Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 3542898, 9 pages
http://dx.doi.org/10.1155/2016/3542898

Research Article

A New Approach for Chaotic Time Series Prediction Using Recurrent Neural Network

Qinghai Li and Rui-Chang Lin

Department of Electronic Engineering, Zhejiang Industry and Trade Vocational College, East Road 717, Wenzhou 325003, China

Correspondence should be addressed to Rui-Chang Lin; rc921.lin@qq.com

Received 4 July 2016; Accepted 31 October 2016

Academic Editor: Lijun Zhu

A self-constructing fuzzy neural network (SCFNN) has been successfully used for chaotic time series prediction in the literature. In this paper, we propose the strategy of adding a recurrent path in each node of the hidden layer of SCFNN, resulting in a self-constructing recurrent fuzzy neural network (SCRFNN). This novel network does not increase complexity in fuzzy inference or learning process. Specifically, the structure learning is based on partition of the input space, and the parameter learning is based on the supervised gradient descent method using a delta adaptation law. This novel network can also be applied for chaotic time series prediction including Logistic and Henon time series. More significantly, it features rapider convergence and higher prediction accuracy.

1. Introduction

A chaotic time series can be expressed as a deterministic dynamical system that however is usually unknown or incompletely understood. Therefore, it becomes important to make prediction from the experimental observation of a real system. The technology was widely studied in many science and engineering fields such as mathematical finance, weather forecasting, and intelligent transport and trajectory forecasting. The early research on nonlinear prediction of chaotic time series can be found in [1]. Most recent work mainly focuses on different methods for improving the prediction performance, such as fuzzy neural networks [2, 3], RBF [4, 5], recurrent neural networks [6, 7], back-propagation recurrent neural networks [8], predictive consensus networks [9, 10], and biologically inspired neuronal network [11, 12].

Both fuzzy logic and artificial neural networks are potentially suitable for nonlinear chaotic series prediction as they can perform complex mappings between their input and output spaces. In particular, the self-constructing fuzzy neural network (SCFNN) [13] is capable of constructing a simple network without the need of knowledge to the chaotic series. This capability is due to SCFNN's ability of self-adjusting the location of the input space fuzzy partition, so there is no need to estimate in advance the series states distribution. Moreover, carefully setting conditions on the increasing demand of fuzzy rules makes the architecture of the constructed SCFNN fairly simple. These advantages motivated researchers to build various chaotic series prediction algorithms by employing an SCFNN structure in, for example, [14, 15].

Neural network has wide applications in various areas; see, for example, [16–21]. In particular, recurrent neural network (RNN) has been proved successful in speech processing and adaptive channel equalization. One of the most important features of RNN is its feedback paths in the circuit that makes it have a sequential rather than a combinational behavior. RNNs were applied not only in the processing of time-varying patterns or data sequences but also in dealing with the dissonance of input pattern when the possibly different outputs are generated by the same set of input patterns due to the feedback paths. Since RNN is a highly nonlinear dynamical system that exhibits rich and complex behaviors, it is expected that RNN possesses better performance than traditional signal processing techniques in modeling and predicting chaotic time series. Some works on improving the performance of chaotic series prediction using RNNs can be found in, for example, [6, 7].

From the above observation, it is a novel idea to combine the SCFNN and RNN techniques for chaotic series prediction, which results in a new architecture called self-constructing recurrent fuzzy neural network (SCRFNN) in this paper. The structure learning and parameter learning algorithms in SCRFNN are inherited from those in SCFNN, which maintains the simplicity in implementation. Nevertheless, the recurrent path in SCRFNN makes it more complex in function deviation and exhibit richer behaviors. Extensive numerical simulation shows that both SCFNN and SCRFNN are effective in predicting chaotic time series including Logistic series and Henon series. But the latter has superior performance in convergence rate and prediction accuracy at the cost of slightly heavier structure (number of hidden nodes) and fuzzy logic rules.

It is noted that a similar neural network structure has been used for nonlinear channel equalizers in [20]. The purpose of an equalizer in, for example, wireless communication systems, is to recover the transmitted sequence or its delayed version using a trained neural network. But the chaotic time series prediction studied in this paper has a completely different mechanism. Chaotic time series prediction is the problem of developing a dynamic model by the observed time series for a nonlinear chaotic system that exhibits deterministic behavior with a known initial condition. A neural network is used to build chaotic time series predication after the so-called phase space reconstruction that is not touched in channel equalizers.

The rest of this paper is organized as follows. In Section 2, the problem of chaotic series prediction is formulated. The structure, the inference output, and the learning algorithms of SCRFNN are described in Section 3. Numerical simulation results on two classes of Logistic and Henon chaotic series predictions are presented in Section 4. Finally, the paper is concluded in Section 5.

2. Background and Preliminaries

The phase space reconstruction theory is commonly used for chaotic time series prediction; see, for example, [22, 23]. The main idea is to find a way for phase space reconstruction from time series and then conduct prediction in phase space. The theory is briefly revisited in this section followed by a general neural network prediction model.

Delay embedding is the main technique for phase space reconstruction. Assume a chaotic time series is represented as $x_1, x_2, \ldots, x_n$ and the phase space vector

$$X_i = \left(x_i, x_{i+\tau}, \ldots, x_{i+(m-1)\tau}\right)^T, \quad i = 1, 2, \ldots, M. \tag{1}$$

The parameter τ is the delay, m the embedding dimension, and $M = n - (m-1)\tau$ the number of phase vectors. A prediction model contains an attractor that warps the observed data, in the phase space, and provides precise information about the dynamics involved. Therefore, it can be used to predict X_{i+1} from X_i in phase space and hence synthesize x_{i+1} using time series reconstruction. The procedure can be summarized as a model with input X_i and output x_{i+1}.

Takens' theorem provides the conditions under which a smooth attractor can be reconstructed from the observations made with a generic function. The theorem states that, if the embedding dimension $m \geq 2d + 1$, where d is the dimension of the system dynamics, then the phase space constituted by the original system state variable and the dynamic behaviors in the one-dimensional sequence of observations are equivalent. It is equivalent because the chaotic attractor differential in the two spaces is homeomorphism. The reconstructed system that includes the evolution information of all state variables is capable of calculating the future state of the system based on its current state, which provides a basic mechanism for chaotic time series prediction.

As explained before, the prediction model with input X_i and output x_{i+1} provides precise information of the nonlinear dynamics under consideration. Neural network (NN) is an appropriate structure to build a nonlinear model of chaotic time series predication. Specifically, a typical three-layer NN is discussed below.

When we apply a three-layer NN to predict a chaotic time series, better prediction performance can be achieved if the number of neurons of the input layer is equal to the embedding dimension of the phase space reconstructed by the chaotic time series. Specifically, let the number of neurons of the input layer be m, that of the hidden layer be p, and that of output layer be 1. Then, the NN describes a mapping $f : \mathbb{R}^m \to \mathbb{R}^1$.

The input for the nodes of the hidden layer is

$$S_j = \sum_{i=1}^{m} \omega_{ij} x_i - \theta_j, \quad j = 1, 2, \ldots, p, \tag{2}$$

where ω_{ij} is the link weight from the input layer to the hidden layer and θ_j is the threshold. Assume the Sigmoid transfer function $f(s) = 1/(1+e^{-s})$ is used for the NN. Then the output of the hidden layer nodes is

$$b_j = \frac{1}{1 + \exp\left(-\sum_{i=1}^{m} \omega_{ij} x_i + \theta_j\right)}, \quad j = 1, 2, \ldots, p. \tag{3}$$

Similarly, the input and the output of the output layer nodes are

$$\begin{aligned} L &= \sum_{j=1}^{p} V_j b_j - \gamma, \\ x_{i+1} &= \frac{1}{1 + \exp\left(-\sum_{j=1}^{p} V_j b_j + \gamma\right)}, \end{aligned} \tag{4}$$

respectively. Here V_j is the link weight from the hidden layer to the output layer and γ is the threshold.

In general, the aforementioned link weights (ω_{ij}, V_j) and the thresholds (θ_j, γ) of the NN can be randomly initialized, for example, within $[0, 1]$. Then, they can be properly trained such that the resulting NN has the capability of precise prediction. The specific training strategy depends on the specific architecture of the NN, which is the main scope of this research line and will be elaborated in the next section.

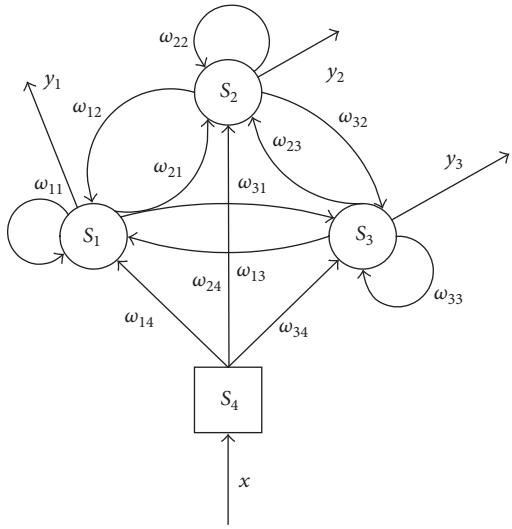

Figure 1: Structure of a recurrent neural network.

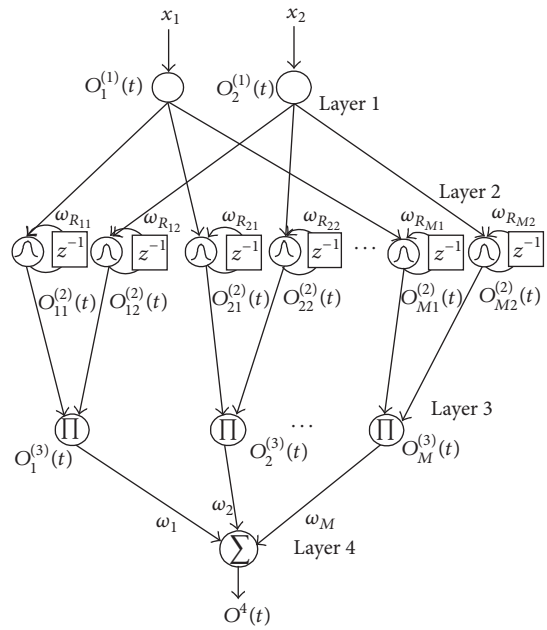

Figure 2: Schematic diagram of SCRFNN.

3. Self-Constructing Recurrent Fuzzy Neural Network

Following the general description of the NN prediction model introduced in the previous section, we aim to propose a specific architecture with the features of fuzzy logic, recurrent path, and self-constructing ability, called a self-constructing recurrent fuzzy neural network (SCRFNN). With these features, the SCRFNN is able to exhibit rapid convergence and high prediction accuracy.

3.1. Background of SCRFNN. The network structure of a traditional fuzzy neural network (FNN) is determined in advance. During the learning process, the structure is fixed and a supervised back-propagation algorithm is applied to adjust the membership function parameters and the weighting coefficients. Such an FNN with a fixed structure usually needs a large number of hidden layer nodes for acceptable performance, which significantly increases the system complexity [24, 25].

To overcome the aforementioned drawback of a fixed structure, the SCRFNN used in this paper exploits a two-phase learning algorithm that includes both structure learning and parameter learning. Specifically, the structure of fuzzy rules is determined in the first phase and the coefficients of each rule are tuned in the second one. It is conceptually easy to sequentially carry out the two phases. However, it is suitable only for offline operations with a large amount of representative data collected in advance [26]. Moreover, independent realization of structure and parameter learning is time-consuming. These disadvantages can be eliminated in SCRFNN as the two phases of structure learning and parameter learning are conducted concurrently [27].

The main feature of the proposed SCRFNN is the novel recurrent path in the circuit. The schematic diagram of a recurrent neural network (RNN) is shown in Figure 1 [28], where, for example, x and y_k, $k = 1, 2, 3$, denote the external input and the unit outputs, respectively. The dynamics of such a structure can be described by, for $k = 1, 2, 3$,

$$s_k[t+1] = \sum_{l=1}^{n} \omega_{kl}[t]\, y_l[t] + \sum_{l=1}^{m} \omega_{k,l+n}[t]\, x_l^{\text{net}}[t],$$
$$y_k[t+1] = f\left(s_k[t+1]\right), \tag{5}$$

where m and n denote the numbers of external inputs and hidden layer units, respectively, $\omega_{ij}[t]$ is the connection weight from the jth unit to the ith unit at the time instant t, and the activation function $f(\cdot)$ can be any real differentiable function. Clearly, the output of each unit depends on the previous external inputs to the network as well as the previous outputs of all units. The training algorithms for the recurrent parameters of RNNs have been well studied, for example, the real-time recurrent learning (RTRL) algorithm [29].

3.2. Structure and Inference Output of SCRFNN. The schematic diagram of SCRFNN is shown in Figure 2. The fuzzy logic rule and the functions in each layer are briefly described as follows.

The fuzzy logic rule adopted in the SCRFNN has the following form:

$$\begin{aligned} R_j:\ & \text{If } \left(x_1 + \omega_{R_{j1}} O_{j1}^{(2)}(t-1)\right) \text{ is } A_{j1}, \\ & \text{and } \left(x_2 + \omega_{R_{j2}} O_{j2}^{(2)}(t-1)\right) \text{ is } A_{j2}, \ldots, \\ & \text{and } \left(x_n + \omega_{R_{jn}} O_{jn}^{(2)}(t-1)\right) \text{ is } A_{jn}. \\ & \text{Then } y(t) = \omega_j \end{aligned} \tag{6}$$

with n input variables and a constant consequence ω_j. In (6), A_{ji} is the linguistic term of the precondition part with

membership function $\mu_{A_{ji}}$, x_i's and y denote the input and output variables, respectively, and $\omega_{R_{ji}}$ and $O_{ji}^{(2)}(t-1)$ represent the recurrent coefficient and the last state output of the jth term associated with the ith input variable.

The nodes in Layer 1, called the input nodes, simply pass the input signals to the next layer. There are two input variables x_1 and x_2 and two corresponding outputs $O_1^{(1)}(t) = x_1$ and $O_2^{(1)}(t) = x_2$ for the problem considered in this paper.

Each node in Layer 2 acts as a linguistic label for the input variables from Layer 1. Let $O_{ji}^{(2)}(t)$ be the output of the jth rule associated with the ith input node in status t and $\omega_{R_{ji}}$'s the recurrent coefficients. Then, the Gaussian membership function is determined in this layer as follows:

$$O_{ji}^{(2)}(t) = \exp\left(-\frac{\left(O_i^{(1)}(t) + \omega_{R_{ji}} O_{ji}^{(2)}(t-1) - m_{ji}\right)^2}{\sigma_{ji}^2}\right) \quad (7)$$

with the mean m_{ji} and the standard deviation σ_{ji}.

Each node in Layer 3, represented by the product symbol Π, works as the precondition part of the fuzzy logic rule. Specifically, the output of the jth rule node in status t is thus expressed as

$$O_j^{(3)}(t) = \prod_i O_{ji}^{(2)}(t). \quad (8)$$

The single node in Layer 4, called the output node, acts as a defuzzifier. As a result, the final output of the SCRFNN is

$$O^{(4)}(t) = \sum_j \omega_j O_j^{(3)}(t), \quad (9)$$

where the link weight ω_j is the output action strength associated with the jth rule.

3.3. Online Learning Algorithm for SCRFNN. As discussed in Section 3.1, a two-phase learning algorithm for structure learning and parameter learning is used for SCRFNN. The initial SCRFNN is composed of the input and output nodes only. The membership and the rule nodes are dynamically generated and adjusted according to the online data by performing the structure and parameter learning processes. These two phases are explained below.

The structure learning algorithm aims to find the proper input space fuzzy partitions and fuzzy logic rules with minimal fuzzy sets and rules. As the initial SCRFNN contains no membership or rule node, the main work in structure learning is to decide whether it is necessary to add a new membership function node in Layer 2 and the associated fuzzy logic rule in Layer 3. The criterion for generating a new fuzzy rule for new incoming data is based on the firing strengths $O_j^{(3)}(t) = \prod_i O_{ji}^{(2)}(t)$, $j = 1, \ldots, M$, where M is the number of existing rules.

Specifically, if the maximum degree $\mu_{\max} = \max_{1 \le j \le M} O_j^{(3)}(t)$ is not larger than a prespecified threshold parameter $\mu_{\min}$, a new membership function needs to be generated. Also, the mean value and the standard deviation of the new membership function are, respectively, assigned as $m_i^{\text{new}} = x_i$ and $\sigma_i^{\text{new}} = \sigma_i$, where x_i is the new incoming data and σ_i is an empirical prespecified constant. The value $\mu_{\min}$ is initially chosen between 0 and 1 and then keeps decaying in order to limit the growing size of the SCRFNN structure.

The parameter learning algorithm aims to minimize a predefined energy function by adaptively adjusting the vector of network parameters based on a given set of input-output pairs. The particular energy function used in the SCRFNN is as follows:

$$E = \frac{1}{2}\left(O^d - O^{(4)}(t)\right)^2, \quad (10)$$

where O^d is the desired output associated with the input pattern and $O^{(4)}(t)$ is the inferred output. The vector is adjusted along the negative gradient of the energy function with respect to the vector. In the four-layer SCRFNN, a back-propagation learning rule is adopted as the gradient vector is calculated in the direction opposite to the data flow, as described below.

First, the link weight ω_j for the output node in Layer 4 is updated along the negative gradient of the energy function; that is,

$$\Delta\omega_j = \frac{\partial E}{\partial \omega_j} = -\left(O^d - O^{(4)}(t)\right) O_j^{(3)}(t). \quad (11)$$

In particular, it is updated according to

$$\omega_j(k+1) = \omega_j(k) - \eta_\omega \Delta\omega_j, \quad (12)$$

where k is the pattern number of the jth link and the factor η_ω is the learning-rate parameter.

Next, in Layer 3, the mean m_{ji} and the standard deviation σ_{ji} of the membership functions are updated by

$$\begin{aligned} m_{ji}(k+1) &= m_{ji}(k) - \eta_m \Delta m_{ji}, \\ \sigma_{ji}(k+1) &= \sigma_{ji}(k) - \eta_\sigma \Delta\sigma_{ji}, \end{aligned} \quad (13)$$

with the learning-rate parameters η_m and η_σ. The terms Δm_{ji} and $\Delta\sigma_{ji}$ are also calculated as the gradient of the energy function as follows:

$$\Delta m_{ji} = \frac{\partial E}{\partial m_{ji}} = -\left(O^d - O^{(4)}(t)\right)\omega_j O_j^{(3)}(t) \cdot \frac{2\left(O_i^{(1)}(t) + \omega_{R_{ji}} O_{ji}^{(2)}(t-1) - m_{ji}\right)}{\sigma_{ji}^2},$$
$$\Delta\sigma_{ji} = \frac{\partial E}{\partial \sigma_{ji}} = -\left(O^d - O^{(4)}(t)\right)\omega_j O_j^{(3)}(t) \cdot \frac{2\left(O_i^{(1)}(t) + \omega_{R_{ji}} O_{ji}^{(2)}(t-1) - m_{ji}\right)^2}{\sigma_{ji}^3}. \quad (14)$$

Finally, the variation of recurrent coefficient $\Delta\omega_{R_{ji}}$ in Layer 2 is updated by

$$\omega_{R_{ji}}(k+1) = \omega_{R_{ji}}(k) - \eta_R \Delta\omega_{R_{ji}}, \quad (15)$$

where

$$\Delta\omega_{R_{ji}} = \frac{\partial E}{\partial \omega_{R_{ji}}} = \left(O^d - O^{(4)}(t)\right)\omega_j \left(\prod_{i,i\neq j} O_{ji}^{(2)}(t)\right) \cdot \frac{2\left(O_i^{(1)}(t) + \omega_{R_{ji}} O_{ji}^{(2)}(t-1) - m_{ji}\right)^2}{\sigma_{ji}^3} O_{ji}^{(2)}(t-1) \quad (16)$$

is again the gradient of the energy function.

4. Simulation Results

In order to evaluate the effectiveness of the proposed SCRFNN, we apply it on two benchmark chaotic time series data sets: Logistic series and Henon series. The number of data used for each benchmark problem is 2000. In particular, we use the first 1000 data for training and the remaining 1000 for validation. It will be shown that both SCFNN and SCRFNN are effective in predicting Logistic series and Henon series. But the latter has superior performance in convergence rate and prediction accuracy at the cost of slightly heavier structure (number of hidden nodes) and rules.

4.1. Logistic Chaotic Series. A Logistic chaotic series is generated by the following equation:

$$x_{k+1} = \mu x_k(1 - x_k), \quad x_k \in [0,1], \ \mu \in [0,4]. \quad (17)$$

In particular, the parameter μ is restricted within the range of $3.57 < \mu \leq 4$ for chaotic behavior.

The training performance for both SCFNN and SCRFNN is demonstrated in Figure 3, where the profiles with 5, 20, and 50 learning cycles are compared in three graphs, respectively. It is observed that the proposed learning algorithm is effective in terms of fast convergence. In all the three cases, the convergence rate for the SCRFNN is faster than that for the SCFNN.

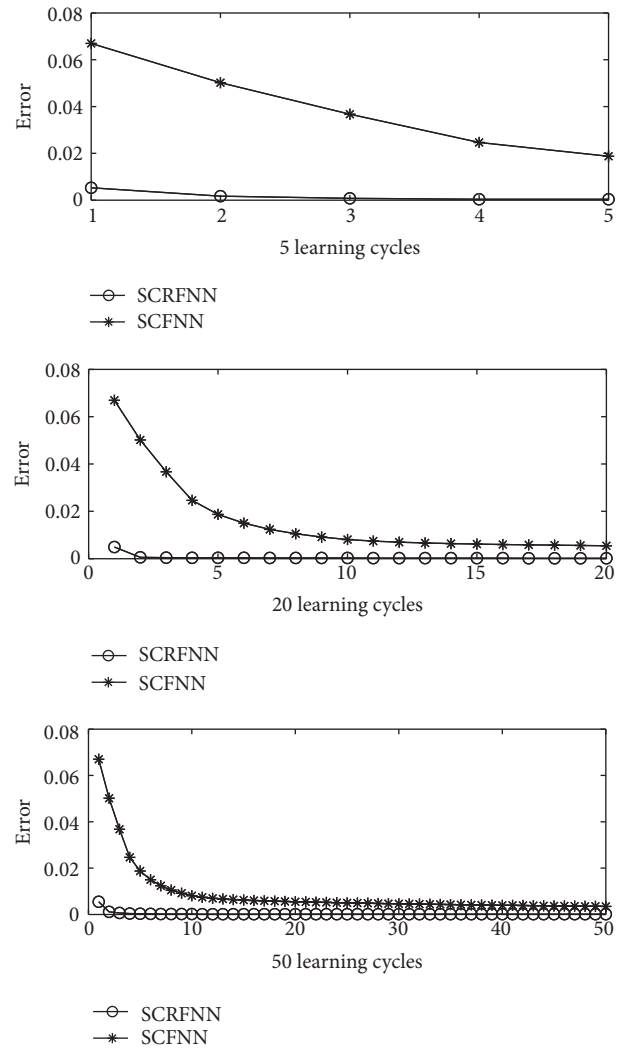

Figure 3: Comparison of convergence for 5, 20, and 50 learning cycles.

Root mean squared error (RMSE) is the main error metrics for describing the training errors. It is explicitly defined as follows:

$$\text{RMSE} = \left(\frac{1}{S-1}\sum_{i=1}^{S}[P_i - Q_i]^2\right)^{1/2}, \quad (18)$$

where S is the number of trained patterns and P_i and Q_i are the FNN derived values and the actual chaotic time series data, respectively. Table 1 shows that the RMSEs of SCFNN and SCRFNN completed 5, 20, and 50 learning cycles. In all the three cases, SCRFNN results in smaller RMSEs than SCFNN.

For the well-trained SCFNN and SCRFNN, the prediction performance is compared and shown in Figures 4 and 5. It is observed in Figure 4 that the predicted outputs from the SCFNN and the SCRFNN well match the real data. More

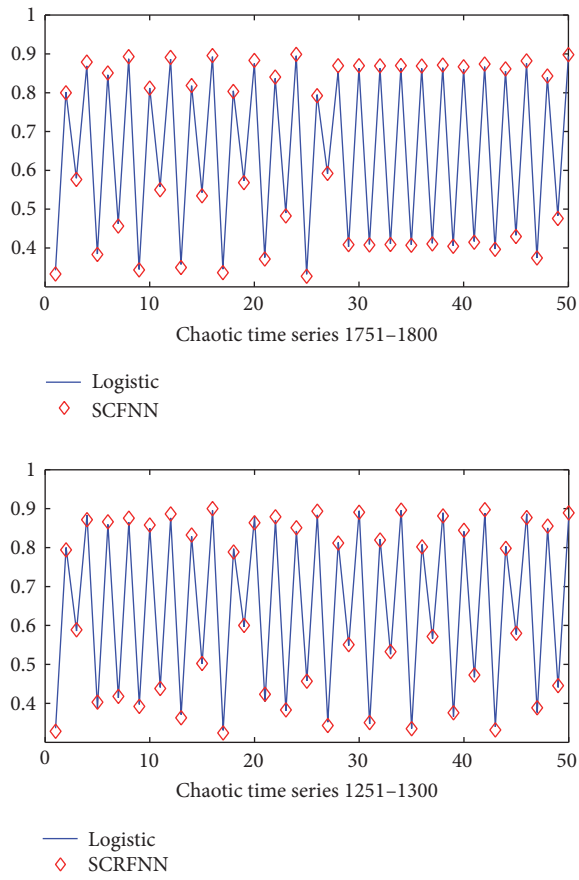

Figure 4: Logistic series and the predictions with SCFNN and SCRFNN.

Table 1: RMSE for SCFNN and SCRFNN completed 5, 20, and 50 cycles.

FNN (learning cycles)	Logistic	Henon
SCFNN (5)	7.65499e − 004	0.0015
SCRFNN (5)	4.51469e − 004	0.0014
SCFNN (20)	5.0209e − 004	0.0011
SCRFNN (20)	4.2344e − 004	6.3094e − 004
SCFNN (50)	4.7764e − 004	7.8083e − 004
SCRFNN (50)	4.189e − 004	4.4918e − 004

Table 2: Number of hidden nodes in SCFNN and SCRFNN completed 5, 20, and 50 cycles.

FNN (learning cycles)	Logistic	Henon
SCFNN (5)	4	7
SCRFNN (5)	5	7
SCFNN (20)	4	7
SCRFNN (20)	6	8
SCFNN (50)	4	7
SCRFNN (50)	5	10

specifically, the prediction errors are less significant in the SCRFNN than those in the SCFNN, as shown in Figure 5.

Finally, Table 2 shows that the numbers of hidden nodes in SCFNN and SCRFNN completed 5, 20, and 50 learning cycles. The SCRFNN has slightly more hidden nodes in all the three cases. This heavier structure of SCRFNN, together with the extra rules for recurrent path, is the cost for the aforementioned performance improvement.

4.2. Henon Chaotic Series. Henon mapping was proposed by the French astronomer Michel Henon for studying globular clusters [30]. It is one of the most famous simple dynamical systems with wide applications. In recent years, Henon mapping has been well studied in chaos theory. Its dynamics are given as follows:

$$x(k+1) = 1 + y_k - ax^2(k),$$
$$y(k+1) = bx(k). \tag{19}$$

For example, chaos is produced with $a = 1.4$ and $b = 0.3$.

Similar simulation is conducted for Henon series with the corresponding results shown in Figure 6 for the comparison of convergence rates in 5, 20, and 50 learning cycles, in Figure 7 for the comparison of prediction performance, and in Figure 8 for the prediction errors. RMSEs and the numbers of hidden nodes for SCFNN and SCRFNN are also listed in Tables 1 and 2, respectively.

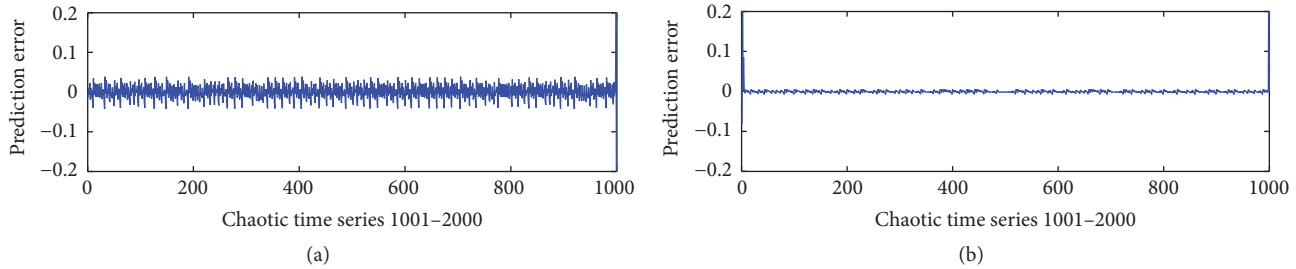

Figure 5: The prediction errors for well-trained SCFNN (a) and SCRFNN (b).

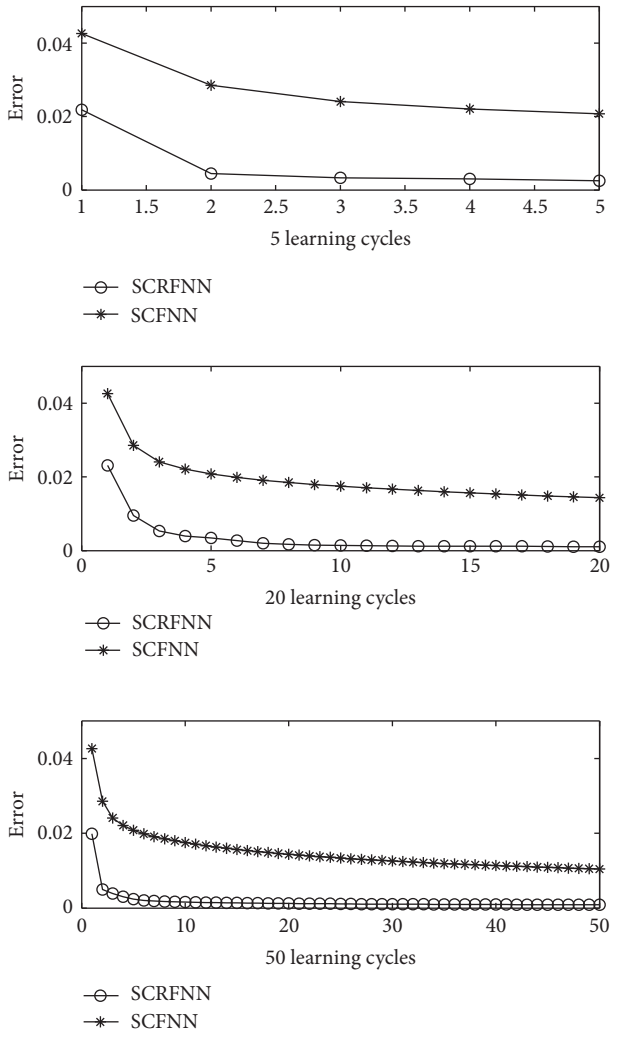

Figure 6: Comparison of convergence for 5, 20, and 50 learning cycles.

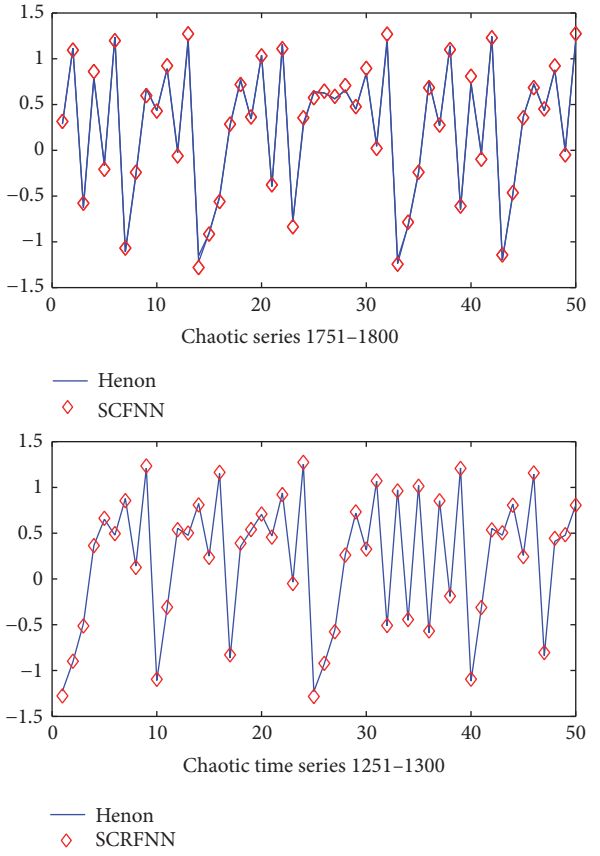

Figure 7: Henon series and the predictions with SCFNN and SCRFNN.

5. Conclusions

A novel type of network architecture called SCRFNN has been proposed in this paper for chaotic time series prediction. It inherits the practically implementable algorithms, the self-constructing ability, and the fuzzy logic rule from the existing SCFNN. Also, it brings new recurrent path in each node of the hidden layer of SCFNN. Two numerical studies have demonstrated that SCRFNN has superior performance in convergence rate and prediction accuracy than the existing SCFNN, at the cost of slightly heavier structure (number of hidden nodes) and extra rules for recurrent path. Hardware implementation of the proposed SCRFNN will be interesting for future research.

Competing Interests

The authors declare that they have no competing interests.

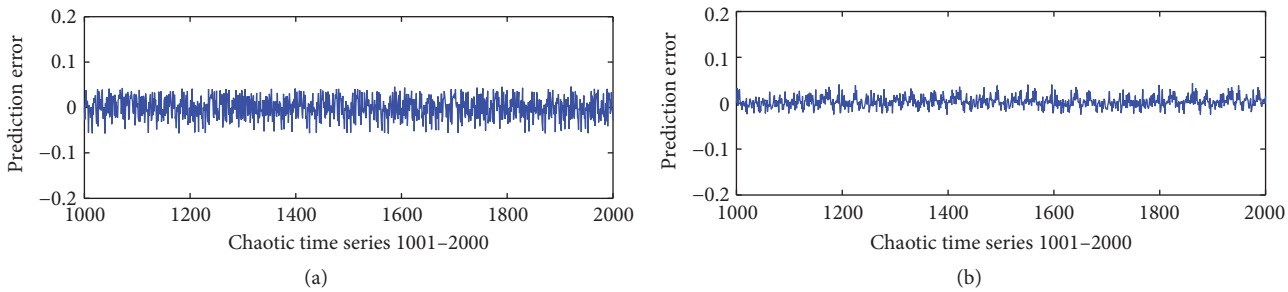

Figure 8: The prediction errors for well-trained SCFNN (a) and SCRFNN (b).

References

[1] M. Casdagli, "Nonlinear prediction of chaotic time series," *Physica D: Nonlinear Phenomena*, vol. 35, no. 3, pp. 335–356, 1989.

[2] J. Zhang, H. S.-H. Chung, and W.-L. Lo, "Chaotic time series prediction using a neuro-fuzzy system with time-delay coordinates," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 7, pp. 956–964, 2008.

[3] Z. Chen, C. Lu, W. Zhang, and X. Du, "A chaotic time series prediction method based on fuzzy neural network and its application," in *Proceedings of the 3rd International Workshop on Chaos-Fractals Theories and Applications (IWCFTA '10)*, pp. 355–359, October 2010.

[4] K. Wu and T. Wang, "Prediction of chaotic time series based on RBF neural network optimization," *Computer Engineering*, vol. 39, pp. 208–216, 2013.

[5] J. Xi and H. Wang, "Modeling and prediction of multivariate series based on an improved RBF network," *Information and Control*, vol. 41, no. 2, pp. 159–164, 2012.

[6] Q.-L. Ma, Q.-L. Zheng, H. Peng, T.-W. Zhong, and L.-Q. Xu, "Chaotic time series prediction based on evolving recurrent neural networks," in *Proceedings of the 6th International Conference on Machine Learning and Cybernetics (ICMLC '07)*, vol. 6, pp. 3496–3500, IEEE, Hong Kong, August 2007.

[7] R. Chandra and M. Zhang, "Cooperative coevolution of Elman recurrent neural networks for chaotic time series prediction," *Neurocomputing*, vol. 86, pp. 116–123, 2012.

[8] Y. Zuo and T. Wang, "Prediction for chaotic time series of BP neural network based on differential evolution," *Computer Engineering and Applications*, vol. 49, pp. 160–163, 2013.

[9] H.-T. Zhang and Z. Chen, "Consensus acceleration in a class of predictive networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 10, pp. 1921–1927, 2014.

[10] Z. Peng, D. Wang, Z. Chen, X. Hu, and W. Lan, "Adaptive dynamic surface control for formations of autonomous surface vehicles with uncertain dynamics," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 2, pp. 513–520, 2013.

[11] Z. Chen and T. Iwasaki, "Circulant synthesis of central pattern generators with application to control of rectifier systems," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 273–286, 2008.

[12] L. Zhu, Z. Chen, and T. Iwasaki, "Oscillation, orientation, and locomotion of underactuated multilink mechanical systems," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 5, pp. 1537–1548, 2013.

[13] F.-J. Lin, C.-H. Lin, and P.-H. Shen, "Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 5, pp. 751–759, 2001.

[14] W.-D. Weng, R.-C. Lin, and C.-T. A. Hsueh, "The design of an SCFNN based nonlinear channel equalizer," *Journal of Information Science and Engineering*, vol. 21, no. 4, pp. 695–709, 2005.

[15] Y. Li, Q. Peng, and B. Hu, "Remote controller design of networked control systems based on self-constructing fuzzy neural network," in *Proceedings of the 2nd International Symposium on Neural Networks: Advances in Neural Networks (ISNN '05)*, pp. 143–149, Chongqing, China, June 2005.

[16] Z. Chen and T. Iwasaki, "Matrix perturbation analysis for weakly coupled oscillators," *Systems and Control Letters*, vol. 58, no. 2, pp. 148–154, 2009.

[17] Z. Chen, T. Iwasaki, and L. Zhu, "Neural control for coordinated natural oscillation patterns," *Systems and Control Letters*, vol. 62, no. 8, pp. 693–698, 2013.

[18] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 3, pp. 328–339, 1989.

[19] G. Kechriotis, E. Zervas, and E. Manolakos, "Using recurrent neural networks for blind equalization of linear and nonlinear communications channels," in *Proceedings of the Military Communications Conference (MILCOM '92) Conference Record. Communications-Fusing Command, Control and Intelligence*, pp. 784–788, IEEE, 1992.

[20] R.-C. Lin, W.-D. Weng, and C.-T. Hsueh, "Design of an SCRFNN-based nonlinear channel equaliser," *IEE Proceedings—Communications*, vol. 152, no. 6, p. 771, 2005.

[21] G. Kechriotis and E. S. Manolakos, "Using recurrent neural network for nonlinear and chaotic signal processing," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP '93)*, pp. 465–469, Minneapolis, Minn, USA, April 1993.

[22] H. Kantz and T. Schreiber, *Nonlinear Time Series Analysis*, Cambridge University Press, New York, NY, USA, 1997.

[23] A. Basharat and M. Shah, "Time series prediction by chaotic modeling of nonlinear dynamical systems," in *Proceedings of the IEEE 12th International Conference on Computer Vision*, pp. 1941–1948, September 2009.

[24] F.-J. Lin, R.-F. Fung, and R.-J. Wai, "Comparison of sliding-mode and fuzzy neural network control for motor-toggle servomechanism," *IEEE/ASME Transactions on Mechatronics*, vol. 3, no. 4, pp. 302–318, 1998.

[25] F.-J. Lin, W.-J. Hwang, and R.-J. Wai, "A supervisory fuzzy neural network control system for tracking periodic inputs," *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 1, pp. 41–52, 1999.

[26] C.-T. Lin, "A neural fuzzy control system with structure and parameter learning," *Fuzzy Sets and Systems*, vol. 70, no. 2-3, pp. 183–212, 1995.

[27] F.-J. Lin, R.-J. Wai, and C.-C. Lee, "Fuzzy neural network position controller for ultrasonic motor drive using push-pull DC-DC converter," *IEE Proceedings: Control Theory and Applications*, vol. 146, no. 1, pp. 99–107, 1999.

[28] G. Kechriotis, E. Zervas, and E. S. Manolakos, "Using recurrent neural networks for adaptive communication channel equalization," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 267–278, 1994.

[29] 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.

[30] M. Hénon, "A two-dimensional mapping with a strange attractor," *Communications in Mathematical Physics*, vol. 50, no. 1, pp. 69–77, 1976.