http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 12 (14): 2815-2819, 2013
ISSN 1812-5638 / DOI: 10.3923/itj.2013.2815.2819

Time Series Prediction with RBF Neural Networks

[1]Dongqing Zhang and [2]Yubing Han
[1]College of Engineering, Nanjing Agricultural University, 210031, Nanjing, China
[2]School of Electronic and Optical Engineering, Nanjing University of Science and Technology, 210094, Nanjing, China

Abstract: Nonlinear time series is quite broadly in various areas such as physical science, social science, economics and so on. The purpose of this study is to forecast these nonlinear time series using Radial Basis Function (RBF) neural networks. Because the prediction errors at previous time are included in the input of RBF networks, classical methods for parameters estimation of RBF networks are not available. Gradient descent and trial method according to Occam's razor are adopted to estimate the parameters and corresponding algorithm is proposed in this study. At last, the data of monthly mean sunspot number are analyzed and the experimental results indicate that the proposed algorithm is effective.

Key words: Prediction, radial basis function neural networks, gradient descent method, occam's razor

INTRODUCTION

At present, many time series are modeled in autoregressive moving average (ARMA) models which are linear with assumption that time series are stationary and follow a stochastic model of white noise (Box *et al*., 1994). However, many time series in real world are nonlinear, so Chen and Billings (1989) proposed nonlinear autoregressive moving average (NARMA) model which is a concise expansion of past inputs and prediction errors and provides a unified representation for a wide class of nonlinear systems. But it is, on most occasions, impossible to derive closed-form analytical expression to represent the nonlinear function in NARMA model, so researchers resort to neural networks for their abilities of approximating functions (Hornik *et al*., 1989), such as feedforward neural networks (Pawlus *et al*., 2013), Multi-Layer Perceptrons (MLPs) (De Freitas *et al*., 2001; Bonnet *et al*., 1997), Radial Basis Function (RBF) neural networks (De Freitas *et al*., 2001a, b; Li *et al*., 2013b; Zhang *et al*., 2012; Gan *et al*., 2012) and so on. Because RBF neural networks tend to be more tractable than other neural networks, they are chosen to approximate the nonlinear function in NARMA model in this study.

PROBLEM STATEMENT

A NARMA(p, q) model for time series $\{y_t, t = 1, 2,..., T\}$ is expressed as (Chen and Billings, 1989):

$$\hat{y}_t = f(y_{t-g}, \cdots, y_{t-1}, e_{t-q}, \cdots, e_{t-1}) + e_t \quad (1)$$

where, $\hat{y}_t$ is predictive value, f(.) is a nonlinear unknown function and g,q are the order of the autoregressive part and the moving part, respectively. $e_{t-q},..., e_{t-1}$ are measurement noise. Since, the noise is generally unobserved, it can be replaced by the prediction error, i.e., $e_{t-q} = y_{t-q} - \hat{y}_{t-q}, \cdots, e_{t-1} = y_{t-1} - \hat{y}_{t-1}$. e_t is measurement noise at time t which can be any specified distribution, it is assumed to be zero-mean Gaussian in this study, i.e., $e_t \sim N(0, \sigma_t^2)$.

The key of NARMA model for prediction is to choose proper expression of nonlinear function f(.) in Eq. 1. However, in many cases, it is impossible to express nonlinear function in exact expression. In this study, a RBF networks model is chosen to approximate nonlinear function because of its simple topological structure and its ability to reveal how learning proceeds in an explicit manner.

We shall consider an approximation scheme consisting of a mixture of k RBFs and a linear regression term (Holmes and Mallick, 1998):

$$\tilde{y}_t = \sum_{j=1}^{k} a_j \phi(\| x_t - c_j \|) + \beta^T x_t, \quad k \geq 0 \quad (2)$$

where, $x_t = (y_{t-g}, \cdots, y_{t-1}, e_{t-q}, \cdots, e_{t-1})^T$ denotes the input of RBF networks. $\tilde{y}_t$ denotes the output of RBF networks. ||.|| denotes a distance metric (usually Euclidean), $\phi(.)$ is the basis functions, c_j denotes the jth RBF centre for a model with k RBFs, a_j denotes the jth RBF amplitude, $\beta = (\beta_1,...,\beta_2)^T$ denotes the linear regression parameters and d = g+q.

In this study, we choose Gaussian basis function $\phi(x) = e^{-\lambda x^2}$, where λ is the width of Gaussian basis function and will be treated as a user-set parameter

Corresponding Author: Dongqing Zhang, College of Engineering, Nanjing Agricultural University, Nanjing 210031, China

(De Freitas *et al*., 2001). For convenience, the approximation model is expressed in vector-matrix form:

$$\tilde{y}_t = D_t a_t \tag{3}$$

where, $D_t = (y_{t-g}, \cdots, y_{t-1}, e_{t-q}, \cdots, e_{t-1}, \phi(x_t, c_1), \cdots, \phi(x_t, c_k))$, $a_t = (\beta_1, \cdots, \beta_d, a_1, \cdots, a_k)^T$.

TIME SERIES PREDICTION BASED ON RBF NETWORKS

Here, our intention is to obtain $\hat{y}_t$ using RBF networks model given all available information $y_{1:T}$, where $y_{1:T} = (y_1, \cdots, y_T)$. However, before forecasting, the following parameters of the RBF networks should be estimated: input neurons g,q, hidden neurons k, RBF centers c_j, width of Gaussian basis function λ, weights a_j and linear regression parameters β.

Because errors $(e_{t-q}, \cdots, e_{t-1})$ are included in the inputs of our RBF networks, RBF centers c_j can't be estimated using classical methods such as clustering method (Li *et al*., 2013a; Montazer *et al*., 2013), resource allocating method (Platt, 1991; Corradini *et al*., 2012) and so on. Therefore, gradient descent and trial method according to Occam's razor are adopted in this study. Occam's razor means that the simplest explanation is best or causes shall not be multiplied beyond necessity (Courtney and Courtney, 2012). Here for RBF parameters estimation, set the threshold ε and the RBF networks start from the simplest architecture to more complex one. Given certain RBF networks architecture p, q, k, gradient descent method is used to optimize the parameters c_j, a_j, β, λ. Once the fitting error is less than ε, the iterative procedure stops and the RBF networks is the best one.

For convenience, the input of RBF networks is divided into two parts, i.e., $x_t = (x_t^d, x_t^e)^T$, where $x_t^d = (y_{t-g}, \cdots, y_{t-1}), x_t^e = (e_{t-q}, \cdots, e_{t-1})$. Accordingly, we have $\beta^T = (\beta^d, \beta^e)$ and $c_j = (c_j^d, c_j^e)^T, j = 1, \cdots, k$. Because $\phi(x) = e^{-\lambda x^2}$ is adopted as basis function, then Eq. 2 can be rewritten as:

$$\begin{aligned}\tilde{y}_t = &\sum_{j=1}^{k} a_j \exp\left\{-\lambda\left(\sum_{i=1}^{g}(y_{t-i} - c_{j,i}^d)^2 + \sum_{m=1}^{q}(e_{t-m} - c_{j,m}^e)^2\right)\right\} \\ &+ \sum_{i=1}^{g}\beta_i^d y_{t-i} + \sum_{m=1}^{q}\beta_m^e e_{t-m}, \quad \max(g,q)+1 \le t \le T\end{aligned} \tag{4}$$

where, $e_{t-m} = y_{t-m} - \tilde{y}_{t-m}, m = 1, \cdots, q$. For simplicity, the implicit function of Eq. 4 is given by:

$$\begin{aligned}&\tilde{y}_t = F_t\left(\lambda, a_{1:k}, \beta_{1:g}^d, \beta_{1:q}^e, c_{1:k,1:g}^d, c_{1:k,1:q}^e, y_{t-1:t-g}, e_{t-1:t-q}\right), \\ &t = \max(g,q)+1:T\end{aligned} \tag{5}$$

In order to train RBF networks, an objective function is constructed as follows:

$$E = \frac{1}{2}\sum_{t=\max(g,q)+1}^{T}(y_t - \tilde{y}_t)^2 \tag{6}$$

The objective of training is to find c_j, a_j, β, λ that determine the global minimum of the objective function E. Unless the model is overfitted, these parameters should provide good generalization. The following iterative procedure of a gradient descent training algorithm is given by:

$$\begin{cases}\lambda(r) = \lambda(r-1) - \eta_\lambda \dfrac{\partial E}{\partial \lambda}(r-1) \\ a_j(r) = a_j(r-1) - \eta_a \dfrac{\partial E}{\partial a_j}(r-1), j = 1, \cdots, k \\ \beta_i^d(r) = \beta_i^d(r-1) - \eta_{\beta^d} \dfrac{\partial E}{\partial \beta_i^d}(r-1), i = 1, \cdots, g \\ \beta_m^e(r) = \beta_m^e(r-1) - \eta_{\beta^e} \dfrac{\partial E}{\partial \beta_m^e}(r-1), m = 1, \cdots, q \\ c_{j,i}^d(r) = c_{j,i}^d(r-1) - \eta_{c^d} \dfrac{\partial E}{\partial c_{j,i}^d}(r-1), j = 1, \cdots, k, i = 1, \cdots, g \\ c_{j,m}^e(r) = c_{j,m}^e(r-1) - \eta_{c^e} \dfrac{\partial E}{\partial c_{j,m}^e}(r-1), j = 1, \cdots, k, m = 1, \cdots, q\end{cases} \tag{7}$$

where, $\eta_\lambda, \eta_a, \eta_{\beta^d}, \eta_{\beta^e}, \eta_{c^d}, \eta_{c^e}$ are the iterative step sizes, r is the number of iteration. Obviously, when the step sizes are small enough, the algorithm is convergence.

Next, we discuss the details of the partial derivatives:

$$\frac{\partial E}{\partial \lambda}, \frac{\partial E}{\partial a_j}, \frac{\partial E}{\partial \beta_i^d}, \frac{\partial E}{\partial \beta_m^e}, \frac{\partial E}{\partial c_{j,i}^d}, \frac{\partial E}{\partial c_{j,m}^e}$$

appeared in Eq. 7. When t<max(g, q), without loss of generality, we set $e_t = 0$ and get:

$$\begin{cases}\dfrac{\partial E}{\partial \lambda} = \displaystyle\sum_{t=\max(g,q)+1}^{T}(\tilde{y}_t - y_t)\frac{\partial \tilde{y}_t}{\partial \lambda} \\ \dfrac{\partial \tilde{y}_t}{\partial \lambda} = \dfrac{\partial F_t}{\partial \lambda} + \displaystyle\sum_{l=1}^{q}\frac{\partial F_t}{\partial e_{t-l}}\frac{\partial e_{t-l}}{\partial \lambda} \\ \qquad = \dfrac{\partial F_t}{\partial \lambda} - \displaystyle\sum_{l=1}^{q}\frac{\partial F_t}{\partial e_{t-l}}\frac{\partial \tilde{y}_{t-l}}{\partial \lambda}\end{cases} \tag{8}$$

$$\begin{cases}\dfrac{\partial E}{\partial a_j} = \displaystyle\sum_{t=\max(g,q)+1}^{T}(\tilde{y}_t - y_t)\frac{\partial \tilde{y}_t}{\partial a_j} \\ \dfrac{\partial \tilde{y}_t}{\partial a_j} = \dfrac{\partial F_t}{\partial a_j} + \displaystyle\sum_{l=1}^{q}\frac{\partial F_t}{\partial e_{t-l}}\frac{\partial e_{t-l}}{\partial a_j} \\ \qquad = \dfrac{\partial F_t}{\partial a_j} - \displaystyle\sum_{l=1}^{q}\frac{\partial F_t}{\partial e_{t-l}}\frac{\partial \tilde{y}_{t-l}}{\partial a_j}\end{cases} \tag{9}$$

$$\begin{cases}\dfrac{\partial E}{\partial \beta_i^d} = \displaystyle\sum_{t=\max(g,q)+1}^{T}(\tilde{y}_t - y_t)\frac{\partial \tilde{y}_t}{\partial \beta_i^d} \\ \dfrac{\partial \tilde{y}_t}{\partial \beta_i^d} = \dfrac{\partial F_t}{\partial \beta_i^d} + \displaystyle\sum_{l=1}^{q}\frac{\partial F_t}{\partial e_{t-l}}\frac{\partial e_{t-l}}{\partial \beta_i^d} \\ \qquad = \dfrac{\partial F_t}{\partial \beta_i^d} - \displaystyle\sum_{l=1}^{q}\frac{\partial F_t}{\partial e_{t-l}}\frac{\partial \tilde{y}_{t-l}}{\partial \beta_i^d}\end{cases} \tag{10}$$

$$\begin{cases} \dfrac{\partial E}{\partial \beta_m^e} = \sum\limits_{t=\max(g,q)+1}^{T} (\tilde{y}_t - y_t)\dfrac{\partial \tilde{y}_t}{\partial \beta_m^e} \\ \dfrac{\partial \tilde{y}_t}{\partial \beta_m^e} = \dfrac{\partial F_t}{\partial \beta_m^e} + \sum\limits_{l=1}^{q} \dfrac{\partial F_t}{\partial e_{t-l}} \dfrac{\partial e_{t-l}}{\partial \beta_m^e} \\ \quad = \dfrac{\partial F_t}{\partial \beta_m^e} - \sum\limits_{l=1}^{q} \dfrac{\partial F_t}{\partial e_{t-l}} \dfrac{\partial \tilde{y}_{t-l}}{\partial \beta_m^e} \end{cases} \tag{11}$$

$$\begin{cases} \dfrac{\partial E}{\partial c_{j,i}^d} = \sum\limits_{t=\max(g,q)+1}^{T} (\tilde{y}_t - y_t)\dfrac{\partial \tilde{y}_t}{\partial c_{j,i}^d} \\ \dfrac{\partial \tilde{y}_t}{\partial c_{j,i}^d} = \dfrac{\partial F_t}{\partial c_{j,i}^d} + \sum\limits_{l=1}^{q} \dfrac{\partial F_t}{\partial e_{t-l}} \dfrac{\partial e_{t-l}}{\partial c_{j,i}^d} \\ \quad = \dfrac{\partial F_t}{\partial c_{j,i}^d} - \sum\limits_{l=1}^{q} \dfrac{\partial F_t}{\partial e_{t-l}} \dfrac{\partial \tilde{y}_{t-l}}{\partial c_{j,i}^d} \end{cases} \tag{12}$$

$$\begin{cases} \dfrac{\partial E}{\partial c_{j,m}^e} = \sum\limits_{t=\max(g,q)+1}^{T} (\tilde{y}_t - y_t)\dfrac{\partial \tilde{y}_t}{\partial c_{j,m}^e} \\ \dfrac{\partial \tilde{y}_t}{\partial c_{j,m}^e} = \dfrac{\partial F_t}{\partial c_{j,m}^e} + \sum\limits_{l=1}^{q} \dfrac{\partial F_t}{\partial e_{t-l}} \dfrac{\partial e_{t-l}}{\partial c_{j,m}^e} \\ \quad = \dfrac{\partial F_t}{\partial c_{j,m}^e} - \sum\limits_{l=1}^{q} \dfrac{\partial F_t}{\partial e_{t-l}} \dfrac{\partial \tilde{y}_{t-l}}{\partial c_{j,m}^e} \end{cases} \tag{13}$$

Then the partial derivatives of the implicit function F_t can be calculated by:

$$\begin{cases} \dfrac{\partial F_t}{\partial \lambda} = -\sum\limits_{j=1}^{k} a_j \left(\sum\limits_{i=1}^{g} (y_{t-i} - c_{j,i}^d)^2 + \sum\limits_{m=1}^{q} (e_{t-m} - c_{j,m}^e)^2 \right) \\ \qquad \times \exp\left[-\lambda \left(\sum\limits_{i=1}^{g} (y_{t-i} - c_{j,i}^d)^2 + \sum\limits_{m=1}^{q} (e_{t-m} - c_{j,m}^e)^2 \right) \right] \\ \dfrac{\partial F_t}{\partial a_j} = \exp\left[-\lambda \left(\sum\limits_{i=1}^{g} (y_{t-i} - c_{j,i}^d)^2 + \sum\limits_{m=1}^{q} (e_{t-m} - c_{j,m}^e)^2 \right) \right] \\ \dfrac{\partial F_t}{\partial \beta_i^d} = y_{t-i} \\ \dfrac{\partial F_t}{\partial \beta_m^e} = e_{t-m} \\ \dfrac{\partial F_t}{\partial c_{j,i}^d} = 2\lambda a_j (y_{t-i} - c_{j,i}^d) \\ \qquad \times \exp\left[-\lambda \left(\sum\limits_{i=1}^{g} (y_{t-i} - c_{j,i}^d)^2 + \sum\limits_{m=1}^{q} (e_{t-m} - c_{j,m}^e)^2 \right) \right] \\ \dfrac{\partial F_t}{\partial c_{j,m}^e} = 2\lambda a_j (e_{t-m} - c_{j,m}^e) \\ \qquad \times \exp\left[-\lambda \left(\sum\limits_{i=1}^{g} (y_{t-i} - c_{j,i}^d)^2 + \sum\limits_{m=1}^{q} (e_{t-m} - c_{j,m}^e)^2 \right) \right] \\ \dfrac{\partial F_t}{\partial e_{t-1}} = \beta_1^e - 2\lambda \sum\limits_{j=1}^{k} a_j (e_{t-1} - c_{j,1}^e) \\ \qquad \times \exp\left[-\lambda \left(\sum\limits_{i=1}^{g} (y_{t-i} - c_{j,i}^d)^2 + \sum\limits_{m=1}^{q} (e_{t-m} - c_{j,m}^e)^2 \right) \right] \end{cases} \tag{14}$$

Substituting:

$$\frac{\partial F_t}{\partial \lambda}, \frac{\partial F_t}{\partial a_j}, \frac{\partial F_t}{\partial \beta_i^d}, \frac{\partial F_t}{\partial \beta_m^e}, \frac{\partial F_t}{\partial c_{j,i}^d}, \frac{\partial F_t}{\partial c_{j,m}^e}, \frac{\partial F_t}{\partial e_{t-1}}$$

into Eq. 8-12, we can obtain:

$$\frac{\partial E}{\partial \lambda}, \frac{\partial E}{\partial a_j}, \frac{\partial E}{\partial \beta_i^d}, \frac{\partial E}{\partial \beta_m^e}, \frac{\partial E}{\partial c_{j,i}^d}, \frac{\partial E}{\partial c_{j,m}^e}$$

The details of the parameters estimation based on gradient descent and trial method according to Occam's razor are summarized as algorithm 1. Here, we introduce C_{RBF} to represent the complexity of RBF networks which is defined as:

$$C_{RBF} = (k+1)(g+q+1) \tag{15}$$

Algorithm 1: Parameters estimation algorithm of RBF networks

Set the threshold ε, the maximum orders of the autoregressive and moving part are g_{max} and q_{max} and the maximum number of hidden nodes k_{max}. We can estimate the parameters $g, q, k, c_j, a_j, \beta, \lambda$ simultaneously.

For:

$$C_{RBF} = 1 : (k_{max}+1)(g_{max}+q_{max}+1)$$

Denote:

$$\Omega = \left\{ (g,q,k) \middle| \begin{array}{l} g \in Z, q \in Z, k \in Z \\ (k+1)(g+q+1) = C_{RBF} \\ g \le g_{max}, q \le q_{max}, k \le k_{max} \end{array} \right\}$$

where Z is the set of positive integer.

If $\Omega \neq \emptyset$

For each (g, q, k)$\in\Omega$

Adopt the gradient descent method to train RBF networks for c_j, a_j, β, λ

Calculate the fitting error E(g, q, k)

end

$E(C_{RBF}) = \min\limits_{(g,q,k)\in\Omega} \{E(g,q,k)\}$

If $E(C_{RBF}) \le \varepsilon$

Return the parameters of RBF networks $g, q, k, c_j, a_j, \beta, \lambda$

end

end

end

If we can't find RBF networks with $E(C_{RBF}) \le \varepsilon$, then ε or $(g_{max}, q_{max}, k_{max})$ should be increased and repeat the Algorithm 1.

After we completed the training of RBF networks, the parameters $g, q, k, c_j, a_j, \beta, \lambda$ have been estimated. Because e_t is assumed to be zero-mean Gaussian, according to Eq. 1, the output of RBF networks $\tilde{y}_t = D_t a_t$ is the prediction value $\hat{y}_t$ at time t.

EXPERIMENTAL RESULTS

The solar activity is characterized by means of the relative Wolf sunspot number. In this section we will concentrate our studies on this index, since it has the largest data set compared to other activity measurements. We will consider the data of monthly mean sunspot number over a period of 502 months from Jan. 1964 to Oct. 2005 and the data come from SIDC (http://sidc.oma.be/).

In our case, we use the sunspot values for Jau.1964-Dec.1996 as the training set (396 samples) and Jau.1997-Oct. 2005 as the test set (106 samples). That is, using the sunspot values from 20th to 22nd solar cycle to predict the ones in 23rd solar cycle. The performance of the RBF networks forecaster is measured in terms of Mean Absolute Percentage Error (MAPE) which is defined by:

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \tag{16}$$

where, n is the length of test data.

At first, the parameters of RBF networks are estimated using Algorithm 1, for the details, refer to section 3. In the experiment, we set $g_{max} = 12$, $q_{max} = 12$, $k_{max} = 20$, $\varepsilon = 0.001$, $\eta_\lambda = \eta_a = \eta_{\beta^d} = \eta_{\beta^s} = \eta_{c^d} = \eta_{c^s} = 0.001$ and the maximum number of iteration is 1000. Figure 1 shows the training error versus iteration time and the fitted monthly mean sunspot number is given in Fig. 2.

For training RBF networks, the iteration time is 243 and the estimated parameters are:

$$g = 10,\ q = 5,\ k = 8,\ \lambda = 1.011$$

$$a = (a_1, \cdots, a_8)^T = (0.0099,\ 0.1389,\ 0.2029,\ 0.1921,\ 0.5990,\ 0.2722,\ 0.1992,\ 0.0153)^T,$$

$$\beta = (\beta_1, \cdots, \beta_{15})^T = (0.0272,\ -0.2215,\ 0.2054,\ -0.2253,\ -0.1081,\ 0.4009,\ 0.1463,\ -0.1432,\ 0.2784,\ 0.6159,\ 0.4618,\ 0.8169,\ 0.8700,\ 1.0336,\ 1.0473)^T$$

Because there are 8 hidden nodes and the center vector of each hidden node has dimension of 15, we omit these center vectors due to the limited space.

After the network is convergence, it is used to predict monthly mean sunspot number of test data and the MAPE is 3.26%. Here, we compare the results against trivial model. The trivial model simply involves using the current value of the option as the next prediction, namely $\hat{y}_{t+1} = y_t$ and its MAPE is 3.98%. The predictive values and errors of two methods are illustrated in Fig. 3.

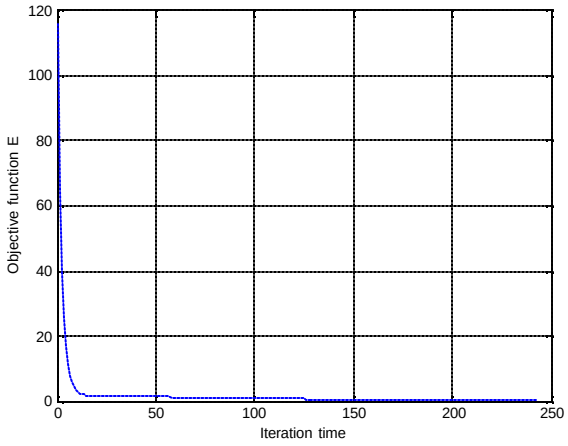

Fig. 1: Objective function E versus iteration time

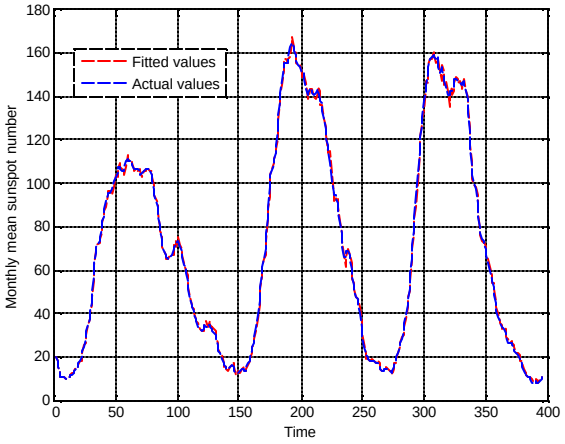

Fig. 2: Actual vs fitted monthly mean sunspot number (from Jan. 1964 to Dec. 1996)

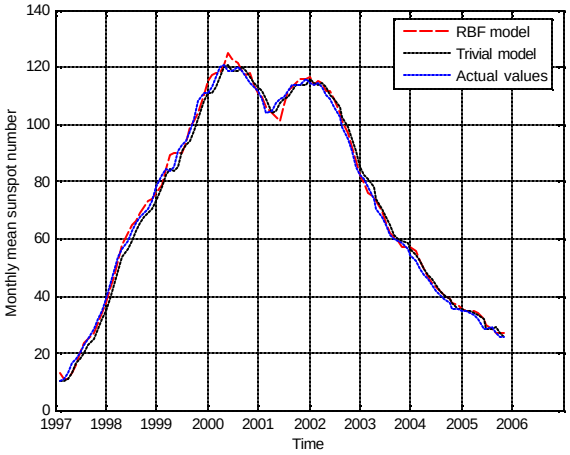

Fig. 3: Actual vs predicted monthly mean sunspot number (from Jan. 1997 to Oct. 2005)

CONCLUSION

A RBF networks model is adopted to forecast nonlinear time series in this study. Because the previous prediction errors are included in the input of RBF networks, classical methods for parameters estimation of RBF networks are not available. Gradient descent and trial method according to Occam's razor are applied to estimate the parameters and corresponding parameters estimation algorithm is proposed in this study. At last, the data of monthly mean sunspot number are analyzed and we compare the results against the trivial model, the experimental results indicate that the proposed algorithm of this study is effective.

AKNOWLEDGMENT

This study is supported by the National Natural Science Foundation of China (71101072, 71301077) and the Natural Science Foundation of Jiangsu Province (BK2011652).

REFERENCES

Bonnet, D., V. Labouisse and A. Grumbach, 1997. δ-NARMA neural networks: A new approach to signal prediction. IEEE Trans. Signal Proces., 45: 2799-2810.

Box, G.E.P., G.M. Jenkins and G.C. Reinsel, 1994. Time Series Analysis: Forecasting and Control. 3rd Edn., Prentice-Hall, Englewood Cliffs, New Jersey.

Chen, S. and S.A. Billings, 1989. Representations of non-linear systems: The NARMAX model. Int. J. Control, 49: 1013-1032.

Corradini, M.L., V. Fossi, A. Giantomassi, G. Ippoliti, S. Longhi and G. Orlando, 2012. Minimal resource allocating networks for discrete time sliding mode control of robotic manipulators. IEEE Trans. Ind. Inform., 8: 733-745.

Courtney, A. and M. Courtney, 2012. Comments regarding on the nature of science. Phys. Can., 64: 7-8.

De Freitas, N., C. Andrieu, P. Hojen-Sorensen, M. Niranjan and A. Gee, 2001. Sequential Monte Carlo Methods for Neural Networks. In: Sequential Monte Carlo Methods in Practice, Doucet, A., N. de Freitas and N. Gordon (Eds.). Springer-Verlag, New York, ISBN-13: 978-1-4419-2887-0, pp: 359-379.

Gan, M., H. Peng and X.P. Dong, 2012. A hybrid algorithm to optimize RBF network architecture and parameters for nonlinear time series prediction. Applied Math. Model., 36: 2911-2919.

Holmes, C.C. and B.K. Mallick, 1998. Bayesian radial basis functions of variable dimension. Neural Computat., 10: 1217-1233.

Hornik, K., M. Stinchcombe and H. White, 1989. Multilayer feedforward networks are universal approximators. Neural Networks, 2: 359-366.

Li, M.S., X.Y. Huang, H.S. Liu, B.X. Liu and Y. Wu, 2013. Prediction of the gas solubility in polymers by a radial basis function neural network based on chaotic self-adaptive particle swarm optimization and a clustering method. J. Applied Polym. Sci., 130: 3825-3832.

Li, T.S., D. Wang, J.F. Li and Y.M. Li, 2013b. Adaptive decentralized NN control of nonlinear interconnected time-delay systems with input saturation. Asian J. Control, 15: 533-542.

Montazer, G.A., H. Khoshniat and V. Fathi, 2013. Improvement of RBF neural networks using Fuzzy-OSD algorithm in an online radar pulse classification system. Applied Soft Comput., 13: 3831-3838.

Platt, J., 1991. A resource-allocating network for function interpolation. Neural Computat., 3: 213-225.

Pawlus, W., H.R. Karimi and K.G. Robbersmyr, 2013. Data-based modeling of vehicle collisions by nonlinear autoregressive model and feed forward neural network. Inform. Sci., 235: 65-79.

Zhang, D.Q., Y.B. Han and X.Y. Tang, 2012. Nonlinear/non-gaussian time series prediction based on RBF-HMM-GMM model. Telkomnika, 10: 1214-1226.