Exploring the Neural State Space Learning from One-Dimension Chaotic Time Series

Zhiwei Shi, Min Han and Jianhui Xi
School of Electronic and Information Engineering,
Dalian University of Technology, Liaoning, 116023, China
E-mail: minhan@dlut.edu.cn

Abstract— **Because the chaotic system is initial condition sensitive, it is difficult to decide a proper initial state for a recurrent neural network to model observed one-dimension chaotic time series. In this paper, a recurrent neural network with feedback composed of internal state is introduced to model one-dimension chaotic time series. The neural network output is a nonlinear combination of the internal state variable. To successfully model a chaotic time series, this paper proves that the recurrent neural network with internal state can start from arbitrary initial state. In the simulation, the neural systems perform multi-step ahead prediction, also, the reconstructed neural state space is compared with the original state space, and largest LEs(Lyapunov Exponents) of the two systems are calculated and compared to see if the two system has similar chaotic invariant.**

Index Terms—**Recurrent Neural Network, Chaotic Time Series, State Space**

I. INTRODUCTION

In the last decade, neural network is widely used in pattern recognition, system identification and system control, and it has been proved that neural network method is an effective method to solve problems of complex nonlinear system, which usually has no analytical approach to work on. Chaotic time series is usually a sequence observed from a complex nonlinear dynamic system. Since nonlinear dynamic systems abound in the real-life and engineering, it is important to analyze system dynamics from the observed time series, so that one can reconstruct the state space and make a prediction about the future. The most common state space reconstruction method in the analysis of chaotic time series is the method of delays. By the analysis of the geometry of the embedded data one can discover the relationship between past and future points in a time series[1].

Feed-forward neural network method is introduced into the analysis of the chaotic time series, and the task is to let the neural network learn a set of time series, and then make a prediction, feed-forward neural network technique usually rely on method of delay coordinate embedding, for examples, by using embedding dimension and delay time as a network structure parameter[2][3][4].

Recurrent neural network is dynamic system, which performs dynamic mapping task. Recent results indicate that such kind of networks is suitable to model time series sampled from chaotic system, for examples, ESN(Echo State Network), ANSS(Autonomous Neural State Space) model and LRNN(Local Recurrent Neural Network), RNN method does not directly rely on the estimation of embedding dimension as the feed-forward neural network counterpart does, and also the time lag is not necessary to be estimated. ESN[5][6] is a kind of recurrent neural network with a huge reservoir as the internal state variable, which has no the usual difficulties of network training as the traditional recurrent neural network. However, the internal state variable in the ESN is complex, and it is difficult to analysis the state variable of the neural dynamics learning from an observed time series. ANSS[7] and LRNN[8] use all the state information carefully from the original dynamic system, and it is difficult to apply to the usual case, in which only a scalar time series is available and it seems not easy to determine a proper initial condition for RNN.

In this paper, an autonomous neural network with an internal state variable is to built to model the usual scalar time series, and the interests are put on the evolution of the state variable, it is proved that the neural network performance is not sensitive to initial condition when modeling a chaotic system, that is to say, the initial state of the recurrent neural network can be chosen freely. Also, the reconstructed neural state space is compared with the original state space, from the simulation it can be seen that the attractor shape of neural system well trained is similar to that of original dynamic system.

In Section II, we will give a brief introduction of the chaotic time series and state space network model. Section III is state transformation of the neural state space, which serve as a theoretic foundation to use the recurrent neural network, and the explanation to use any arbitrary initial state. Section IV gives the results of simulation study, we will discuss the state evolution of the neural network, and the neural state space is compared with the original state space, iterative

0-7803-8812-7/05/$20.00 ©2005 IEEE

predictions are made to show the network performance, and one of chaotic invariant—Largest LEs(Lyapunov Exponents) are calculated to compare the two systems. In section V we will give the conclusions.

II. Chaotic Time Series and State Space Neural Network

State equation of deterministic dynamical systems describes the state evolution of a system. They can be expressed for example by ordinary differential equations:

$$\dot{\mathbf{X}}(t) = \mathbf{A}(\dot{\mathbf{X}}(t), \mathbf{X}(0)) \tag{1}$$

or by discrete maps with the form:

$$\mathbf{X}(k+1) = \mathbf{F}(\mathbf{X}(k), \mathbf{X}(0)) \tag{2}$$

A time series can be thought of as an observed sequence $\{y(k), k = 0,1,2,...\}$, which is performed with some measurement scalar function $y(k) = h(\mathbf{X}(k))$. Since the time series in itself does not properly represent the phase space of the dynamic system, one has to employ some technique to unfold the multidimensional structure using the available data.

The phase space reconstruction of a dynamic system from the knowledge of a single scalar variable is one of the fundamental problems of the nonlinear time series analysis. In a system with complex dynamics, the variables interact with each other, so each component contains information on the system. Consequently, even by only one of the system observed time series, it is possible to reconstruct a space which is topologically equivalent to the real phase space.

Delay coordinate embedding is popular method to the reconstruction, and one can find a delay vector $\mathbf{S}(k) = [y(k)\ y(k+\tau)\ y(k+2\tau)\dots y(k+(m-1)\tau)]$, with proper embedding dimension m and delay time τ, so that the observed time can be unfolded to a new state space which is topologically equivalent to the original system and therefore contains the dynamic information about real system.

There are many existing recurrent neural networks in the literature, among them, a kind of state space neural network can be described as follows[9-12]:

$$\begin{bmatrix} \hat{\mathbf{X}}(k+1) \\ \hat{\mathbf{Y}}(k) \end{bmatrix} = \mathbf{W}_1 \cdot \boldsymbol{\sigma}(\mathbf{W}_2 \cdot \hat{\mathbf{X}}(k) + \mathbf{W}_3 \cdot \mathbf{U}(k) + \boldsymbol{\theta}) \tag{3}$$

in which $\boldsymbol{\sigma}(\cdot)$ is a sigmoid function, $\hat{\mathbf{X}}(k)$ and $\hat{\mathbf{Y}}(k)$ serve as state variable and output variable respectively, and $\mathbf{U}(k)$ is an input variable if the system is excited by some external force. $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$ and $\boldsymbol{\theta}$ are the connection weights and bias in the network, which can be trained by several algorithms, for example, gradient descent methods.

Chaotic time series is generated from an autonomous system, so there is an autonomous neural system, as shown in Figure 1:

$$\begin{bmatrix} \hat{\mathbf{X}}(k+1) \\ \hat{y}(k) \end{bmatrix} = \mathbf{W}_1 \cdot \boldsymbol{\sigma}(\mathbf{W}_2 \cdot \hat{\mathbf{X}}(k) + \boldsymbol{\theta}) \tag{4}$$

where $\{\hat{y}(k), k = 0,1,2,...\}$ is an one-dimension time series.

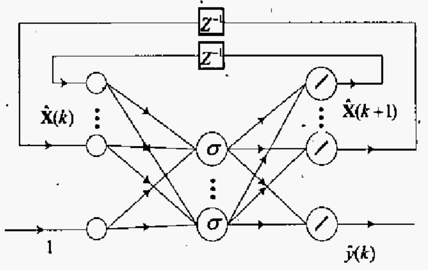

Figure 1 Structure of Autonomous Neural System

It is clear that a time series $\{\hat{y}(k), k = 0,1,2,...,Q\}$ can be produced by equation(4) with some initial state $\hat{\mathbf{X}}(0)$. For a given chaotic time series $\{y(k), k = 0,1,2,...\}$, one hope to let the neural network(4) learn from it. In fact, network training is carried out to minimize the error between $\hat{y}(k)$ and the desired system output is $y(k)$, for k=0, 1, 2, ... , Q, and then the neural state space (4) is compared with the original state space(1) or (2).

III. The State Transformation of the Neural State Space

A key element of deterministic chaos is the sensitive dependence of the trajectory on the initial conditions, which means that two such systems with however small a difference in their initial state eventually will end up with a big difference between their states. However, it is usually difficult to know the initial state of the equation (1) or (2), from which the time series is produced, so many difficulties appear when learning from a time series using neural network (4). However, the hesitation could be avoided when determine the initial state if one considered the following result. In fact, one can choose any initial condition.

For the given dynamic system as described as follows:

$$\begin{aligned} \mathbf{X}(k+1) &= \mathbf{F}(\mathbf{X}(k), \mathbf{X}(0)) \\ y(k) &= h(\mathbf{X}(k)) \end{aligned} \tag{5}$$

where $\mathbf{F}(\cdot)$ is vector continuous function, and $h(\cdot)$ is scalar continuous function. There exists another dynamic system (6):

$$\begin{aligned} \tilde{\mathbf{X}}(k+1) &= \tilde{\mathbf{F}}(\tilde{\mathbf{X}}(k), \tilde{\mathbf{X}}(0)) \\ \tilde{y}(k) &= \tilde{h}(\tilde{\mathbf{X}}(k)) \end{aligned} \tag{6}$$

where $\tilde{\mathbf{F}}(\cdot)$ is another vector continuous function, and $\tilde{h}(\cdot)$ is another scalar continuous function. $\tilde{\mathbf{X}}(k) = \mathbf{G}^{-1}(\mathbf{X}(k))$, where $\mathbf{G}(\cdot)$ is a bijective invertible continuous function that maps one space to the other, so that:

$y(k) = \tilde{y}(k)$, for all k.

Proof: From (5), we have:

$$\begin{aligned} \tilde{\mathbf{X}}(k+1) &= \mathbf{G}^{-1}(\mathbf{X}(k+1)) \\ &= \mathbf{G}^{-1}(\mathbf{F}(\mathbf{X}(k), \mathbf{X}(0))) \\ &= \mathbf{G}^{-1}(\mathbf{F}(\mathbf{G}(\tilde{\mathbf{X}}(k)), \mathbf{G}(\tilde{\mathbf{X}}(0))) \end{aligned}$$

and

$$y(k) = h(\mathbf{X}(k)) = h(\mathbf{G}(\tilde{\mathbf{X}}(k))$$

so we can find a vector continuous function $\tilde{\mathbf{F}}(\tilde{\mathbf{X}}(k), \tilde{\mathbf{X}}(0)) = \mathbf{G}^{-1}(\mathbf{F}(\mathbf{G}(\tilde{\mathbf{X}}(k)), \mathbf{G}(\tilde{\mathbf{X}}(0)))$ an a scalar continuous function $\tilde{h}(\tilde{\mathbf{X}}(k)) = h(\mathbf{G}(\tilde{\mathbf{X}}(k))$, so that $y(k) = \tilde{y}(k)$, for all k.

Remark: Actually, the network training is carried to find a proper continuous mapping between two spaces, and in the neural state space one can specify the initial state freely. If such a bijective invertible continuous function $\tilde{\mathbf{X}}(k) = \mathbf{G}^{-1}(\mathbf{X}(k))$ is found, in fact, two deterministic systems with identical initial conditions are found. While $y(k) = \tilde{y}(k)$ can not be ensured when one uses neural network(for example equation 4), in neural network, one usually uses the term: $\|y(k) - \tilde{y}(k)\| < \varepsilon$, for all the $0 \le k \le Q$, where Q is the number of the training examples. That is to say, one can find neural network (6) to approximate the system (5) with arbitrary accuracy, if one chooses proper network scale and learning algorithm, together with sufficient training examples.

IV. Simulation

The state space in (6) is a reconstructed state space from an observed one-dimension time series, and it is different from the reconstructed state space by using delay coordinate method, so it is called reconstructed neural state space. Two issues will be addressed on the reconstructed neural state space: the one is the evolution of state variable and the other is the prediction ability of the neural system after the neural network is trained by an observed time series from state equation.

Typically, two simulation examples are investigated, the one is Rossler system and the other is Lorenz system. For each system, an observed time series is presented to the neural network (4) for training, and the neural network tries to learn a segment of the time series(training set), and then makes an iterative prediction from the beginning of the training time series, additional prediction is also made after the tail of the training time series presented, so that one can view the attractor reconstructed clearly and check the prediction ability of the autonomous neural system.

A. Rossler Attractor

In the Rossler equation(7), chaotic character appears if the parameters *a, b* and *c* are properly chosen. A trajectory has been generated for the initial condition [-2.2, 1.1, 3.3] by using Runge-Kutta method with fixed step-length(0.1), and in this simulation let *a*=0.2, *b*=0.2 and *c*=10.

The number of hidden layer neurons is 8, and so the total number of adjustable parameters is 64. The initial state of the neural network is set to [0.1, 0.1, 0.1], and this is for the example and not limited to this value. The first 300 data points are used for trajectory learning until the error on the training sequence drops to a satisfying level.

$$\begin{aligned} \dot{x} &= -(y+z) \\ \dot{y} &= x + a \cdot y \\ \dot{z} &= b + z \cdot (x-c) \end{aligned} \tag{7}$$

When the network is well trained, the reconstructed neural state spaces are plotted in the same coordination as the original Rossler state space. There are three successful stochastic trainings(a-c) with the initial weights selected at random, and the three reconstructed neural state spaces are shown from the Figure 2 to 4. In these figures, the original state space is scaled down to 1:100 so that the comparisons could be clear and obvious.

All these neural systems are starting from the same initial state[0.1, 0.1, 0.1], which is different from the initial state variable of original rossler attractor[-2.2, 1.1, 3.3]. The reconstructed neural state spaces have some common characters, for example, similarity in the attractor shape.

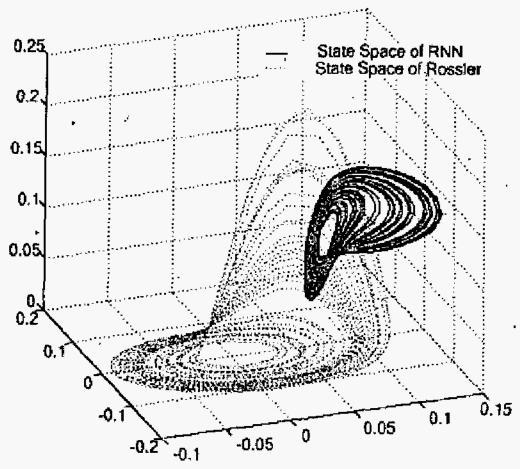

Figure 2 The reconstructed neural state space (a) and original state space(be scaled to 1:100)

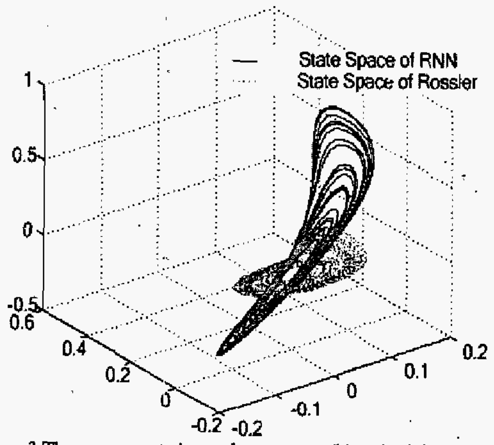

Figure 3 The reconstructed neural state space (b) and original state space(be scaled to 1:100)

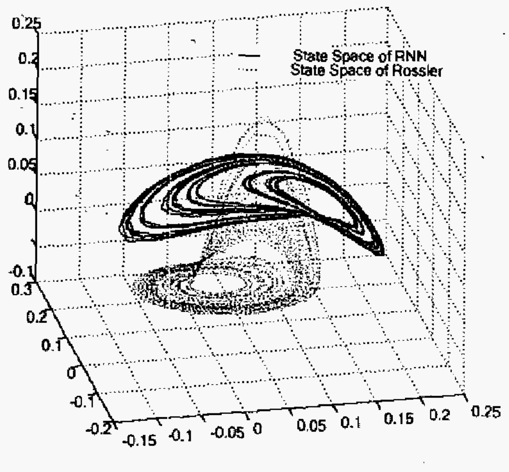

Figure 4 The reconstructed neural state space (c) and original state space(be scaled to 1:100)

From Figure 2 to 4, one can find 4 completely different dynamic systems, which have similar outputs. In fact, we can find more such dynamic systems which are produced by network training. The produced systems have close relationships with the dynamic system being learned. From the results of section III, we can say that there is an approximate nonlinear transformation between state variables of each produced system and the state variables of the original system.

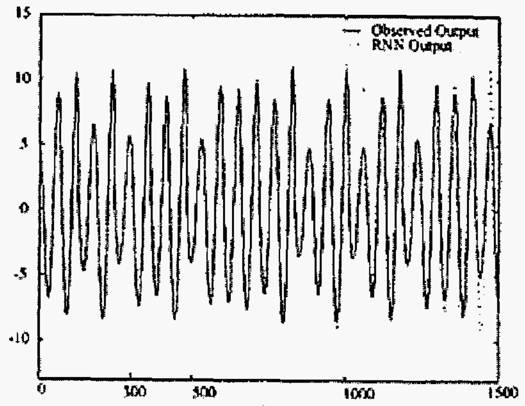

Figure 5 The comparison of the x-component from Rossler attractor and neural network output

Table 1 The comparison of the Largest LEs of the time series between Rossler system and the RNN

Rossler system	RNN
0.0714	0.0751

The sum squared error on the training set is 1.10e-4 as shown in Figure 5 after a successful training. To check the prediction ability of the RNN, the RNN is run continuously until time step 1500, as given in Figure 5. The RNN make a wonderful prediction on the following testing set, and accurate prediction is still available to time step 1000. It is also interesting to check if some chaotic invariant is the same between the two systems. The Largest LE of the Rossler system is 0.0714[14], and the largest LE of the neural system is calculated by program Tisean[13](as shown in Table 1), and it can be seen that the LE of RNN is close to that of Rossler system, and it can be seen that the LE of RNN is close to that of Rossler system.

B. Lorenz Attractor

In the Lorenz equation(8), chaotic character appears if the parameters *a*, *b* and *c* are properly chosen. A trajectory has been generated for the initial condition [13, -13, 33] by using Runge-Kutta method with fixed step-length(0.02), and in this simulation let a=10, b=28 and c=8/3.

The number of hidden layer neurons is 10, and so the total number of adjustable parameters is 80. The initial state of the neural network is set to [0.1, 0.1, 0.1], and this is for the example and not limited to this value. The first 300 data points are used for trajectory learning until the error on the training sequence drops to a satisfying level.

$$\begin{aligned} \dot{x} &= -a(y-x) \\ \dot{y} &= (b-z)x-y \\ \dot{z} &= xy-cz \end{aligned} \qquad (8)$$

When the network is well trained, the reconstructed neural state spaces are plotted in the same coordination with the original Lorenz state space. There are three successful stochastic trainings(d-f), and the initial weights are selected at random. From the Figure 6 to 8, it can be seen that the three neural state spaces reconstructed by RNN are similar with the original state space. Though the RNN started learning from different initial weights, it could learn the chaotic character well. In these figures, the original state space is scaled down to 1:50 so that the comparisons could be clear and obvious.

All these neural systems are starting from the same initial state[0.1, 0.1, 0.1], which is different from the initial state variable of original Lorenz attractor[13, -13, 33].

As the attractor is plotted, a strand is drawn from initial state, and will start weaving the outline of the two butterfly wings. The attractor will continue weaving back and forth between the two wings. The neural state space as shown in the figure also has similar geometrical shape and spreads out like the original attractor.

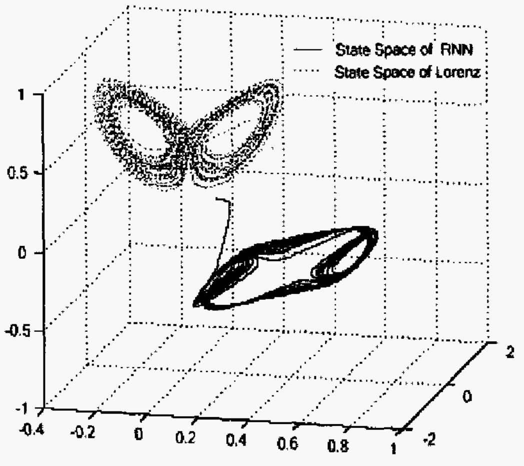

Figure 6 The reconstructed neural state space(d) and original Lorenz state space(be scaled to 1:50)

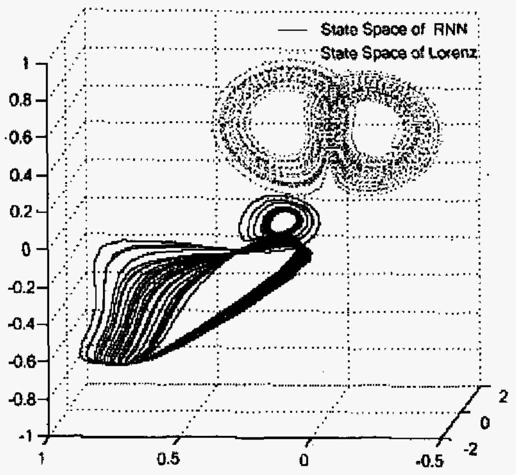

Figure 7 The reconstructed neural state space(e) and original Lorenz state space(be scaled to 1:50)

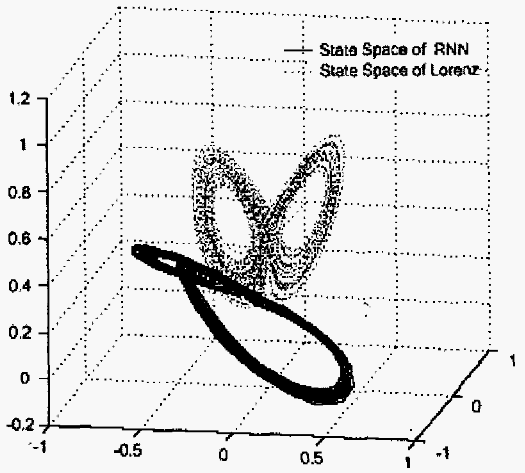

Figure 8 The reconstructed neural state space(f) and original state space(be scaled to 1:50)

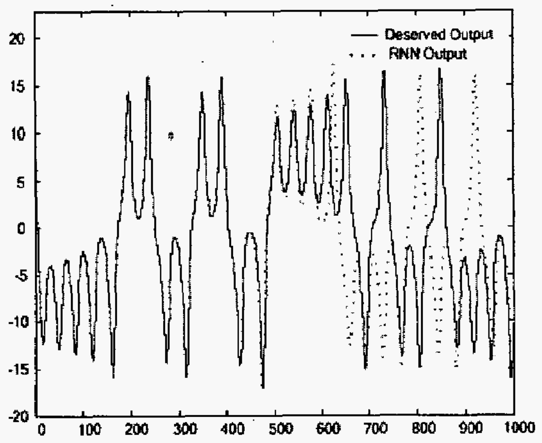

Figure 9 The comparison of the x-component from Lorenz attractor and neural network output

The sum squared error on the training set is 2.53e-3 as shown in Figure 9 after a successful training. To check the prediction ability of the RNN, the RNN is run continuously until time step 1000, as given in Figure 9. The RNN make a wonderful prediction on the following testing set, and accurate prediction is still available to time step 500. The Largest LE of Lorenz system is 0.906[15], and the largest LE of the neural system is calculated(calculated by program Tisean[13]), and it can be seen that the LE of RNN is close to that of Lorenz system(as shown in Table 2).

Table 2 The comparison of the largest LEs of the time series between Lorenz system and the RNN

Lorenz system	RNN
0.906	1.002

From the two simulation cases, it can be seen that the neural network learning from Lorenz system does not make a wonderful job as the one learning from Rossler time series. Obviously, it can be explained from the largest LE of the system, Lorenz system has a bigger largest LE than Rossler system and relatively hard to prediction. The prediction is still influenced by chaotic character, and the reason is in model error rather than the difference of initial condition, since the neural network is trained by finite training examples, and the training set does not contain all the information in the original attractor, so the neural network model does not equal the original attractor precisely. That is to say, better results will be obtained if the RNN can learn more data points and has an optimal network structure.

V. CONCLUSIONS

To learn from a scalar chaotic time series, one can use a neural dynamic system with different state space, so that one can choose the initial state freely. Learning from a scalar time series, an autonomous neural network with arbitrary initial state performs a satisfactory prediction performance.

The reconstructed neural state space is compared with the original state space, the reconstructed Rossler and Lorenz attractors look much like the original state space, and the largest lyapunov exponents of the two systems are calculated and this chaotic invariant is close to that of original system. The prediction is still influenced by chaotic character, and the reason is in model error rather than the difference of initial condition

ACKNOWLEDGMENT

This research is supported by the project (60374064) of the National Nature Science Foundation of China, and this support is appreciated.

REFERENCES

[1] H. D. I. Abarbanel, R. Brown and J. J. Sidorowich, "The analysis of observed chaotic data in physical systems," *Rev. Mod. Phys.*, Vol. 65, pp.1331-1392, 1993.

[2] R.J. Frank, N. Davey and S.P. Hunt, "Time series prediction and neural networks," *Journal of Intelligent and Robotic Systems: Theory and Applications*, Vol. 31, No. 1-3, pp.91-103, 2001.

[3] S. Tronci, M. Giona, and R. Baratti, "Reconstruction of chaotic time series by neural models: A case study," Vol. 55, pp. 581, 2003.

[4] K. A. de Oliveira, A. Vannucci, and E. C. da Silva, "Using artificial neural networks to forecast chaotic time series," Vol. 284, pp. 393, 2000.

[5] H. Jaeger, and H. Haas, "Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication," *Science*, Vol.304, No.5667, pp.78-80, 2004.

[6] H. Jaeger, "Adaptive Nonlinear System Identification with Echo State Networks," in Advances in Neural Information Processing Systems, S. Becker, S. Thrun, and K. Obermayer, Eds. Cambridge, MA: MIT Press, 2003, pp. 593-600.

[7] J.A.K. Suykens, and J. Vandewalle, "Learning a simple recurrent neural state space model to behave like Chua's double scroll," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, Vol.42, No.8, pp.499-502, 1995.

[8] S. Cincotti, M. Marchesi, and F. Pilo, "Learning of Chua's circuit attractors by locally recurrent neural networks," *Chaos, Solitons and Fractals*, Vol.12, No.11, pp.2109-2115, 2001.

[9] J.M. Zamarreno, and P. Vega. "State space neural network. Properties and application," *Neural Networks*, Vol. 11, No.6, pp. 1099-1112, 1998.

[10] J.A.K Suykens, B.L.R. De Moor and J. Vandewalle, "Nonlinear system identification using neural state space models, applicable to robust control design," *International Journal of Control*, Vol. 62, No.1, pp. 129-152, 1995.

[11] M. Han, Z. W. Shi, W. Wang, "Modeling Dynamic System by Recurrent Neural Network with State Variables", Lecture Notes in Computer Science, Vol.3174, pp.200-205, 2004.

[12] C. Alippi, and V. Piuri, "Neural modeling of dynamic systems with nonmeasurable state variables," IEEE Transactions on Instrumentation and Measurement. Vol.48, No.6, pp.1073-1080, 1999.

[13] R. Hegger, H. Kantz, and T. Schreiber, Practical implementation of nonlinear time series methods: The TISEAN package, *Chaos*, Vol. 9, No. 2, pp.413-435, 1999

[14] O.E. Rössler, "An Equation for Continuous Chaos", *Phys.Lett. A*, Vol. 57, No.5, pp.397-398, 1976

[15] E. N.Lorenz, "Deterministic Nonperiodic Flow." *Journal of the Atmospheric Sciences*. Vol.20, pp.130-141, 1963.