ELSEVIER

Available online at www.sciencedirect.com

SCIENCE DIRECT•

Mathematical and Computer Modelling 41 (2005) 227–239

MATHEMATICAL AND COMPUTER MODELLING

www.elsevier.com/locate/mcm

Recurrent Wavelet-Based Neuro Fuzzy Networks for Dynamic System Identification

CHENG-JIAN LIN* AND CHENG-CHUNG CHIN
Department of Computer Science and Information Engineering
Chaoyang University of Technology
No. 168, Jifong E. Rd., Wufong Township
Taichung County 41349, Taiwan, R.O.C.
cjlin@mail.cyut.edu.tw

(Received May 2003; revised and accepted May 2004)

Abstract—A recurrent wavelet-based neuro fuzzy network (RWNFN) is proposed in this paper. The proposed RWNFN integrates wavelet transforms with fuzzy rules. Temporal relations are embedded in the network by adding feedback connections from memory units in the third layer of a feedforward wavelet-based neuro fuzzy network (WNFN). The RWNFN augments the basic ability of the WNFN to overcome temporal problems. Moreover, an online learning algorithm is proposed to automatically construct the RWNFN. Computer simulations were conducted to illustrate the performance and applicability of the proposed system. © 2005 Elsevier Ltd. All rights reserved.

Keywords—Recurrent network, TSK-type fuzzy model, Wavelet neural networks.

1. INTRODUCTION

The backpropagation network is a multilayer feedforward network combined with a gradient-descent-type learning algorithm called the backpropagation learning rule. The backpropagation network has had a major impact on the field of neural networks and is the primary method employed in most of the applications. However, it is difficult to understand the meaning of each neuron and each weight in the network. Generally, fuzzy systems are easy to appreciate because they use linguistic terms and if-then rules. However, fuzzy systems lack the learning procedures to fine-tune the fuzzy rules and membership functions. Therefore, neuro fuzzy networks aim at providing fuzzy systems with automatic tuning abilities. With this approach, we witness the use of the backpropagation learning rule in learning or tuning membership functions and fuzzy rules of fuzzy systems. The backpropagation learning techniques can thus substantially reduce development time and cost while improving the performance of fuzzy systems [1]. Recently, neuro fuzzy networks have been shown to obtain successful results in the identification and control of dynamic systems [2–6].

The authors would like to thank the National Science Council of the Republic of China, Taiwan for financially supporting this research under Contract No. NSC 92-2213-E-324-002.

*Author to whom all correspondence should be addressed.

As is widely known, problems arise in the identification and control of dynamic systems [7]. Since, for a dynamic system, the output is a function of a past output or a past input or both, identification and control of this kind of system is not as straightforward as of a static system. If existing feedforward neuro fuzzy networks are used to deal with dynamic systems, then we must know the number of delayed inputs and outputs in advance. The problem with this approach is that the exact order of the dynamic system is usually unknown. In addition, the usage of the long tapped delay input will increase the input dimension and will result in a large network size. To summarize the problems mentioned above, a major drawback of feedforward neuro fuzzy networks [2–6] is that their application is limited to static mapping problems. However, to identify dynamic systems or to recognize temporal sequences, recurrent neuro fuzzy networks should be employed [8–14].

In this paper, we propose a recurrent wavelet-based neuro fuzzy network (RWNFN). The RWNFN model is based on our previous research [2]. In [2], we presented a wavelet-based neuro fuzzy network (WNFN) that deals with the problems of static system identification. Processing temporal problems using WNFN is inefficient. Hence, we propose a RWNFN model that deals with temporal problems and is more efficient for solving dynamic systems than the WNFN.

The RWNFN model integrates the traditional TSK-type fuzzy model and the wavelet neural network (WNN). The goal of combining the RWNFN model with the WNN model is to improve the accuracy of function approximation. Each fuzzy rule corresponding to a WNN consists of single-scaling wavelets. We adopt nonorthogonal and compactly supported functions as the bases of the wavelet neural network. The online structure/parameter learning algorithm is performed concurrently in the RWNFN. Structure learning is based on the degree measure to determine the number of fuzzy rules and wavelet functions, and parameter learning is based on the gradient descent method to adjust the shape of the membership function and the connection weights of WNN.

Overall, the proposed model addresses the following objectives:

(1) to achieve recurrent properties for dynamic systems identification;
(2) to obtain dynamic composition for patterns clustering and to adjust free parameters by the learning process from input-output data;
(3) to achieve quick convergence and to require as small a number of tuning parameters as possible.

2. STRUCTURE OF THE RECURRENT WAVELET-BASE NEURO FUZZY NETWORK

2.1. Description of Wavelet Neural Network

To generate the novel form of the TSK model, the RWNFN integrates the traditional TSK-type fuzzy model and the WNN [15]. The goal of integrating the RWNFN model with the WNN model is to improve the accuracy of function approximation [16]. Each fuzzy rule corresponding to a WNN consists of single-scaling wavelets. In this paper, we adopt nonorthogonal and compactly supported functions in a finite range as the wavelet bases [17]. The shape and position of the wavelet bases are shown in Figure 1.

A new type of WNN is shown in Figure 2. An ordinary wavelet neural network model is often used to normalize input vectors in the interval $[0,1]$. We then calculate the $\phi_{a.b}(x_i)$ functions which are used to input vectors to fire up the wavelet interval. (This means the value of the wavelet function should be zero.) Obviously, we would obtain a value $\psi_{a.b}$, as follows:

$$\psi_{a.b} = \frac{\sum_{i=1}^{n} \phi_{a.b}(x_i)}{|X|}, \qquad \text{where } b = 1,\ldots,a \text{ and } a = 1,\ldots,m, \tag{1}$$

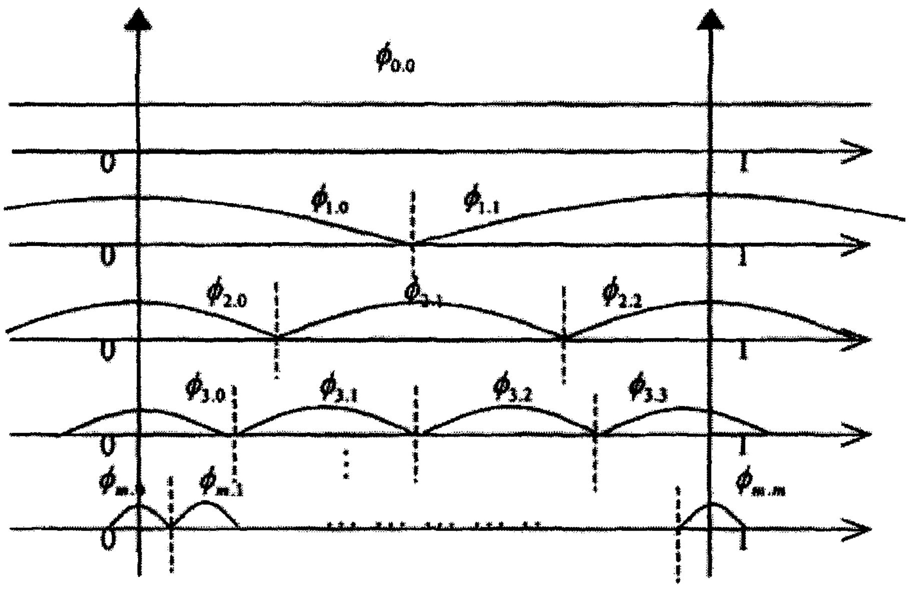


Figure 1. Wavelet bases are over-complete and compactly supported.

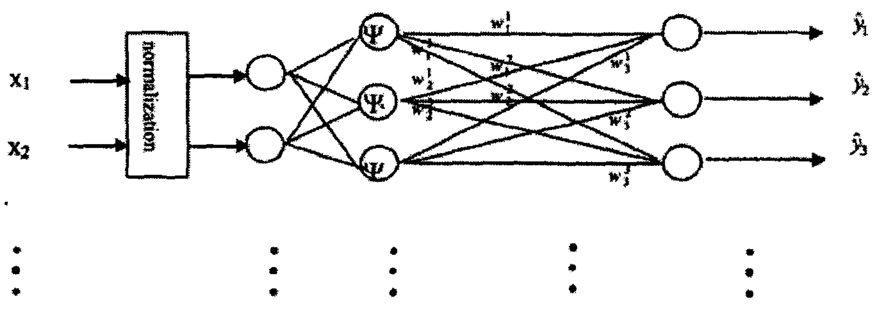


Figure 2. Schematic diagram of the WNN.

where $|X|$ is the number of input dimensions and b is a shifting parameter, the maximum value of which is equal to the corresponding scaling parameter a. The final output of the wavelet neural network is:

$$\hat{y}_j = \sum_{i=1}^{k} w_i^j \psi_{a.b} = w_1^j \psi_{0.0} + w_2^j \psi_{1.0} + \cdots + w_k^j \psi_{m.m}, \tag{2}$$

where $\hat{y}_j$ is the output of the local model for the jth rule, and the link weight w_k^j is the output action strength associated with the jth rule and kth $\psi_{a.b}$.

2.2. Description of the RWNFN Model

The suggested recurrent wavelet-based neuro fuzzy network (RWNFN) is given as follows:

$$\begin{aligned} R_j : &\text{ If } x_1(t) \text{ is } A_1^j \text{ and } \ldots x_i(t) \text{ is } A_i^j \text{ and } \ldots \text{ and } x_n(t) \text{ is } A_n^j \text{ and } h_j(t) \text{ is } G \\ &\text{then } \hat{y}_j(t+1) \text{ is } \sum_{i=1} w_i^j \psi_{a.b} \text{ and } h_j(t+1) \text{ is } \theta_j, \end{aligned} \tag{3}$$

where R_j is the jth rule; $x = [x_1, x_2, \ldots, x_n]^T$ is the input vector to the model; h_j is the internal variable;$\hat{y}_j$ is the jth output of the local model for rule R_j; A_i^j and G are fuzzy sets. The θ_j and w_i^j are the consequent parameters for output h_j and $\hat{y}_j$, respectively. Dynamic reasoning implies that the inference output $Y(t+1)$ is affected by the internal variable $h_j(t)$, and the current

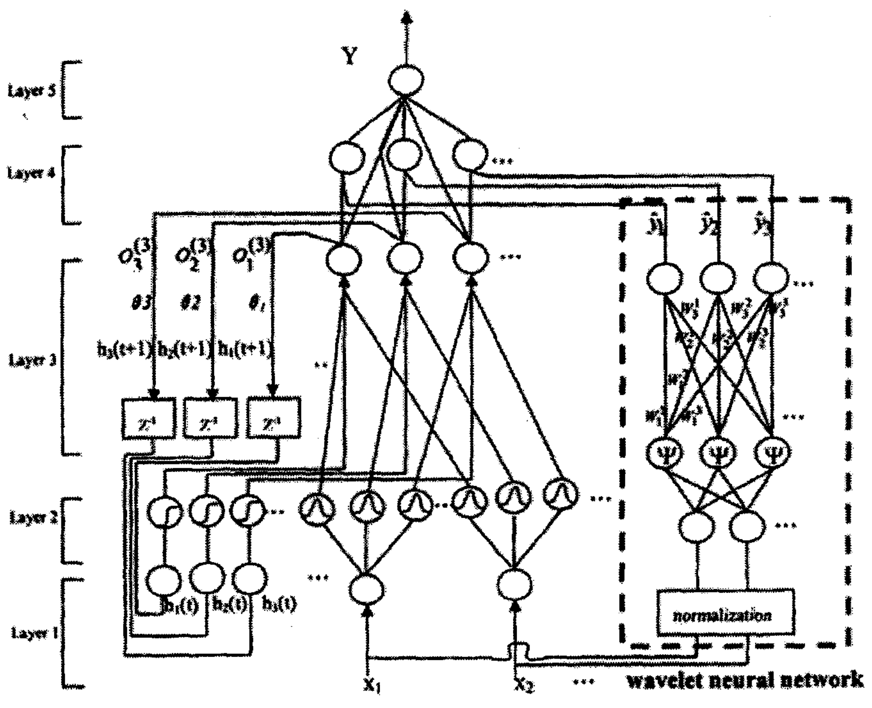


Figure 3. Schematic diagram of the RWNFN model.

internal output $h_j(t+1)$is a function of the previous output value $h_j(t)$; that is, the internal value h_j itself forms the dynamic reasoning.

The structure of the RWNFN model will now be introduced and is shown in Figure 3. Basically, it is a five-layer neural fuzzy network embedded with dynamic feedback connections (the feed back layer in Figure 3). With the dynamic feedback connections, the feedforward neural fuzzy network has temporal processing ability. Nodes in Layer 1 are input nodes for input linguistic variables. Nodes in Layer 2 are called input term nodes and act as membership functions representing the terms of the respective input linguistic variables. Two types of membership functions are used in this layer. For the external variable x, a local membership function, the Gaussian membership function is adopted. For the internal variable h, a global membership function, the sigmoid function is adopted. Each internal variable has a single corresponding fuzzy set. Each node in Layer 3 is called a rule node. The number of rule nodes in this layer is equal to the number of fuzzy sets corresponding to each external linguistic input variable. Note that the number of internal variables h is equal to the rule nodes. Nodes in Layer 4 are called consequent nodes. Each rule node has a corresponding consequent node which performs a weighted nonlinear combination of the input variables x. In Layer 5, the nodes are called defuzzification nodes.

To give a clear understanding of the mathematical function of each node, we will describe the function of RWNFN layer by layer. The notation $I_i^{(l)}$ signifies the net input to the ith node in Layer l, and the notation $O_i^{(l)}$ signifies the net output.

LAYER 1. No computation is done in this layer. These nodes only pass the input signal to the next layer

$$O_i^{(1)} = x_i. \tag{4}$$

LAYER 2. Each node in this layer acts as a membership function representing the term of the respective input-linguistic variables; that is, the membership value specifying the degree to which an input value belongs to a fuzzy set is determined in this layer. The Gaussian function given below is adopted as the membership function

$$O_{ij}^{(2)} = \exp\left(-\frac{(I_i^{(2)} - m_{ij})^2}{\sigma_{ij}^2}\right) \qquad \text{and} \qquad I_i^{(2)} = O_i^{(1)}, \tag{5}$$

where m_{ij} and σ_{ij} are the mean and standard deviation, respectively, of the j^{th} term associated with the i^{th} input variable . For the internal variable h_j, the following sigmoid membership function is used

$$O_j^{(f)} = \frac{1}{1+\exp(-h_j)}, \tag{6}$$

where

$$h_j = O_j^{(3)} \cdot \theta_j \tag{7}$$

are the feedback units acting as memory elements, and θ_j is the feedback weight. As in Figure 3, the delayed value of h_j is fed back to Layer 1 and act as an input variable to the precondition part of a rule. Each rule has a corresponding internal variable h_j and is used to decide the influence degree of temporal history to the current rule.

LAYER 3. Each node in this layer is a rule node representing the preconditioning part of one fuzzy logic rule. Therefore, each node in this layer is denoted by Π, which multiplies the incoming signals from Layer 2 and the feedback layer and outputs the product result, that is, the firing strength of a rule. For the j^{th} rule node

$$O_j^{(3)} = O_j^{(f)} \cdot \prod_{i=1}^{n} O_{ij}^{(2)} = \frac{1}{1+\exp(-h_j)} \cdot \exp\left(-\frac{(I_i^{(2)} - m_{ij})^2}{\sigma_{ij}^2}\right), \tag{8}$$

where n is the number of external inputs.

LAYER 4. Nodes in this layer receive the signals, which are $\hat{y}_j$ from the output of the wavelet neural network model and $O_j^{(3)}$ from the output of Layer 3. The mathematical function of each Node j is

$$O_j^{(4)} = \hat{y}_j \cdot O_j^{(3)} = \left(\sum_{i=1}^{k} w_i^j \psi_{a.b}\right) \cdot O_j^{(3)}. \tag{9}$$

LAYER 5. The node in this layer computes the output signal Y. The output node together with links connected to it acts as a defuzzifier. The mathematical function is

$$\begin{aligned} Y &= \frac{\sum_{j=1}^{M} O_j^{(4)}}{\sum_{j=1}^{M} O_j^{(3)}} = \frac{\sum_{j=1}^{M} \hat{y}_j \cdot O_j^{(3)}}{\sum_{j=1}^{M} O_j^{(3)}} \\ &= \frac{\sum_{j=1}^{M} (w_1^j \psi_{0.0} + w_2^j \psi_{1.0} + \cdots + w_k^j \psi_{m.m}) \cdot O_j^{(3)}}{\sum_{j=1}^{M} O_j^{(3)}}, \end{aligned} \tag{10}$$

where the link weight $\hat{y}_j$ is the output of the local model of the wavelet neural network model for the j^{th} rule, $O_j^{(3)}$ is the output of Layer 3, M denotes the number of existing fuzzy rules, which equals the number of wavelwt bases and Y is the output of the RWNFN.

3. AN ONLINE LEARNING ALGORITHM FOR RWNFN

For dynamic neuro fuzzy models, the major problem is in determining the structure of a model. Recently, many neuro fuzzy models were proposed by [3,4,10,11]. The two steps in the learning method were adopted to determine the proper fuzzy rules and the adjustable parameters for their models. For the initialization of the parameters in their models, the rule number needs to be given in advance. What makes our proposal different from [3,4,10,11] is that an online learning algorithm is adopted for constructing the RWNFN. Users need not give any a priori knowledge or even any initial information in our proposed model. More notably, in our learning method, only the training data need to be provided from the outside world.

The proposed learning algorithm consists of structure learning and parameter learning. The objective of structure learning is to make sure that proper fuzzy partitioning, membership of the rule nodes, weights of the feedback, and link weights in the wavelet neural network are generated dynamically. On the other hand, the objective of parameter learning is to tune an adjustable parameter that is generated from structure learning. Notice that the structure and parameter learning are performed concurrently to construct the RWNFN. Details of the two learning algorithm are described as follows.

3.1. Structure Learning Algorithm

Since there are no rules initially in the RWNFN, the first step is to generate a new rule from the input space, which represents the spatial information. For this reason, the spatial information is used for clustering due to its local mapping. Geometrically, a rule corresponds to a cluster in the input space, with m_{ij} and σ_{ij} representing the mean and variance of that cluster. For each incoming pattern x_i, the strength with which a rule is fired can be interpreted as the degree of the incoming pattern that belongs to the corresponding cluster. The firing strength obtained from equation (8) is used as the *degree measure* [8]

$$D_j = O_j^{(3)}, \qquad j = 1, \dots, Q(s), \tag{11}$$

where $Q(s)$is the number of existing rules at time s. According to the degree measure, the criterion for generating a new fuzzy rule for new incoming data is described as follows.

Find the maximum degree $D_{\max}$

$$D_{\max} = \max_{1 \le j \le Q(s)} D_j. \tag{12}$$

If $D_{\max} \le \bar{D}$, then a new rule and new wavelet base are generated, where $\bar{D} \in (0,1)$ denotes a prespecified threshold that should decay during the learning process, thus limiting the size of RWNFN.

$$m_i^{(\text{new})} = x_i, \tag{13}$$

$$\sigma_i^{(\text{new})} = \sigma_{\text{prespecified}}, \tag{14}$$

$$w_k^{i(\text{new})} = \theta_i^{(\text{new})} = r, \qquad \text{where } r \in [-1,1], \tag{15}$$

where x_i is the new incoming data; σ_i is a prespecified constant; and the weight of the feedback θ_i and the link weight of WNN w_k^i are selected with random values r in $[-1,1]$.

The concise online structure learning algorithm of the RWNFN model is given as follows:

```
Initialization;
do{
    IF $x$ is the first incoming pattern,
    do{
        Generate new nodes;
        with mean $m_{ij} = x_i$;
        deviation $\sigma_{ij} = \sigma_{prespecified}$;
        link weight $w_k^i$ and of WNN and weight of feedback $\theta_i$ are selected with random values;
        }
        ELSE for each newly incoming $x$
        do{
            Execution structure learning scheme
            If $D_{\max} \leq \bar{D}$
            do{
                Generate new nodes;
                with mean $m_{ij} = x_i$;
                deviation $\sigma_{ij} = \sigma_{\text{prespecified}}$;
                link weight $w_k^i$ and of WNN and weight of feedback $\theta_i$ are selected with random
                values;
            }
        }
} until task completion
```

3.2. Parameter Learning Algorithm

After the network structure has been adjusted according to the current training pattern, the network then begins parameter learning to adjust the parameters of the membership functions optimally with the same training pattern. The problem in parameter learning can be stated as follows: given the training input data $x = [x_1, \ldots, x_n]$ and the desired output value Y, we want to optimally adjust the parameters of the membership functions, feedback weights and link weights in the wavelet neural network optimally. These fuzzy logic rules and wavelet nodes are learned in the structure learning. Basically, the idea of backpropagation algorithm is used for this parameter learning to find the output errors of the node in each layer. Then, these errors are analyzed to perform parameters adjustment. The goal is to minimize the error function

$$E(t+1) = \frac{1}{2}\left(Y(t+1) - Y^d(t+1)\right)^2, \tag{16}$$

where $Y(t+1)$ is the model output and $Y^d(t+1)$ is the desired output at time $t+1$.

When the backpropagation learning algorithm is used, the weighting vector of the RWNFN is adjusted such that the error defined in equation (16) is less than the desired threshold value after a given number of training cycles. The well-known backpropagation learning algorithm may be written briefly as

$$W(t+1) = W(t) + \Delta W(t) = W(t) + \eta\left(-\frac{\partial E(t+1)}{\partial W}\right), \tag{17}$$

where η and W represent the learning rate and tuning parameters of the WRFNN respectively. Let $e(t+1) = (Y^d(t+1) - Y(t+1))$ and $W = [m, \sigma, \theta, w]^{\mathrm{T}}$ denote the training error and weighting vector of the RWFNN, respectively. Then the gradient of error $E(.)$ in equation (16) with respect to an arbitrary weighting vector W is

$$\frac{\partial E(t+1)}{\partial W} = e(t+1) \cdot \frac{\partial Y(t+1)}{\partial W}. \tag{18}$$

With the above equation defined, we can derive the updated rules for the free parameters in the RWNFN as follows.

The link weight of wavelet neural network is updated by

$$w_k^j(t+1) = w_k^j(t) + \Delta w_k^j(t), \tag{19}$$

where

$$\Delta w_k^j(t) = -\eta_w \frac{\partial E(t+1)}{\partial w_k^j} = \eta_w \cdot e(t+1) \cdot \frac{O_j^{(3)}(t)\psi_{a.b}(x)}{\Sigma_j O_j^{(3)}}(t). \tag{20}$$

Similarly, the update laws of m_{ij}, σ_{ij}, and θ_j are

$$m_{ij}(t+1) = m_{ij}(t) + \Delta m_{ij}(t), \tag{21}$$
$$\sigma_{ij}(t+1) = \sigma_{ij}(t) + \Delta \sigma_{ij}(t), \tag{22}$$
$$\theta_j(t+1) = \theta_j(t) + \Delta \theta_j(t), \tag{23}$$

where

$$\begin{aligned}\Delta m_{ij}(t) &= -\eta_m \frac{\partial E(t+1)}{\partial m_{ij}} = \eta_m \cdot e(t+1) \cdot 2 \cdot \left[\frac{\left(O_i^{(1)} - m_{ij}\right)}{\sigma_{ij}^2}(t) \right] \\ &\cdot O_j^{(3)}(t) \cdot \frac{\left(\hat{y}_j \sum_j O_j^{(3)} - \sum_j \hat{y}_j\right)}{\left(\sum_j O_j^{(3)}\right)^2}(t),\end{aligned} \tag{24}$$

$$\begin{aligned}\Delta \sigma_{ij}(t) &= -\eta_\sigma \frac{\partial E(t+1)}{\partial \sigma_{ij}} = \eta_\sigma \cdot e(t+1) \cdot 2 \cdot \left[\frac{\left(O_i^{(1)} - m_{ij}\right)^2}{\sigma_{ij}^3}(t) \right] \\ &\cdot O_j^{(3)}(t) \cdot \frac{\left(\hat{y}_j \sum_j O_j^{(3)} - \sum_j \hat{y}_j\right)}{\left(\sum_j O_j^{(3)}\right)^2}(t),\end{aligned} \tag{25}$$

$$\begin{aligned}\Delta \theta_{ij}(t) &= -\eta_\theta \frac{\partial E(t+1)}{\partial \theta_{ij}} = \eta_\theta \cdot e(t+1) \cdot O_j^{(3)}(t) \cdot \frac{\left(\hat{y}_j \sum_j O_j^{(3)} - \sum_j \hat{y}_j\right)(t)}{\left(1 + \exp\left(-O_j^{(3)}(t-1) \cdot \theta_j(t)\right)\right)^2} \\ &\cdot \exp\left(-O_j^{(3)}(t-1) \cdot \theta_j(t)\right) \cdot O_j^{(3)}(t-1).\end{aligned} \tag{26}$$

4. ILLUSTRATIVE EXAMPLES

In this section, three examples are given to demonstrate the validity of the proposed RWNFN to cope with temporal problems. In the following simulations, the parameters and number of training epochs were determined based on the desired result.

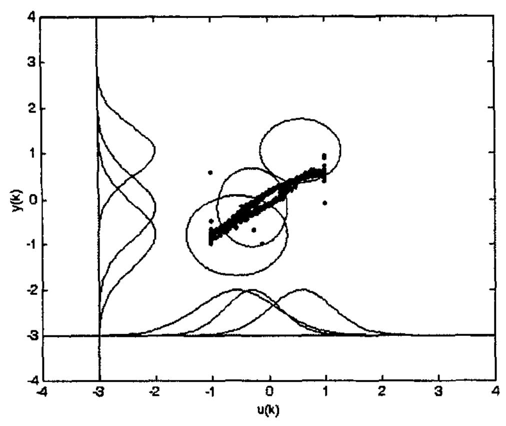
EXAMPLE 1. In this example, a nonlinear plant with multiple time delay was guided by the following difference equation:

$$y_p(t+1) = f\left(y_p(t), y_p(t-1), y_p(t-2), u_p(t), u_p(t-1)\right), \tag{27}$$

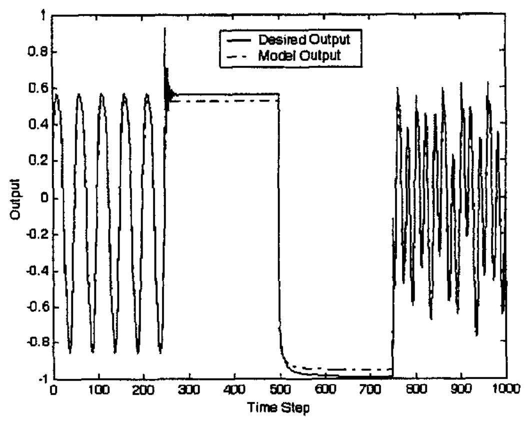
where

$$f\left(x_1, x_2, x_3, x_4, x_5\right) = \frac{x_1 x_2 x_3 x_5 \left(x_3 - 1\right) + x_4}{1 + x_2^2 + x_3^2}. \tag{28}$$

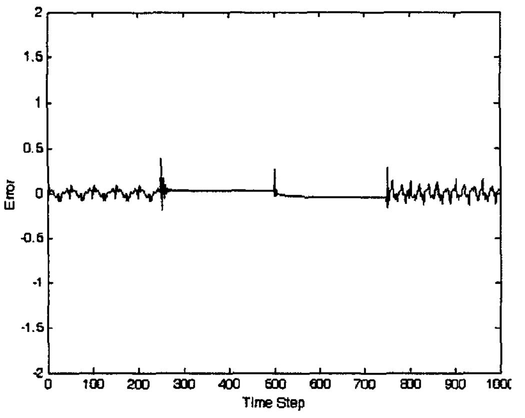
Here, the current output of the plant depends on three previous outputs and two previous inputs. In [7], the feedforward neural network, with five input nodes for feeding the appropriate past values of y_p and u were used. In the RWNFN, only the current state $y_p(t)$ and the control input $u(t)$ were fed to our model, and the output $y_p(t+1)$ was determined. In this simulation, we used only ten epochs. There were 900 time steps in each epoch. The training inputs were independently and identically distributed (i.i.d.) with a uniform sequence over $[-2, 2]$ for about half of the training time. A single sinusoid signal was given by $1.05 \sin(\pi t/45)$ for the remaining training time. There was no repetition of these 900 training data; that is, we had different



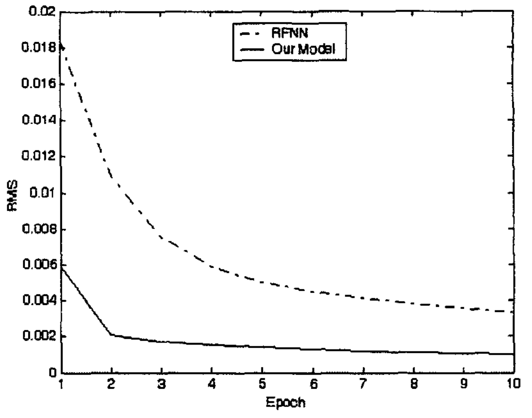
(a). The input training patterns and the final assignment of rules for the distribution of the membership functions on the y(t) and u(t) dimensions.



(b). The outputs of the plant and the RWNFN.



(c). The error between the RWNFN output and the desired output.



(d). Learning curves of the RWNFN and the RFNN [10].

Figure 4. Simulation results of the RWNFN for dynamic system identification in Example 1.

training sets for each epoch. The test input signal $u(t)$ in the following equation was used to determine the identification results

$$u(t) = \begin{cases} \sin\left(\frac{\pi t}{25}\right), & 0 < t < 250, \\ 1.0, & 250 \le t < 500, \\ -1.0, & 500 \le t < 750, \\ 0.3\sin\left(\frac{\pi t}{25}\right) + 0.1\sin\left(\frac{\pi t}{32}\right) + 0.6\sin\left(\frac{\pi t}{10}\right), & 750 \le t < 1000. \end{cases} \tag{29}$$

The initial parameters $\eta_m = \eta_\sigma = \eta_\theta = \eta_w = 0.05$, $\sigma_{\text{init}} = 0.8$, and $\bar{D} = 0.08$ were chosen. After training, three fuzzy rules grew for the incoming training data. Figure 4a shows the distribution of the input training patterns and the final assignment of the rules, (i.e., distribution of the input membership functions). In this figure, the boundary of each ellipse represents a rule with firing strength 0.5. The maximum firing strength 1 occurred in the center of the ellipse. The membership functions on the $u(t)$ and $y(t)$ dimensions are also shown in Figure 4a. The obtained dynamic fuzzy rules after online learning were

R_1 : if $u(t)$ is $\mu(-0.28, 0.69)$ and $y_p(t)$ is $\mu(-0.18, 0.97)$ and $h_1(t)$ is G,
then $\hat{y}_1(t+1)$ is $0.48\psi_{0.0} - 0.08\psi_{1.0} - 0.014\psi_{1.1}$ and $h_1(t+1)$ is 0.01;
R_2 : if $u(t)$ is $\mu(-0.55, 1)$ and $y_p(t)$ is $\mu(-0.78, 0.98)$ and $h_2(t)$ is G,
then $\hat{y}_2(t+1)$ is $-0.6\psi_{0.0} - 0.69\psi_{1.0} - 0.55\psi_{1.1}$ and $h_2(t+1)$ is -0.39;
R_3 : if $u(t)$ is $\mu(0.57, 0.8)$ and $y_p(t)$ is $\mu(1.05, 0.79)$ and $h_3(t)$ is G,
then $\hat{y}_3(t+1)$ is $0.18\psi_{0.0} + 0.76\psi_{1.0} + 0.38\psi_{1.1}$ and $h_3(t+1)$ is -0.46.

In the above rules, h_1, h_2, and h_3 are the generated internal variables; $\hat{y}_1$, $\hat{y}_2$, and $\hat{y}_3$ are the outputs of the wavelet neural networks; $\mu(m_{ij}, \sigma_{ij})$ represents a Gaussian membership function with center m_{ij} and width σ_{ij}; and G is the sigmoid function stated previously in equation (6). Figure 4b shows the outputs of the plant and the RWNFN model for the testing data. The results show the perfect identification capability of the RWNFN model. Figure 4c illustrates the error between the desired output and the RWNFN output. The learning curves of the RWNFN model and the RFNN [10] model are shown in Figure 4d. As shown in this figure, we obtained a smaller rms error and a quicker convergence.

We now compare the performance of our model with that of other existing recurrent neuro fuzzy methods (RSONFIN [8], TRFN-S [9], and RFNN [10]). The comparison results are tabulated in Table 1. As shown in Table 1, the proposed RWNFN model produces smaller rms error than other recurrent neuro fuzzy methods by using adjustable parameters. Clearly, the RWNFN is more effective than other existing recurrent neuro fuzzy networks.

Table 1. Performance comparison of various recurrent methods in Example 1.

	Parameters	RMS Error (Train)	RMS Error (Test)	Epochs
RWNFN	24	0.001	0.0014	10
RSONFIN [8]	36	0.0248	0.078	10
TRFN-S [9]	33	0.0084	0.0346	10
RFNN [10]	24	0.00346	0.0033	10

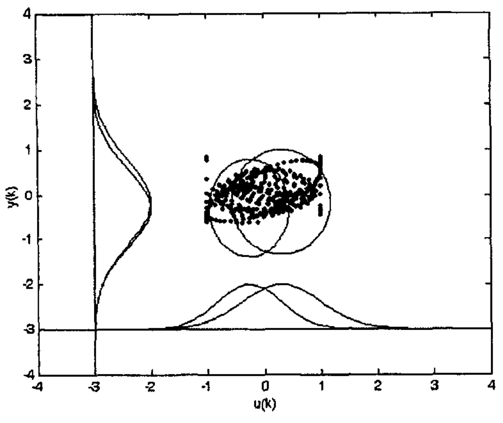
EXAMPLE 2. Consider next the following dynamic plant with time delay inputs

$$y_p(t+1) = 0.72y_p(t) + 0.025y_p(t-1)u(t-1) + 0.01u^2(t-2) + 0.2u(t-3). \tag{30}$$

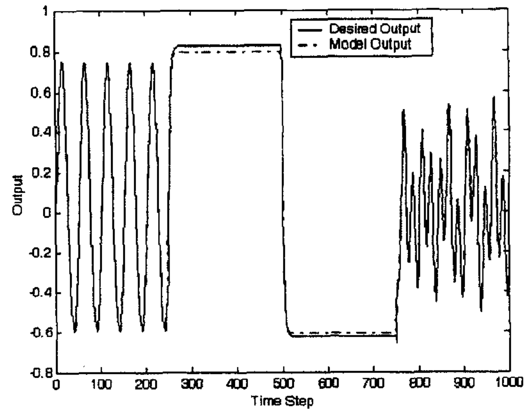
This plant was the same as that used in [9]. The current output of the plant depends on two previous outputs and three previous inputs, as in Example 1 of the identification model, where only two external input values were fed to the input of the RWNFN. The training data and time steps were the same as those used in Example 1. In the training process, we also used 10 epochs. Each epoch consisted of 900 time steps. The initial parameters $\eta_m = \eta_\sigma = \eta_\theta = \eta_w = 0.05$, $\sigma_{\text{init}} = 0.95$, and $\bar{D} = 0.08$ were chosen. After training, two recurrent fuzzy logic rules were generated, and the corresponding obtained dynamic fuzzy rules after online learning were

R_1 : if $u(t)$ is $\mu(0.62, 0.92)$ and $y_p(t)$ is $\mu(0.04, 1.3)$ and $h_1(t)$ is G,
then $\hat{y}_1(t+1)$ is $1.11\psi_{0.0} + 0.31\psi_{1.0}$ and $h_1(t+1)$ is -0.12;
R_2 : if $u(t)$ is $\mu(-0.13, 0.81)$ and $y_p(t)$ is $\mu(-0.19, 1.2)$ and $h_2(t)$ is G,
then $\hat{y}_2(t+1)$ is $-0.92\psi_{0.0} - 0.1\psi_{1.0}$ and $h_2(t+1)$ is -0.79.

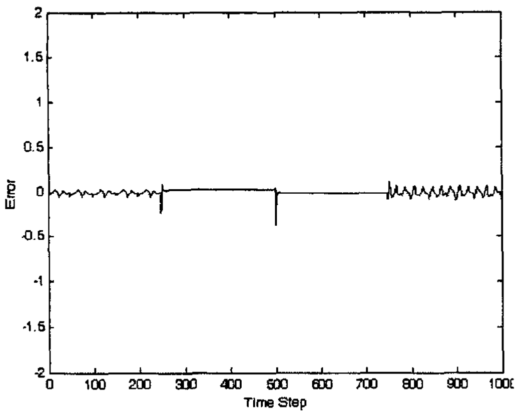
The test signal used to test the identification result was the same as used in Example 1. Figure 5a shows the distribution of the training patterns and the final task of the rules in the $[u(t), y(t)]$ plane. The membership functions on the $u(t)$ and $y(t)$ dimension are also shown in



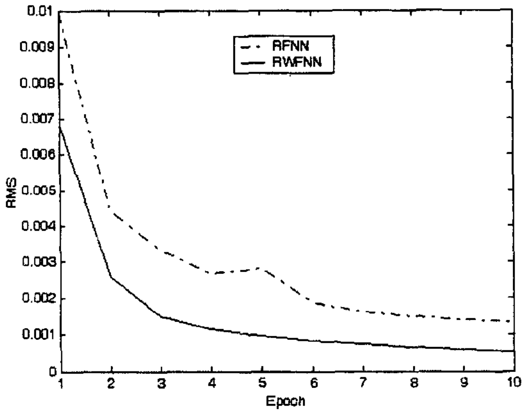
(a). The input training patterns and the final assignment of rules for the distribution of the membership functions on the y(t) and u(t) dimensions.



(b). The outputs of the plant and the RWNFN.



(c). The error between the RWNFN output and the desired output.



(d). Learning curves of the RWNFN and the RFNN [10].

Figure 5. Simulation results of the RWNFN for dynamic system identification in Example 2.

Table 2. Performance comparison of various recurrent methods in Example 2.

	Parameters	RMS Error (Train)	RMS Error (Test)	Epochs
RWNFN	18	0.00048	0.00093	10
RSONFIN [8]	49	0.03	0.06	10
TRFN-S [9]	33	0.0067	0.0313	10
RFNN [10]	24	0.0014	0.0026	10

Figure 5a. Figure 5b shows the outputs of the plant and the RWNFN model. The results also showed the perfect identification capability of the RWNFN model. Figure 5c illustrates the error between the desired output and the RWNFN output. The learning curves of the RWNFN model and the RFNN [10] model are shown in Figure 5d. In this figure, we obtained a smaller rms error and convergence more quickly than RFNN [10].

Finally, we compared the performance of our model with that of other existing recurrent neuro fuzzy methods (RSONFIN [8], TRFN-S [9], and RFNN [10]). The comparison results are tabulated in Table 2. As shown in Table 2, the numbers of adjustable parameters and rms error in our model are smaller than other recurrent methods with the same training epochs.

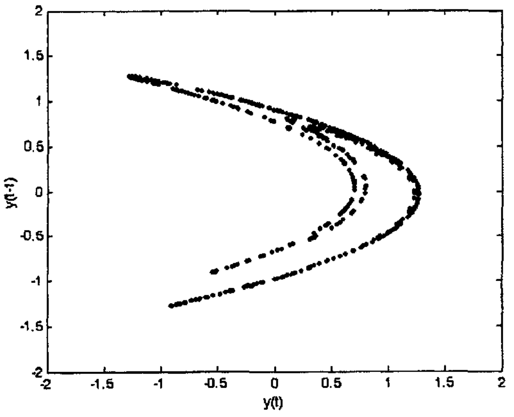
EXAMPLE 3. The discrete time Henon system is repeatedly used in the study of chaotic dynamics and is not too simple in the sense that it is of the second order with one delay and two parameters [10]. This chaotic system is described by

$$y(t+1) = -P \cdot y^2(t) + Q \cdot y(t-1) + 1.0, \qquad \text{for } t = 1, 2, \dots, \tag{31}$$

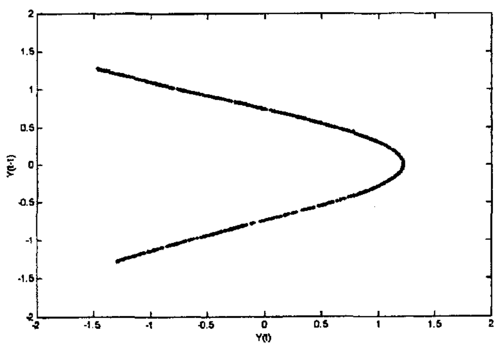
which, with $P = 1.4$ and $Q = 0.3$, produces a strange chaotic attractor, as shown in Figure 6a. For this training, the input of the RWNFN was $y(t-1)$ and the output was $y(t)$. We used the training input patterns sampled randomly (1000 pairs) from the system over the interval $y(t) \in [-1.5, 1.5]$. Then the RWNFN was used to approximate the chaotic system.

In applying the RFWNN to this example, we used only 100 epochs. Here, the initial point was $[y(1), y(0)]^{\mathrm{T}} = [0.4, 0.4]^{\mathrm{T}}$. The learning rate $\eta_m = \eta_\sigma = \eta_\theta = \eta_w = 0.05$, $\sigma_{\text{init}} = 0.3$, and the prespecified threshold $\bar{D} = 5 \times 10^{-5}$ were used. After training, three recurrent fuzzy logic rules were generated. The obtained fuzzy rules were

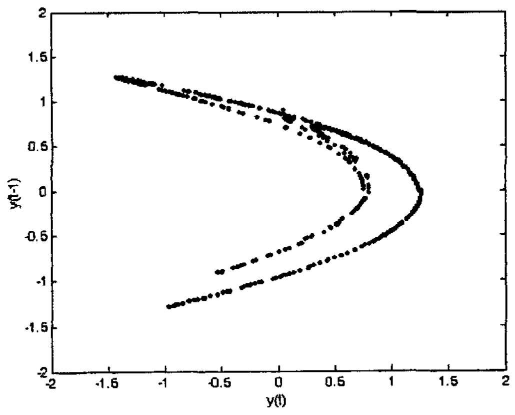
$$\begin{aligned}
R_1 &: \text{if } y(t-1) \text{ is } \mu(2.98, -2.83) \text{ and } h_1(t) \text{ is } G, \\
&\quad \text{then } \hat{y}_1(t) \text{ is } -2.97\psi_{0.0} - 0.63\psi_{1.0} - 1.71\psi_{1.1} \text{ and } h_1(t+1) \text{ is } -2.52; \\
R_2 &: \text{if } y(t-1) \text{ is } \mu(-0.11, 1.4) \text{ and } h_2(t) \text{ is } G, \\
&\quad \text{then } \hat{y}_2(t) \text{ is } 2.71\psi_{0.0} + 1.41\psi_{1.0} + 1.41\psi_{1.1} \text{ and } h_2(t+1) \text{ is } 0.07;
\end{aligned}$$



(a). Check data of this chaotic system.



(b). Result of identification using the FNN [6] for the chaotic system.



(c). Result of identification using the RWNFN for the chaotic system.

Figure 6. Simulation results for identification of a chaotic system.

Table 3. Performance comparison of various recurrent methods in Example 3.

	Rule Numbers	Parameters	RMS Error (Train)	RMS Error (Test)	Epochs
RWNFN	3	16	0.0020	0.0023	100
RFNN [10]	8	32	0.0141	0.0145	100
FNN [6]	8	24	0.1338	0.1557	100

$$\begin{aligned} R_3 : &\text{ if } y(t-1) \text{ is } \mu(-3.31, 2.78) \text{ and } h_3(t) \text{ is } G, \\ &\text{then } \hat{y}_3(t) \text{ is } -3.44\psi_{0.0} - 1.37\psi_{1.0} - 0.54\psi_{1.1} \text{ and } h_3(t+1) \text{ is } 3.11. \end{aligned}$$

The phase plane of this chaotic system after training for the FNN [6] and the RWNFN are shown in Figure 6b and Figure 6c. From the simulation results shown in Figure 6b, we can see that the FNN is inappropriate for chaotic dynamics system because of its static mapping. From Table 3, a comparison shows that the rms error (training and testing) of the proposed model is smaller than the RFNN model and the FNN [6] model with fewer fuzzy rules and training epohs.

5. CONCLUSION

The proposed RWNFN, which is a modified version of the WNFN [2], was used to identify nonlinear dynamic systems. Adding feedback connections in the third layer of the WFNN, where the feedback units act as memory elements, develops the temporal relations embedded in the RWNFN. Using dynamic composition for patterns clustering and free parameters adjusted by learning process from numeric input-output data. Finally, the RWNFN model was tested on three temporal examples. Simulations demonstrated that the RWNFN model was quite effective in many temporal problems.

REFERENCES

1. C.T. Lin and C.S.G. Lee CSG, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent System.*, Prentice-Hall, NJ, (1996).
2. C.J. Lin and C.C. Chin, A wavelet-based neuro-fuzzy system and its applications, *Proc. IEEE Int. Joint Conference on Neural Networks*, disk, Portland, OR, July 20–24, 2003.
3. J.S.R. Jang, ANFIS: Adaptive-network-based fuzzy inference system, *IEEE Trans. on Systems, Man, and Cybernetics* **23** (3), 665–685, (1993).
4. F.S. Wang and C.S.G. Lee, Self-adaptive neuro-fuzzy inference systems for classifications, *IEEE Trans. on Fuzzy Systems* **10** (6), 790–801, (2002).
5. C.F. Juang and C.T. Lin, An on-line self-constructing neural fuzzy inference network and its applications, *IEEE Trans. on Fuzzy Systems* **6** (1), 12–31, (1998).
6. F.J. Lin, C.H. Lin and P.H. Shen, Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drives, *IEEE Trans. on Fuzzy Syst.* **9**, 751–759, (2001).
7. K.S. Narendra and K. Parthasarathy, Identification and control of dynamical systems using neural networks, *IEEE Trans. on Neural Networks* **1**, 4–27, (1990).
8. C.F. Juang and C.T. Lin, A recurrent self-organizing neural fuzzy inference network, *IEEE Trans. on Neural Networks* **10** (4), 828–845, (1999).
9. C.F. Juang, A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms, *IEEE Trans. on Fuzzy Systems* **10** (2), 155–170, (2002).
10. C.H. Lee and C.C. Teng, Identification and control of dynamic systems using recurrent fuzzy neural networks, *IEEE Trans. on Fuzzy Systems* **8** (4), 349–366, (2000).
11. G. Mouzouris and J.M. Mendel, Dynamic non-singleton fuzzy logic systems for nonlinear modeling, *IEEE Trans. on Fuzzy Systems* **5**, 199–208, (1997).
12. F.J. Lin and R.J. Wai, Hybrid control using recurrent fuzzy neural network for linear-induction motor servo drive, *IEEE Trans. on Fuzzy Systems* **9** (1), 102–115, (2001).
13. J. Zhang and A.J. Morris, Recurrent neuro-fuzzy networks for nonlinear process modeling, *IEEE Trans. on Neural Networks* **10** (2), 313–326, (1999).
14. P.A. Mastorocostas and J.B. Theocharis, A recurrent fuzzy-neural model for dynamic system identification, *IEEE Trans. on Syst., Man, Cybern.* **32** (2), 176–190, (2002).
15. T. Yamakawa T, E. Uchino and T. Samatsu, Proc. of IEEE Conf. on Neural Networks **3**, 1391–1396, (1994).
16. D.W.C. Ho, P.A. Zhang and J. Xu, Fuzzy wavelet networks for function learning, *IEEE Trans. on Fuzzy Systems* **9** (1), 200–211, (2001).
17. I. Daubechies, Orthonormal bases of compactly supported wavelets, *Comm. Pur. Appl. Math.* **41**, (1998).