Turkish Journal of Forecasting vol. 01 no. 1 (2017) pp. 07-15

Content list available at JournalPark

Turkish Journal of Forecasting

Journal Homepage: tjforecast.xyz

Time Series Prediction with Direct and Recurrent Neural Networks

L. M. L. De Campos*

[1]*Universidade Federal do Pará - Faculdade de Computação*

A R T I C L E I N F O

Article history:
Received 18 April 2017
Accepted 20 July 2017
Published online 23 August 2017

Keywords:
Recurrent neural networks
Time series prediction
NARX network

A B S T R A C T

This article presents a comparative study of the prediction of time series for the Consumer Price Index (CPI) using a recurrent neural network (RNN). For this, three models are designed for recurrent networks, with changes in backpropagation applied to allow them to incorporate the ARX (auto-regressive with external input) and NARX (nonlinear auto-regressive with external input) models. We also present a third architecture, re-fed with the hidden layer, called ARXI, which is a special case of the Elman Network. Training is carried out for all networks and tests the ability to generalize them (identification stage), in order to select the best architectures of recurrent networks for prediction of the CPI. Following this stage, the models are validated by testing the extrapolation capacity of the networks, i.e., the presented data were not used during the training phase and obtain responses that indicate the capacity to predict the CPI at various times in the future (validation phase). We conclude that the NARX network shows the best performance and that the hybrid system proposed in [1] constitutes an excellent tool when minimal networks are required that make a series of perdition satisfactorily.

1. Introduction

A growing demand for forecasts and the trend analysis of variables has recently been observed in many professional areas. Many companies and government agencies are seeking to develop computational tools that provide automatic predictions for the values of certain variables that are being tracked; this helps in defining strategic policies within these organizations, and in the process of decision making and planning in the short to medium term. These prediction methods vary greatly and depend on data availability, the quality of models and the types of assumptions made, among other things. In general, these predictions are not straightforward, and this has attracted many researchers to this field.

The Artificial Neural Networks (ANN) have been used increasingly for the prediction of time series in various lines of business, economic and financial [2,3] chaotic time series prediction [4], energy consumption [5] among others. ANNs can learn from examples, recognize a hidden pattern in historical observations and use them to predict future values. Moreover, they are able to deal with incomplete information or noisy data and can be very effective, especially in situations where you cannot set the rules or steps that lead to the solution of a problem.

The task of designing architecture of an ANN to simulate problems of predicting time series isn't a simple task, since the inclusion of recurrences in the models increases the complexity of learning algorithms used in these networks. Thus, recurrent neural networks haven't yet been fully exploited due to its tedious and training for their complex structures [6] discusses several approaches that seek heuristics to limit the complexity of design, computer time, parameterization and selection of ANNs, which causes additional problems of decision and a trial and error approach for modeling the network.

* Corresponding author.
E-mail address: lidio@ufpa.br (Lídio Mauro Lima De Campos)

Some researchers are seeking methods for the automatic design of artificial neural networks (ANNs) using evolutionary strategies; however, there are few studies that seek to generate architectures based on recurrent artificial neural networks (RANNs). The authors of [7] have proposed a method that uses evolutionary algorithms for the automatic design of ANN architectures; their paper presents an evaluation of eight combinations of evolutionary strategies for 16 sets of public domain data, and the methodology gives excellent results for direct neural networks. Methods for the automatic construction of RANNs have been hampered by the fact that the training algorithms for such networks are more complex than those used for direct networks.

In [1], a biologically plausible methodology was proposed which is capable of generating recurrent artificial neural networks with an optimal number of neurons and adequate topology. With this objective in mind, three biological metaphors were used: genetic algorithms (GA), Lindenmayer systems (L-systems) and ANNs. The methodology attempts to imitate the natural process of the growth and evolution of the nervous system, using L-systems as a recipe for the development of neurons and their connections, and GA to develop and optimize the architecture of a nervous system suitable for a specific task. The technique was tested on three simple, known problems (XOR, Parity and Tomita's Languages [8]) where recurrent network topologies can be used. A more complex problem involving the learning of time series was also studied. The results show that the proposal is very promising and can generate direct and recurrent neural network architectures with an optimal number of neurons and connections, good generalization capacity, less error and high tolerance to noise.

We are currently seeking improvements to the methodology proposed in [1], through a deepening of the theoretical and methodological aspects: firstly, the study of mechanisms that can imitate the biological process of the development of neurons, considering aspects of the biological plausibility of the chromosomal coding [9], according to evolutionary strategies that can evolve and select the best connection architecture for the ANNs generated; and secondly the possibility of exploring RANNs with different architectures. In this respect, a detailed study of the various possibilities of recurrent connections is necessary, given the complexity of training algorithms for these architectures.

With these goals in mind, Section 2 presents three architectures used to predict the Consumer Price Index: the ARX and NARX networks, and a third architecture, where the intermediate layer has recurrences, referred to here as ARXI. Section 3 describes simulation results for the prediction of a time series that provides the Consumer Price Index, and finally, the conclusions are presented in Section 4.

2. Nonlinear Presentations

The prediction of time series has been performed with the use of traditional models Auto Regressive and/or moving average. These are parametric models through which the forecasting of future observations is obtained from the linear combination of past values and, when appropriate, with the noise components of the series of interest weighted by a set of parameters. Dynamical systems are nonlinear, this way these applications should be chosen models with nonlinearity, since they produce some dynamic schemes that linear systems cannot represent.

The ANNs are good examples of widespread linear representation which are currently increasingly used for prediction of time series, due to its capacity to add knowledge in its structure, with examples, a predictor based on neural network is able to estimate the future behaviour of a series only from its past samples. A prediction model of this type is called non-parametric, since there is no need to know the parameters the process that generates the signal. The process model is estimated using a learning algorithm where the samples are presented to the neural network and their weights are updated according with the prediction error. The following are some representations based on nonlinear ANNs, as well as changes in the algorithm "back propagation" for each case.

2.1. Presentation 1 - ARX network

This section presents architecture of recurrent neural network with the output feedback, Figure 1. This ANN is based on the ARX model [10], which is nothing more than an MLP network, whose inputs consists of the outputs recurrences with delays. This neural network is equivalent to the ARX model (autoregressive with exogenous inputs), given by equation (1). Where $x(n)$ is the system input and $y(n)$ the output, where the function $f(.)$ is a nonlinear function, usually unknown $x(n)$, $y(n)$ correspond to the input and output at time n, while $dy > 0$, is the order of the input buffer. When this function is approximated by a "perceptron" multilayer network, the resulting topology is called ARX recurrent network, which is a special case of the network shown in section 2.3.

$$y(n) = f[y(n-1) + a2y(n-dy) + x(n)] \quad (1)$$

For deductions of the models presented in this section and in subsequent, consider that "A" is the number of units of input layer, as determined by the length of input vectors for training, "C" is the number of units of output layer and "B" is the number of units in the hidden layer. The input and hidden layers, each have an extra unit used as a limit, so the units of these layers are sometimes indexed by the intervals (0, ..., A) and (0, ..., B). Denote the levels of activation of units in the input layer by x_j the hidden layer by h_j and the output layer by o_j. The weights connecting of the input layer to hidden layer are denoted by $w1_{ij}$, where 'i' index the input units and 'j' index the hidden units. Similarly, the weights connecting the hidden layer to output layer are denoted by $w2_{ij}$ with i and j indexing the hidden units to output units.

The modifications made in the "backpropagation" algorithm to the ARX model for the recurrent neural model were as follows: the outputs in intermediate layer are now given by equation (2). It was included the contributions of the recurrences given by the sum of $m = 0$ to C for the terms $o_m(t-1)$ in it.

$$h_k(t) = \frac{1}{1+e^{-\sum_{m=0}^{C} o_m(t-1) \times w3(t)_{mk} + \sum_{j=0}^{A} x_{nj}(t) \times w1(t)_{jk}}} \quad (2)$$

For $t > 0$, $h_0 = 1$, which is the bias value. The term $o_m(t-1)$ refers to each output that is feedback to the input layer, "C" is the number of neurons in the output layer, in $t = 0$ the value of $o_m(t-1) = 0$. The updating of the weights of the feedback is given by equation (3):

$$\Delta w3_{mk}(t+1) = \Delta w3_{mk}(t) - \eta \frac{\partial \varepsilon_n}{\partial w3(t)_{mk}} \quad (3)$$

After mathematical deductions came the conclusion that the updates of the weights of intermediate layer to an output layer are given by equations (4) and (5). The development of equation (3) is given by equations (6) to (21).

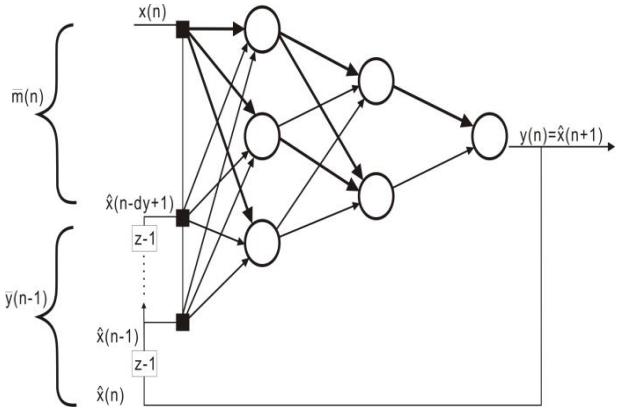

Figure 1. Network ARX model approximation the (1)

$$\Delta w2_{km}(t+1) = \Delta w2_{km}(t) + \eta \times [(y_{nm} - o_m) \times (1-o_m) \times o_m] \times h_k \quad (4)$$

$$\Delta w1_{jk}(t+1) = \Delta w1_{jk}(t) + \eta \times [(1-h_k) \times h_k \times \sum_{m=0}^{C} (y_{nm} - o_m)(1-o_m) \times o_m \times w2_{km}] \times x_{nj} \quad (5)$$

$$\frac{\partial \varepsilon_n}{\partial w3_{mk}(t)} = \frac{1}{2} \sum_{m=0}^{C} \frac{\partial \varepsilon_{nm}}{\partial o_m} \frac{\partial o_m}{\partial h_k} \frac{\partial h_k}{\partial w3_{mk}} \quad (6)$$

$$\frac{\partial \varepsilon_{nm}}{\partial o_m} = -2(y_{nm}(t) - o_m(t)) \quad (7)$$

$$\frac{\partial o_m}{\partial h_k} = (1-o_m).o_m.w2_{km} \quad (8)$$

$$\frac{\partial h_k}{\partial w3_{mk}} = \frac{\partial}{\partial w3_{mk}}\left[\frac{1}{1+e^{-\sum_{m=0}^{C} o_m(t-1).w3(t)_{mk} + \sum_{j=0}^{A} x_{nj}(t).w1(t)_{jk}}}\right] \tag{9}$$

$$u_k(t) = 1 + e^{-\sum_{m=0}^{C} o_m(t-1).w3(t)_{mk} + \sum_{j=0}^{A} X_{nj}(t).w1(t)_{jk}} \tag{10}$$

$$u_k(t) - 1 = e^{-\sum_{m=0}^{C} o_m(t-1)\times w3(t)_{mk} + \sum_{j=0}^{A} X_{nj}(t)\times w1(t)_{jk}} \tag{11}$$

$$h_k(t) = \frac{1}{u_k(t)} \tag{12}$$

$$\frac{\partial h_k(t)}{\partial u(t)} = -\frac{1}{u(t)^2} \tag{13}$$

$$\frac{\partial u_k(t)}{\partial w3(t)_{mk}} = \frac{\partial h_k(t)}{\partial u_k(t)}\frac{\partial u_k(t)}{\partial w3(t)_{mk}} \tag{14}$$

$$\frac{\partial u_k(t)}{\partial w3(t)_{mk}} = (u_k(t)-1)(-o_m(t-1)) \tag{15}$$

$$\frac{\partial h_k(t)}{\partial w3_{mk}(t)} = -\frac{1}{u_k(t)^2}(u_k(t)-1).(om(t-1)) \tag{16}$$

$$\frac{\partial h_k(t)}{\partial w3_{pk}(t)} = \frac{(\frac{1}{h_k(t)}-1)}{(\frac{1}{h_k(t)})^2}\times o_m(t-1) \tag{17}$$

$$\frac{\partial h_k(t)}{\partial w3_{mk}(t)} = (1-h_k(t))h_k(t) \tag{18}$$

$$\frac{\partial \varepsilon_n}{\partial w3_{mk}(t)} = (1-h_k(t))h_k(t) * P \tag{19}$$

P is given by equation (20)

$$P = \sum_{m=0}^{C}(o_m(t)-y_{nm}(t))(1-o_m(t))\times o_m(t)\times o_m(t-1).w2_{km}(t) \tag{20}$$

The equation (3) is given by

$$\Delta w3_{mk}(t+1) = \Delta w3_{mk}(t) + \eta(1-h_k(t))h_k(t)\sum_{m=0}^{C}(y_{nm}(t)-o_m(t))(1-o_m(t)).o_m(t).om(t-1)w2_{km}(t) \tag{21}$$

2.2. Representation 2 - Network with recurrent hidden layer

What is intended in the present section is to present the formulation of an algorithm based on generalized delta rule for a network with recurrent hidden layer, the network architecture is shown in Figure 2. This same is a special case of the Elman network, in the sense that exogenous input, shows no delays $x(n)$, but presents the context unit replenished with delays $vq(n-1)$. The outputs in the hidden layer are now given by equation (22).

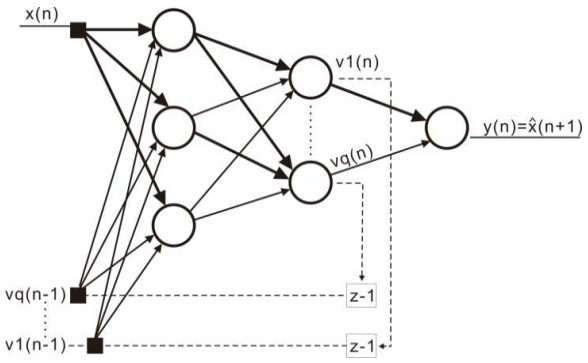

Figure 2. Network with recurrent hidden layer

$$h_k(t) = \frac{1}{1 + e^{-\sum_{p=1}^{B} hp(t-1).w3(t)_{pk} + \sum_{j=0}^{A} X(t).w1(t)_{jk}}} \tag{22}$$

for $t > 0$, $h(0) = 1$, which is the bias value. The terms of $h_p(t-1)$ are equal to the feedback $h_k(t-1)$, for $t = 0$ has a value of $h_p(t-1) = 0$. The updating of the weights is given by:

$$\Delta w3_{pk}(t+1) = \Delta w3_{pk}(t) - \eta \frac{\partial \varepsilon_n}{\partial w3(t)_{pk}} \tag{23}$$

Updating the weights of the hidden layer and output layer are given by equations (4) and (5). The update of the weights of the feedback is calculated by equation (24). The development of equation (23) is shown by equations (24) to (38).

$$\frac{\partial \varepsilon_n}{\partial w3_{pk}(t)} = \frac{1}{2} \sum_{m=0}^{C} \frac{\partial \varepsilon_{nm}}{\partial o_m} \frac{\partial o_m}{\partial h_k} \frac{\partial h_k}{\partial w3_{pk}} \tag{24}$$

$$\frac{\partial \varepsilon_{nm}}{\partial o_m} = -2(y_{nm}(t) - o_m(t)) \tag{25}$$

$$\frac{\partial o_m}{\partial h_k} = (1 - o_m).o_m.w2_{km} \tag{26}$$

$$\frac{\partial h_k}{\partial w3_{pk}} = \frac{\partial}{\partial w3_{pk}} \left[\frac{1}{1 + e^{-\sum_{p=1}^{B} hp(t-1).w3(t)_{pk} + \sum_{j=0}^{A} X(t).w1(t)_{jk}}} \right] \tag{27}$$

$$u_k(t) = 1 + e^{-\sum_{p=1}^{B} hp(t-1).w3(t)_{pk} + \sum_{j=0}^{A} X(t).w1(t)_{jk}} \tag{28}$$

$$u_k(t) - 1 = e^{-\sum_{p=1}^{B} hp(t-1).w3(t)_{pk} + \sum_{j=0}^{A} X(t).w1(t)_{jk}} \tag{29}$$

$$h_k(t) = \frac{1}{u_k(t)} \tag{30}$$

$$\frac{\partial h_k(t)}{\partial u(t)} = -\frac{1}{u(t)^2} \tag{31}$$

$$\frac{\partial u_k(t)}{\partial w3(t)_{pk}} = \frac{\partial h_k(t)}{\partial u_k(t)} \frac{\partial u_k(t)}{\partial w3(t)_{pk}} \tag{32}$$

$$\frac{\partial u_k(t)}{\partial w3(t)_{pk}} = (u_k(t)-1)(-hp(t-1)) \tag{33}$$

$$\frac{\partial h_k(t)}{\partial w3_{pk}(t)} = -\frac{1}{u_k(t)^2}(u_k(t)-1).(hp(t-1)) \tag{34}$$

$$\frac{\partial h_k(t)}{\partial w3_{pk}(t)} = \frac{(\frac{1}{h_k(t)}-1)}{(\frac{1}{h_k(t)})^2}.hp(t-1) \tag{35}$$

$$\frac{\partial h_k(t)}{\partial w3_{pk}(t)} = (1-h_k(t))h_k(t).hp(t-1) \tag{36}$$

$$\frac{\partial \varepsilon_n}{\partial w3_{pk}(t)} = (1-h_k(t))h_k(t).hp(t-1)\sum_{m=0}^{C}(o_m(t)-y_{nm}(t))(1-o_m(t)).o_m(t).w2_{km}(t) \tag{37}$$

$$\Delta w3_{pk}(t+1) = \Delta w3_{pk}(t) + \eta(1-h_k(t))h_k(t).hp(t-1)\sum_{m=0}^{C}(y_{nm}(t)-o_m(t))(1-o_m(t)).o_m(t).w2_{km}(t) \tag{38}$$

2.3. Representation 3 - NARX network

The model studied in this section is the NARX model that is nothing more than an MLP whose input is the output fed back to itself with time delays and an exogenous input, also with delays. In this architecture, the estimated outputs of the network are introduced again to entries, this representation is equivalent to the statistical NARX model (Nonlinear Autoregressive model with exogenous input), given by:

$$y(n) = f[y(n-1), \dots, y(n-dy), u(n-1), \dots, u(n-du+1)] \tag{39}$$

Where the function $f(.)$ is a nonlinear function, usually unknown $u(n)$ and $y(n)$ correspond to the input and output at time *n*, while $du > 0$ and $du \leq dy$, are orders of memory input and output. When this function is approximated by a multilayer perceptron network, the resulting topology is called NARX recurrent network. Figure 3 illustrates a NARX recurrent network. The same illustrates a NARX network with one hidden layer and a global feedback loop. In this research we considered the use of the NARX network in parallel mode of identification.

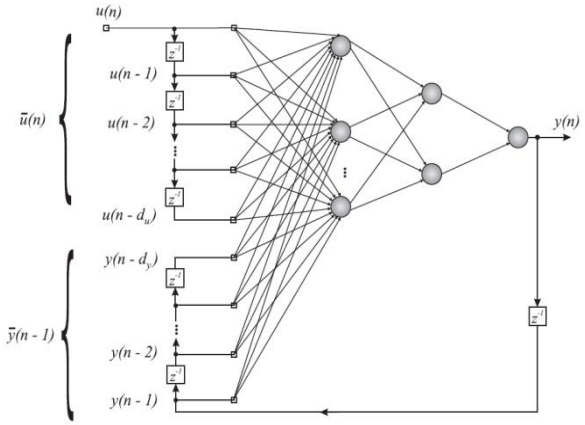

Figure 3. NARX network (Source [3]) with input and output delays

3. Experiments

The experiments performed to test the models presented in Section 2 were based on a time series representing the CPI between January 1998 and December 2009 (source: Central Bank of Brazil; data are shown in Table 1). The CPI quantifies the cost of products at different times; in other words, it is a measure of the price level of goods and services purchased by households over time and is useful for calculating inflation.

Table 1. CPI period Jan-1998 to December 2009 normalized values

	Jan	**Feb**	**Mar**	**Apr**	**May**	**Jun**	**Jul**	**Aug**	**Sep**	**Oct**	**Nov**	**Dec**
1998	0.0126	0.0014	0.0033	0.0023	0.0014	0.0041	-0.0025	-0.0052	-0.0017	0.0020	-0.0019	0.0009
1999	0.0064	0.0141	0.0095	0.0052	0.0008	0.0065	0.0120	0.0048	0.0019	0.0092	0.0112	0.0060
2000	0.0101	0.0005	0.0051	0.0025	0.0040	-0.0001	0.0191	0.0086	0.0004	0.0002	0.0040	0.0062
2001	0.0064	0.0040	0.0056	0.0086	0.0041	0.0052	0.0136	0.0054	0.0012	0.0071	0.0085	0.0070
2002	0.0079	0.0014	0.0042	0.0071	0.0028	0.0055	0.0103	0.0076	0.0066	0.0114	0.0314	0.0194
2003	0.0232	0.0137	0.0106	0.0112	0.0069	-0.0016	0.0034	0.0013	0.0076	0.0021	0.0033	0.0043
2004	0.0108	0.0028	0.0046	0.0031	0.0071	0.0078	0.0059	0.0079	0.0001	0.0010	0.0037	0.0063
2005	0.0085	0.0043	0.0070	0.0088	0.0079	-0.0005	0.0013	-0.0044	0.0009	0.0042	0.0057	0.0046
2006	0.0065	0.0001	0.0022	0.0034	-0.0019	-0.0040	0.0006	0.0016	0.0019	0.0014	0.0024	0.0063
2007	0.0069	0.0034	0.0048	0.0031	0.0025	0.0042	0.0028	0.0042	0.0023	0.0013	0.0027	0.0070
2008	0.0097	0.0056	0.0045	0.0072	0.0087	0.0077	0.0053	0.0014	-0.0009	0.0047	0.0056	0.0052
2009	0.0083	0.0021	0.0061	0.0047	0.0039	0.0012	0.0034	0.0020	0.0018	0.0001	0.0026	0.0024

Initially, a study was carried out of the generalization capacity of the architectures proposed in Section 2 and of the MLP network, selected for this part of the data in Table 1, for generalization (identification) and partly to test the extrapolation capacity of the networks (validation). Thus, values of the CPI from the period Jan 1998 to Dec 2002 were chosen for training the ANNs and analysing the ability to generalize from them, which is the identification stage of the system. The parameters used in the training algorithms were as follows: one neuron in the input layer, four in the intermediate layer and one in the output layer, a learning rate of 1.98 and 240,000 epochs. Figure 4 shows these results. It is notable that the best performance in generalization was obtained by the NARX network and secondly by ARX (a special case of NARX); the MLP generalizes with lower quality. The NARX network showed good performance, fast convergence and better generalization than the other models, although the ARX network showed comparable performance. This occurs because the input vectors of NARX models are built through a delay line with derivation slid over the input signal with a delay line, with derivation formed by the feedback of the output signal of the network [11].

For validation of the models we tested the NARX network for various parameters, initially it was simulated a network with the following number of neurons one (1) in the input layer, six (6) in the hidden layer and one (1) in the output, learning rate of 1.98 and 240000 epochs. The figure 5 illustrates the results considering three steps ahead, i.e. considered in the validation data for the years 2003, 2004 and 2005 that were not used in the identification stage (generalization). It was considered that these results were satisfactory for an error of 0.0001. However, we moved some parameters of the training were achieved several experiments in order to obtain a better validation of the model NARX. Was isolated one of the best solutions were obtained considering the following parameters: number of neurons one (1) in the input layer, eight (8) in the hidden layer and one (1) in the output, learning rate of 30.98 and 240000 epochs. The figure 6 illustrates the results, noticed that as we increase the number of neurons in the hidden layer from 6 to 8 that network performance improves considerably, i.e. the NARX model presented in section 2.3, works as an excellent predictor (red line).

The figure 7 shows the validation tests for the network ARX, with the same parameters of the NARX network presented earlier, i.e., eight (8) neurons in the hidden layer, we noticed a good performance of the network, but the network NARX still is superior. Aiming to find a minimal network capable of predicting the CPI, we used the hybrid system proposed by [1,12], thus a larger universe of networks were tested by the evolutionary algorithm, driven by the fitness function, search now smaller networks capable of predicting well the CPI, the figure 8 shows the results of a network with one (1) neuron in the input layer, two (2) in the hidden layer and one (1) in the output, learning rate of 1.98 and 240000 epochs.

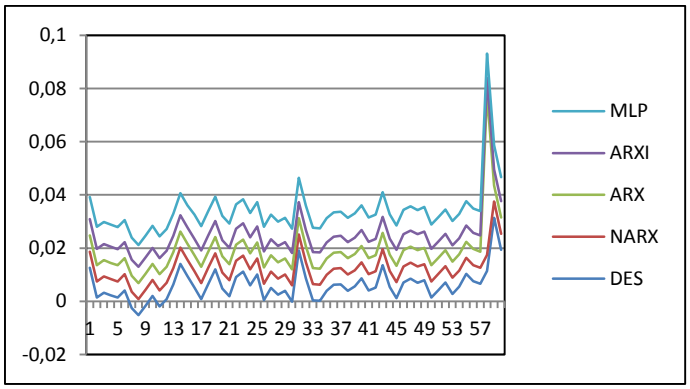

Figure 4. Stage of identification - generalization capability network

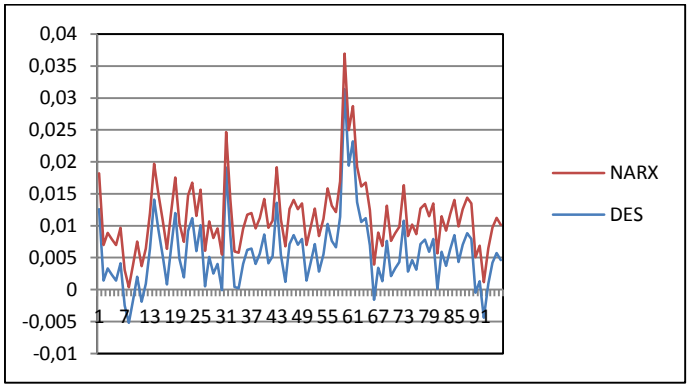

Figure 5. NARX network validation N=3 steps

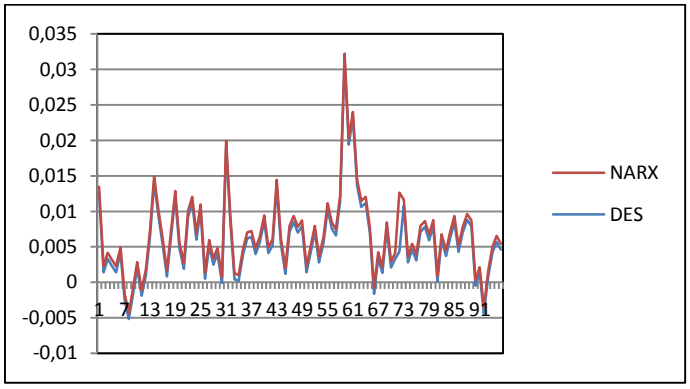

Figure 6. NARX network validation N=3 steps

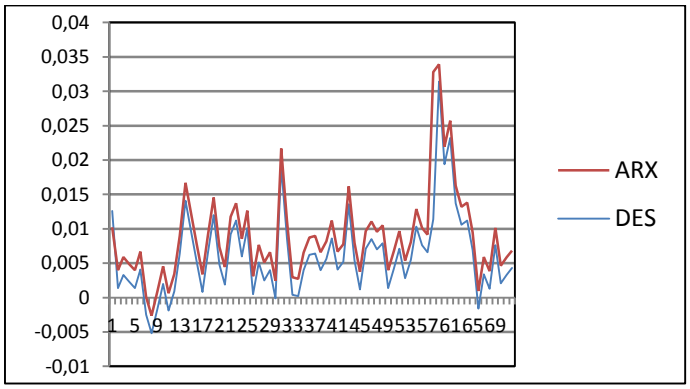

Figure 7. ARX network validation N=1 step

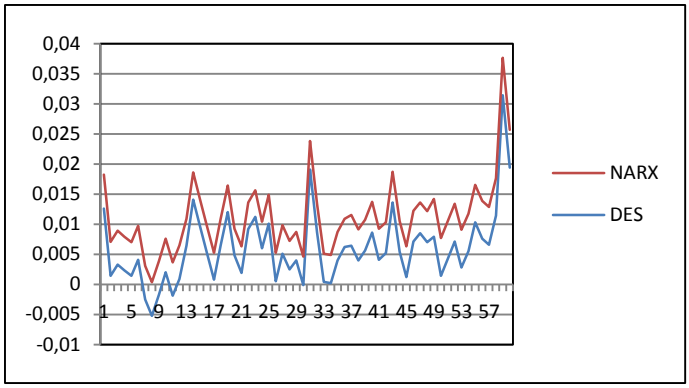

Figure 8. NARX network model generalization / hybrid system

4. Conclusions

The experiments indicate that the NARX network, besides showing good performance in the problem of predicting the CPI, offered faster convergence and a better generalization capability than the other networks. This occurs because the input vectors of NARX models are built through a delay line with derivation slid over the input signal with a delay line with derivation formed by the feedback of the output signal of the network [11]. The performance of the ARX network is similar to that of NARX for the problem studied; however, the network described in Section 2.2 did not show a good generalization capability in the simulations, and was discarded from studies of future predictions. The MLP also did not generalize well for the problem under study. The hybrid system in [1] is an excellent option for testing a population of networks that can generalize the problem well, without the intervention of the designer. Furthermore, it is possible to direct the search, as in the case of the simulation shown in Figure 8, where we sought a minimum network which was able to generalize well, as this is sometimes necessary. The fitness function used in the hybrid system proposed in [1] directs the search by considering the following aspects: the neurons in the hidden layer, the production rules, recurrent connections and residual error.

References

[1] L.M.L. De Campos, M. Roisenberg, J.M. Barreto, A biologically inspired methodology for neural networks design, in: IEEE Conf. Cybern. Intell. Syst. 2004., IEEE, Singapore, 2004: pp. 620–625. doi:10.1109/ICCIS.2004.1460487.

[2] M. Lam, Neural network techniques for financial performance prediction: Integrating fundamental and technical analysis, Decis. Support Syst. 37 (2004) 567–581. doi:10.1016/S0167-9236(03)00088-5.

[3] J.T. Yao, Towards a Better Forecasting Model for Economic Indices, in: Proc. 6th Jt. Conf. Inf. Sci., North Carolina, 2002: pp. 299–303.

[4] J.M.P. Menezes Jr., G.A. Barreto, A New Look at Nonlinear Time Series Prediction with NARX Recurrent Neural Network, in: 2006 Ninth Brazilian Symp. Neural Networks, IEEE, Sao Paulo, 2006: pp. 28–28. doi:10.1109/SBRN.2006.7.

[5] D.C. Park, M.A. El-Sharkawi, R.J. Marks, L.E. Atlas, M.J. Damborg, Electric Load Forecasting Using An Artificial Aeural Network, IEEE Trans. Power Syst. 6 (1991) 442–449. doi:10.1109/59.76685.

[6] S.F. Crone, Stepwise Selection of Artificial Neural Network Models for Time Series Prediction, J. Intell. Syst. 14 (2005) 99–122. doi:10.1515/JISYS.2005.14.2-3.99.

[7] E. Cantu-Paz, C. Kamath, An Empirical Comparison of Combinations of Evolutionary Algorithms and Neural Networks for Classification Problems, IEEE Trans. Syst. Man Cybern. Part B. 35 (2005) 915–927. doi:10.1109/TSMCB.2005.847740.

[8] M. Tomita, Dynamic Construction of Finite Automata From Examples Using Hill-Climbing, in: Proc. Fourth Annu. Conf. Cogn. Sci. Soc., Michigan, 1982: pp. 105–108.

[9] N. Feng, G. Ning, X. Zheng, A framework for simulating axon guidance, Neurocomputing. 68 (2005) 70–84. doi:10.1016/j.neucom.2005.01.007.

[10] L.A. Aguirre, Introdução a Identificação de Sistemas, Minas Gerais, 2007.

[11] T. Lin, B.G. Horne, P. Tino, C.L. Giles, Learning Long-Term Dependencies in NARX Recurrent Neural Networks, IEEE Trans. Neural Networks. 7 (1996) 1329–1338. doi:10.1109/72.548162.

[12] L.M.L. De Campos, M. Roisenberg, R.C.L. De Oliveira, Automatic design of Neural Networks with L-Systems and genetic algorithms - A biologically inspired methodology, in: 2011 Int. Jt. Conf. Neural Networks, IEEE, 2011: pp. 1199–1206. doi:10.1109/IJCNN.2011.6033360.