The use of NARX Neural Networks to predict Chaotic Time Series

EUGEN DIACONESCU, PhD
Electronics, Communications and Computer Science Faculty
University of Pitesti
Targu din Vale, Nr. 1
ROMANIA
eugend@upit.ro

Abstract:
The prediction of chaotic time series with neural networks is a traditional practical problem of dynamic systems. This paper is not intended for proposing a new model or a new methodology, but to study carefully and thoroughly several aspects of a model on which there are no enough communicated experimental data, as well as to derive conclusions that would be of interest. The recurrent neural networks (RNN) models are not only important for the forecasting of time series but also generally for the control of the dynamical system. A RNN with a sufficiently large number of neurons is a nonlinear autoregressive and moving average (NARMA) model, with "moving average" referring to the inputs. The prediction can be assimilated to identification of dynamic process. An architectural approach of RNN with embedded memory, "Nonlinear Autoregressive model process with eXogenous input" (NARX), showing promising qualities for dynamic system applications, is analyzed in this paper. The performances of the NARX model are verified for several types of chaotic or fractal time series applied as input for neural network, in relation with the number of neurons, the training algorithms and the dimensions of his embedded memory. In addition, this work has attempted to identify a way to use the classic statistical methodologies (R/S Rescaled Range analysis and Hurst exponent) to obtain new methods of improving the process efficiency of the prediction chaotic time series with NARX.

Key-Words: - Chaotic Time Series, Hurst Exponent, Prediction, Recurrent Neural Networks, NARX Model

1 Introduction

Many processes in domains as physics, technique, biology, and economics are described by time series. In formal terms, a time series is a sequence of vectors, depending on time t:

$$y(t),\ t = 0,\ 1,\ 2,\ \ldots \qquad (1)$$

The applications of type prediction or forecasting of time series is largely presented in the literature about time series [10][11][12][13]. The prediction of futures values of vector y is helpful or strict required to decide upon a strategy of control or to optimize the activity, production, selection, etc. Formally, the problem of prediction can be formulated as finding a function Γ so as to obtain an estimate $\hat{y}(t+D)$ of the vector y at time $t+D$ ($D = 1,\ 2,\ \ldots$), given the values of y up to time t, plus a number of additional time-independent variables (exogenous features) u_i:

$$\hat{y}\ (t+D) = \Gamma(y(t),\ \ldots\ y(t-d_y),\ u(t),\ \ldots,\ u(t-d_u)) \qquad (2)$$

where $u(t)$ and $y(t)$ represent the input and output of the model at time t, d_u and d_y are the *lags* of the input and output of the system and Γ a nonlinear function. Typically, $D = 1$, meaning *one-step ahead*, but can take any value larger than 1 (*multi-step ahead*) [2].

Viewed this way, prediction becomes a problem of function approximation, where the purpose of the method is to approximate the continuous function Γ as closely as possible. Therefore, in the case of function approximation or regression problems, many methods from that domain can be applied here.

Usually, the evaluation of prediction performance is done by computing an error measure E over a number of time series elements, such as a validation or test set:

$$E = \sum_{k=0}^{N} (\hat{y}(t-k),\ y(t-k)) \qquad (3)$$

E is a function measuring the error between the estimated (predicted) and actual sequence element. Typically, a distance measure (Euclidean or other) is used, but depending on the problem, any function can be used (e.g. a function computing the cost resulting from an incorrect prediction of $y(t+D)$)[2].

The problem of chaotic time series prediction is studied in various disciplines now including engineering, medical and econometric applications. Chaotic time series are the output of a deterministic system with positive Liapunov exponent. Therefore, unless the initial condition are specified with infinite precision, the time series behavior becomes

unpredictable, and the prediction of chaotic time series is a difficult task.

From a historical point of view, before the 1980s, prediction of time series used linear parametric autoregressive (AR), moving-average (MA) or autoregressive moving-average (ARMA) models introduced by Box and Jenkins [11][13]. These models are linear and are not able to cope with certain non stationary signals, and signals whose mathematical model is not linear. An obvious drawback is that these algorithms are linear, and are not able to cope with certain nonstationary signals and signals whose model is chaotic nonlinear. On the other hand, neural networks (NN) are powerful when applied to problems whose solutions require knowledge which is difficult to specify, but for which there is an abundance of examples.

The prediction of chaotic processes implies finding the interdependences between time series components. The dependences are minimal in random time series and maximal in a complete deterministic process. But, random and deterministic are only margin of the large set of chaotic time series signals with weak dependences between components on short or long term. A special case is represented by the fractal time series characterized by auto similarity or non-periodic cycles.

1.1 Prediction with neural networks

After 1980, there has been resurgence in the field of time series prediction, when it becomes clear that this type of prediction is a suitable application for a neuronal network predictor.

The NN approach to time series prediction is non-parametric, in the sense that it is not necessary to know any information regarding the process that generates the signal. It is shown that the recurrent NN (RNN) with a sufficiently large number of neurons is a realization of the nonlinear ARMA (NARMA) process. [1][12][5].

Neural Networks (NN) based prediction has been explored from their beginning and development, because of NNs approximation and generalization property. Many research papers are published in scientific literature and some commercial companies claim or market the so-called advanced statistical programs, using neural networks, for modeling and prediction.

However, some difficulties and limitations remain nevertheless of current actuality and cause researches for new NN models and learning techniques to be conducted[4][5][7][8].

-Outliers make it difficult for NNs (and other prediction models) to model the true underlying functional. Although NN had been shown to be universal approximators, it has found that NN had difficulty modeling seasonal patterns in time series. When a time series contains significant seasonality, the data need to be deseasonalized.

-The number of samples in time series. Researchers have found that increasing observation frequency does not always help to improve the accuracy of prediction.

-Stationarity – the classical techniques for time series prediction, require a stationary time series, while most real time series are not stationary (stationarity refers to a stochastic process whose mean value, variances and covariances – first and second order moments do not change in time). After NNs have been introduced one can use original time series as forecasting targets.

-The problem of long time dependencies - is related to the problem of vanishing gradient or forgetting behavior.

Time series prediction is the same as system identification; this paper shows that the dynamics of nonlinear system that produce complex time series can be captured in a model system. The model system is an artificial RNN. The main idea of RNN is providing a weighted feedback connection between layers of neurons and adding time significance to entire artificial NN. Therefore, RNNs are the most suitable for Time Series Analysis

1.2 Chaotic time series

1.2.1 A short characterization of some example of chaotic time series

To evaluate the prediction capability of proposed algorithms, the best data are the chaotic time series [14], generated by some linear dynamical systems. The degree of irregularity is different, from on type of series to another, depending on the sort of iterated difference equation, chaotic map or flows. In this paper, four time series were been tested: logistic, Mackey-Glass, fractal Weirstrass and BET index.

The logistic time series (4) generate a chaotic map with extremely short memory length. It is a difficult test for prediction algorithm. They do not exhibit cycles as we see sometimes in system practice.

$$y'(t) = a \cdot y(t) \cdot (1 - y(t)) \qquad (4)$$

The Mackey-Gloss equation (5) is classified as a simpler system generating chaotic flows. This type of chaotic time series is a relatively easy mission for prediction algorithms. Non-periodical cycles appear due the delay included in the equation. Chaotic

flows are frequently encountered models in real life practice.

$$y' = \frac{a \cdot y(t-\tau)}{1 + y^c(t-\tau)} - b \cdot y(t) \quad (5)$$

The fractal function of Weirstrass (6) is an infinite sum of a series of sine or cosine waves in which amplitude decreases, while the frequency increases according to different factors. The Weirstrass function used has a limited sum of four terms.

$$y(t) = \sum_{n=0}^{n=3} ((\frac{1}{a^n}) \cos(b^n \cdot \omega \cdot t)) \quad (6)$$

On the financial market, multivariate time series are encountered, for example the market indices BET (figure 1), BET-C, BET-FI from the Romanian stock market. These indexes also have, although they are presented only by their value, other important components, such as the daily transaction volume or number of transactions.

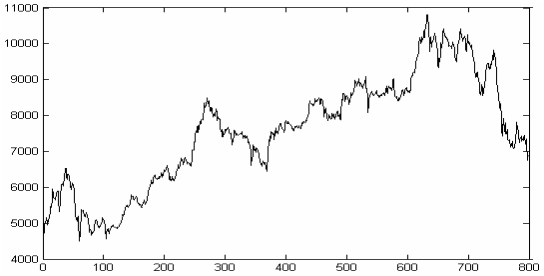

Fig. 1 BET index for 2005-03.2008 years

1.2.2 Preprocessing time series

Usually, the time series $y(t)$ and the input sequence $u_i(t)$ are analyzed in a process named "data selection" for the prediction optimization, or computing time minimization. The rule of input variable selection is that the input variables should be as predictive as possible [21]. Selection consists in partitioning the data series in prediction intervals about contents of measured information with autocorrelation coefficient r_k. For time series, the autocorrelation coefficients r_k are given by:

$$R_k = \frac{\sum_{t=1}^{N} (y(t) - \bar{y})(y(t+k) - \bar{y})}{\sum_{t=1}^{N} (y(t) - \bar{y}))^2} \quad (7)$$

where $y(t)$ is a data value at time step t, k is the lag, and the overall mean is given by:

$$\bar{y} = \sum_{t=1}^{N} \frac{y_t}{N} \quad (8)$$

This aspects do not are an objective for this work.

The time series can be used immediately for processing in only few cases. In most cases, it is necessary to preprocess the time series to ensure an optimal outcome of the processing. The most common operations are removing known systematicies as linear or nonlinear trends (constantly rising or descending of the average value), seasonality (periodic patterns due to a periodic influencing factor)[2][9][12]. In this work, the data were only normalized to be included in [-1, 1] range.

1.2.3 Stochasticity of time series. Long memory processes

Long memory process is a process with a random component, where a past event has a decreasing/decaying effect on future events. The process has some memory of past events, which is "forgotten" as time moves forward.

The mathematical definition of long memory process is given in terms of autocorrelation: when a data set exhibit autocorrelation, a value y_t at time t_i is correlated with a value y_{t+d} at time t_{i+d} , where d is some time increment in the future. In a long memory process autocorrelation decrease over time and the decreasing follows a power low [10].

In a long memory process the decrease of the autocorrelation function (*ACF*) for a time series is a power low:

$$ACF_{TS}(k) = Ck^{-\alpha}, \quad (9)$$

where C is a constant, and $ACF_{TS}(k)$ is the autocorrelation function with log k. The Hurst exponent is related to the exponent α in the equation by:

$$H = 1 - \alpha/2, \quad (10)$$

The value of the Hurst exponent ranges between 0 and 1. A value of 0.5 indicates a random evolution in time (a Brownian time series). In a random process there is no correlation between any element and a future element.

A Hurst exponent value $0.5 < H < 1$ indicates "persistent behavior" (e.g. a positive autocorrelation). Persistence means that if the curve has been increasing for a period, it is expected to continue for another period. If there is an increase from step t_{i-1} to t_i, there will probably be an increase

from t_i to t_{i+1}, or a decrease will be followed by a decrease.

A Hurst exponent of $0 < H < 0.5$ shows antipersistent behavior. After a period of decreases, a period of increases tends to show up. The antipersistent behavior has a rather high fractal dimension, corresponding to a very "noisy" profile-like curve (which highly fills up the plane). This behavior is sometimes called "mean reversion"[10].

In principle, fractal dimension D and Hurst coefficient H are independent of each other because the D is considered a local propriety, and long-memory dependence is a global characteristic [22]. For self-affine processes (e.g. fractals), the local properties are reflected in the global ones, and it is possible the relationship $D + H = n + 1$, where the n is dimension of self-affine space [22]. The long-memory dependence (persistence) is linked with the case $0.5 < H < 1$ and a feature of the surfaces with low fractal dimensions. The antipersistent processes are linked with the surfaces with higher fractal dimensions (rougher) with $0 < H < 0.5$.

2 NARX networks

In this paper, the architectural approach proposed to deal with chaotic time series is one based upon "Nonlinear Autoregressive models with eXogenous input (NARX model)", which are therefore called NARX recurrent neural networks [1][4][5]. This is a powerful class of models which has been demonstrated that they are well suited for modeling nonlinear systems and specially time series. One principal application of NARX dynamic neural networks is in control systems. Also, is a class computationally equivalent to Turing Machines [1].

Some important qualities about NARX networks with gradient-descending learning gradient algorithm have been reported: (1) learning is more effective in NARX networks than in other neural network (the gradient descent is better in NARX) and (2) these networks converge much faster and generalize better than other networks [4][5].

The simulated results show that NARX networks are often much better at discovering long time – dependences than conventional recurrent neural networks. An explanation why output delays can help long-term dependences can be found by considering how gradients are calculated using the back-propagation-through-time (BPTT) algorithm.

Recently, several empirical studies have shown that when using gradient-descent learning algorithms, it might be difficult to learn simple temporal behavior with long time dependencies [7][9], in other words those problems for which the output of a system at time instant k depends on network inputs presented at times $r << k$. The researchers have analyzed learning algorithms for systems with long time dependencies and showed that for gradient-based training algorithms, the information about the gradient contribution m steps in the past vanishes for large m. This effect is referred to as the problem of vanishing gradients, which partially explains why gradient descent algorithms are not very suitable to estimate systems and signals with long time dependencies. For instance, common recurrent neural networks encounter problems when learning information with long time dependencies, a problem in the prediction of nonlinear and no stationary signals. The vanishing gradients problem makes the learning of long-term dependencies in gradient-based training algorithms difficult if not virtually impossible in certain cases [1].

A state space representation of recurrent NARX neural networks can be expressed as[12]:

$$z_k(k+1) = \begin{Bmatrix} \Phi(u(k), z_i(k)), & i = 1, \\ z_i(k), & i = 2,3,...N, \end{Bmatrix} \qquad (11)$$

where the output $y(k) = z_i(k)$ and z_i, i=1,2, … N, are state variables of recurrent neural network. The recurrent network exhibits forgetting behavior, if:

$$\lim_{m\to\infty} \frac{\partial z_i(k)}{\partial z_j(k-m)} = 0 \quad \forall k, m \in K,\ i \in \mathrm{O},\ j \in \mathrm{I}, \qquad (12)$$

where z is state variable, "I" denotes the set of input neurons. "O" denotes the set of output neurons and K denotes the time index set.

Several approaches have been suggested to get around the problem of vanishing gradient in training RNNs. Most of them rest on including embedding memory in neural networks, whereas several others propose improved learning algorithms, such as the extended Kalman filter algorithm, Newton type algorithm, annealing algorithm, etc.

Embedded memory is particularly significant in recurrent NARX and NARMAX neural networks. This embedded memory can help to speed up propagation of gradient information, and hence help to reduce the effect of vanishing gradient. There are various methods of introducing memory and temporal information into neural networks. These include creating a spatial representation of temporal pattern, putting time delays into the neurons or their connections, employing recurrent connections, using neurons with activations that sum input over time, etc.

3 Architecture and learning

3.1 The NARX models

The NARX model for approximation of a function Γ can be implemented in many ways, but the simpler seems to be by using a feedforward neural network with the embedded memory (a first tapped delay line), as is shown in figure 2, plus a delayed connexion from the output of the second layer to input (a second tapped delay line). Making the network dependent on d_u previous sequence elements is identical to using d_u input units being fed with d_u adjacent sequence elements. This input is usually referred to as a *time window* since it provides a limited viewed on part of the series. It can also be viewed as a simple way of transforming the temporal dimension into another spatial dimension.

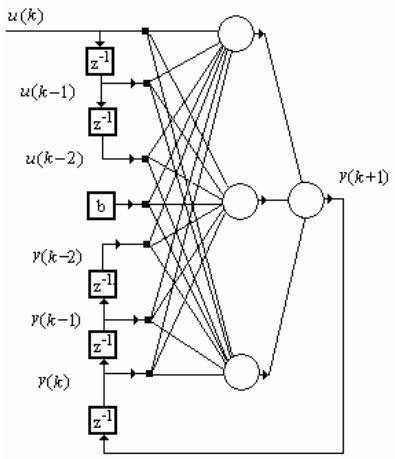

Fig. 2 NARX model with tapped delay line at input

In practice it was observed that forecasting of a time series will be enhanced by simultaneously analyzing related time series. For example, electrical power consumption for the next day will be better predicted if taken together, last p_c day consumptions and last p_t environment temperatures are simultaneously applied as inputs to the neural networks. The architectural model in figure 3 is make to test this hypothesis A generalized implementation of this model allows the input and output to be multidimensional, and thus applying to the "multivariate" type of time series.

For the architectural model in figure 2 the notation used is NN(d_u, d_y; N) to denote the NN with d_u input delays, d_y output delays and N neurons in layer 1. Similarly, for the architectural model in figure 2 the notation used is NN(d_{u1}, d_{u2}, d_y; N).

For the NN models used in this work, with two levels (level 1 surnamed *input* layer and level 2 or *output* layer), the general prediction equations for computing the next value of time series $y(k+1)$ (output) using model in figure 2, the past observation $u(k)$, $u(k\text{-}1)$, …, $u(k\text{-}d_u)$ and the past outputs $y(k)$, $y(k\text{-}1)$, …, $y(k\text{-}d_y)$ as inputs, may be written in the form:

$$y(k+1) = \Phi_o\left\{w_{b0} + \sum_{h=1}^{N} w_{ho} \cdot \Phi_h\left(w_{ho} + \sum_{i=0}^{d_u} w_{ih} u(k-i)\right) + \right. \quad (13)$$

$$\left. + \sum_{j=0}^{d_y} w_{jh} \cdot y(k-j)\right)\right\}$$

For the model in figure 3, the prediction equations for computing the output value $y(k+1)$ using the past observations $u_1(k)$, $u_1(k\text{-}1)$, …, $u_1(k\text{-}d_u)$ for the first time series, the past observations $u_2(k)$, $u_2(k\text{-}1)$, …, $u_2(k\text{-}d_u)$ for the second time series and the past outputs $y(k)$, $y(k\text{-}1)$, …, $y(k\text{-}d_y)$ as inputs, may be written in the form:

$$y(k+1) = \Phi_o\left\{w_{b0} + \sum_{h=1}^{N} w_{ho} \cdot \Phi_h\left(w_{ho} + \sum_{i1=0}^{d_{u1}} w_{i1h} u_1(k-i1) + \right.\right. \quad (14)$$

$$\left.\left. + \sum_{i2=0}^{d_{u2}} w_{i2h} u_2(k-i2) + \sum_{j=0}^{d_y} w_{jh} \cdot y(k-j)\right)\right\}$$

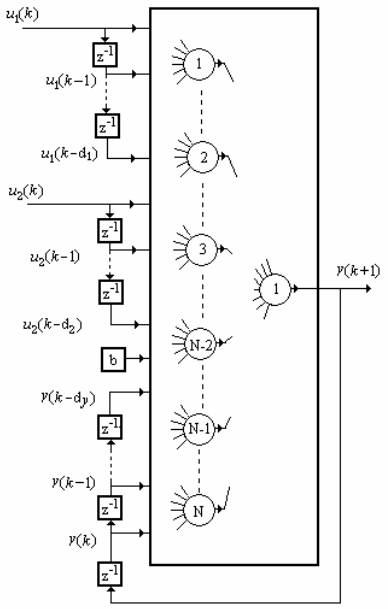

Fig. 3 NARX with two tapped delay lines for two time series applied at NARX input

3.2 Learning algorithms

For learning purposes, a dynamic back-propagation algorithm is required to compute the gradients, which is more computationally intensive than static back-propagation and takes more time. In addition, the error surfaces for dynamic networks can be more complex than those for static networks. Training is more likely to be trapped in local minima [6][12].

The selected training method in this work uses the advantage of availability at the training time of the true real output set. It is possible to use the true output instead of the estimated output to train the network witch has the feedback connections decoupled (cut). The decoupled network has a common feedforward architecture which can be trained with classical static back-propagation algorithm. In addition, during training, the inputs to the feedforward network are just the real/true ones – not estimated ones, and the training process will be more accurate.

The training process has some difficulties. One is related to the number of parameters, which refers to how many connections or weights are contained in network. Usually, this number is large and there is a real danger of "overtraining" the data and producing a false fit which does not lead to better forecasts. For NARX neural network model the number is given by $p = (d_u + d_y + 2)N$. A solution is penalizing the parameter increase [15]. This fact motivates the use of an algorithm including the regularization technique, which involves modifying the performance function for reducing parameters value. Practically, the typical performance function used in training, MSE, is replaced by a new one, MSEreg, as follows:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(t_i - y_i)^2 \qquad (15)$$

$$MSW = \frac{1}{n}\sum_{j=1}^{n} w_j^2 \qquad (16)$$

$$MSEreg = \xi MSE + (1-\xi)MSW \qquad (17)$$

where t_i is the target and ξ is the performance ratio. The new performance function causes the network to have smaller weights and biases, and in this way forces the network response to be smoother and less likely to overfit.

The network training function that updates the weight and bias values according to Levenberg-Marquardt optimization was modified to include the regularization technique. It minimizes a combination of squared errors and weights and, then determines the correct combination so as to produce a network which generalizes well. The process is called Bayesian regularization.

In general, in function approximation problems, for networks that contain up to a few hundred weights, the Levenberg-Marquardt algorithm will have the fastest convergence. This advantage is especially noticeable if very accurate training is required. However, as the number of weights in the network increases, the advantage of this algorithm decreases.

Other training algorithms [15] were tested, but with a less good result:

- The network training function that updates weight and bias values according to gradient descent with momentum.
-The network training function that updates weight and bias values according to the gradient descent with adaptive learning rate.
-The network training function that updates weight and bias values according to the gradient descent momentum and an adaptive learning rate.

The neural network training can be made more efficient if certain preprocessing steps on the network inputs and targets are performed. The *normalization* of the input and target values mean to mapping them into the interval [-1, 1]. This simplifies the problem of the outliers for the network. The normalized inputs and targets that are returned will all fall in the interval [-1, 1].

4 Experimental results

In all the experiments performed, a one-step-ahead prediction is considered; that is, the actual observed values of all lagged samples are used as inputs. (If multistep-ahead predictions are required then, it is possible to proceed by adding the first one-step-ahead prediction to time series, and then the new time series is used to predict the second step-ahead, and so on).

Various NN models having different numbers of lagged input steps (or time windows), different number of lagged output connected steps as inputs, and different number of neurons in layer 1 have been compared. All the models had layer 1 with N neurons and a single neuron in layer 2 (output layer). The input has been rescaled in most cases to be included in [-1, 1] range.

An interesting comparison about length of input lags in feedforward neural network and the NARX model presented in this paper can be made with [3]. It is well known from practice of NN that the input variables of NN should not to be much correlated, because the correlated input variables may degrade

the forecasting by interacting with each other as well as other elements and producing a biased effect [23]. In [3] it is shown that a feedforward model as NN(1-13;4) lead to poor prediction. The notation (1-13) means 13 inputs, and (1, 13) means 2 inputs. In general, the success model in [3] have only 3 or 4 inputs, as in corresponding notation NN(1,12,13;1) or NN(1,2,12,13;2). By his architecture, the NARX model applies simultaneously all the values stored in the tapped delay line as inputs to the RNN. It is easy to observe from the table 1-5 that, in general, the optimum number of lags is in 12-30 range, the best value being found from one case to another case. A bigger number of lags at input do not improve the prediction, it is even worse. That is, a conflicting condition results in selecting the number of lags of the delay line at input: a bigger number should assure input sequence as predictive as possible, but many variables are too correlated to each other to produce unconfused outputs.

4.1 Criteria to estimate prediction

Usually, the performance of a trained network can be measured by the errors on the training, validation, and test sets. In this work, the correlation coefficient R between the outputs and targets of neural network is used. R is a measure of the variation between the generated outputs and targets. If this number is equal to 1, then there is perfect correlation between the targets and outputs [15]. In the following examples, the fact that the number is very close to 1 (R>0.98 or better R>99), indicates a good fit.

4.2 Example 1: Chaotic Mackey-Glass time series

Table 1 depicts the experimental results with chaotic Mackey-Glass time series as input, for a total of 1000 observations, test set being different from training set. Column labeled R (correlation coefficient between input and output of neural network) shows the results in different runs respectively. The rows 3 and 4 from table 1 show that for two different time-series but with identical H, the same NN model gives different predictions. The rows 6, 7, 8 and 9 indicate the influence of architecture on prediction quality: for the same time-series, only the NN(24,1;5) model gives the best prediction.

Table 1

a	b	c	τ	H	sse	ssw	N	d_u	d_y	R
0.2	0.1	10	30	0,25	0.22	13.6	3	12	1	0.99545
0.2	0.12	10	30	0.36	0.62	27.2	3	12	1	0.99966
0.2	0.1	5	75	0.37	0.62e-	142.8	3	12	1	0.94592
0.2	0.1	10	50	0.37	0,102	29.7	3	12	1	0.9931
0.2	0.1	10	30	0.41	0.15	14	3	12	1	0.99185
0.2	0.1	10	80	0.51	2.93	45.39	3	12	1	0.74984
0.2	0.1	10	80	0.51	2.41	75	5	12	1	0.06015
0.2	0.1	10	80	0.51	0.02	59	5	24	1	0.99968
0.2	0.1	10	80	0.51	21,3	10	2	5	1	0.29047
0.2	0.1	8	70	0.50	0.014	43.9	3	12	1	0.99981
0.2	0.1	5	70	0.72	2.4e-5	2413	3	12	1	0.9244

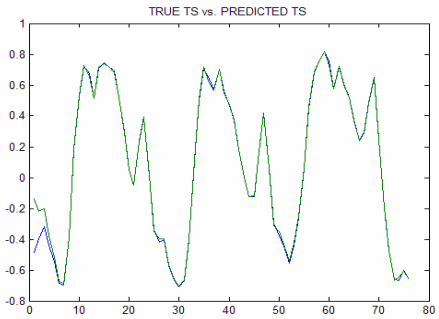

Fig. 4 The traces for original and predicted chaotic Mackey-Glass time series are very close; H=0.51, a=0.2, b=0.1, c=10, τ=80, NN(24, 5; 5) model. R=0.99573

4.3 Example 2: Fractal Weirstrass time series

The data contained in table 2 give a motivation about the conclusion that in that case of Weirstrass fractal time series the values of exponent Hurst do not represent an indication about the prediction success. The column labeled H shows a strong variation of about more 300% of the values of Hurst coefficient which do not can be put in relation with the rate of success represented by the value from the column labeled R.

Table 2

a	b	ω	H	sse	ssw	N	d_u	d_y	R
3.2	6.1	25	0.25	2e-11	5.2	18	15	2	0.99998
2.7	2.1	11.9	0.26	11e-8	4,9	10	30	2	0.99988
3.2	5.1	32	0.3	1e-11	8.32	10	30	1	0.99739
3.2	5.1	32	0.3	1.6e-9	5.61	10	30	2	0.99744
2.3	1.5	15	0.36	3.8e-8	5.28	10	30	2	0.99988
3.2	4.1	50	0.46	1.1e-8	4.07	10	30	2	0.99821
3.2	4.1	50	0.46	7e-10	10.66	10	30	2	0.99906
2.1	2.07	15	0.53	1.8e-9	21	10	35	1	0.99964
3.2	5.1	38	0.57	1.2e-7	24.77	5	15	2	0.9980
3.2	5.1	38	0.57	7e-10	10.66	10	30	2	0.99906
3.2	6.1	25	0.66	2.3e-7	6.58	2	10	1	0.99971
3.2	6.1	25	0.66	0.6	30	2	4	1	0.9655
1.2	1.4	10	0.82	2e-7	61.52	2	10	1	0.99929
2.7	2.1	11	0.84	1.2e-8	5,38	10	15	8	0.98409
2.7	2.1	11	0.84	1.8e-8	5,39	10	30	2	0.99996

4.4 Example 3: BET time series

The BET-index time-series is an average of daily closed prices for nine representative, most liquid listed companies at Bucharest Stock Market.

Although the Hurst coefficient of BET-index time series was close to 1, it was proved that prediction in the case of BET index time series, (figure 1) is a much more very difficult task. For 800 daily observations, the result of simulations was obtained including the test set in training set (Table 3), or excluding the test set from training set (Table 4).

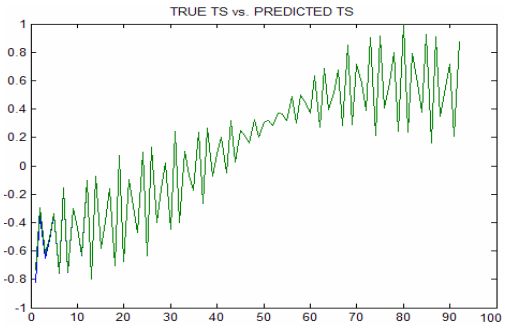

Fig. 5 Weirstrass fractal function, easy to be predicted: H=0.82, a=1.2, b=1.4, ω=10, R=0.99971. Original and predicted time series are approximately the same, NN(8, 2;1).

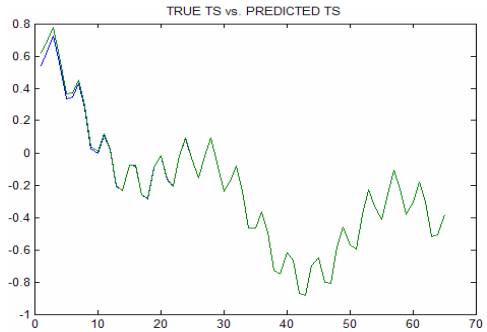

Fig. 6 A difficult Weirstrass case: H=0.53, a=2.1, b=2.07, ω=15, R = 0,99964, NN(35,1;10)

From the first set of simulations, one of the best prediction is graphic represented in figure 7 with the correlation R=0.96116. The training algorithm with the best results was the back-propagation with Levenberg - Marquardt optimization and Bayesian regularization. To be compared with the results in Table 3, some results obtained after training with an algorithm that updates weight and bias values according to the gradient descent momentum and an adaptive learning rate, were presented in Table 5.

Table 3

N	d_u	d_y	R
6	30	2	0.95185
7	30	2	0.96206
8	30	2	0.9535
10	30	2	0.72639
10	35	2	0.72639
11	30	2	0.54259
11	45	2	0.92045
11	45	3	0.92947
12	30	2	0.54337
12	50	3	0.9592
12	50	4	0.82927
15	50	3	0.96116
18	50	3	0.86318

Table 4

N	d_u	d_y	R
6	30	3	0.25758
7	15	2	0.53296
9	24	2	0.88662
10	24	1	0.83436
10	24	2	0.2868
10	30	2	0.87045
11	40	1	0.64629
11	40	2	0.57197
15	50	3	0.07878

Table 5

N	d_u	d_y	R
6	30	2	0.8207
7	30	2	0.9299
8	30	2	0.8281
11	45	2	0.8933
11	45	3	0.8690
12	30	2	0.4266
12	50	3	0.4266
15	50	3	0.7904

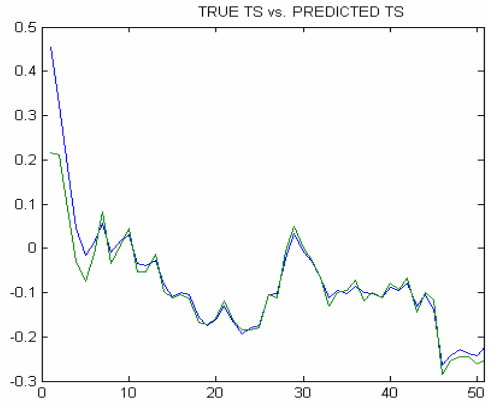

Fig. 7 BET prediction, NN(50, 3;15)

The RNN model presented in figure 3, for two inputs presented simultaneously to the network, was tested for two related time series: BET - index value and BET - index volume (figure 9 and 10). TDL means time delay line. The volume (figure 8) should be enhanced the prediction for the time series values. The results are in table 6. Another test was made by using as second input the delayed time series (with 15 days) of the same BET-values time series (figure 10). The results are in the table 7.

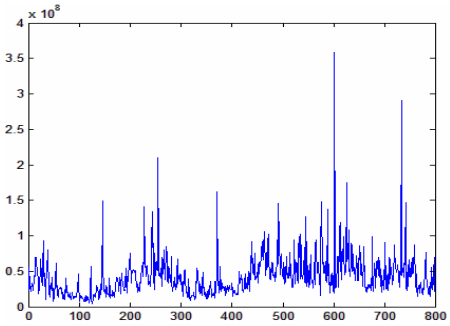

Fig. 8 BET index volume 2005-03.2008

Table 6

N	d_{u1}	d_{u2}	d_y	R
10	25	25	3	0.8147
12	6	6	2	0.5607
12	25	25	3	0.8093
14	14	14	2	0.5916
15	25	25	3	0.7145

Table 7

N	d_{u1}	d_{u2}	d_y	R
10	25	25	3	0.2432
12	6	6	2	0.6143
12	25	25	3	0.5320
14	14	14	2	0.0301
15	25	25	3	0.7308

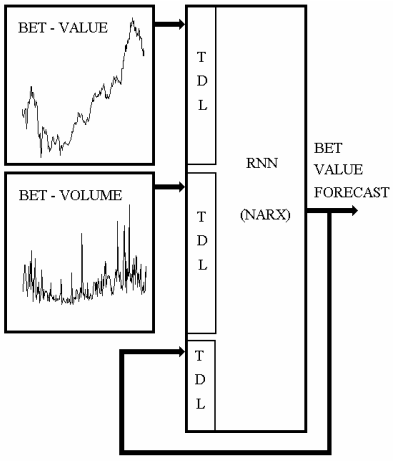

Fig. 9 NARX RNN with simultaneous inputs: BET-index value and BET- index volume

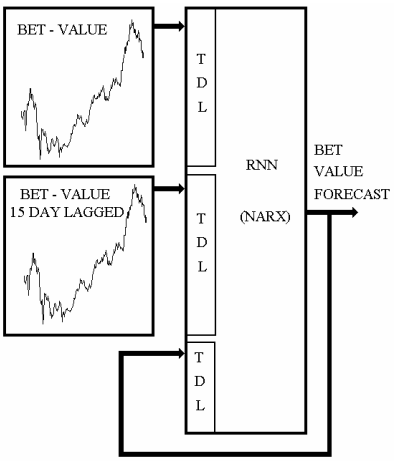

Fig. 10 NARX RNN with simultaneous inputs: BET-index value and BET- index value 15 days lagged

5 Conclusions

In this paper, the performance of the prediction for different time series was tested using a NARX dynamic recurrent neural network. Comparative experiments with real and artificial chaotic time series from diverse domains have been made.

The first conclusion of this paper is that NARX recurrent neural networks have the potential to capture the dynamics of nonlinear dynamic system such as in the examples shown, for the Mackey-Glass system with different delays. This affirmation is based on the fact that correlation coefficient R estimated for the original and generated (1000 points) time series is close to 1 in many cases, and the prediction can be considered of real interest or significance if R>0.98.

The paper has attempted to use traditional statistical methodologies (R/S Rescaled Range, from where Hurst coefficient) to obtain indications to make efficient the process of prediction chaotic time series with RNN. To certain extent, the Hurst coefficient may give a clue, otherwise vague, about existence of long time memory in the analyzed time series. The prediction may fails however, even the values of Hurst coefficient are encouraging, in conformity with R/S theory.

The second conclusion is that the nonlinear NARX models are not without problems, they have limitation in learning long time dependences due to the "vanishing gradient", and like any dynamical system are affected by instability, and have lack of a procedure of optimizing embedded memory.

The last conclusion, and the most important, is that the architecture of the tested RNN model affects the performance of prediction. The most favorable behavior of NARX model is dependent upon the dimension of embedded memory of input and output and the number of neurons in the input layer. The determination of these architectural elements, in an optimal way, is a critical and difficult task for the NARX model, and remains an objective for future works.

The followings lines contain several directions to explore:

- The avoiding of saturation and over-fitting because of too many neurons in network. Too many hidden neurons lead to poor prediction. A solution is the including in NARX models the penalizing terms as Bayessian Information Criteria (BIC), Akaike Information Criteria (AIC) – a process named *regularization*.
- The data input analysis may show how the length of sequences and their correlation influence the

interval and value of predictability (selection the length of lags).
- The finding other training algorithms than back-propagation, for the NARX models. Consecutive restarts of the program with back-propagation training function, give different results, indicating the end of iteration process in local optima.

References:

[1] Simon Haykin, *Neural Networks*, Second Edition, Pearson Education, 1999

[2] Georg Dorffner, Neural Networks for Time Series Processing, *Neural Network World*, Vol. 6, No. 4, 447-468, 1996

[3] J.Farawey, C.Chatfield, Time Series Forecasting with Neural Networks: A Comparative Study Using the Airline Data, *Applied Statistics*, Volume 47, No.2, 1998, pp. 231-250

[4] Tsungnan Lin, Bill G. Horne, Peter Tino, C. Lee Giles, Learning long-term dependencies in NARX recurrent neural networks, *IEEE Transactions on Neural Networks*, Vol. 7, No. 6, 1996, pp. 1329-1351

[5] Yang Gao, Meng Joo Er, NARMAX time series model prediction: feedforward and recurrent fuzzy neural network approaches, *Fuzzy Sets and Systems*, Vol. 150, No. 2, 2005, pp.331-350

[6] M. T. Hagan, O. D. Jesus, R. Schultz, Training Recurrent Networks for Filtering and Control, in (editors) L.R. Medsker, L.C. Jain, *Recurrent Neural Networks – Design and Applications*, CRC Press, 2001

[7] Tsungnan Lin, C. Lee Giles, Bill G. Horne, S.Y. Kung, A Delay Damage Model Selection Algorithm for NARX Neural Networks, *IEEE Transactions on Signal Processing, "Special Issue on Neural Networks"*, Vol. 45, No. 11, 1997, pp. 2719-2730

[8] H. T. Siegelmann, B. G. Horne and C. Lee Giles, Computational capabilities of recurrent NARX neural networks*, IEEE Transactions on Systems, Man and Cybernetics*, Part B, Vol. 27, No.2, 1997, 208-215

[9] Jingtao Yao, Chew Lim Tan, A case study on using neural networks to perform technical forecasting of forex, *Neurocomputing*, 34, 2000, pp. 79-98

[10] Edgar E. Peters, *Fractal Market Analysis*, John Wiley & Sons, 2001

[11] D.S.G. Pollok, *A Handbook of Time – Series, Signal Processing and Dynamics*, Academic Press, 1999

[12] D.P. Mandic, J.A. Chambers, *Recurrent Neural Networks for Prediction*, JohnWiley&Sons, 2001

[13] M. Tertisco, P. Stoica, T. Petrescu, *Modeling and forecasting of time series*, Publ. House of Romanian Academy, 1985

[14] Garnet P. Williams, *Chaos Theory Tamed*, Joseph Henry Press, 1999

[15] H.B. Demuth, M. Beale, *Users' Guide for the Neural Network Toolbox for Matlab*, The Mathworks, Natica, MA, 1998

[16] L.F. Mingo, J. Castellans, G. Lopez, F. Arroyo, *Time Series Analysis with Neural Network*, WSEAS Transactions on Business and Economics, Issue 4, Volume 1, October, ISSN 1109-9526, pp. 303-310, 2004

[17] A.C. Tsakoumis, P. Fessas, V. M. Mladenov, N. E. Mastorakis, *Application of Neural Networks for Short Term Electric Load Prediction*, WSEAS Transaction on systems, Issue 3, Volume 2, July 2003, ISSN 1109-2777, pp. 513-516

[18] A.C. Tsakoumis, P. Fessas, V. M. Mladenov, N. E. Mastorakis, *Application of Chaotic Time Series for Short-Time Load Prediction*, WSEAS TRANSACTION on SYSTEMS, Issue 3, Volume 2, July 2003, ISSN 1109-2777, pp. 517-523

[19] Theodor D. Popescu, *New method for time series forecasting*, WSEAS TRANSACTIONS on CIRCUITS and SYSTEMS, Issue 3, Volume 2, July 2003, ISSN 1109-2734, pp. 582-587

[20] O. Valenzuela, I. Rojas, I. Marquez, M. Pasados, WSEAS TRANSACTION on CIRCUITS and SYSTEMS, *A novel Aproach to ARMA Time Series Model Identification by Neural Network*, Issue 2, Volume 3, April 2004, ISSN 1109 – 2737, pp. 342-347

[21] Wei Huang, Shouyang Wang, Lean Yu, Yukun Bao, L. Wang, *A New Computational Method of Input Selection for Stock Market Forrecasting with Neural Networks*, International Conference on Computational Science, 2006, pp. 308-315

[22] Tilmann Gneiting, Martin Schlather, *Stochastic Models That Separate Fractal Dimension and Hurst Effect*, SIAM review, 46, 2004, pp. 269-282

[23] G. P. Zhang, *Neural networks in Business Forecasting*, Ideea Group Inc., 2003