter adjustments.

6. Experimental Results

Our experiments have been carried out on three real-life time series. The first one describes the daily number of passengers on a particular railroad line. The second is the total monthly number of railroad passengers in France. The last one describes the monthly evolution of the expenditure of a given budget. Benchmarking practice experimentally suggests that evaluating models on three time series with significantly various characteristics gives a good approximation of their overall performance. The chosen comparison criterion is the Average Relative Variance (ARV) which is roughly model and problem independent [8]. Results are shown in table 1.

7. Discussion

As in this case Mean Absolute Percent Error (MAPE) and ARV are consistent and as error average value is almost null, the hypothesis of null mean Gaussian noise is roughly fulfilled. Thus confidence intervals for the error are easily deduced from its standard deviation. Results confirm that using error-based recurrent connections increases performance, particularly when comparing GLFN networks and δ-NARMA networks. Moreover, using only local multiply-delayed recurrent connections seems to be more efficient since it favors useful recurrent connections, as shown by the comparison of GLFN versus Elman networks. Surprisingly, the ARARMA model lack of computational power is strongly counterbalanced by its efficient temporal behavior. This shows that using local recurrent error connections is more important than using models with strong non-linear behavior. The overall performance of δ-NARMA networks is definitely better than all the other models in the case of these series.

An extension of δ-NARMA networks is δ-NARMA networks with eXogenous variables (δ-NARMAX). For these networks, inputs are not only the past values of the time series (endogenous variables) but also past – and maybe future – values of explicative variables (exogenous variables).

References

[1] G.E. Box and J.E. Jenkins, *Time series analysis : Forecasting and control*, Holden-Day, San-Francisco, CA, USA, 1970.

[2] E. Parzen, "ARARMA models for time series analysis and forecasting," *Journal of Forecasting*, vol. 1, pp. 67–82, 1982.

[3] T. Lin, B.G. Horne, P. Tiňo, and C.L. Giles, "Learning long-term dependencies is not difficult with NARX recurrent neural networks," technical report UMIACS-TR-95-78, Institute for Advanced Studies, University of Maryland, College Park, MD, USA, 1995.

[4] J. Connor, L.E. Atlas, and D.R. Martin, "Recurrent network and NARMA modelling," in *Advances in Neural Information Processing Systems*, S.J. Hanson, R.P. Lippmann, J.E. Moody, and D.S. Touretzky, Eds., vol. 4, pp. 301–308. Morgan Kaufmann, San Mateo, CA, USA, 1992.

[5] P. Frasconi, M. Gori, and G. Soda, "Local feedback multilayered networks," *Neural Computation*, vol. 4, no. 2, pp. 120–130, 1992.

[6] T. Catfolis, "Mapping a complex temporal problem into a combination of static and dynamic neural networks," *SIGART*, vol. 5, no. 3, pp. 23–28, 1994.

[7] P. Werbos, "Backpropagation through time : What it does and how it do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[8] A.S. Weigend, "Time series analysis and prediction," technical report CU-CS-744-94, Computer Science Department and Institute of Cognitive Science, University of Colorado at Boulder, Boulder, CO, USA, 1994.