Recurrent NN Model for Chaotic Time Series Prediction

Jun Zhang, K. S. Tang and K F. Man
Department of Electronic Engineering
City University of Hong Kong
Tat Chee Avenue, Kowloon, Hong Kong
Tel: (852) 2788-7783
Fax: (852) 2788-7791
e-mail: eekstang@cityu.edu.hk

Abstract ----- A new Elman neural network learning algorithm is proposed for chaotic time series prediction. This method has a number of advantages over the use of a standard Back-Propagation (BP) algorithm. It is not only its capability for handling a much higher complexity time data series, but its superiority in time convergence can prove to be a valuable asset for time critical application. Furthermore, this method is also very accurate in prediction as it can reach global minimum in a much attainable manner.

keywords : Recurrent Neural Network, Chaos, Time series model, Training algorithms.

I. INTRODUCTION

The aim of time series prediction is to accurately predict the evolution of a system's behavior. However, due to certain unpredictable dynamics as well as its nonlinearity, the trend to apply non-linear time series prediction and analysis has drawn much attention recently. With the development of the advanced computer technology, artificial neural networks (ANN) has become a viable alternative for achieving the required time series prediction.

The technique of Recurrent Neural Network (RNN) is a specific type of ANN, of which the activity pattern passes through the network more than once before it generates an output. In this way, unlike the static feed-forward networks, RNN exhibits some intrinsic dynamics which is capable of performing more complex computations than static feed-forward networks [1-5]. This phenomenon can be further reinforced by introducing Elman net [6] for the standard BP training algorithm. This paper is to describe the composite training algorithm and at the same time to demonstrate its capability in predicting some chaotic time series.

II. TIME SERIES MODELS

One of the most important research areas of time series analysis is to deal with the problem of "predicting" a future value of a series, given a set of observations of its past information. Consider that the given record $X_t, X_{t-1}, .. X_{t-n}$, on a discrete time process, and wishes to predict the value of $X_{t+m}(m>0)$. The predictor, $\hat{X}_{t+m}$, will be computed based on the past history. This can be expressed as

$$\hat{X}_{t+m} = \theta(X_t, X_{t-1}, .. X_{t-n}) \qquad (1)$$

for a function θ. The problem is to choose a suitable θ so that $\hat{X}_{t+m}$ approaches closely to X_{t+m}. A measure by means of mean-square error is generally applied which takes the form:

$$\varepsilon(m) = E[\{X_{t+m} - \hat{X}_{t+m}\}^2] \qquad (2)$$

then the problem is now reduced to the selection of θ so that ε is therefore minimized. In practice, θ takes the following three forms:

- If θ is a linear function, then (1) is the linear time series model;
- If θ is a non-linear function ,then (1) is the non-linear time series model; and
- If θ is a chaotic function, then (1) is a chaotic time series.

III. ELMAN NEURAL NETWORKS ARCHITECTURE AND TRAINING

Elman[7] introduced a simple RNN, as shown in Fig.1

$$a_1(t) = F[\sum w_1^R(t-1)\cdot a_1(t-1) + w_1(t-1)\cdot p(t)] \qquad (3)$$

$$a_2(t) = \sum w_2(t-1)\cdot a_1(t) + b_2 \qquad (4)$$

0-7803-3932-0

where $F[\bullet]$ is a sigmoid function; the external input to the network is represented by $p(t)$; $a_2(t)$ is Elman RNN output ; w_1 and w_2 are the weights . w_1^R is the weight of recurrent connection and b_2 is the bias.

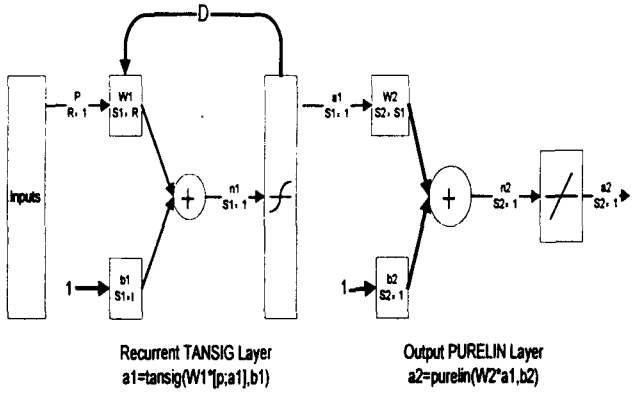

Fig.1. The Structure of Elman Net

A. Dynamics

From Fig.1 it can be seen that there are hidden units, output unit and also the context units in a basic Elman network. The input and output units interact with the outside environment, whereas the hidden and context units are contained within. The input units form a buffer which passes the signals without changing, while the output unit is a linear unit which sums up the feed-in signals.

The hidden units may have linear or nonlinear activation functions. The context units are used only to memorize the previous activation of the hidden units and can be considered to be a function as one-step time delay. The feed-forward connections are modifiable; while the recurrent connections are fixed.

At a specific time *k*, the previous activation of the hidden units (at time *k-1*) and the current input (at time *k*) are used as inputs to the network. At this stage, the network acts as a feed-forward network and propagates these inputs forward to produce the output. The standard BP learning rule can be employed to train the network. Once this training step has been started, the activation of the hidden units at time *k* are sent back through the recurrent links to the context units and stored the information for the next training step (*k+1*.)

B. Learning Rule and Algorithm

The learning mechanism is the traditional gradient descent learning method. The activation functions *f* is differentiable as the weight updates are based on the gradient of the error which is defined in terms of the weights and activation functions. The form of the update rule is given as a solution to an equation as:

$$\Delta w_{ij} = \eta \frac{\partial E}{\partial w_{ij}} \qquad (5)$$

where η is a learning rate parameter and w_{ij} is the weight on the connection between unit i and j.

In BP network, global active sigmoid functions are often adopted. The impact to each node is substantial which is frequently resulting to a canyon-shaped error chamber. Thus, the learning convergence speed is rather slow and training course is long.

On the other hand, RNN is based on the gradient descent method and proceeds in a manner similar to the BP. Hence it suffers from the same major drawback caused by BP, i.e. *slow learning speed.* This slowness is largely due to the lack of knowledge of the right step size being taken to reach the global minimum. For certain prediction problem, we do need the learning time to be very short.

In order to improve the speed of learning, there is a number of learning algorithms can be used. e.g. gradient reuse algorithm, composite training algorithm etc. In this case, composite training algorithm is adopted due to its efficiency of the learning convergence speed for Elman networks. The composite training algorithm is largely based on the error function which is:

$$J(w) = \frac{1}{2}\sum_k \sum_t e_k^2(t) \qquad (6)$$

where w is the vector of the network weights; $e_k(t)$ is the prediction error in time t , where $[e_k(t) = d_k(t) - y_k(t)]$; k is the dimension of system output; $d_k(t)$ is the expected output; and $y_k(t)$ is the actual output.

At the beginning of training process, the gradient descent learning method is adopted. The weight update equation is derived according to

$$w(t+1) = w(t) - \frac{\frac{\partial J}{\partial w}}{\zeta + \frac{\partial^2 J}{\partial w^2}} \qquad (7)$$

where ζ is a very small constant.

In the vicinity of the convergence area, the convergence speed of the gradient descent learning method becomes very slow. The search direction and step are easy to get stuck in local optimum. So it is difficult to find the real global optimization point. In order to improve its performance, another method which has the adaptive ability to change the

search step and direction is employed. The algorithm can be described as follows:

step 1. define the initial weight w_0 and initial step size λ ;

step 2. solve the gradient vector ∇w_0 of object function in w_0, let $S_0 = -\nabla w_0$;

step 3. from w_0 to w_1, along the direction of S_0, the step size is λ. Solve ∇w_0, let $S_1 = -\nabla w_1$;

step 4. Assume α and β are the parameters controlling the direction and step size of convergence, respectively. Let $\alpha = 0$, $\beta = 0.01$, solve $J(w_1)$ the object function of w_1, let $TAR = J(w_1)$;

step 5. $\alpha^{(l+1)} = \alpha^{(l)} + \beta^{(l)}$, $S^{(l+1)} = S_0 + \alpha^{(l)} \times S^{(l)}$, from w_0 to w_2, the step size is λ;

step 6. if $J(w_2) < J(w_1)$, then $w_1 = w_2$, $\beta^{(l+1)} = 2\beta^{(l)}$, return to Step 5,

if $J(w_2) > J(w_1)$ and $|\alpha| < \xi_1$ or $|\beta| > \xi_2$, go to step 7, otherwise $\beta^{(l+1)} = \frac{\beta^{(l)}}{3}$, return Step 5; ξ_1 and ξ_2 are limit value ;

step 7. $D = (TAR - J(w_0)) / J(w_0)$,

if $D > \xi_3$, then $S_1 = -\nabla w_1$, return to step 4.

if $D < \xi_3$, then go to Step 8; ξ_3 is limit value; and

step 8. if $J(w_1) < J(w_2)$, then $w_0 = w_1$, return to step 2;

if $\lambda > \xi_4$, then $\lambda = \frac{\lambda}{2}$,

if $\lambda < \xi_4$, then stop calculate. ξ_4 is limit value.

IV. RECURRENT NEURAL NETWORKS MODEL FOR TIME SERIES PREDICTION

The Elman RNN has some special feature, such as the dynamic and nonlinear behavior. Considering that the basic form $y(k) = \hat{y}(k) + e(k)$ is to be retained, while the estimate is taken as the output of a Elman networks which is driven by past values of the sequence. This can be written as:

$$\begin{aligned} y(k) &= \hat{y}(k) + e(k) \\ &= G[y(k-1), y(k-2), \ldots, y(k-T)] + e(k) \end{aligned} \quad (8)$$

where $G[\bullet]$ is an RNN system function. In order to create different type of time series pattern, e.g. chaotic, periodic types, etc., the simple but highly nonlinear Duffing equation and/or the canonical Chua's circuit are used for generating the time series.

A. Duffing system

The Duffing oscillator is one of the prototype systems of nonlinear dynamics. It first became popular for studying harmonic oscillations and, later, chaotic nonlinear dynamics in the wake of early studies by the engineer Georg Duffing. The system has been successfully used to model a variety of physical processes such as stiffening springs, beam bucking, nonlinear electronic circuits, super-conducting Josephson parametric amplifiers, and ionization waves in plasmas. Despite the simplicity of the Duffing oscillator, the dynamical behavior is extremely rich and research is still going on today [10].

The Duffing oscillator [6] can be described by the following equations

$$\begin{cases} \dot{x} = y \\ \dot{y} = -p_2 x - x^3 - p_1 y + q\cos(wt) \end{cases} \quad (9)$$

Chaotic time series data can be obtained using the following parameters: $p_1 = 0.4, p_2 = -1.10, q = 2.10, w = 1.80$.

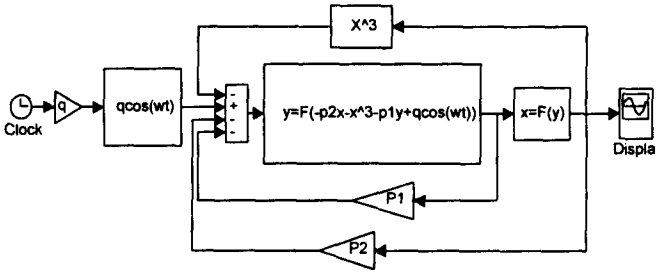

Fig. 2. Simulink Simulation Flowchart

These data can be altered with a different set of values for a different pattern. By the use of MATLAB's simulink [8] which is shown as Fig 2, the nonlinear time series data $x(t)$ is obtained.

B. Duffing Time Series Simulation

Having formulated the structure of Elman network as described in Fig.3 in the training mode , the prediction of the chaotic and periodic behavior of Duffing Equation can be proceeded.

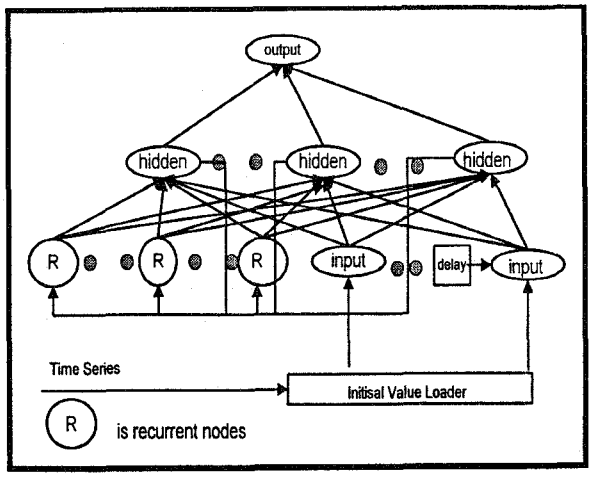

Figure 3. Proposal Recurrent neural networks architecture

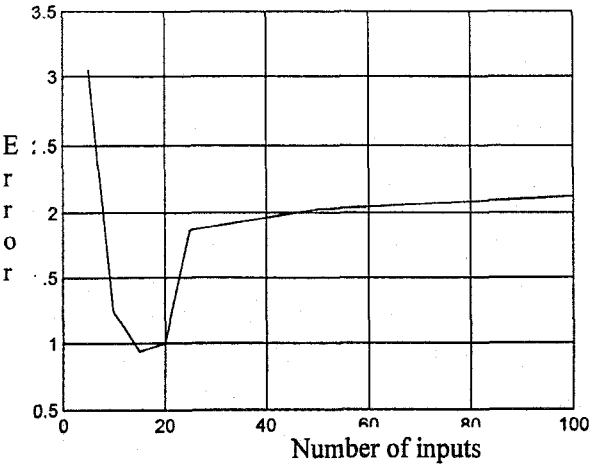

Fig. 4 Error-number of input Graph of RNN

Fig.3 shows an RNN with one hidden layer (For the time series that we propose to analyze, it turns out that one hidden layer sufficed). The abbreviation denotes the following network:

- The N input units are given the values $x_{t-1}, x_{t-2}, \ldots, x_{t-N}$.
- The k=10 hidden nodes are fully connected to the input nodes;
- The (1+k+N) input nodes, where N is the number of inputs. From Fig.4, the number of inputs nodes as N =15 is chosen due to a thorough study on the trade-off between error and quantity of inputs;
- The calculation model of RNN is expressed as Eq.3 and Eq.4;
- The linear output unit is fully connected to the hidden units, producing the prediction value $\hat{x}_t$ as the weighted sum of the activation of the hidden units; and
- The weights can be positive , negative , or zero .

The prediction process is that :

(a) Use the first-half data $x(t), x(t+\Delta t), .., x(t+N\Delta t)$ as a input variable $p(t)$, (refer to Eq.(3),(4)), the $x(t+(N+m)\Delta t)$ is termed as output variable $a_2(t)$ (m is the predict steps, here m=3) to train the recurrent neural networks;

(b) After training, RNN can keep weight w_1 and w_2;

(c) Then , we use the second-half time series data as input variable $p(t)$ to predict next N-step time series.

The prediction results are depicted in Figs.5. (The solid line is real data. The dotted line is prediction result). It can be seen that the proposed algorithm predicts the chaotic motion very accurately.

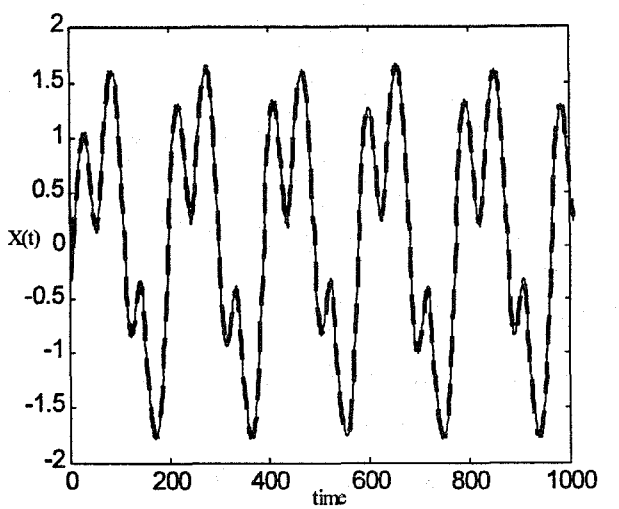

Fig.5 Short-term Controlling Duffing Chaotic time series prediction

C. The Canonical Chua's Circuit

To able to generalize the RNN, it has been used to predict another Chaos time series which is obtained from Chua's Circuit. This circuit is a well known physical system whose mathematical model can be accurately determined. It provides an unique feature for studying the chaotic phenomena not only from computer simulations, but it also can be physically determined experimentally by simple

electronic circuits. Moreover, its chaotic nature has been proven mathematically due to Shil'nikov theory[9].

The state equation for the canonical Chua's circuit is given by:

$$\begin{cases} \frac{dv_1}{dt} = \frac{1}{C_1}[\frac{v_2 - v_1}{R} - f(v_1)] \\ \frac{dv_2}{dt} = \frac{1}{C_2}[\frac{v_1 - v_2}{R} + i_3] \\ \frac{di_3}{dt} = \frac{1}{L}[-v_2 - R_0 i_3] \end{cases} \qquad (10)$$

Where $f(v_1)$ is given by $i_R = f(v_R) = av_R + cv_R^3$. The parameter and variable are explained in [11].

Chaotic time series data can be obtained using the following parameters:

$a = -0.5995 \times 10^{-3}; c = 0.0218 \times 10^{-3}; C_1 = 7 \times 10^{-9};$

$C_2 = 7.8 \times 10^{-9}; \quad L = 18.9 \times 10^{-3}; \quad R = 1960;$

$R_0 = 14.99.$

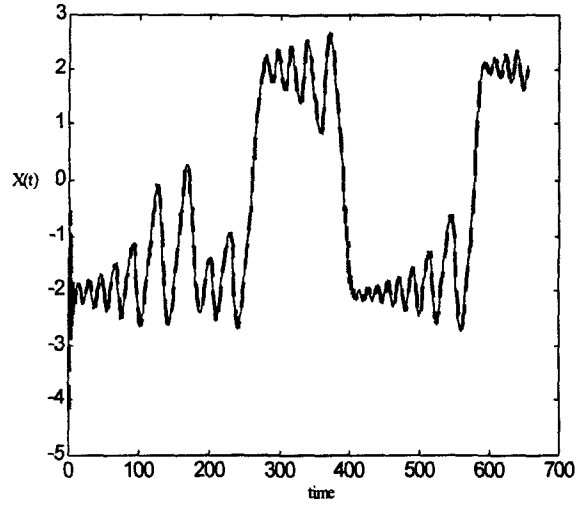

Fig.6 Short-term Chua's Circuits Chaotic time series prediction

Based on the above Chua's Circuit, we use the same RNN structure to predict the dynamics of Chua's circuit, whose accurate predicting result is depicted in Fig.6.

V. CONCLUSION

Based on the standard Elman network, a composite learning algorithm can be used to predict a chaotic time series. The advantages of this approach be summarized as follows:

1. The Elman composite training algorithm can handle more complicate computation than the standard BP in the order of magnitude while the time convergence is only a fraction (tenth) of its speed; and

2. Due to its ability of being more attainable to reach the global optimum, Elman RNN has better prediction ability for time series modeling.

ACKNOWLEDGMENT

We would like to thank Prof. G. Q. ZHONG for provide Chua's Circuits simulation program .

REFERENCES

[1]Martine Casdagli , " Nonlinnear Prediction Of Chaotic Time Series" *Physica D* 25 (1989) 335-356

[2]Jerome T. Connor, " Recurrent Neural Networks And Robust Time Series Prediction" *IEEE Transactions On Neural Networks* , Vol.5, No.2 March 1994

[3]Liming W. Salvino , " Predictability In Time Series" *Physics Letters A* 209 (1995) 327-332

[4]Jerome T. Connor, R. Douglas Martin , and L.E. Atlas , MARCH 1994, "Recurrent Neural Networks and Robust Time series Prediction", *IEEE Transactions on neural networks*, vol 5. No.2. pp240-253

[5]Ronald J. Williams , 1989, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks" *Neural Computation* VOL.1, pp.270-280

[6]Duffing,G., *"Erzwungene Schwingungen bei Veranderlicher Eigenfrequenz,"* Braunschweig, Germany: F. Vieweg u. Sohn, 1918.

[7]Elman, J.L., ``*Finding Structure In Time*", Cognitive Science, **14**, pp. 179--211, 1990.

[8]MATHWORKS, ``*MATLAB User's Guide*", The MathWorks, Inc, 1991.

[9]Rabinder N. Madan, *"Chua's circuit: A paradigm for CHAOS"* World Scientific Series on Nonlinear Science .

[10]H.J.Korsch, H.-J.Jodl, *"Chaos, a program collection for PC"* Springer-Verlag, 1994

[11]L.O. Chua, *"A Zoo of Strange Attractors from The canonical Chua's Circuits"* 1992

[12]MATHWORKS, *"Simulink Dynamic system simulation software , User Guide"*, The MathWorks, Inc., 1991.