Prediction of Chaotic Time Series Based on the Recurrent Predictor Neural Network

Min Han, *Member, IEEE*, Jianhui Xi, Shiguo Xu, and Fu-Liang Yin

***Abstract*—Chaos limits predictability so that the long-term prediction of chaotic time series is very difficult. The main purpose of this paper is to study a new methodology to model and predict chaotic time series based on a new recurrent predictor neural network (RPNN). This method realizes long-term prediction by making accurate multistep predictions. This RPNN consists of nonlinearly operated nodes whose outputs are only connected with the inputs of themselves and the latter nodes. The connections may contain multiple branches with time delays. An extended algorithm of self-adaptive back-propagation through time (BPTT) learning algorithm is used to train the RPNN. In simulation, two performance measures [root-mean-square error (RMSE) and prediction accuracy (PA)] show that the proposed method is more effective and accurate for multistep prediction. It can identify the systems characteristics quite well and provide a new way to make long-term prediction of the chaotic time series.**

***Index Terms*—Chaos, prediction, recurrent neural network.**

I. Introduction

PREDICTION for chaotic time series is to approximate the unknown nonlinear functional mapping of a chaotic signal. The laws underlying the chaotic time series can be expressed as a deterministic dynamical system, but these deterministic equations are usually not given explicitly. Farmer and Sidorowich [1] suggest reconstructing the dynamics in phase space by choosing a suitable embedding dimension and time delay. Takens' [2] theorem ensures that the method is reliable, based on the fact that the interaction between the variables is such that every component contains information on the complex dynamics of the system. Owing to the presence of positive Lyapunov exponents, short-term prediction [3] has been proved feasible by many methods. On the other hand, however, the possibility of making long-term prediction has been prevented. Linear-based methods are inadequate to model most nonlinear dynamical processes. Jinno *et al.* [4] and Xu *et al.* [5] proposed that the extended Kalman filter can be used to identify parameters and update predictions based on observed chaotic time series, but suitable system equation and initial values are necessarily acquired in this method. It usually cannot be realized in practical problem, which limits the applications of Kalman filter. So, advanced identification tools are badly needed for chaotic system.

In recent years, particular interest has been put into predicting chaotic time series using neural networks because of their universal approximation capabilities [6]. Most applications in this field are based on feed-forward neural networks, such as the Back Propagation (BP) network [7], Radial Basis Function (RBF) network [8], and so on. Defining a suitable number of hidden nodes is a difficult problem of these networks. Besides, being static networks, these networks have limitations to identify chaotic dynamical systems. Recurrent neural networks (RNNs), with the consideration of the internal feedback, are more general models than the feed-forward networks. It is widely used tool for the prediction of time series [9], [10]. Although the gradient descent algorithms commonly used to train RNNs exhibit certain problems during training, such as lack of stability and being the difficulty of dealing with long-term dependencies in the time series, RNNs are computationally more powerful than feed-forward networks [11], [12], and valuable approximation results were obtained for prediction problems. Furthermore, recently, a number of researchers have added connections with time delays to the RNNs [13], [14], which often allows gradient descent algorithms to find better solutions in these cases. Hirasawa *et al.* [15], [16] proposed a Universal learning network (ULN), which consists of a number of nodes connected by multiple branches with arbitrary time delays. This network has a generalized framework to model and control complex systems. However, fully connection with multiple branches may increase the computational cost and make network memory redundant information. In the field of prediction, the main problem is how to use the past states and the current inputs to decide the next states. Considering that most of the practical systems are causal systems, the current state point will influence the state values at a later moment, and it cannot affect state points appearing earlier. A new recurrent predictor neural network (RPNN) is constructed in this paper.

RPNN tries to decrease redundant branches between nodes before the training. The nodes represent the processing elements or time instants in series, and the branches between the nodes describe the relation among the processes. RPNN has feedback branches existing between 1) nodes and nodes themselves and 2) nodes and those nodes representing state points at latter moment. When an RNN is used to predict a chaotic time series, the number of input nodes can be selected according to the estimated attractor dimension of a chaotic signal [17]. Learning of RPNN is to determinate its optimal parameter values that minimize a criterion function of the network. According to Hirasawa *et al.* [15], the derivatives of the node outputs can be obtained by

Manuscript received May 7, 2002; revised November 17, 2003. This work was supported by Project 60374064 of the National Natural Science Foundation of China and Project G1999043602 of the grand basic research development plan. The associate editor coordinating the review of this paper and approving it for publication was Prof. Derong Liu.

M. Han, J. Xi, and F. Yin are with the School of Electronic and Information Engineering, Dalian University of Technology, Dalian, China, 116023 (e-mail: minhan@dlut.edu.cn; xjhui@student.dlut.edu.cn; flyin@dlut.edu.cn).

S. Xu is with the School of Civil Engineering and Architecture, Dalian University of Technology, Dalian, China, 116023 (e-mail: sgxu@dlut.edu.cn).

Digital Object Identifier 10.1109/TSP.2004.837418

propagating the changes in the criterion function caused by the changes in the other node outputs backward in time/space. The idea is an extension of the idea used in back-propagation through time (BPTT) [18]. Along this line, a generalized self-adaptive learning rate backward propagation algorithm is used to train RPNN.

In this paper, the principles of nonlinear time series prediction based on RPNN are set forth to try to exploit a new methodology that embraces the entire and long-term behavior of the observed time series. In order to attain accurate prediction results, when new information is presented to the network, the parameters should be accordingly adjusted. So, before each prediction, a set of the latest signal records is presented to the network as a Teaching Signal and use the above-mentioned gradient-based learning algorithm to train the network. In the following sections, Section II describes the structure and the learning algorithm of RPNN. Section III considers the problem of chaotic time series prediction using RPNN. In Section IV, the prediction simulation using RPNN is described. Finally, Section V gives summary and discussion.

II. RPNN

The RPNN is a special RNN. It consists of two kinds of elements: nodes and branches. The main purpose of RPNN is to provide a more direct link between the activity of a neuron and the output of the network at a later moment. The features of RPNN have listed as follows.

- The nodes of RPNN correspond to the instant points of the variables present in the phenomenon. The output of a node is fed back into the inputs of itself and those nodes representing the instant points at latter moment.
- RPNN has multiple branches with time delays between corresponding nodes, which represent the relations between those nodes.

A. Description of RPNN Structure

The general architecture of RPNN is shown in Fig. 1. The salient property is that the node activation functions (internal states) are fed back into themselves and their latter nodes at every time step to provide an additional input. This recurrence gives the network dynamical properties that make it possible for the network to possess internal memory. So the output function of the RPNN network can thus be represented as

$$h_j(t)=f_j\left(r_j(t)+\theta_j+\sum_{i=1}^{j}\sum_{p=1}^{B_{ij}}h_i(t-D_{ij}(p))\times\omega_{ij}^{p}(t)\right),\quad t\in T \tag{1}$$

where $h_j(t)$ is the output value of node j at time t, $f_j(\cdot)$ is a nonlinear function, $r_j(t)$ is value of the jth external input variable at time t, θ_j is the threshold parameter, $D_{ij}(p)$ is the time delay on the pth branch between node i and j, B_{ij} is number of branches from node i to j, $\omega_{ij}^{p}(t)$ is the weight of the pth branch from node i to j at time t, and T is the discrete set of sampling instants.

Functions $f_j(\cdot)$ that govern the operation of the nodes can be any continuously differentiable functions, for example, Tanh functions and linear functions. Here, bipolar sigmoidal function is employed, and (1) can be expressed specifically as

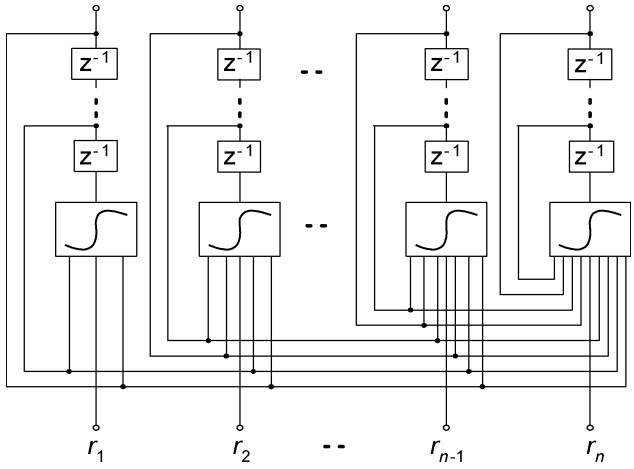

Fig. 1. General architecture of the RPNN.

$$h_j(t)=A\frac{1-e^{-\phi_j\alpha_j(t)}}{1+e^{-\phi_j\alpha_j(t)}} \tag{2}$$

$$\alpha_j(t)=r_j(t)+\theta_j+\sum_{i=1}^{j}\sum_{p=1}^{B_{ij}}h_i\left(t-D_{ij}(p)\right)\times\omega_{ij}^{p}(t) \tag{3}$$

where ϕ_j is the slope parameter of node j.

B. Description of RPNN Algorithm

From Fig. 1, assuming that the network consists of n nodes, the objective of the RPNN algorithm is to adjust the weights of each branch between nodes to minimize a criterion function E_S of the network.

$$E_S=\frac{1}{2}\sum_{t=1}^{S}\sum_{j=1}^{M}\left(h_j(t)-\hat{h}_j(t)\right)^2 \tag{4}$$

where S is the total number of samples, M is the output dimension, $\hat{h}_j(t)$ is the target value, and $h_j(t)$ is the output of network computed by (2) and (3).

Multiple branches with time delays make it deficient for traditional BPTT to identify RPNN. In this paper, the ordered derivative [15] and [19] is exploited to adjust the weights. This method can be represented as follows:

$$\omega_{ij}^{p}(t')=\omega_{ij}^{p}(t'-1)-\gamma\frac{\partial^{+}E_S}{\partial\omega_{ij}^{p}(t'-1)} \tag{5}$$

where t' is learning times, and γ is the learning rate coefficient.

$$\frac{\partial^{+}E_S}{\partial\omega_{ij}^{p}(t')}=\sum_{t\in T}\left[\frac{\partial h_j(t)}{\partial\omega_{ij}^{p}(t')}\delta_j(t)\right]+\frac{\partial E_S}{\partial\omega_{ij}^{p}(t')} \tag{6}$$

$$\begin{aligned}\delta_j(t)&=\frac{\partial^{+}E_S}{\partial h_j(t)}\\&=\sum_{k\in K}\sum_{p\in B_{jk}}\left[\frac{\partial h_k\left(t+D_{jk}(p)\right)}{\partial h_j(t)}\times\delta_k\left(t+D_{jk}(p)\right)\right]\\&\quad+\frac{\partial E_S}{\partial h_j(t)}\end{aligned} \tag{7}$$

where K is the set of numbers of nodes whose inputs are connected to node j. The second term of (6) is the direct influence on E_S of $\omega_{ij}^p(t')$, assuming that all the other variables are constant. The first term of (6) corresponds to the indirect effects on E_S of $\omega_{ij}^p(t')$. Its evaluation requires the help of some intermediate variables such as $h_j(t)$, which is the output of node j directly affected by the parameter $\omega_{ij}^p(t')$. Since $h_j(t)$ possibly affects E_S directly or indirectly, the ordered derivative $\delta_j(t) = \partial^+ E_S/\partial h_j(t)$ is also considered. Therefore, (6) takes into consideration both the direct and indirect paths that lead to the change of criterion function E_S. Equation (7) computes $\delta_j(t)$ taking the outputs of the downstream nodes k as the intermediate variables. Supposing MO is the set of output nodes, the above equations can be further described by

$$\begin{aligned}\frac{\partial h_j(t)}{\partial \omega_{ij}^p(t')} &= \frac{\partial h_j(t)}{\partial \alpha_j(t)} \frac{\partial \alpha_j(t)}{\partial \omega_{ij}^p(t')} \\ &= A \frac{2\phi e^{-\phi\alpha_j(t)}}{\left(1+e^{-\phi\alpha_j(t)}\right)^2} \cdot h_i\left(t - D_{ij}(p)\right) \\ &= A \cdot \left[1 - \left(\frac{h_j(t)}{A}\right)^2\right] \\ &\quad \cdot \frac{\phi}{2} \cdot h_i\left(t - D_{ij}(p)\right) \qquad (8)\end{aligned}$$

$$\begin{aligned}\frac{\partial h_k\left(t + D_{jk}(p)\right)}{\partial h_j(t)} &= \frac{\partial h_k\left(t + D_{jk}(p)\right)}{\partial \alpha_k\left(t + D_{jk}(p)\right)} \frac{\partial \alpha_k\left(t + D_{jk}(p)\right)}{\partial h_j(t)} \\ &= A \cdot \left[1 - \left(\frac{h_j\left(t + D_{jk}(p)\right)}{A}\right)^2\right] \\ &\quad \cdot \frac{\phi}{2} \cdot \omega_{ij}^p(t') \qquad (9)\end{aligned}$$

$$\frac{\partial E_S}{\partial h_j(t)} = \begin{cases} h_j(t) - \hat{h}_j(t), & j \in MO \\ 0, & \text{others.} \end{cases} \qquad (10)$$

From these equations, it can be seen that once the output of each node is computed, the adjustment of weights can be implemented directly using the node outputs and does not need more exponential computation. The learning process is simple.

The learning coefficient γ is usually initialized a small positive value and is adjusted according to the information presented to the network. If E_S increases, γ should be adjusted to reduce the movement of weights along the gradient descent direction, namely, $\gamma = \gamma \cdot a_1$, $0 < a_1 < 1$; contrarily, if E_S decreases, the movement of weights may increase to speed up the process of learning $\gamma = \gamma \cdot a_2$, $a_2 > 1$. It is noted that the weights only are substituted by the new weights when E_S decreases. This can assure the convergence of the RPNN.

Repeat the above process until E_S is either sufficiently low or zero.

III. Chaotic Time Series Prediction Using RPNN

Chaotic time series prediction is an important investigative tool for the dynamics of a natural phenomenon. In fact, the deterministic nature of chaotic dynamics and the universal approximation capabilities of neural networks make it possible to make reliable forecasts in complex systems [7], [8].

A. Multistep Prediction

If the time series of one of the variables $x(t)(t = 1, \cdots, N)$ is available, based on the fact that the interaction between the variables is such that every component contains information on the complex dynamics of the system, a smooth function can be found to model the portraits of time series. Choosing a delay time τ [20], [21], usually, a multiple of the sampling period Δt, the method entails the construction of a series of $S = N - (n-1)\tau/\Delta t$ vectors, of dimension n, of the form

$$X(t) = \{x(t), x(t-\tau), \cdots, x\left(t-(n-1)\tau\right)\} \qquad (11)$$

where $n (n = 2, 3, \cdots)$ is called the embedding dimension, and $t = 1, \cdots, S$. In the n-dimensional space, such vectors topologically describe over time an object that is equivalent to the conjectured attractor of the physical system from which the variable $x(t)$ was measured, provided that $n \geq 2D + 1$, where D is the fractal dimension of the attractor calculated according to [22].

Theorem: Letting η be the ahead steps of prediction and $\eta\tau$ the prediction lead time, for any $n \geq 2D + 1$, there exists a smooth function defined on the reconstructed manifold in R^n to interpret the dynamics

$$x(t + \eta\tau) = F_\tau\left(X(t)\right) \qquad (12)$$

where $\eta = 1, 2, \cdots$, and $X(t)$, shown in (11), is the state of the system at time t.

Proof: According to Takens theorem [2], the one-step prediction can be defined by

$$X(t + \tau) = f_\tau^1\left(X(t)\right).$$

Then

$$\begin{aligned} X(t+2\tau) &= f_\tau^2\left(X(t+\tau)\right) = f_\tau^2\left(f_\tau^1\left(X(t)\right)\right) \\ &\vdots \\ X(t+\eta\tau) &= f_\tau^{(\eta-1)}\left(X\left(t+(\eta-1)\tau\right)\right) \\ &= f_\tau^{(\eta-1)}\left(\cdots\left(f_\tau^1\left(X(t)\right)\right)\right) \\ &= f_\tau\left(X(t)\right). \end{aligned} \qquad (13)$$

Let $g : R^n \to R^1$ be a projection defined by

$$g\left(X(t)\right) = x(t). \qquad (14)$$

Then, we have

$$\begin{aligned} x(t+\eta\tau) &= g\left(X(t+\eta\tau)\right) = g\left(f_\tau\left(X(t)\right)\right) \\ &= F_\tau\left(X(t)\right) \end{aligned}$$

where the domain of definition of F_τ is the same as the domain of definition of the f_τ^1. □

Obviously, making long-term prediction possible, a good representation $\hat{F}_\tau(\cdot)$ has to be found to approximate $F_\tau(\cdot)$. In this paper, RPNN is used as a predictor to model the unknown mapping $F_\tau(\cdot)$.

B. RPNN Model

As discussed above, the element $x(t)$ may directly influence the elements at later n sampling moments, i.e.,

$$\begin{aligned} x(t+\tau) &= F_\tau\left(\underline{x(t)}, x(t-\tau), \cdots, x(t-(n-1)\tau)\right) \\ x(t+2\tau) &= F_\tau\left(x(t+\tau), \underline{x(t)}, \cdots, x(t-(n-2)\tau)\right) \\ &\vdots \\ x(t+n\tau) &= F_\tau\left(x(t+(n-1)\tau), \right. \\ &\qquad \left. x(t+(n-2)\tau), \cdots, \underline{x(t)}\right). \end{aligned} \tag{15}$$

We use RNN to predict chaotic time series if the network structure can be designed to reflect the relations described in (15), which consequently results in that the necessary prediction information is able to be included in the network to make accurate prediction. Along this line, the RPNN is designed as follows.

1) The number of nodes can be defined as the embedding dimension n. Each node has one external input which represents the elements of $X(t)$, i.e., $x(t), x(t-\tau), \cdots, x(t-(n-1)\tau)$.

2) The number of branches is also defined as the embedding dimension n, which only exists between the nodes and themselves or the nodes and those nodes representing the instant points at latter moment. In addition, the delays on each branch could be $\tau, 2\tau, \cdots, n\tau$. In some cases, zero-delay is also considered.

From the description of RPNN in Section II-A and the above discussion, it can be seen that besides the output nodes, the outputs of other nodes also satisfy the relations of (15). This network structure can better describe the correlation of a single time series in a temporal sense.

RPNN tries to include all useful information for making long-term prediction via reasonable network structure. The output of RPNN, which means the predicted value with different lead time, can be obtained by means of the map

$$\hat{X}(t+\eta\tau) = \hat{F}_\tau\left(X(t)\right) \tag{16}$$

where

$$\begin{aligned} \hat{X}(t+\eta\tau) = [&\hat{x}(t+\eta\tau), \hat{x}\left(t+(\eta-1)\tau\right) \\ &\cdots, \hat{x}\left(t+(\eta-n+1)\tau\right)] \end{aligned}$$

and the innovation $\hat{x}(t+\eta\tau)$ is the predicting values.

For practical systems, the error between F_τ and $\hat{F}_\tau$ is unavoidable. It mainly originates from computation errors, noise, difference between the network model and the practical system, and so on. These errors may increase at exponent speed with the development of prediction lead times $\eta\tau$. Consequently, chaotic prediction can only be made during finite prediction lead times.

IV. Simulation

Modeling of sunspots is important since they indicate the relative activity of the Sun. In order to quantitatively measure the performance of prediction algorithm, prediction accuracy (PA) and root-mean-square error (RMSE) are computed [6].

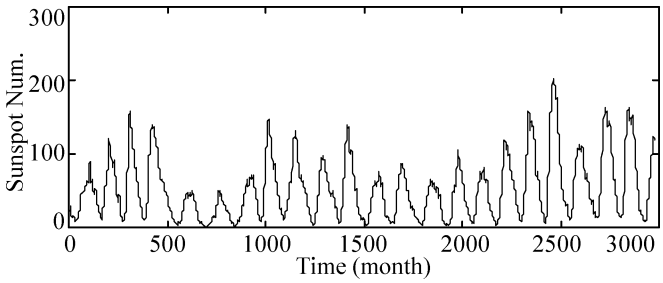

Fig. 2. Monthly sunspot time series from 1753 to 2000.

TABLE I
Simulation Condition

Initial value of ω_{ij}^p		-0.2~0.2
Number of nodes		4
Number of branches between nodes		4
$D_{ij}(p)$	between node i and j, $i \neq j$	p (p = 0,1,2,3)
	between node i and i	p+1 (p = 0,1,2,3)
Initial γ		0.001
Learning number of each prediction		1000

$$E_{\mathrm{RMSE}} = \left(\frac{1}{S-1}\sum_{t=1}^{S}\left[\hat{h}(t)-h(t)\right]^2\right)^{\frac{1}{2}} \tag{17}$$

$$E_{\mathrm{PA}} = \frac{\sum_{t=1}^{S}\left[\left(\hat{h}(t)-\hat{h}_m\right)\left(h(t)-h_m\right)\right]}{(S-1)\sigma_{\hat{h}}\sigma_h} \tag{18}$$

where $\hat{h}(t)$ is the target value, $h(t)$ is the output of network, h_m is averaged network output, $\hat{h}_m$ denotes averaged practical output, and σ_h and $\sigma_{\hat{h}}$, respectively, denote the standard deviations of the observed output and the network output. In the ideal situation, if there were no error in prediction, these parameters would indicate that $E_{\mathrm{RMSE}} = 0$ and $E_{\mathrm{PA}} = 1$.

A. Prediction of Monthly Sunspot Time Series Using RPNN

Recent studies of sunspots have revealed its chaotic behavior and qualities [23]. Fig. 2 shows the monthly mean Wolf sunspot numbers over 248 years [24], [25], totaling 2976 records. Mundt *et al.* [23] found the fractal dimension of sunspots $D \approx 2.3$. Berndtsson [26] computed $D < 2$. As discussed in Section III, assume the number of RPNN nodes is $n = 4$. In order to make any month ahead prediction, choose the time delay $\tau = 1$. That τ is small may lead to redundancy, but the use of noise-reduced data (shown in Fig. 2) can weaken this infection. Besides, it cannot influence the comparison results because different modeling methods are built based on the same data set.

Then, sunspot nonlinear system is modeled by the RPNN shown in Fig. 1, which has four nodes and has recurrent connections with four branches between the corresponding nodes. Each node has one external input that represents the sunspot number of month t, month $t-1$, month $t-2$, and month $t-3$. It is supposed that output of RPNN can be obtained from the fourth node. The simulation conditions are shown in Table I.

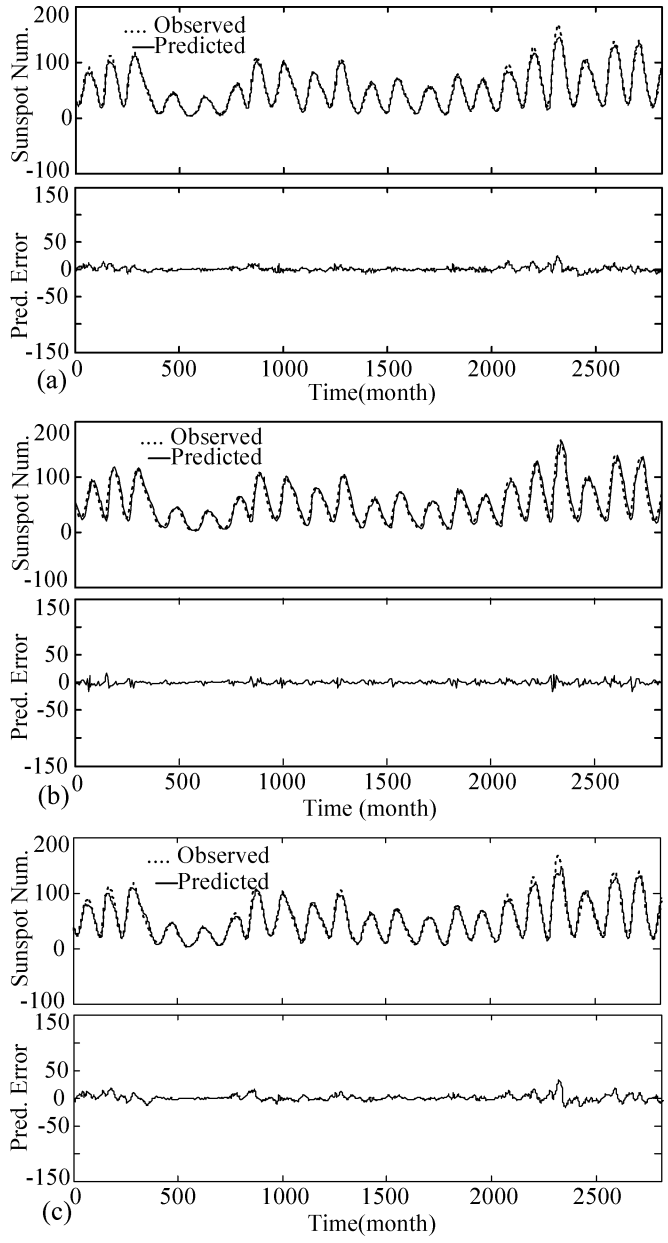

Fig. 3. Comparison of predicted error with six-month lead times between the (a) RPNN model, (b) the Kalman filter model, and (c) the ULN model.

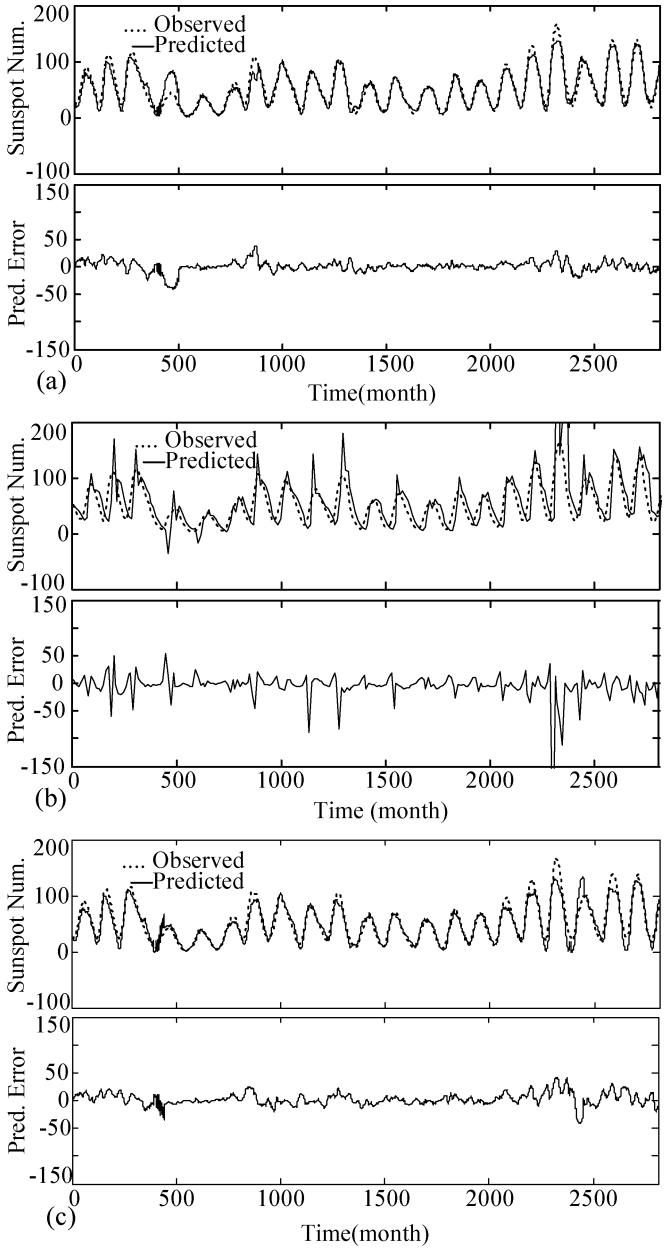

Fig. 4. Comparison of predicted error with twelve-month lead times between (a) the RPNN model, (b) the Kalman filter model, and (c) the ULN model

The prediction is mainly made up of two steps. First, before each prediction, update the learning samples with the latest 150 records of monthly sunspots and use them as the Teaching Signal to train the RPNN network. The second step is that on the basis of the trained RPNN model, the current state is presented to the network, and the multistep ahead prediction is made. Six- and 12-step, corresponding to six- and 12-month ahead predictions, with the observed values, are, respectively, shown in Figs. 3(a) and 4(a). To detect the performance of RPNN model, compare the simulation results with those of the ULN [15] and Kalman filter models [4]. The ULN model is trained at the same simulation conditions shown in Table I. The Kalman filter model is mainly built based on following procedure: 1) Select a modified form of the Rössler equation that shows similar chaotic features with those of the observed time series as the reference equation, and 2) use the reference equation as an initial state in the extended Kalman filter to estimate the structure of the governing system equations. The procedure was specially discussed in [4]. The predictions are shown in Figs. 3 and 4(b) and (c).

From Figs. 3 and 4, the predictions of Kalman filter model, the ULN model, and the RPNN model follow similar trends, that is, with the prediction lead times grows, the prediction error increases accordingly. When short-term prediction with lead times of less then six months was made, the Kalman filter follows the observed values a little more closely because of its good real-time updating capability. When prediction lead times are greater than six months, the prediction error of the Kalman filter increases remarkably. At the same time, better and more acceptable results can be reached by using the ULN and RPNN, as shown in Fig. 4(a) and (c). Therefore, RNN is a convenient tool

TABLE II
COMPARISON OF KALMAN FILTER MODEL, ULN MODEL, AND RPNN MODEL FOR MONTHLY SUNSPOTS

Lead Time	Kalman model		ULN model		RPNN model	
	E_{PA}	E_{RMSE}	E_{PA}	E_{RMSE}	E_{PA}	E_{RMSE}
6-m	0.993	3.971	0.986	5.816	0.992	4.419
10-m	0.873	22.292	0.956	10.584	0.980	7.050
15-m	0.609	55.124	0.857	19.613	0.922	13.658
20-m	0.223	228.908	0.804	22.292	0.886	16.793

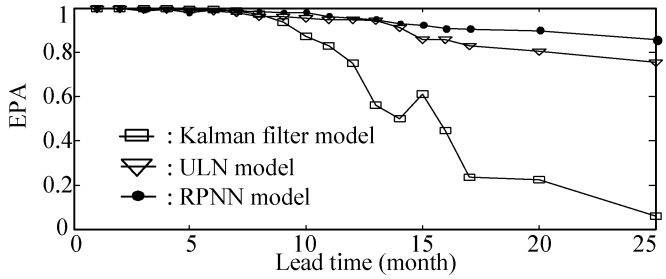

Fig. 5. E_{PA} from RPNN model, ULN model, and Kalman model between observed and predicted sunspot time series versus lead times for monthly data.

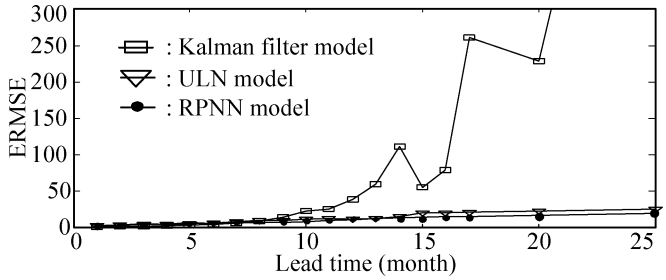

Fig. 6. E_{RMSE} from RPNN model, ULN model, and Kalman model between observed and predicted sunspot time series versus lead times for monthly data.

for approximating the values of unknown real-valued functions without regard to the mathematic equation of the system.

Quantitative measures between prediction models are shown in Table II. Fig. 5 shows the prediction accuracy E_{PA} versus the prediction lead times of three models. E_{PA} is the correlation coefficient between observed and predicted sunspot time series. It is seen that the E_{PA} is comparable for the three models at lead predictions less than six months. After that, the E_{PA} of the RPNN model reduces more slowly. From Table II, for the 10-month ahead prediction, the E_{PA} for RPNN model is 0.980, whereas the E_{PA} for the ULN model is 0.956 and, for the Kalman filter model, is 0.873. When the lead times go up to 20 months, the E_{PA} for the RPNN model is 0.886, whereas the E_{PA} for the ULN model is 0.804, and for the Kalman filter model, it is 0.223. Clearly, for larger prediction lead times, the RPNN model is a better choice.

The E_{RMSE} can be used to describe how well the model accounts for the variation in the observed data. Fig. 6 shows the E_{RMSE} versus the prediction lead times. For 20-month ahead prediction, the E_{RMSE} for RPNN model is 16.793, whereas the E_{RMSE} for the ULN model is 22.292 and, for the Kalman filter model, is 228.908. The RPNN model is better. It is seen that the two criteria from the three models are similar in trend, and their values are comparable for shorter prediction lead times, but most values of the E_{RMSE} and the E_{PA} from the RPNN model are closer to the ideal value, especially for larger prediction lead times.

As is known, the Kalman filter model is built based on an initial mathematic equation. The estimate of the state vector at time step $t+\eta\tau$ is calculated using the observation obtained at time t, which implies that it updates prediction by following only one of the neighbor trajectories near the present state. Since, with the development of prediction lead time the divergence of two initially closed trajectories increases at exponential speed, it leads to the sharply decay of prediction performance. On the other hand, RNN, such as ULN and RPNN, modeling is based on the learning of a great deal of the Teacher signal, in which the detailed information of attractor trajectories is included. To make a prediction η steps ahead, all the paths of the nearest neighbors η steps ahead can be tracked. The prediction output is obtained by an underlying relationship between the present state and its nearest neighbors. By continuously updating the learning samples and effectively studying these samples, RNN can adaptively find out the closest neighbors and furthermore adjust the relationship between the present state and its nearest neighbors so that a more accurate prediction can be made. If the learning samples are sufficient and effective, RNN can make accurate prediction. From simulations, the prediction lead times influence RPNN and ULN less seriously than they do for the Kalman filter model. Therefore, the emphasis of RNN is how to simplify the network structure and improve the network algorithm in order to enhance the learning ability and generalization ability. RPNN is constructed along this line. Its network structure reflects the relationship between state vectors more directly. In simulation process, because of simple network structure and adaptive learning rate, RPNN learns faster than ULN does. For example, for monthly sunspot numbers, for the same simulation conditions as shown in Table I, the proposed RPNN requires about 7 s of computation time per 1000 circles and about 13 s for the ULN model. The simulation results also prove that the RPNN has better performance.

Since drawing techniques are limited, we reconstruct the phase space portrait of monthly sunspot series in a three-dimensional space. Fig. 7(a) reconstructs the phase space portrait using the observed values, and Fig. 7(b) and (c) use the predicted values from the RPNN model. Comparing Fig. 7(a) with Fig. 7(b) and (c), it is obvious that the trained RPNN remembers the features of the strange attractor. The same result that prediction accuracy is reduced with the development of prediction lead times can also been seen, which is known as a typical feature of chaotic systems [27].

B. Prediction of Annual Sunspot Time Series Using RPNN

Based on above discussion, use RPNN and ULN to model annual sunspot time series, where one-step prediction represents one-year prediction. Fig. 8 shows the annual mean Wolf sunspot numbers from the years 1753 to 2000. The simulation conditions are same as those in Table I. The learning samples with the latest 110 records of annual sunspots are updated before each prediction. E_{PA} and E_{RMSE} versus the prediction lead times is shown in Figs. 9 and 10, respectively. Table III shows the comparison of the ULN model and the RPNN model. From these simulation results, it can be seen that the E_{PA} for the RPNN is generally

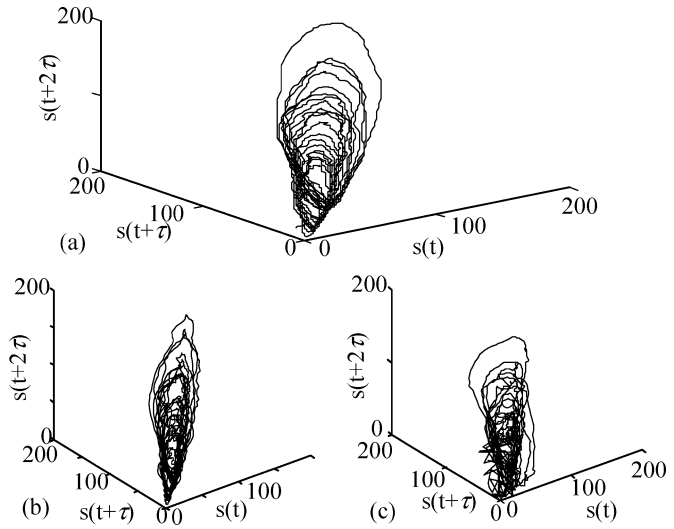

Fig. 7. Reconstructed phase space portrait of monthly sunspot time series with (a) $\tau = 10$ months from the observed values and (b) the RPNN model with six-month prediction lead time and (c) 12 -month prediction lead time.

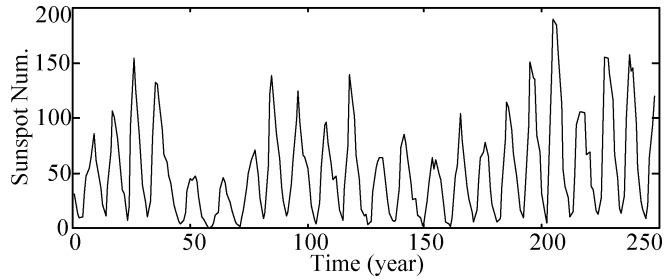

Fig. 8. Annual sunspot time series from 1753 to 2000.

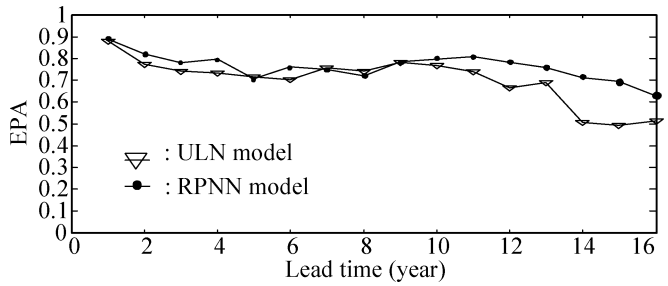

Fig. 9. E_{PA} from RPNN and ULN model between observed and predicted sunspot time series versus lead time for annual data.

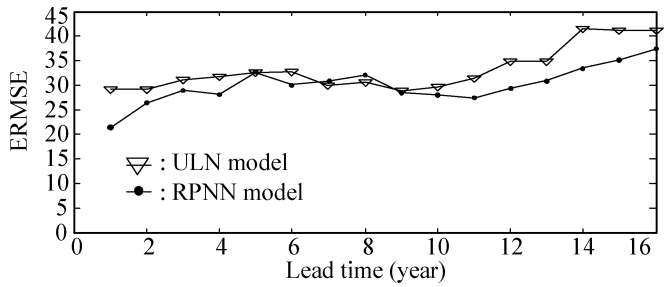

Fig. 10. E_{RMSE} from RPNN and ULN model between observed and predicted sunspot time series versus lead time for annual data.

higher than the ULN model, and the E_{RMSE} for the RPNN is generally lower than ULN model, especially for larger prediction lead times. The RPNN effectively makes long-term prediction by realizing multistep prediction. Fig. 11(a)–(c) reflect the same tendency as Fig. 7 wherein the prediction accuracy has a descending tendency with the development of prediction lead time.

TABLE III
COMPARISON OF ULN MODEL AND RPNN MODEL FOR ANNUAL SUNSPOTS

Lead time	ULN model		RPNN model	
	E_{PA}	E_{RMSE}	E_{PA}	E_{RMSE}
3-year	0.7405	31.1452	0.7806	29.0686
6-year	0.7039	32.7470	0.7609	30.0788
12-year	0.6663	34.8286	0.7820	29.2812
15-year	0.4922	41.1510	0.6942	35.1376

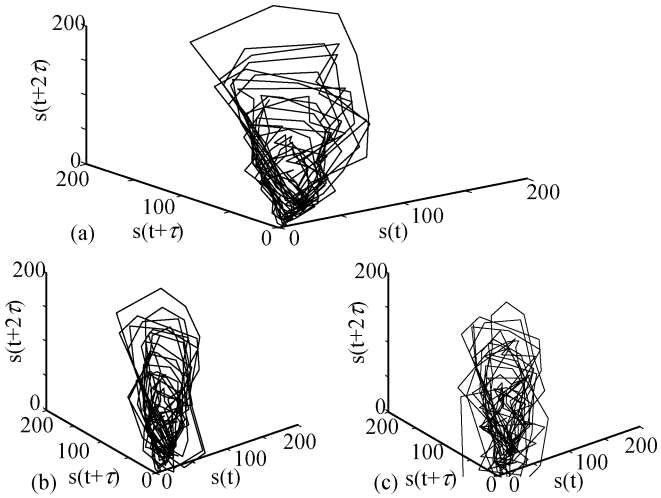

Fig. 11. Reconstructed phase space portrait of annual sunspot time series with (a) $\tau = 1$ year from the observed values, (b) RPNN model with one-year prediction lead time, and (c) three-year prediction lead time.

V. CONCLUSION

This paper describes a new methodology, based on RPNN, to model and predict chaotic time series. The RPNN structure is designed according to the reconstructing phase space, and a gradient-based and self-adaptive algorithm is applied. Prediction is made by examining trajectories on the reconstructed phase space. Most of previously published prediction methods are consistent for making repeated short-term forecasts. The method presented here tries to realize accurate multistep prediction that leads to the encompassing of the long-term information by the RPNN. Compared with prediction by Kalman filter and ULN models, the study could lead to the following conclusions.

1) The proposed RPNN model is applied to monthly sunspot time series, which possess the chaotic characteristics. Reasonable ahead predictions including multistep prediction are made by the RPNN model. Compared with the model based on the Kalman filter and the ULN, when making short-term predictions, these three models are consistent. However, with the development of prediction lead time, the output from the RPNN model follows more closely with observed values and, hence, are qualitatively better than output from other two models. The performance measures (PA and RMSE) yield similar results.

2) The ability of the proposed RPNN to identify underlying dynamics makes it a useful tool to model and predict the annual sunspot time series, in which one-step prediction denotes one-year prediction.
3) The chaotic system is highly sensitive on all kinds of interruptions from initial values of variables, parameters and noises, etc. From results of monthly sunspots using the Kalman filter model, when prediction lead time is greater than ten months, there is a sharp decay of performance indexes with no periodic trend. However, when using RPNN model, the performance indexes are impaired more slowly. That means that on the basis of strong self-learning ability, the advanced neural network methods are robust to those interruptions that may influence dynamical methods greatly. Therefore, RPNN will provide a new way to study the chaotic system.

On the other hand, there are some points that should be improved in further work, such as improving the network algorithm, enhancing the generalization ability, etc.

REFERENCES

[1] J. D. Farmer and J. J. Sidorowich, "Predicting chaotic time series," *Phys. Rev. Lett.*, vol. 59, pp. 845–848, 1987.

[2] F. Takens, "On the numerical determination of the dimension of an attractor [A]," in *Dynamical Systems and Turbulence, Lecture Notes in Mathematics. 898[C]*. Berlin, Germany: Springer-Verlag, 1981, pp. 230–241.

[3] P. Akritas, I. Antoniou, and V. V. Ivanov, "Identification and prediction of discrete chaotic maps applying a Chebyshev neural network," *Chaos, Solitons and Fractals*, vol. 11, pp. 337–344, 2000.

[4] K. Jinno, S. Xu, R. Berndtsson, A. Kawamura, and M. Matsumoto, "Prediction of sunspots using reconstructed chaotic system equations," *J. Geophys. Res.*, vol. 100, pp. 14 773–14 781, 1995.

[5] S. Xu, K. Jinno, A. Kawamura, R. Berndtsson, and J. Olsson, "Reconstructing systems from chaotic numerical time series," in *Proc. Hydraul. Eng., Jpn. Soc. Civ. Eng.*, 1993, pp. 853–856.

[6] C. Jiann-Long, I. Shafiqul, and B. Pratim, "Nonlinear dynamics of hourly ozone concentrations: nonparametric short term prediction," *Atmospheric Environment*, vol. 32, no. 11, pp. 1839–1848, 1998.

[7] R. Bakker, J. C. Schouten, and C. L. Giles *et al.*, "Learning chaotic attractors by neural networks," *Neural Comput.*, vol. 12, pp. 2355–2383, 2000.

[8] H. Leung, T. Lo, and S. Wang, "Prediction of noisy chaotic time series using an optimal radial basis function neural network," *IEEE Trans. Neural Networks*, vol. 12, pp. 1163–1172, Sept. 2001.

[9] J. T. Conner, D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE Trans. Neural Networks*, vol. 5, pp. 240–253, Mar. 1994.

[10] P. A. Mastorocostas and J. B. Theocharis, "A recurrent fuzzy-neural model for dynamic system identification," *IEEE Trans. Neural Networks*, vol. 32, pp. 176–190, Mar. 2002.

[11] R. Geqay and T. Liu, "Nonlinear modeling and prediction with feedforward and recurrent networks," *Physica D*, vol. 108, pp. 119–134, 1997.

[12] J. Wang and S. Hu, "Global asymptotic stability and global exponential stability of continuous-time recurrent neural networks," *IEEE Trans. Automatic Control*, vol. 47, pp. 802–807, May 2002.

[13] R. Boné, M. Crucianu, and J.-P. A. de Beauville, "Learning long-term dependencies by the selective addition of time-delayed connections to recurrent neural network," *Neurocomput.*, vol. 48, no. 1–4, pp. 229–250, 2002.

[14] T. Lin, B. G. Horne, P. Tino, and C. L. Giles, "Learning long-term dependencies in NARX recurrent neural networks," *IEEE Trans. Neural Networks*, vol. 7, pp. 1329–1337, Nov. 1996.

[15] K. Hirasawa, X. Wang, J. Murata, J. Hu, and C. Jin, "Universal learning network and its application to chaos control," *Neural Networks*, vol. 13, pp. 239–253, 2000.

[16] K. Hirasawa, S. Kim, J. Hu, J. Murata, M. Han, and C. Jin, "Improvement of generalization ability for identifying dynamical systems by using universal learning networks," *Neural Networks*, vol. 14, pp. 1389–1404, 2001.

[17] M. Casdagli, "Nonlinear prediction of chaotic time series," *Phys. D*, vol. 35, pp. 335–356, 1989.

[18] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, no. 2, pp. 270–280, 1989.

[19] P. J. Werbos, "Backpropagation through time: what it dose and how to do it," *Proc. IEEE*, vol. 78, pp. 1550–1560, Oct. 1990.

[20] H. D. I. Abarbanel, M. Naoki, M. I. Rabinovich, and T. Evren, "Distribution of mutual information," *Phys. Lett. A*, vol. 281, no. 5–6, pp. 368–373, 2001.

[21] M. T. Rosenstein, J. J. Collins, and C. J. De Luca, "Reconstruction expansion as a geometry-based framework for choosing proper delay time," *Physica D*, vol. 73, pp. 82–98, 1994.

[22] R. Esteller, G. Vachtsevanos, J. Echauz, and B. Litt, "A comparison of waveform fractal dimension algorithms," *IEEE Trans. Circuits Syst. I*, vol. 48, pp. 177–183, Feb. 2001.

[23] M. D. Mundt, W. B. Maguire II, and R. R. P. Chase, "Chaos in the sunspot cycle: analysis and prediction," *J. Geophys. Res.*, vol. 96, pp. 1705–1716, 1991.

[24] E. J. Chernosky and M. P. Hagan, "The Zurich sunspot number and its variations for 1700–1957," *J. Geophys. Res.*, vol. 63, pp. 755–788, 1958.

[25] L. Hong, Y. Fengmin, and Z. Yuandong, "Observation and study for the sunspot of the 22-nd solar activity cycle" (in Chinese), *J. CUN (Natural Sciences Edition)*, vol. 8, no. 2, pp. 135–139, 1999.

[26] R. Berndtsson, K. Jinno, A. Kawamura, J. Olsson, and S. Xu, "Dynamical systems theory applied to long-term temperature and precipitation time series," in *Trends in Hydrology*, J. Menon, Ed. Trivandrum, India, 1994, pp. 291–297.

[27] G. Sugihara and R. M. May, "Nonlinear forecasting as a way of distinguishing chaos from measurement error in time series," *Nature*, vol. 344, pp. 734–741, 1990.

Min Han (M'02) was born in Beijing, China, in 1959. She received the B.S. and M.S degrees from Department of Electrical Engineering, Dalian University of Technology, Dalian, China, in 1982 and 1993, respectively, and the M.S and Ph.D. degrees from Kyushu University, Kyushu, Japan, in 1996 and 1999, respectively.

She is a Professor with the School of Electronic and Information Engineering, Dalian University of Technology. Her current research interests are neural networks, chaos, and their applications to control and identification.

Jianhui Xi was born in Guizhou, China, in 1975. She received the B.S. degree from the Department of Automation, Beijing University of Aeronautics and Astronautics, Beijing, China, in 1996 and the M.S. degree from Dalian University of Technology, Dalian, China, in 2002, where she is currently a doctoral student.

Her current research interests are chaos theory and neural networks.

Shiguo Xu was born in Shandong, China, in 1958. He received the B.S., M.S., and Ph.D. degrees from Dalian University of Technology, Dalian, China, in 1982, 1984, and 1989, respectively.

He is a professor with the School of Civil and Hydraulic Engineering, Dalian University of Technology, Dalian, China, From 1992 to 1995, he studied with the Engineering Faculty, Kyushu University, Kyushu, Japan. His research interests have been expanded to the use of floodwater resources, water issues of wetlands, and water environment restoration.

Fu-Liang Yin was born in Liaoning, China, in 1962. He received the B.S. and M.S. degrees in electronic engineering from Dalian University of Technology, Dalian, China, in 1984 and 1987, respectively.

From 1991 to 1994, he was an Associate Professor at Dalian University of Technology, where he has been a Professor since 1995 and the dean of the School of Electronic and Information Engineering since 2000. His research interests include theories and applications of digital signal processing, speech processing, speech processing and pattern recognition, and digital communication.