Available online at www.sciencedirect.com

European Journal of Operational Research 172 (2006) 696–715

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/ejor

Computing, Artificial Intelligence and Information Management

A compensation-based recurrent fuzzy neural network for dynamic system identification

Cheng-Jian Lin [*], Cheng-Hung Chen

Department of Computer Science and Information Engineering, Chaoyang University of Technology, No. 168, Jifong E. Rd., Wufong Township, Taichung County 41349, Taiwan, ROC

Received 9 June 2003; accepted 18 November 2004
Available online 8 January 2005

Abstract

In this paper, a type of compensation-based recurrent fuzzy neural network (CRFNN) for identifying dynamic systems is proposed. The proposed CRFNN uses a compensation-based fuzzy reasoning method, and has feedback connections added in the rule layer of the CRFNN. The compensation-based fuzzy reasoning method can make the fuzzy logic system more adaptive and effective, and the additional feedback connections can solve temporal problems. The CRFNN model is proven to be a universal approximator in this paper. Moreover, an online learning algorithm is proposed to automatically construct the CRFNN. The results from simulations of identifying dynamic systems have shown that the convergence speed of the proposed method is faster than the convergence speed of conventional methods and that only a small number of tuning parameters are required.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Identification; Chaotic system; Fuzzy neural networks; Compensatory operator; Recurrent networks

1. Introduction

In the field of artificial intelligence, neural networks are essentially low-level computational structures and algorithms that offer good performance when they deal with sensory data. However, it is difficult to understand the meaning of each neuron and each weight in the networks. Generally, fuzzy systems are easy to appreciate because they use linguistic terms and if-then rules. However, they lack the learning capacity to fine-tune fuzzy rules and membership functions. Therefore, fuzzy neural networks combine the benefits of neural networks and fuzzy systems to solve the problems they are given [1]. Recently, fuzzy neural networks have been shown to obtain successful results in the identification of dynamic systems [1–5].

[*] Corresponding author. Fax: +886 4374 2375.
E-mail address: cjlin@mail.cyut.edu.tw (C.-J. Lin).

0377-2217/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2004.11.007

Fuzzy neural networks have emerged as a powerful approach to solving many engineering problems [1–5]. Lin and Lee [1] brought the low-level learning and computational power of neural networks to fuzzy systems and integrated the high-level human-like thinking and reasoning capability of fuzzy systems into neural networks. Jang [2] proposed a model, called the adaptive-network-based fuzzy inference system (ANFIS) architecture, to represent fuzzy models. Through backpropagation training, ANFIS is adapted to refine, or derive; the fuzzy if-then rules using system input–output data. A modified fuzzy ART mechanism is employed for domain partitioning. The idea of utilizing fuzzy ART concepts for structure learning was proposed by Lin and Lin [3]. The self-constructing neural fuzzy inference network (SONFIN) and the self-constructing fuzzy neural network (SCFNN) were proposed in [4,5] to perform the structure and parameter learning phases concurrently. However, a major disadvantage of the existing fuzzy neural networks is that their application domain is limited to static problems because of their internal feedforward network structure, which causes inefficiency for temporal problems. Hence, a recurrent fuzzy neural network capable of solving temporal problems is needed.

For a dynamic system, the output is a function of past inputs or past outputs or both; identification of this system is not as direct as identification of a static system. To deal with temporal problems of dynamic systems, the commonly used model is a neural network [6] or a fuzzy neural network [2–5]. If a feedforward network is adopted for this task, we should 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. To solve this problem, recurrent networks for processing dynamic systems can be used. Interest in these networks has been steadily growing in recent years [7–9]. However, recurrent networks deal with optimal fuzzy membership functions and defuzzification schemes for applications by using learning algorithms to adjust the parameters of fuzzy membership functions and defuzzification functions. Unfortunately, for optimal fuzzy logic reasoning and fuzzy operators, only static fuzzy operators are often used to make fuzzy reasoning. For example, commonly used fuzzy operators are MIN, MAX, product, and algebraic sum. Intuitively, it is not adaptive and not optimal for a complete fuzzy system to use an unchangeable pair of fuzzy operators.

Zimmermann and Zysno [10] first defined the essence of compensatory operations. Zhang and Kandel [11] proposed more extensive compensatory operations based on the pessimistic operation and the optimistic operation. Recently, many researchers [12–14] have used the compensatory operation on fuzzy systems successfully. Therefore, in this paper, we propose a fuzzy neural network model that cannot only adaptively adjust fuzzy membership functions but can also dynamically optimize the adaptive fuzzy operators.

In this paper, a compensation-based recurrent fuzzy neural network (CRFNN) is proposed. The CRFNN is a recurrent multilayer connectionist network for fuzzy reasoning and can be constructed from a set of fuzzy rules. At the same time, the compensatory fuzzy inference method uses adaptive fuzzy operations of fuzzy neural networks to make the fuzzy logic system more adaptive and effective.

An online learning algorithm is proposed to automatically construct the CRFNN. It consists of structure learning and parameter learning. The structure learning algorithm determines whether to add a new node to satisfy the fuzzy partition of the input data. The backpropagation learning algorithm is used for tuning input membership functions.

Therefore, the proposed CRFNN model has three advantages. First, it does not require a human expert's assistance, and its structure is obtained from the input data. Second, it converges quickly, and only a small number of tuning parameters are required. Third, it can deal with temporal problems for dynamic system identification.

2. The compensatory operation

Zhang and Kandel [11] proposed compensatory operations based on the pessimistic operation and the optimistic operation. The pessimistic operation could map the inputs x_i to the pessimistic output by making a conservative decision for the pessimistic situation or for even the worst case. For example, $p(x_1, x_2, \ldots, x_N) = \mathrm{MIN}(x_1, x_2, \ldots, x_N)$ or Πx_i. Actually, the *t*-norm fuzzy operation is a pessimistic operation.

The optimistic operation can map the inputs x_i to the optimistic output by making an optimistic decision for the optimistic situation or for even the best case. For example, $o(x_1, x_2, \ldots, x_N) = \mathrm{MAX}(x_1, x_2, \ldots, x_N)$. Actually, the *t*-conorm fuzzy operation is an optimistic operation. The compensatory operation can map the pessimistic input x_1 and the optimistic input x_2 to make a relatively compromised decision for the situation between the worst case and the best case. For example, $c(x_1, x_2) = x_1^{1-\gamma} x_2^{\gamma}$, where $\gamma \in [0, 1]$ is called the compensatory degree.

The general fuzzy if-then rule is as follows:

$$\text{Rule-}j: \text{IF } x_1 \text{ is } A_{1j} \text{ and } \ldots\ldots \text{ and } x_N \text{ is } A_{Nj} \text{ THEN } y \text{ is } w_j, \tag{1}$$

where x_i and y are the input dimensions and output variables, respectively; A_{ij} is the linguistic term of the precondition part with membership function $\mu_{A_{ij}}$; w_j is the constant consequent; i is the input dimension, $i = 1, \ldots, N$; N is the number of existing dimensions; j is the number of rules, $j = 1, \ldots, R$; and R is the number of existing rules.

For an input fuzzy set A' in U, the *j*th fuzzy rule (1) can generate an output fuzzy set b'_j in v by using the sup-dot composition

$$\mu_{b'_j} = \sup_{\underline{x} \in U} [\mu_{A_{1j} \times \cdots \times A_{Nj} \to b_j}(\underline{x}, y) \bullet \mu_{A'}(\underline{x})], \tag{2}$$

where $\underline{x} = (x_1, x_2, \ldots, x_N)$. $\mu_{A_{1j} \times \cdots \times A_{Nj}}(\underline{x})$ is defined in a compensatory operation,

$$\mu_{A_{1j} \times \cdots \times A_{Nj}}(\underline{x}) = (u_j)^{1-\gamma_j} (v_j)^{\gamma_j}, \tag{3}$$

where $\gamma_j \in [0, 1]$ is a compensatory degree. The pessimistic operation and the optimistic operation are as follows:

$$u_j = \prod_{i=1}^{N} \mu_{A_{ij}}(x_i), \tag{4}$$

$$v_j = \left[\prod_{i=1}^{N} \mu_{A_{ij}}(x_i) \right]^{1/N}. \tag{5}$$

For simplicity, we can rewrite

$$\mu_{A_{1j} \times \cdots \times A_{Nj}}(\underline{x}) = \left[\prod_{i=1}^{N} \mu_{A_{ij}}(x_i) \right]^{1-\gamma_j+\gamma_j/N}. \tag{6}$$

Since $\mu_{A'}(x_i) = 1$ for the singleton fuzzifier and $\mu_{b'_j}(y) = 1$, according to (2) we have

$$\mu_{b'_j}(y) = \left[\prod_{i=1}^{N} \mu_{A_{ij}}(x_i) \right]^{1-\gamma_j+\gamma_j/N}. \tag{7}$$

Therefore, we can rewrite the fuzzy if-then rule as follows:

$$\text{Rule-}j: [\text{IF } x_1 \text{ is } A_{1j} \text{ and } \ldots\ldots \text{ and } x_N \text{ is } A_{Nj}]^{1-\gamma_j+\gamma_j/N} \text{ THEN } y \text{ is } w_j. \tag{8}$$

3. Structure of the compensation-based recurrent fuzzy neural network

The structure of the CRFNN is shown in Fig. 1, in which A_{ij} by a Gaussian-type membership function, $\mu_{A_{ij}}(x_i)$, defined by

$$\mu_{A_{ij}}(x_i) = \exp\left\{-\frac{[x_i - m_{ij}]^2}{\sigma_{ij}^2}\right\}, \tag{9}$$

where m_{ij} and σ_{ij} are, respectively, the mean and variance of the Gaussian membership function of the jth term of the ith input variable x_i. Defining the number and the locations of the membership functions leads to a partition of the premise space $\aleph = \aleph_1 \times \cdots \times \aleph_n$.

The collection of fuzzy sets $A_j = \{A_{1j}, \ldots, A_{nj}\}$ pertaining to the premise part of *Rule-j* forms a fuzzy region $\boldsymbol{A}_j$ that can be regarded as a multi-dimensional fuzzy set whose membership function is determined by

$$\mu_{A_j}(\underline{x}(t)) = \prod_{i=1}^{n} \mu_{A_{ij}}(x_i(t)). \tag{10}$$

The above equation provides the degree to which a particular input vector $\underline{x}(t)$ belongs to the fuzzy region A_j. For the internal variable s_j, the following sigmoid function is used:

$$s_j = \frac{1}{1 + \exp\{-h_j\}}, \tag{11}$$

where $h_j = \mu_{A_j}(\underline{x}(t-1)) \cdot \theta_j$ is the feedback units acting as memory elements, and θ_j is the feedback weight. Therefore, we can rewrite Eq. (10) as follows:

$$\mu_{A_j}(\underline{x}(t)) = \prod_{i=1}^{n} \mu_{A_{ij}}(x_i(t)) \cdot s_j. \tag{12}$$

Due to the compensatory operation of the grades of the membership functions $\mu_{A_{ij}}(x_i(t))$ in Eq. (12), for simplicity, we can rewrite it as

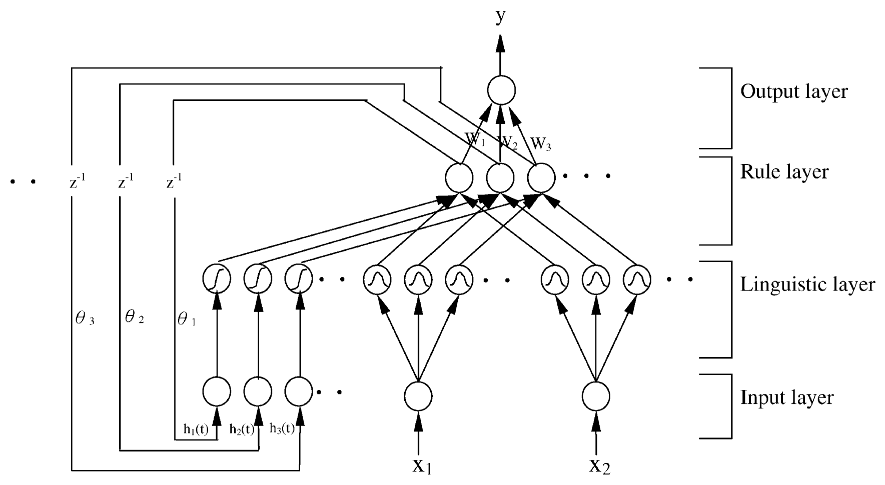

Fig. 1. Structure of the proposed CRFNN.

$$\mu_{A_j}(\underline{x}(t)) = \left[\prod_{i=1}^{n} \mu_{A_{ij}}(x_i) \cdot s_j\right]^{1-\gamma_j+\gamma_j/n}. \tag{13}$$

Therefore, we can rewrite the fuzzy if-then rule as follows:

$$\begin{aligned}&\text{Rule-}j: [\text{IF } x_1(t) \text{ is } A_{1j} \text{ and } \ldots\ldots \text{ and } x_n(t) \text{ is } A_{nj} \text{ and } h_j(t) \text{ is } G]^{1-\gamma_j+\gamma_j/n}\\ &\quad \text{THEN } y'(t+1) \text{ is } w_j \text{ and } h_j(t+1) \text{ is } \theta_j,\end{aligned} \tag{14}$$

where *Rule-j* denotes the *j*th fuzzy rule, $\underline{x}(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^{\mathrm{T}}$ is the input vector to the model at time t with $x_i(t) \in \aleph_i \subset \Re(i = 1, \ldots, n)$, $h_j(t)$ is the internal variable at time t, $y'(t+1)$ is the *j*th output of the local model for *Rule-j*, A_{ij} and G are fuzzy sets, w_j is the consequent part of *Rule-j*, and θ_j is the fuzzy singleton. The output of the model at time t is $y(t)$ and is labeled $\sum$. It is the sum of all incoming signals:

$$y(t) = \sum_{j=1}^{R} \mu_{A_j}(x(t)) \cdot w_j, \tag{15}$$

where the weight w_j is the output action strength of the output associated with the *j*th rule.

Finally, the overall representation of the input x and the output y is

$$y(t) = \sum_{j=1}^{R} w_j \cdot \left\{ \frac{\prod_{i=1}^{n} \exp\left[-\frac{(x_i+(t)-m_{ij})^2}{\sigma_{ij}^2}\right]}{1+\exp[-\mu_{A_j}(x_i(t-1)) \cdot \theta_j]} \right\}^{1-\gamma_j+\gamma_j/n}, \tag{16}$$

where $\gamma_j = c_j^2/c_j^2 + d_j^2$ is the compensatory degree, m_{ij}, σ_{ij}, θ_{ij}, c_j, d_j, and w_j are the tuning parameters, and

$$\mu_{A_j}(\underline{x}(t-1)) = \left\{ \frac{\prod_{i=1}^{n} \exp\left[-\frac{(x_i(t-1)-m_{ij})^2}{\sigma_{ij}^2}\right]}{1+\exp[-\mu_{A_j}(x_i(t-2)) \cdot \theta_j]} \right\}^{1-\gamma_j+\gamma_j/n}. \tag{17}$$

Explicitly, using the CRFNN, the same inputs at different times yield different outputs.

As above, the CRFNN consists of four layers and $R \times (N \times 2 + 3 + M)$ parameters, where N, R, and M denote the number of inputs, rules, and outputs, respectively. The proposed CRFNN can be shown to be a universal uniform approximation for continuous functions over compact sets. The detailed proof is shown in the Appendix A. We have the following result.

Theorem 1: *Universal approximation theorem. For any real continuous function g in a compact set* $U \subset \Re^N$ *and for any given arbitrary* $\varepsilon > 0$, *there is a model f such that*

$$\sup_{x \in U} \|f(x) - g(x)\| < \varepsilon.$$

Here $\|\bullet\|$ *can be any norm.*

This theorem shows that if the CRFNN has a sufficiently large number of compensatory fuzzy rules, then it can approximate any continuous function in $C(\Re^N)$ over a compact subset of $\Re^N$. For system over identification, the theorem means that for any given continuous output trajectory $y^d(t)$ of any nonlinear dynamic system over any compact time-interval $t \in [t_0, T]$, the output $y(t)$ of the CRFNN can approximate $y^d(t)$ uniformly with arbitrarily high accuracy.

4. Learning algorithms of CRFNN

In this section, we present an online learning algorithm for constructing the CRFNN. The learning algorithm consists of a structure learning phase and a parameter learning phase. The flow diagram of the learning scheme for the CRFNN model is shown in Fig. 2. The structure learning algorithm determines whether to add a new node, which would satisfy the fuzzy partitioning of the input data. The parameter learning is based upon supervised learning algorithms. The backpropagation algorithm minimizes a given cost function by adjusting the weights in the consequent part, the weights of the feedback, the compensatory degree, and the parameters of the membership functions. Initially, there are no nodes in the network except the input–output nodes, that is, there are no rules or memberships. They are created dynamically and automatically as learning proceeds when online incoming training data are received and when the structure and parameter

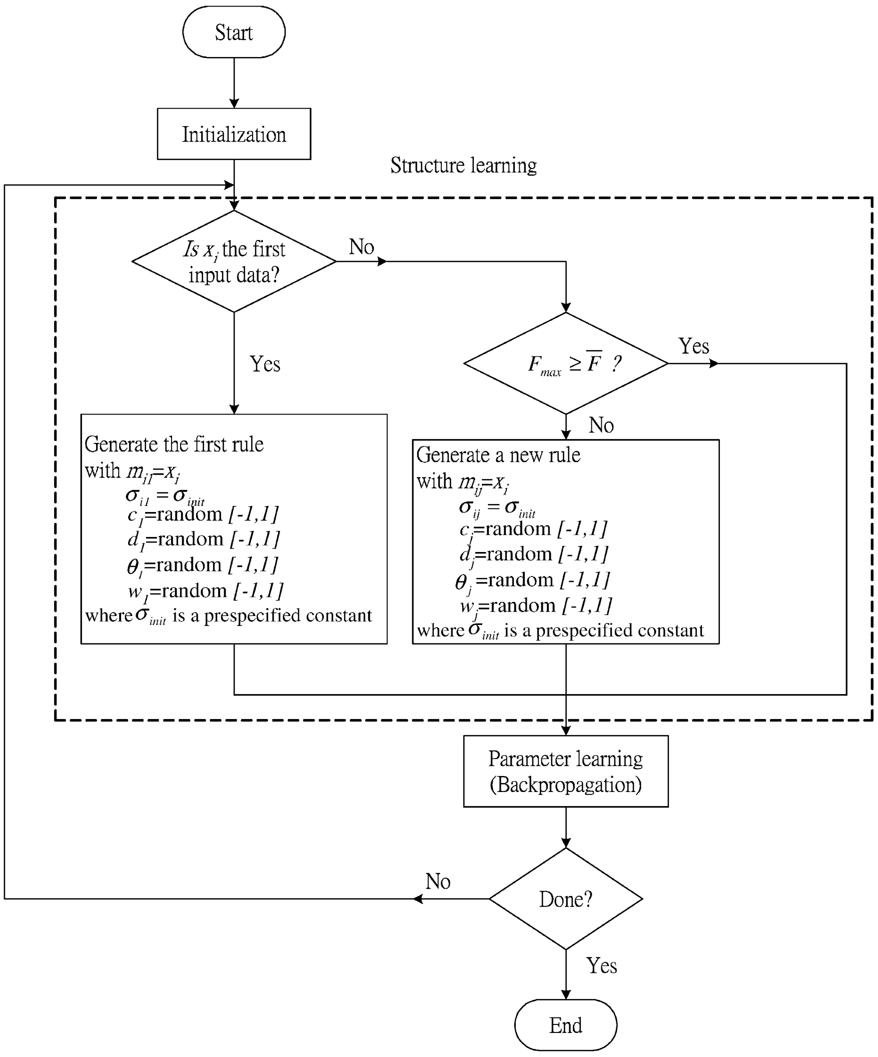
Start
Initialization
Structure learning
Is x_i the first input data?
No
Yes
$F_{max} \geq \overline{F}$?
Yes
No
Generate the first rule with $m_{i1}=x_i$, $\sigma_{i1}=\sigma_{init}$, c_1=random [-1,1], d_1=random [-1,1], θ_1=random [-1,1], w_1=random [-1,1], where σ_{init} is a prespecified constant
Generate a new rule with $m_{ij}=x_i$, $\sigma_{ij}=\sigma_{init}$, c_j=random [-1,1], d_j=random [-1,1], θ_j=random [-1,1], w_j=random [-1,1], where σ_{init} is a prespecified constant
Parameter learning (Backpropagation)
Done?
No
Yes
End

Fig. 2. The flow diagram of the structure/parameter learning for the CRFNN model.

learning processes are performed. The details of the structure learning phase and the parameter learning phase are described in the rest of this section.

4.1. The structure learning phase

The first step in the structure learning is to determine whether to extract a new rule from the training data, as well as to determine the number of fuzzy sets in the universal of discourse of each input variable, since one cluster in the input space corresponds to one potential fuzzy logic rule, with m_{ij} and σ_{ij} representing the mean and variance of that cluster. For each incoming pattern $\underline{x}$, the strength a rule is fired can be interpreted as the degree to which the incoming pattern belongs to the corresponding cluster. For computational efficiency, we can directly use a compensatory operator of the firing strength obtained from $\mu_{A_j}(\underline{x}(t))$ as the degree measure

$$F_j = \mu_{A_j}(\underline{x}(t)) = \left[\prod_{i=1}^{n} \mu_{A_{ij}}(x_i) \cdot s_j\right]^{1-\gamma_j+\gamma_j/n}, \tag{18}$$

where $F_j \in [0,1]$. Using this degree measure, we can obtain the following criterion for the generation of a new fuzzy rule for new incoming data, described as follows. Find the maximum degree $F_{\max}$,

$$F_{\max} = \max_{1 \leqslant j \leqslant R_{(t)}} F_j, \tag{19}$$

where $R_{(t)}$ is the number of existing rules at time t. If $F_{\max} \leqslant \overline{F}$, then a new rule is generated, where $\overline{F} \in (0,1)$ is a prespecified threshold that should decay during the learning process limits the size of the CRFNN.

In the structure learning step, the threshold parameter $\overline{F}$ is an important parameter. The threshold value is set to between zero and one. A low threshold value leads to the learning of coarse clusters (i.e., less rules are generated), whereas a high threshold value leads to the learning of fine clusters (i.e., more rules are generated). If the threshold value is equal to zero, all the training data belong to the same cluster in the input space. Therefore, the selection of the threshold value $\overline{F}$ will critically affect the simulation results, and the value will be based on practical experimentation or on trial-and-error tests.

Once a new rule is generated, the next step is to assign an initial mean and variance for the new membership function. Since our goal is to minimize an objective function, the mean and variance are all adjustable according to the current training pattern in the parameter learning. The mean and variance for the new membership function are set as follows:

$$m_{ij}^{(R_{(t+1)})} = x_i, \tag{20}$$

$$\sigma_{ij}^{(R_{(t+1)})} = \sigma_{\text{init}}, \tag{21}$$

where x_i is the new data and σ_{init} is a prespecified constant. Since the generation of a membership function corresponds to the generation of a new fuzzy rule, the compensatory degree $c_j^{(R_{(t+1)})}$, $d_j^{(R_{(t+1)})}$, i.e., $r_j^{(R_{(t+1)})} = (c_j^{(R_{(t+1)})})^2/(c_j^{(R_{(t+1)})})^2 + (d_j^{(R_{(t+1)})})^2$, the weight of the feedback $\theta_j^{(R_{(t+1)})}$, and the weight of the link $w_j^{(R_{(t+1)})}$ associated with a new fuzzy rule have to be determined. Generally, the compensatory degree $c_j^{(R_{(t+1)})}$, $d_j^{(R_{(t+1)})}$, the weight of the feedback $\theta_j^{(R_{(t+1)})}$, and the weight of the link $w_j^{(R_{(t+1)})}$ are selected with random values in $[-1,1]$.

The whole algorithm for the generation of new fuzzy rules as well as of fuzzy sets in each input variable is as follows. Suppose no rules initially exist:

```
Step 1: IF $x_i$ is the first incoming pattern THEN do
        {Generate a new rule
        with mean $m_{i1} = x_i$, variance $\sigma_{i1} = \sigma_{\text{init}}$, compensatory degree $c_1 = random$,
        $d_1 = random$, weight of feedback $\theta_1 = random$, weight of link $w_1 = random$
        where $\sigma_{\text{init}}$ is a prespecified constant.
        }
Step 2: ELSE for each newly incoming $x_i$, do
        {Find $F_{\max} = \max_{1 \leqslant j \leqslant R_{(t)}} F_j$
        IF $F_{\max} \geqslant \overline{F}$
           do nothing
        ELSE
        {$R_{(t+1)} = R_{(t)} + 1$
         generate a new rule
         with mean $m_{ij}^{R_{(t+1)}} = x_i$, variance $\sigma_{ij}^{R_{(t+1)}} = \sigma_{\text{init}}$, compensatory degree
        $c_j^{R_{(t+1)}} = random$, $d_j^{R_{(t+1)}} = random$, weight of feedback $\theta_j^{R_{(t+1)}} = random$,
        weight of link $w_j^{R_{(t+1)}} = random$
         where $\sigma_{\text{init}}$ is a prespecified constant.}
        }
```

4.2. The parameter learning phase

After the network structure is adjusted according to the current training pattern, the network then enters the parameter learning phase to adjust the parameters of the network optimally based on the same training pattern. Notice that the following parameter learning is performed on the whole network after structure learning, regardless of whether the nodes (links) are newly added or originally exist. Since the learning process involves the determination of the vector that minimizes a given cost function, the gradient of the cost function with respect to the vector is computed, and the vector is adjusted along the negative gradient. When the single output case is considered for clarity, our goal is to minimize the cost function $E(t)$, defined as

$$E(t) = \frac{1}{2}[y^d(t) - y(t)]^2, \tag{22}$$

where $y^d(t)$ is the desired output and $y(t)$ is the current output for each discrete time t. In each training cycle, starting at the input variables, a forward pass is used to calculate the activity of all the current output $y(t)$.

When the backpropagation learning algorithm is used, the weighting vector of the CRFNN is adjusted such that the error defined in Eq. (22) 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

$$W(t+1) = W(t) + \Delta W(t) = W(t) + \eta\left(-\frac{\partial E(t)}{\partial W}\right), \tag{23}$$

where, in this case, η and W represent the learning rate and the tuning parameters of the CRFNN, respectively. Let $e(t) = y^d(t) - y(t)$ and $W = [m, \sigma, \theta, c, d, w]^{\mathrm{T}}$ be the training error and weighting vector of the CRFNN, respectively. Then the gradient of error $E(\cdot)$ in Eq. (22) with respect to an arbitrary weighting vector W is

$$\frac{\partial E(t)}{\partial W} = -e(t)\frac{\partial y(t)}{\partial W}. \tag{24}$$

From recursive applications of the chain rule, the error term for each layer is first computed. Then the parameters in the corresponding layers are adjusted. With the CRFNN and the cost function as defined in Eq. (22), we can derive the update rule for w_j, c_j, d_j, and θ_j as follows:

$$w_j(t+1) = w_j(t) - \eta_w \frac{\partial E(t)}{\partial w_j}, \tag{25}$$

$$c_j(t+1) = c_j(t) - \eta_c \frac{\partial E(t)}{\partial c_j}, \tag{26}$$

$$d_j(t+1) = d_j(t) - \eta_d \frac{\partial E(t)}{\partial d_j}, \tag{27}$$

$$\theta_j(t+1) = \theta_j(t) - \eta_\theta \frac{\partial E(t)}{\partial \theta_j}, \tag{28}$$

where

$$\frac{\partial E(t)}{\partial w_j} = -e(t) \cdot \mu_{A_j}(x(t)),$$

$$\frac{\partial E(t)}{\partial c_j} = -e(t) \cdot w_j \cdot \mu_{A_j}(x(t)) \cdot \ln\left[\prod_{i=1}^{n} \mu_{A_{ij}}(x_i) \cdot s_j\right] \cdot \left(\frac{1}{n} - 1\right) \cdot \left[\frac{2c(t)d^2(t)}{(c^2(t) + d^2(t))^2}\right],$$

$$\frac{\partial E(t)}{\partial d_j} = -e(t) \cdot w_j \cdot \mu_{A_j}(x(t)) \cdot \ln\left[\prod_{i=1}^{n} \mu_{A_{ij}}(x_i) \cdot s_j\right] \cdot \left(\frac{1}{n} - 1\right) \cdot \left[-\frac{2c^2(t)d(t)}{(c^2(t) + d^2(t))^2}\right],$$

$$\frac{\partial E(t)}{\partial \theta_j} = -e(t) \cdot w_j \cdot \left(1 - \gamma_j + \frac{\gamma_j}{n}\right) \cdot \left\{\frac{\prod_{i=1}^{n} \mu_{A_{ij}}(x_i)}{1 + \exp[-\mu_{A_j}(x(t-1)) \cdot \theta_j]}\right\}^{-\gamma_j + \gamma_j/n}$$
$$\cdot \left\{\frac{\prod_{i=1}^{n} \mu_{A_{ij}}(x_i) \cdot \left[\mu_{A_j}(x(t-1)) \cdot \exp[-\mu_{A_j}(x(t-1)) \cdot \theta_j]\right]}{[1 + \exp[-\mu_{A_j}(x(t-1)) \cdot \theta_j]]^2}\right\}.$$

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

$$m_{ij}(t+1) = m_{ij}(t) - \eta_m \frac{\partial E(t)}{\partial m_{ij}}, \tag{29}$$

$$\sigma_{ij}(t+1) = \sigma_{ij}(t) - \eta_\sigma \frac{\partial E(t)}{\partial \sigma_{ij}}, \tag{30}$$

where

$$\frac{\partial E(t)}{\partial m_{ij}} = -e(t) \cdot w_j \cdot \left(1 - \gamma_j + \frac{\gamma_j}{n}\right) \cdot \mu_{A_j}(x(t)) \cdot \left\{\frac{2[x_i - m_{ij}]}{\sigma_{ij}^2}\right\},$$

$$\frac{\partial E(t)}{\partial \sigma_{ij}} = -e(t) \cdot w_j \cdot \left(1 - \gamma_j + \frac{\gamma_j}{n}\right) \cdot \mu_{A_j}(x(t)) \cdot \left\{\frac{2[x_i - m_{ij}]^2}{\sigma_{ij}^3}\right\}.$$

5. Simulation results

We evaluated the performance of the CRFNN for temporal problems. Several examples and performance contrasts with some other recurrent fuzzy neural networks are presented in this section. The parameters (η_m, η_σ, η_θ, η_c, η_d, η_w, σ_{init}, $\overline{F}$) were set in advance, and the number of training epochs for the CRFNN in each example was determined based on the desired accuracy.

Example 1. *Identification of dynamic system.* In this example, a nonlinear plant with multiple time delays is guided by the following differential equation:

$$y_p(t+1) = f(y_p(t), y_p(t-1), y_p(t-2), u_p(t), u_p(t-1)) \tag{31}$$

where

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

Here, the current output of the plant depends on three previous outputs and two previous inputs. In [6], the feedforward neural network, with five input nodes for feeding the appropriate past values of y_p and u were used. In our model, only two input values, $y_p(t)$ and $u(t)$, were fed to the CRFNN to determine the output $y_p(t+1)$. The training inputs were independent and had an identically distributed (i.i.d.) uniform sequence over $[-2, 2]$ for about half of the training time and a single sinusoid signal 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 training sets for each epoch. The check input signal $u(t)$, as shown in the equation below, was used to determine the identification results.

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

In training the CRFNN, we used 10 epochs. Each epoch consisted of 900 time steps. The learning rate $\eta_w = \eta_c = \eta_d = \eta_m = \eta_\sigma = \eta_\theta = 0.05$, $\sigma_{\text{init}} = 0.2$, and the prespecified threshold $\overline{F} = 10^{-4}$ were used. After training, the final root-mean-square error (rms error) was 0.0011, and three fuzzy logic rules were generated. These designed three rules are

Rule 1: IF [$u(t)$ is $\mu(-0.0313, 0.3149)$ and $y(t)$ is $\mu(0.1539, 0.7908)$ and $h(t)$ is $G]^{0.8138}$
THEN $y(t+1)$ is 0.4686 and $h(t+1)$ is 0.4686.
Rule 2: IF [$u(t)$ is $\mu(0.6041, 0.3120)$ and $y(t)$ is μ $(1.0319, 0.4989)$ and $h(t)$ is $G]^{0.6453}$
THEN $y(t+1)$ is 0.9018 and $h(t+1)$ is -0.5938.
Rule 3: IF [$u(t)$ is $\mu(-1.3234, 0.6565)$ and $y(t)$ is $\mu(-1.1754, 0.7360)$ and $h(t)$ is $G]^{0.5324}$
THEN $y(t+1)$ is -1.3784 and $h(t+1)$ is 0.0816.

Fig. 3(a) illustrates the distribution of some of the training patterns and the final assignment of the rules in the $[u(t), y(t)]$ plane. This is due to the parameter learning process, which adjusts the mean and variance

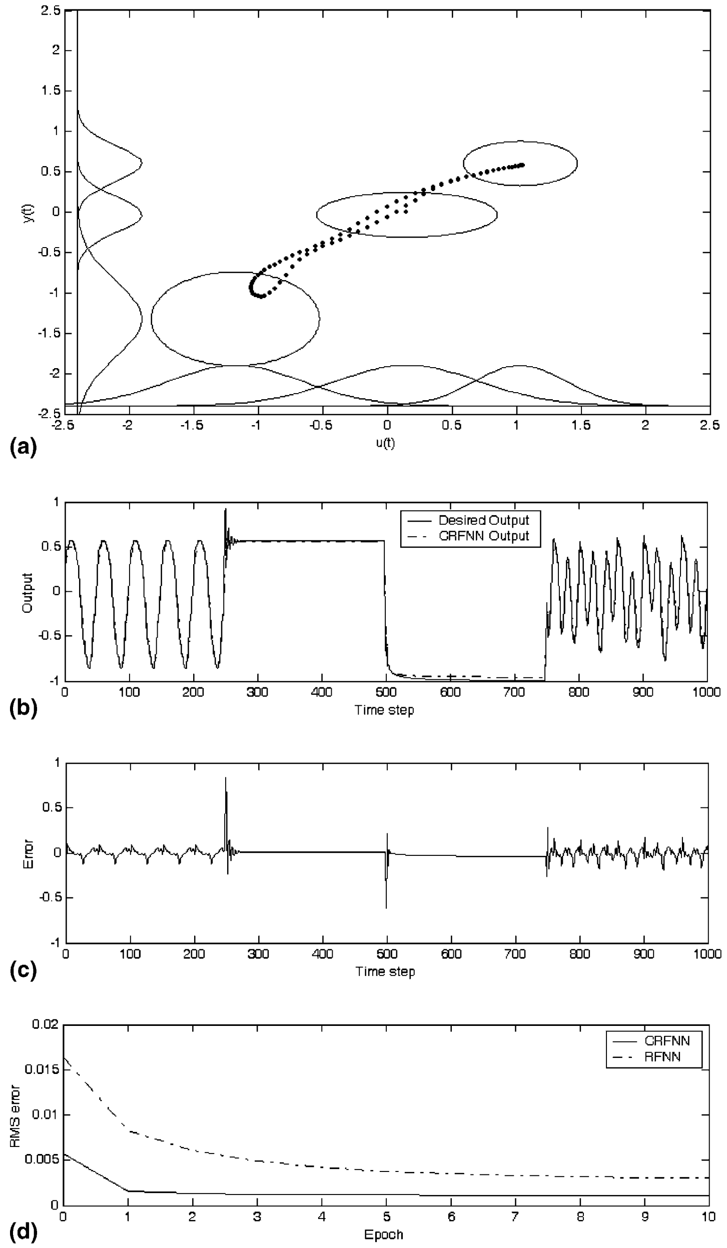

Fig. 3. Simulation results of the CRFNN for dynamic system identification in Example 1. (a) The input training patterns and the final assignment of rules for the distribution of the membership functions on the $y(t)$ and $u(t)$ dimensions. (b) The outputs of the plant and the CRFNN. (c) The error between the CRFNN output and the desired output. (d) The learning curve of the CRFNN and the RFNN [8].

of each membership function at each time step to minimize the output cost function. The membership functions in the $u(t)$ and $y(t)$ dimensions are shown in Fig. 3(a). Fig. 3(b) shows results using the CRFNN for identification. The results show that the CRFNN model has perfect identification capability. Fig. 3(c) illustrates the error between the desired output and the CRFNN output.

Table 1
Performance comparison of various recurrent methods with respect to the identification problem in Example 1

	CRFNN	RSONFIN [7]	RFNN [8]	TRFN-S [9]
Number of parameters	24	36	24	33
RMS error (training)	0.0011	0.0248	0.0030	0.0084
RMS error (testing)	0.0019	0.0780	0.0033	0.0346
Epochs	10	10	10	10

Recently, Lin and Wai [8] proposed a model called the recurrent fuzzy neural network (RFNN) architecture for learning and tuning a fuzzy identifier. Our model is similar to the RFNN except for the rule layer. The rule layer of the RFNN performs the product operation while the rule layer of our model performs compensatory and product operations. Furthermore, users need not give any *a priori* knowledge or even any initial information in our proposed model.

Fig. 3(d) shows the learning curves of the CRFNN model and the RFNN [8] model. We compared the speed (CPU time) of our model with that of the RFNN [8]. The learning time of the CRFNN and RFNN model needed 7.609 and 7.672 seconds, respectively, for ten epochs. The computation time was measured on a personal computer with an Intel Pentium 4 (1500 MHz) CPU inside. In this simulation, the proposed CRFNN model converged faster than the RFNN [8] model.

We also compared the performance of our model with that of other existing recurrent methods (RSONFIN [7], RFNN [8], TRFN-S [9]). The performance indices considered included the numbers of adjustable parameters, the rms error (training and testing), and the numbers of epochs. The comparison results are tabulated in Table 1. As shown in Table 1, the numbers of adjustable parameters and the rms error of the CRFNN are smaller than other recurrent models under the same training epochs.

Example 2. *Identification of a chaotic system.* 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 [15]. This chaotic system is described by

$$y(t+1) = -P\cdot y^2(t) + Q\cdot y(t-1) + 1.0 \quad \text{for } t = 1,2,\ldots \tag{32}$$

which, with $P = 1.4$ and $Q = 0.3$, produces a chaotic strange attractor, as shown in Fig. 4(a). For this training, the input of the CRFNN was $y(t-1)$ and the output was $y(t)$. We first randomly took the training data (1000 pairs) from a system over the interval $[-1.5, 1.5]$. Then the CRFNN was used to approximate the chaotic system.

In applying the CRFNN 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 $\eta_w = \eta_c = \eta_d = \eta_m = \eta_\sigma = \eta_\theta = 0.05$, $\sigma_{\text{init}} = 0.1$, and the prespecified threshold $\overline{F} = 10^{-4}$ were used. After training, 8 fuzzy logic rules were generated. The phase plane of this chaotic system after training for the FNN [5] and the CRFNN are shown in Fig. 4(b) and (c), respectively. From the simulation results shown in Fig. 4(b), we can see that the FNN is inappropriate for chaotic dynamics system because of its static mapping. In Table 2, the comparison shows that the rms error (training and testing) of the proposed model is smaller than the RFNN model and the FNN model. We also compared the speed (CPU time) of our model with those of the RFNN [8] and FNN [5]. The computation time was measured on a personal computer with an Intel Pentium 4 (1500 MHz) CPU inside. From the comparisons, we can find that the proposed CRFNN model only needs fewer CPU time than the compared networks [5,8] under the same adjustable parameters and training epochs.

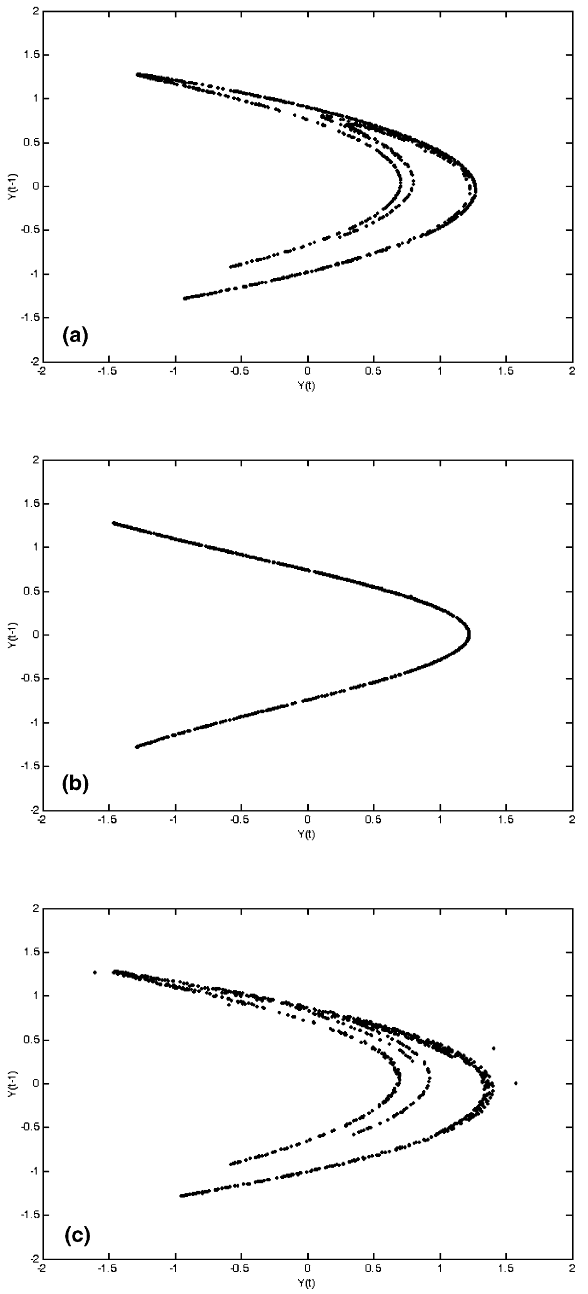

Fig. 4. Simulation results for identification of a chaotic system. (a) Check data of this chaotic system. (b) Result of identification using the FNN for the chaotic system. (c) Result of identification using the CRFNN for the chaotic system.

Example 3. *Control of water bath temperature system.* The goal of this experiment was to control the temperature of a water bath system given by

$$\frac{\mathrm{d}y(t)}{\mathrm{d}t} = \frac{u(t)}{C} + \frac{Y_0 - y(t)}{RC}, \tag{33}$$

Table 2
Performance comparison of various methods with respect to the identification problem in Example 2

	CRFNN	RFNN [8]	RFNN [8]	FNN [5]	FNN [5]
Number of rules	8	8	12	8	16
Number of parameters	48	32	48	24	48
RMS error (training)	0.0034	0.0141	0.0094	0.1338	0.0733
RMS error (testing)	0.0036	0.0145	0.0112	0.1577	0.0748
CPU time (second)	2835.1	2713.5	2995.7	101.6	2836.3
Epochs	100	100	100	100	100

where $y(t)$ is the system output temperature in °C; $u(t)$ is the heat flowing inward the system; Y_0 is the room temperature; C is the equivalent system thermal capacity; and R is the equivalent thermal resistance between the system borders and surroundings.

Assuming that R and C are essentially constant, we can rewrite the system in Eq. (33) as a discrete-time form with some reasonable approximation. The system

$$y(t+1) = \mathrm{e}^{-\alpha Ts} y(k) + \frac{\frac{\delta}{\alpha}(1-\mathrm{e}^{-\alpha Ts})}{1+\mathrm{e}^{0.5y(k)-40}} u(k) + [1-\mathrm{e}^{-\alpha Ts}] y_0 \tag{34}$$

is obtained, where α and δ are some constant values describing R and C. The system parameters used in this example were $\alpha = 1.0015\mathrm{e}^{-4}$, $\delta = 8.67973\mathrm{e}^{-3}$, and $Y_0 = 25.0$ (°C), which were obtained from a real water bath plant in [16]. The input $u(k)$ was limited to 0 and 5 V. The sampling period was $Ts = 30$. The system configuration is shown in Fig. 5, where y_{ref} is the desired temperature of the controlled plant.

When the online training scheme for the CRFNN model was implemented, a sequence of random input signals $u_{rd}(k)$ limited to 0 and 5 V was injected directly into the simulated system described in Eq. (34). The 120 training patterns were chosen from the input–output characteristics in order to cover the entire reference output. The initial temperature of the water was 25 °C, and the temperature rose progressively when random input signals were injected. For the CRFNN model, after 10,000 training iterations, 9 fuzzy rules were generated (see Fig. 5).

For the CRFNN model, two groups of computer simulations were conducted on the water bath temperature control system. Each simulation was performed over 120 sampling time steps. The first task was to control the simulated system to follow three set-points.

$$y_{ref}(k) = \begin{cases} 35\ ^\circ\text{C}, & \text{for } k \leqslant 40, \\ 55\ ^\circ\text{C}, & \text{for } 40 < k \leqslant 80, \\ 75\ ^\circ\text{C}, & \text{for } 80 < k \leqslant 120. \end{cases} \tag{35}$$

The regulation performance of the CRFNN model is shown in Fig. 6.

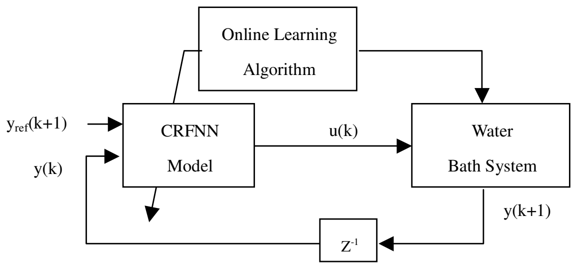

Fig. 5. Flow diagram for using CRFNN model to solve the temperature control problem.

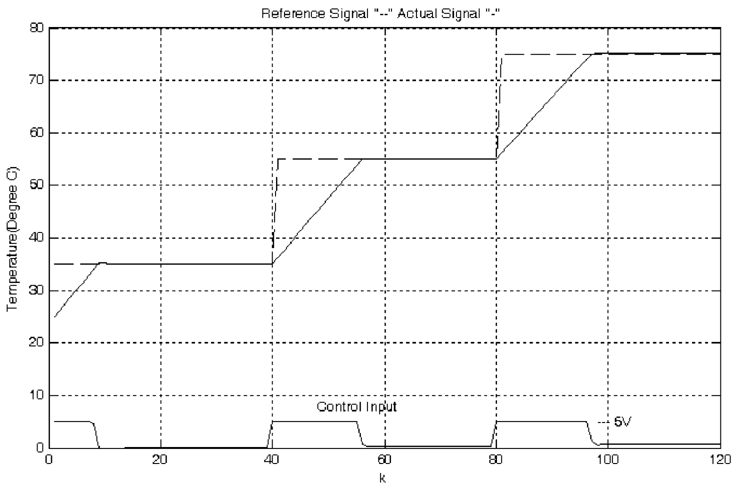

Fig. 6. Final regulation performance of the CRFNN model for a water bath system.

A second set of simulations was carried out to study the noise-rejection ability of the CRFNN model when unknown impulse noise was imposed on the process. An impulse noise value of −5 °C was added to the plant output at the sixtieth sampling instant. A set point of 50 °C was performed in this set of simulations. For the CRFNN model, the same training scheme, training data, and learning parameters used in the first set of simulations were used. The behaviors of the CRFNN model under the influence of impulse noise are shown in Fig. 7.

For the manually designed fuzzy controller [19], the input variables are chosen as $e(t)$ and $ce(t)$, where $e(t)$ is the performance error indicating the error between the desired water temperature and the actual measured temperature and $ce(t)$ is the rate of change in the performance error $e(t)$. The output or the controlled linguistic variable is the voltage signal $u(t)$ to the heater. Seven fuzzy terms are defined for each linguistic variable. These fuzzy terms consist of negative large (NL), negative medium (NM), negative small (NS),

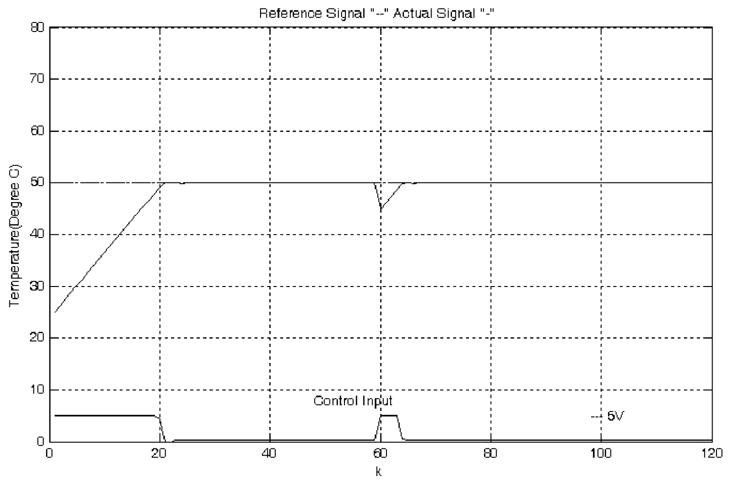

Fig. 7. Behavior of the CRFNN model under the impulse noise for a water bath system.

zero (ZE), positive small (PS), positive medium (PM), and positive large (PL). Each fuzzy term is specified by a Gaussian membership function. According to common sense and engineering judgment, 25 fuzzy rules are specified.

For the manually designed fuzzy controller [19], the numbers of rules and membership functions have to be decided and tuned by hand. For the manually designed fuzzy controller, therefore, they usually require a long time in design for achieving good performance. In the CRFNN controller, however, no controller parameters have to be decided in advance. We only need to choose propose training patterns of the CRFNN controller. Although the structure of CRFNN controller is more complicated than the manually designed fuzzy controller, in general, the CRFNN controller usually spends a relatively short time in design for achieving good performance. This study attempted to emphasize the methodology and control abilities of the proposed CRFNN model. In the future, we will apply the proposed CRFNN model to a real water bath temperature control system.

6. Discussion

In this section, we summarize the features of the proposed CRFNN model. First, compensatory fuzzy operators are used in the CRFNN model. A new adaptive fuzzy reasoning method using compensatory fuzzy operators makes the fuzzy logic system more adaptive and effective. Since the compensatory fuzzy logic system is a universal approximator, we developed a compensatory learning algorithm for the CRFNN model. The CRFNN model cannot only adaptively adjust fuzzy membership functions but can also dynamically optimize the adaptive fuzzy reasoning by using the compensatory learning algorithm. Because the conventional fuzzy neural network can only adjust fuzzy membership functions by using fixed fuzzy operations such as MIN and MAX, the CRFNN model with adaptive fuzzy reasoning is more effective and adaptive than the conventional fuzzy neural network [8] with non-adaptive fuzzy reasoning. Since the compensatory parameters in a fuzzy neural network have physical meanings, these parameters can be initialized by a heuristic algorithm so as to train the model more quickly.

The second feature of the CRFNN model is its distributed representation of the input patterns. This is achieved by the fuzzification process through the adaptive input membership functions. With the adaptive input membership functions, the input space is divided into overlapping small regions and, more importantly, this partitioning is not performed in advance, but is dynamically and appropriately adjusted during the learning process. As a result, each region varies in size and the degree of overlapping between regions is also adjustable. This is in contrast to [2,18,19].

The third feature of the CRFNN model is its dynamic online learning ability to find proper fuzzy logic rules. There are no rules in the CRFNN model initially. They are created dynamically as learning proceeds when online incoming training data are received by performing the following learning processes simultaneously: (1) input space partitioning; (2) fuzzy rules construction; and (3) parameter identification.

The fourth feature of the proposed CRFNN model is its ability to solve temporal problems. Recurrent networks have self-loops and backward connections in their topologies, and these feedback loops are used to memorize past information. Therefore, they can be used to deal with temporal problems. From the simulation results (see Table 2), we can see that the non-recurrent network [5] is inappropriate for recurrent dynamics system because of its static mapping.

7. Conclusions and future works

In this paper, a compensation-based recurrent fuzzy neural network was proposed for identifying dynamic systems. The CRFNN is a recurrent multilayered connectionist network for realizing fuzzy inference using dynamic fuzzy rules. The network consists of four layers, including two hidden layers and a feedback network. The CRFNN was proved to be a universal approximator. An online learning algorithm that consists of structure learning and parameter learning was proposed to automatically construct the CRFNN. The structure learning algorithm determines whether to add a new node, which would satisfy the fuzzy partition of the input data. The backpropagation learning algorithm is used for tuning input membership functions. Simulations demonstrated that the proposed CRFNN model is quite effective in many temporal problems.

Two advanced topics on the proposed CRFNN model should be addressed in future research. First, since the backpropagation technique is used to minimize the error function, the results may reach the local minima solution. We will adopt genetic algorithms (GA) to solve the local minima problem. GA is a parallel and global search technique. Because it simultaneously evaluates many points in the search space, it is more likely to converge toward the global solution. Second, it would be better if the CRFNN model has the ability to delete unnecessary or redundant rules. The fuzzy similarity measure [1] determines the similarity between two fuzzy sets in order to prevent existing membership functions from being too similar.

Acknowledgment

This research is supported by the National Science Council of ROC under grant NSC 93-2213-E-324-008.

Appendix A. Proof of the universal approximation theorem

Theorem 1 will be proven using the Stone–Weierstrass theorem. The structure of the proposed CRFNN is illustrated in Fig. 1. The single output of the CRFNN can be expressed as

$$y(\underline{x}) = \sum_{j=1}^{R} \mu_{A_j}(\underline{x}) \cdot w_j, \tag{A.1}$$

where $\underline{x} \in \Re^N$ is the input variable of the CRFNN,

$$\mu_{A_j}(\underline{x}) = \left[\prod_{i=1}^{N} \mu_{A_{ij}}(x_i) \cdot s_j\right]^{1-\gamma_j+\frac{\gamma_j}{N}}, \tag{A.2}$$

where

$$\mu_{A_{ij}}(x_i) = \exp\left(-\frac{[x_i(t) - m_{ij}]^2}{\sigma_{ij}^2}\right), \tag{A.3}$$

$$s_j = \frac{1}{1+\exp(-h_j)}, \tag{A.4}$$

where $h_j = \mu_{A_j}(\underline{x}(t-1)) \cdot \theta_j$ denotes the input of layer 3, the link weight w_j is the output action strength, the $m_{ij}, \sigma_{ij} \in \Re$, and Y is the family of function $y : \Re^N \to \Re$.

Theorem 2: *Stone–Weierstrass Theorem* [17]. *Let A be a set of real continuous functions on a compact set U. If* (1) *U is an algebra. That is, if* $f_1, f_2 \in A$, *and* $c \in \Re$, *then* $f_1 + f_2 \in A$, $f_1 \cdot f_2 \in A$, *and* $cf_1 \in A$; (2) *A separates points on U, i.e., for* $x, y \in U$, $x \neq y$, *there exists* $f_1 \in A$ *such that* $f_1(x) \neq f_2(y)$; *and* (3) *A vanishes at* *no point of U, i.e., for each $x \in U$ there exists $f_1 \in A$ such that $f_1(x) \neq 0$, then the uniform closure of A consists of all real continuous functions on U.*

Lemma 1. *Let Y be the family of y defined in* (A.1), *then $Y \subset U$, where U is a compact set.*

Proof of Lemma 1. Here, the membership function

$$0 < \mu_{A_{ij}}(x) = \exp\left[-\frac{(x_i - m_{ij})^2}{(\sigma_{ij})^2}\right] \leqslant 1$$

and, therefore, the continuous function $\mu_{A_j}(x)$ is closed and bounded for all $x \in \Re^N$. That is, $Y \subset A$. □

Proof of Theorem 1. First, we prove that Y is an algebra. Let $f_1, f_2 \in Y$, so that we can write them as

$$f_1(x) = \sum_{j_1=1}^{R_1} w1_{j_1} \cdot \mu1_{A_{j_1}}(\underline{x_1}) = \sum_{j_1=1}^{R_1} w1_{j_1} \cdot \left[\prod_{i_1=1}^{N_1} \mu1_{A_{i_1 j_1}}(x_{i_1}) \cdot s_j\right]^{1-\gamma1_{j_1}+\gamma1_{j_1}/N_j}$$
$$= \sum_{j_1=1}^{R_1} w1_{j_1} \cdot \left[\frac{\prod_{i_1=1}^{N_1} \exp\left[-\frac{\left(x_{i_1}(t) - m1_{i_1 j_1}\right)^2}{\sigma1^2_{i_1 j_1}}\right]}{1 + \exp\left[-\mu1_{A_{j_1}}(x_{i_1}(t-1)) \cdot \theta1_{j_1}\right]}\right]^{1-\gamma1_{j_1}+\gamma1_{j_1}/N_1}, \tag{A.5}$$

$$f_2(x) = \sum_{j_2=1}^{R_2} w2_{j_2} \cdot \mu2_{A_{j_2}}(\underline{x}) = \sum_{j_2=1}^{R_2} w2_{j_2} \cdot \left[\prod_{i_2=1}^{N_2} \mu2_{A_{i_2 j_2}}(x_{i_2}) \cdot s_j\right]^{1-\gamma2_{j_2}+\gamma2_{j_2}/N_2}$$
$$= \sum_{j_2=1}^{R_2} w2_{j_2} \cdot \left[\frac{\prod_{i_2=1}^{N_2} \exp\left[-\frac{\left(x_{i_2}(t) - m2_{i_2 j_2}\right)^2}{\sigma2^2_{i_2 j_2}}\right]}{1 + \exp\left[-\mu2_{A_{j_2}}(x_{i_2}(t-1)) \cdot \theta2_{j_2}\right]}\right]^{1-\gamma2_{j_2}+\gamma2_{j_2}/N_2}, \tag{A.6}$$

where $w1_j$ and $w2_j \in \Re$, we therefore have

$$f_1(x) + f_2(x) = \sum_{j_1=1}^{R_1} w1_{j_1}\mu1_{A_{j_1}}(\underline{x}) + \sum_{j_2=1}^{R_2} w2_{j_2}\mu2_{A_{j_2}}(\underline{x}) = \sum_{j_1=1}^{R_1}\sum_{j_2=1}^{R_2}\left(w1_{j_1}\mu1_{A_{j_1}}(\underline{x}) + w2_{j_2}\mu2_{A_{j_2}}(\underline{x})\right). \tag{A.7}$$

Since $\mu1_{A_j}$ and $\mu2_{A_j}$ are Gaussian in form, i.e., this can be verified by straightforward algebraic operations; hence, (A.7) is in the same form as (A.1), so that $f_1 + f_2 \in Y$. Similarly, we have

$$f_1(x)f_2(x) = \sum_{j_1=1}^{R_1} w1_{j_1}\mu1_{A_{j_1}}(\underline{x}) \cdot \sum_{j_2=1}^{R_2} w2_{j_2}\mu2_{A_{j_2}}(\underline{x}) = \sum_{j_1=1}^{R_1}\sum_{j_2=1}^{R_2}\left(w1_{j_1}\mu1_{A_{j_1}}(\underline{x}) \cdot w2_{j_2}\mu2_{A_{j_2}}(\underline{x})\right) \tag{A.8}$$

which is also in the same form of (A.1); hence, $f_1 f_2 \in Y$. Finally, for arbitrary $c \in \Re$

$$c \cdot f_1(x) = \sum_{j_1=1}^{R_1} c \cdot w1_{j_1}\mu1_{A_{j_1}}(\underline{x}) \tag{A.9}$$

which is again in the form of (A.1); hence, $c \cdot f_1 \in Y$. Therefore, Y is an algebra.

Next, we prove that Y separates points on U. We prove this by constructing a required f; i.e., we specify $f \in Y$ such that $f(\underline{x}') \neq f(\underline{y}')$ for arbitrarily given $\underline{x}', \underline{y}' \in U$ with $\underline{x}' \neq \underline{y}'$. We choose two fuzzy rules in the form of Eq. (14) for the fuzzy rule base. Let $\underline{x}' = (x'_1, x'_2, \ldots, x'_N)$ and $\underline{y}' = (y'_1, y'_2, \ldots, y'_N)$. If $x'_i \neq y'_i$, we can choose two fuzzy rules as the fuzzy rule base. Furthermore, let the Gaussian membership functions be

$$\mu_{A_{i1}}(x_i) = \exp\left(-\frac{(x_i - x'_i)^2}{\sigma^2}\right), \tag{A.10}$$

$$\mu_{A_{i2}}(x_i) = \exp\left(-\frac{(x_i - y'_i)^2}{\sigma^2}\right). \tag{A.11}$$

Then f can be expressed as

$$f = w_1 \cdot \left[\prod_{i=1}^{N} \exp\left(-\frac{x_i - x'_i)^2}{\sigma^2}\right)\right]^{1-\gamma_1+\frac{\gamma_1}{N}} + w_2 \cdot \left[\prod_{i=1}^{N} \exp\left(-\frac{(x_i - y'_i)^2}{\sigma^2}\right)\right]^{1-\gamma_2+\frac{\gamma_2}{N}}, \tag{A.12}$$

where w_1, w_2 are the link weights. With this system, we have

$$f(\underline{x}') = w_1 + w_2 \cdot \left[\prod_{i=1}^{N} \exp\left(-\frac{(x_i - y'_i)}{\sigma^2}\right)\right]^{1-\gamma_2+\frac{\gamma_2}{N}}, \tag{A.13}$$

$$f(\underline{y}') = w_1 \cdot \left[\prod_{i=1}^{N} \exp\left(-\frac{(y'_i - x'_i)}{\sigma^2}\right)\right]^{1-\gamma_1+\frac{\gamma_1}{N}} + w_2. \tag{A.14}$$

Since $\underline{x}' \neq \underline{y}'$, there must be some i such that $x'_i \neq y'_i$, hence $f(\underline{x}') \neq f(\underline{y}')$. Therefore, Y separates points on U.

Finally, we prove that Y vanishes at no point of U. By (A.1), $\mu_{A_j}(x)$ are constant and not equal to zero. That is, for all $x \in \Re^N$, $\mu_{A_j}(x) > 0$. If we choose $w_j > 0$ $(j = 1, 2, \ldots, R)$, then $y > 0$ for any $x \in \Re^N$. That is, any $y \in Y$ with $w_j > 0$ can serve as the required f. $\square$

In summary, the CRFNN is a universal approximation. Using the *Stone–Weierstrass theorem* and the fact that Y is a set of real continuous on U, we have proven the theorem.

References

[1] C.T. Lin, C.S.G. Lee, Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent System, Prentice-Hall, Englewood Cliffs, NJ, 1996.

[2] J.S.R. Jang, ANFIS: Adaptive-network-based fuzzy inference system, IEEE Transactions on Systems, Man and Cybernetics 23 (3) (1993) 665–685.

[3] C.J. Lin, C.T. Lin, Reinforcement learning for an ART-based fuzzy adaptive learning control network, IEEE Transactions on Neural Networks 7 (1996) 709–731.

[4] C.F. Juang, C.T. Lin, An on-line self-constructing neural fuzzy inference network and its applications, IEEE Transactions on Fuzzy Systems 6 (1) (1998) 12–31.

[5] F.J. Lin, C.H. Lin, P.H. Shen, Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive, IEEE Transactions on Fuzzy Systems 9 (2001) 751–759.

[6] K.S. Narendra, K. Parthasarathy, Identification and control of dynamical systems using neural networks, IEEE Transactions on Neural Networks 1 (1) (1990) 4–27.

[7] C.F. Juang, C.T. Lin, A recurrent self-organizing neural fuzzy inference network, IEEE Transactions on Neural Networks 10 (4) (1999) 828–845.

[8] F.J. Lin, R.J. Wai, Hybrid control using recurrent fuzzy neural network for linear-induction motor servo drive, IEEE Transactions on Fuzzy Systems 9 (1) (2001) 102–115.

[9] C.F. Juang, A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms, IEEE Transactions on Fuzzy Systems 10 (2) (2002) 155–170.
[10] H.J. Zimmermann, P. Zysno, Latent connective in human decision, Fuzzy Sets and Systems 4 (1998) 31–51.
[11] Y.Q. Zhang, A. Kandel, Compensatory neurofuzzy systems with fast learning algorithms, IEEE Transactions on Neural Networks 9 (1) (1998) 83–105.
[12] H. Seker, D.E. Evans, N. Aydin, E. Yazgan, Compensatory fuzzy neural networks-based intelligent detection of abnormal neonatal cerebral Doppler ultrasound waveforms, IEEE Transactions on Information Technology in Biomedicine 5 (3) (2001) 187–194.
[13] C.J. Lin, C.H. Chen, Nonlinear system control using compensatory neuro-fuzzy networks, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E86-A (9) (2003) 2309–2316.
[14] C.J. Lin, W.H. Ho, A pseudo-Gaussian-based compensatory neural fuzzy system, Proceedings of the IEEE International Conference on Fuzzy Systems (2003) 214–220.
[15] G. Chen, Y. Chen, H. Ogmen, Identifying chaotic system via a Wiener-type cascade model, IEEE Transactions on Control Systems Technology (1997) 29–36.
[16] J. Tanomaru, S. Omatu, Process control by on-line trained neural controllers, IEEE Transactions on Industrial Electronics 39 (1992) 511–521.
[17] W. Rudin, Principles of Mathematical Analysis, third Ed., McGraw-Hill, New York, 1976.
[18] L.X. Wang, J.M. Mendel, Generating fuzzy rules by learning from examples, IEEE Transactions on Systems, Man and Cybernetics 22 (6) (1992) 1414–1427.
[19] E.H. Mamdani, S. Assilian, An experiment in linguistic synthesis with a fuzzy logic controller, International Journal of Man-Machine Studies 7 (1) (1975) 1–13.