DYNAMIC SYSTEM IDENTIFICATION USING PSEUDO-GAUSSIAN-BASED RECURRENT COMPENSATORY FUZZY NEURAL NETWORKS


ABSTRACT

In this paper, a Pseudo-Gaussian-based Recurrent Compensatory Fuzzy Neural Network (PG-RCFNN) is proposed for identification of dynamic systems. The recurrent network is embedded in the PG-RCFNN by adding feedback connections in the second layer, where the feedback units act as memory elements. The compensatory-based fuzzy reasoning method is using adaptive fuzzy operations of fuzzy neural networks that can make the fuzzy logic systems more adaptive and effective. The Pseudo-Gaussian membership function can provide the proposed model, which own a higher flexibility and can approach the optimal result more accurately. An on-line learning algorithm that consists of structure learning and parameter learning is proposed to automatically construct the PG-RCFNN. The structure learning is based on the degree measure and the parameter learning is based on the supervised gradient decent method. Computer simulations have been conducted to illustrate the performance and applicability of the proposed model.

Key Words: recurrent fuzzy neural networks, identification, compensatory operation, Pseudo-Gaussian, backpropagation.

I. INTRODUCTION

In recent years, using the learning abilities of neural networks to automate and realize the design of fuzzy logic systems has become a very active research area (Lin and Lee, 1996; Paul and Kumar, 2002; Lin et al., 2001; Jang et al., 1997). The integration brings the low-level learning and computation power of neural networks into fuzzy logic systems and provides the high-level human-like thinking and reasoning of fuzzy logic systems into neural networks. Such synergy of integrating neural networks and fuzzy logic into a functional system provides a new direction for the realization of intelligent systems for various applications. However, a major drawback of the fuzzy neural network (FNN) is that its application domain is limited to static problems due to its feedforward network structure. Processing temporal problems using the FNN is inefficient. Hence, many researchers (Lee and Teng, 2000; Zhang and Morris, 1999; Juang and Lin, 1999; Mastorocostas and Theocharis, 2002; Juang, 2002) have proposed recurrent fuzzy neural networks (RFNN) for various temporal problems.

Many papers (Lin and Lee, 1996; Paul and Kumar, 2002; Lin et al., 2001; Jang et al., 1997; Lee and Teng, 2000; Zhang and Morris, 1999; Juang and Lin, 1999; Mastorocostas and Theocharis, 2002; Juang, 2002) have dealt with how to optimize fuzzy membership functions and how to choose an optimal defuzzification scheme for an application by using learning algorithms to adjust the parameter of fuzzy membership functions and defuzzification functions. Unfortunately, for the problem of how to optimize fuzzy logic reasoning and select optimal fuzzy operators, the static fuzzy operators are often used to make fuzzy reasoning. Commonly, while the conventional FNN can only adjust fuzzy membership...
functions by using fixed fuzzy operations such as Min and Max, the compensatory fuzzy neural network (Lin and Chen, 2003; Zhang and Kandel, 1998; Seker et al., 2001) with adaptive fuzzy reasoning is more effective and more adaptive than the conventional FNN with non-adaptive fuzzy reasoning.

In spite of the wide use of the standard symmetric Gaussian membership functions, we choose the asymmetric function acting as input linguistic node in this paper. Since the asymmetric Gaussian membership function’s variability and malleability are higher than the traditional one, the Pseudo Gaussian (PG) membership function can provide a fuzzy neural network which owns a higher flexibility and can approach the true result more easily (Rojas et al., 1999; Rojas et al., 2000; Ouyang and Lee, 1999).

In this paper, a Pseudo-Gaussian-based Recurrent Compensatory Fuzzy Neural Network (PG-RCFNN) is proposed, which is a multilayered connectionist network. The Pseudo-Gaussian (PG) membership function is an asymmetric Gaussian membership function acting as input linguistic node. Its variability and malleability are higher than traditional functions that can partition input space more certainly. Adding feedback connections in the second layer of the PG-RCFNN develops the temporal relations embedded in the PG-RCFNN. It provides the memory elements for the PG-RCFNN and expands the ability of the FNN to include temporal problems. The concept of compensation is used in the third layer of the PG-RCFNN to make the system more adaptive and more effective. An on-line learning algorithm is proposed to automatically construct the PG-RCFNN. It consists of structure learning and parameter learning. The structure learning algorithm decides to add a new node which is satisfying the fuzzy partition of the input data. The backpropagation learning algorithm is used for tuning input/output membership functions. Therefore, the proposed PG-RCFNN model has three advantages. First, it does not require the human expert’s assistance and its structure is obtained from the input data. Second, only a small number of tuning parameters is required. Third, it can obtain a smaller RMS error for dynamic system identification.

This paper is organized as follows. In Section II, we shall describe the structure of the proposed Pseudo-Gaussian-based Recurrent Compensatory Fuzzy Neural Networks (PG-RCFNN). In Section III, the on-line learning algorithm of the PG-RCFNN is presented. Simulation results are discussed in Section IV. Section V gives the conclusion of this paper.

II. STRUCTURE OF THE PG-RCFNN MODEL

The general fuzzy if-then rule form (Lin and Lee, 1996) is shown as follows

Rule—j: IF $x_1$ is $A_{ij}$ and ... and $x_n$ is $A_{nj}$, THEN $y$ is $b_j$  \hspace{1cm} (1)

where for $j=1,2,\ldots,R$, $x_i$ and $y$ are the input for $i=1,2,\ldots,n$ and output variable, respectively, $n$ is the number of input variables, $A_j$ is the linguistic term of the precondition part, $b_j$ is the constant consequent part and $R$ is the number of existing rules.

Zhang and Kandel (1998) proposed more extensive compensatory operations based on pessimistic operation and optimistic operation. The pessimistic operation (such as $t$-norm) can map the inputs $x_i$ to the pessimistic output by making a conservative decision for the pessimistic situation or even the worst case. For example, $p(x_1, x_2, \ldots, x_n)=\min(x_1, x_2, \ldots, x_n)$ or $\Pi x_i$. The optimistic operation (such as $t$-conorm) can map the inputs $x_i$ to the optimistic output by making an optimistic decision for the optimistic situation or even the best case. For example, $o(x_1, x_2, \ldots, x_n)=\max(x_1, x_2, \ldots, x_n)$. The compensatory operation can map the pessimistic input $x_1$ and the optimistic input $x_2$ to make a compromise decision for the situation between the worst case and the best case. For example, $c(x_1, x_2)=x_1\bar{x}_2^\gamma$, where $\gamma \in [0, 1]$ is called the compensatory degree. Many researchers (Lin and Chen, 2003; Zhang and Kandel, 1998; Seker et al., 2001) have used the compensatory operation for fuzzy systems successfully.

For a preconditioned fuzzy set $A^\prime$ in $U$, the $j$th fuzzy rule (1) can map a consequent fuzzy set $b^\prime_j$ in $V$ by using the sup-dot composition

$$\mu_{b_j} = \sup_{\bar{x} \in U} \left[ \mu_{A_{ij} \times \cdots \times A_{nj}}(\bar{x}) \cdot \mu_A(\bar{x}) \right]$$  \hspace{1cm} (2)

Suppose $\bar{x}=(x_1, x_2, \ldots, x_n)$ and the $\mu_{A_{ij} \times \cdots \times A_{nj}}(\bar{x})$ is defined in a compensatory operation form (3) using the pessimistic operation (4) and the optimistic operation (5).

$$\mu_{A_{ij} \times \cdots \times A_{nj}}(\bar{x})=\left(\gamma u_j^\gamma \right)^\gamma$$ \hspace{1cm} (3)

with a compensatory degree $\gamma \in [0, 1]$

$$u_j = \prod_{i=1}^n \mu_{A_{ij}}(x_i)$$ \hspace{1cm} (4)

$$v_j = \left(\prod_{i=1}^n \mu_{A_{ij}}(x_i)\right)^{1/n}$$ \hspace{1cm} (5)

For simplicity, we have

$$\mu_{A_{ij} \times \cdots \times A_{nj}}(\bar{x}) = \left(\prod_{i=1}^n \mu_{A_{ij}}(x_i)\right)^{1-\gamma}
\gamma \mu_{b_{\gamma_j}}(y)$$ \hspace{1cm} (6)

Since $\mu_A(x_i)=1$ for the singleton fuzzifier and $\mu_{b_j}(y)=1$, according to (2) we have
\[ \mu_{b_j}(y) = \left( \prod_{i=1}^{n} \mu_{A_{ij}}(x_i) \right)^{1-\gamma_j/n} \]  
\[ (7) \]

Therefore, the new fuzzy if-then rule is shown as follows:

**Rule–j**: IF \( x_1 \) is \( A_{l_1} \) and ... and \( x_n \) is \( A_{l_n} \),

THEN \( y \) is \( b_j \)  
\[ (8) \]

Actually, the input of each membership function is the network input \( x_i \) plus the temporal term \( O_{ij}^{(2)}(t-1) \theta_{ij} \). Therefore, the fuzzy system with its memory, i.e., terms feedback units, can be considered a dynamic fuzzy inference system. Thus, the fuzzy if-then rule can be rewritten as follows:

**Rule–j**: IF \( u_{ij}^{(2)} \) is \( A_{l_1} \) and ... and \( u_{nj}^{(2)} \) is \( A_{l_n} \),

THEN \( y \) is \( w_j \)  
\[ (9) \]

where \( u_{ij}^{(2)}(t)=x_i(t)+O_{ij}^{(2)}(t-1) \theta_{ij} \), \( A_{l_1} \) is the linguistic term of the precondition part, \( w_j \) is the constant consequent part, \( \gamma \) is called the compensatory degree, and \( n \) is the number of input dimensions.

The membership function of the preconditioned part in this paper is different from the typical Gaussian membership function. We adopt the Pseudo-Gaussian (PG) membership function to approximate excellent results. The definition of PG membership function is as follows:

\[ \mu_{A_{l_j}}(x_i) = \exp\left( -\frac{(x_i-m_{ij})^2}{2\sigma_{ij}^2} \right)U(x_i; m_{ij}, \infty) + \exp\left( -\frac{(x_i-m_{ij})^2}{2\sigma_{ij}^2} \right)U(x_i; -\infty, m_{ij}) \]
\[ (10) \]

where \( U(x_i; a, b) = \begin{cases} 1, & \text{if } a \leq x_i < b \\ 0, & \text{otherwise} \end{cases} \)

The \( m_{ij}, \sigma_{ij}^-, \) and \( \sigma_{ij}^+ \) are the mean, negative deviation and positive deviation of PG membership function. The PG membership function is asymmetric and it has great flexibility.

Next, we shall introduce the operation functions of the node in each layer for the PG-RCFNN. The structure of the proposed PG-RCFNN model is shown in Fig. 1. In the following description, \( u^{(0)} \) and \( O^{(0)} \)
denote the input and output of a node in the lth layer, respectively.

Layer 1 (Input Layer): Each node in this layer only transmits input signals to the next layer directly, i.e.,
\[ O^{(1)}_i = u^{(1)}_i = x_i \]  (11)
where \( i = 1, 2, \ldots, n \). The link weight at this layer is unity.

Layer 2 (Linguistic Term Layer): Each node in this layer performs a PG membership function (11) which can specify the strength of the input-linguistic variables. The function is represented as
\[ O^{(2)}_{ij} = \exp\left(-\frac{(u^{(2)}_{ij} - m_{ij})^2}{\sigma_{ij}^2}\right)U(u^{(2)}_{ij}; -\infty, m_{ij}) \]
\[ + \exp\left(-\frac{(u^{(2)}_{ij} - m_{ij})^2}{\sigma_{ij}^2}\right)U(u^{(2)}_{ij}; m_{ij}, \infty) \]  (12)
where the subscript \( ij \) indicates the \( j \)th term of the \( i \)th input \( x_i \). In addition, the inputs of this layer for discrete time \( t \) can be defined by
\[ u^{(2)}_{ij}(t) = O^{(1)}_i(t) + O^{(2)}_{ij}(t-1) \cdot \theta_{ij} \]  (13)
where \( O^{(2)}_{ij}(t-1) \) are the memory terms which are the input of this layer, which store the past information of the network, and \( \theta_{ij} \) denotes the link weight of the feedback unit.

Layer 3 (Compensatory Rule Layer): Each node in this layer represents the preconditioned part of one fuzzy logic rule that can receive the one-dimensional membership degrees of the associated rule from nodes of a set in Layer 2. Then we use a compensatory, all ready operator, mentioned to perform condition matching of fuzzy rules. As a result, the output function of each inference node is
\[ O^{(3)}_j = (\prod_i u^{(3)}_i)^{1 - \gamma_j \cdot \gamma_j/n} \]  (14)
where the \( \prod_i u^{(3)}_i \) of a rule node represents the firing strength of its corresponding rule. By tuning \( \gamma_j \), the compensatory operator becomes more adaptive.

Layer 4 (Output Layer): Each node in this layer is labeled \( \Sigma \) and its sums all incoming signals to obtain the final inferred result
\[ y_k = O^{(4)}_k = \sum_j u^{(4)}_j w_{jk} \]  (15)
where the weight \( w_{jk} \) is output action strength of the \( k \)th output associated with the \( j \)th rule and \( O^{(4)}_k \) is the \( k \)th output of the PG-RCFNN.

III. THE ON-LINE LEARNING ALGORITHM

In this section, we present an on-line learning algorithm, which consists of structure learning phase and parameter learning phase, to construct the PG-RCFNN. The structure learning is used to decide proper fuzzy partitions and find fuzzy logic rules subject to two objectives: to minimize the number of rules generated and to minimize the number of fuzzy sets in the universe of discourse of each input variable. The parameter learning is based on supervised learning algorithms. The link weights in the consequent part, the parameters of PG membership functions, the link weights of the feedback unit, and the compensatory degrees are adjusted by the backpropagation algorithm to minimize a given cost function. Initially, there are not rule nodes or any membership except the input and output nodes in the proposed PG-FCFNN model. They are created dynamically and automatically in the learning process, receiving the on-line incoming training data by performing the structure and parameter learning processes.

1. The Structure Learning Phase

The structure learning phase is to determine whether or not to extract a new rule from training data as well as the number of fuzzy sets in the universe of discourse of each input variable. For computational efficiency, we can use a compensatory operation of the firing strength obtained directly as this degree measure
\[ P_j = (\prod_i u^{(3)}_i)^{1 - \gamma_j \cdot \gamma_j/n} \]  (16)
where \( P_j \in [0, 1] \). According to the degree measure, the criterion for generating a new fuzzy rule for new incoming data is described as follows. We are to find the maximum degree \( P_{\text{max}} \)
\[ P_{\text{max}} = \max_{1 \leq j \leq R(t)} P_j \]  (17)
where \( R(t) \) is the number of existing rules at time \( t \). If \( P_{\text{max}} \leq P \), then a new rule is generated where \( P \in (0, 1) \) is a prespecified threshold that should decay during the learning process to limit the size of PG-RCFNN. Once a new rule is generated, it assigns initial mean, deviation, and weight of feedback for the new membership function. Thus, the mean, deviation, and weight of feedback for the new membership function are selected as follows:
\[ m^{R(t), i} = x_i \]  (18)
\[ \sigma^{R(t), i} = \sigma^{R(t), i} = \sigma_{\text{init}} \]  (19)
The parameter learning phase

After the network structure is adjusted according to the current training pattern, the network enters the parameter learning phase to adjust the parameters of the PG-RCFNN optimally, based on the same training pattern. The learning process involves the determination to minimize a given cost function. The gradient of the cost function is adjusted along the negative gradient. The online parameter learning algorithm of the PG-RCFNN uses the supervised gradient descent method by backpropagation algorithm. Considering the single output case for clarity, our goal is to minimize the cost function $E$ that is defined as

$$E = \frac{1}{2}(y^d(t) - y(t))^2 = \frac{1}{2}e(t)^2$$

where $y^d(t)$ is the desired output and $y(t)$ is the current output. Then the parameter learning algorithm is based on backpropagation, described in the following:

**Layer 4:** The link weight is updated by the amount

$$\Delta w_j = -\frac{\partial E}{\partial w_j} = -(\frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial w_j}) = e \cdot u^{(4)}_j$$

The link weights in layer 4 are updated according to the following equation:

$$w_j(t+1) = w_j(t) + \eta_n \Delta w_j$$

where $\eta_n$ is the learning rate parameter of the link weight and $t$ denotes the iteration number of the $j$th link.

**Layer 3:** To eliminate the constraint $\gamma \in [0, 1]$, we redefine $r$ as follows:

$$\gamma_j = \frac{c_j^2}{c_j^2 + d_j^2}$$

then we have

$$\Delta \theta_j = -\frac{\partial E}{\partial \theta_j} = -(\frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial u^{(4)}_j} \cdot \frac{\partial u^{(4)}_j}{\partial \theta_j})$$

$$= e \cdot w_j \cdot u^{(4)}_j \cdot (1 - \frac{r}{n}) \cdot \ln\left(\prod_{i=1}^{n} u^{(3)}_i\right)$$

The compensatory degree $c_j$ and $d_j$ are updated according to the following equation:

$$c_j(t + 1) = c_j(t) + \eta_c (\frac{2c_j(t)d_j(t)}{(c_j^2(t) + d_j^2(t))^2}) \Delta \gamma_j$$

$$d_j(t + 1) = d_j(t) - \eta_d (\frac{2c_j^2(t)d_j(t)}{(c_j^2(t) + d_j^2(t))^2}) \Delta \gamma_j$$

$$\gamma(t + 1) = \frac{c_j^2(t + 1)}{c_j^2(t + 1) + d_j^2(t + 1)}$$

In all above formulas, $\eta_c$ and $\eta_d$ are the learning rate of the $c_j$ and $d_j$, respectively.

**Layer 2:** The mean is updated by the amount

$$\Delta m_{ij} = -\frac{\partial E}{\partial m_{ij}} = -(\frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial m_{ij}} \cdot \frac{\partial m_{ij}}{\partial u^{(3)}_j} \cdot \frac{\partial u^{(3)}_j}{\partial \theta_j})$$

$$= e \cdot w_j \cdot (1 - \gamma_j + \gamma_j/n) \cdot \prod_{i=1}^{n} u^{(3)}_i \cdot (u^{(2)}_i - m_{ij}) \frac{2u^{(2)}_i - m_{ij}}{\sigma_{ij}^2} \cdot \exp(-\frac{u^{(2)}_i - m_{ij}}{\sigma_{ij}^2})$$

$$\cdot U(u^{(2)}_i; -\infty, m_{ij}) + \frac{2u^{(2)}_i - m_{ij}}{\sigma_{ij}^2} \cdot \exp(-\frac{u^{(2)}_i - m_{ij}}{\sigma_{ij}^2}) U(u^{(2)}_i; m_{ij}, \infty))$$

The negative deviation is updated by the amount

$$\Delta \sigma_{ij} = -\frac{\partial E}{\partial \sigma_{ij}} = -(\frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial m_{ij}} \cdot \frac{\partial m_{ij}}{\partial u^{(3)}_j} \cdot \frac{\partial u^{(3)}_j}{\partial \sigma_{ij}})$$

$$= e \cdot w_j \cdot (1 - \gamma_j + \gamma_j/n) \cdot \prod_{i=1}^{n} u^{(3)}_i \cdot (u^{(2)}_i - m_{ij}) \frac{2u^{(2)}_i - m_{ij}}{\sigma_{ij}^2} \cdot \exp(-\frac{u^{(2)}_i - m_{ij}}{\sigma_{ij}^2})$$

$$\cdot U(u^{(2)}_i; -\infty, m_{ij})$$

$$\cdot U(u^{(2)}_i; \infty, m_{ij})$$

$$\cdot U(u^{(2)}_i; \infty, m_{ij})$$

$$\cdot U(u^{(2)}_i; -\infty, m_{ij})$$

$$\cdot U(u^{(2)}_i; \infty, m_{ij})$$
The positive deviation is updated by the amount

\[
\Delta \sigma_{ij+} = - \frac{\partial E}{\partial \sigma_{ij+}} = - \left( \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial u_{ij}^{(4)}} \cdot \frac{\partial u_{ij}^{(3)}}{\partial \sigma_{ij+}} \right)
\]

\[
= e \cdot w_j \cdot (1 - \gamma_j^+ + \gamma_j^-) \cdot \left( \prod_{i \neq j} u_{ij}^{(3)} \right)^{-\gamma_j^+ + \gamma_j^-} \cdot U(u_{ij}^{(2)}; m_{ij}, \infty)
\]

\[
\cdot \frac{2(u_{ij}^{(2)} - m_{ij})^2}{\sigma_{ij+}^2} \exp\left(- \frac{(u_{ij}^{(2)} - m_{ij})^2}{\sigma_{ij+}^2}\right)
\]

\[
\cdot U(u_{ij}^{(2)}; m_{ij}, \infty)
\]

(32)

The weight of feedback is updated by the amount

\[
\Delta \theta_{ij} = - \frac{\partial E}{\partial \theta_{ij}} = - \left( \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial \sigma_{ij+}} \cdot \frac{\partial \sigma_{ij+}}{\partial \theta_{ij}} \right)
\]

\[
= - e \cdot w_j \cdot (1 - \gamma_j^+ + \gamma_j^-) \cdot \left( \prod_{i \neq j} u_{ij}^{(3)} \right)^{-\gamma_j^+ + \gamma_j^-} \cdot U(u_{ij}^{(2)}; -\infty, m_{ij})
\]

\[
\cdot \frac{2(u_{ij}^{(2)} - m_{ij})^2}{\sigma_{ij+}^2} \exp\left(- \frac{(u_{ij}^{(2)} - m_{ij})^2}{\sigma_{ij+}^2}\right)
\]

\[
\cdot \frac{2(u_{ij}^{(2)} - m_{ij})^2}{\sigma_{ij+}^2} \exp\left(- \frac{(u_{ij}^{(2)} - m_{ij})^2}{\sigma_{ij+}^2}\right)
\]

\[
\cdot U(u_{ij}^{(2)}; m_{ij}, \infty)
\]

(33)

The mean and deviation of the PG membership functions and the weight of feedback are updated according to the following equation:

\[
m_{ij}(t+1) = m_{ij}(t) + \eta_m \Delta m_{ij}
\]

(34)

\[
\sigma_{ij}(t+1) = \sigma_{ij}(t) + \eta_\sigma \Delta \sigma_{ij}
\]

(35)

\[
\sigma_{ij}(t+1) = \sigma_{ij}(t) + \eta_\sigma \Delta \sigma_{ij}
\]

(36)

\[
\theta_{ij}(t+1) = \theta_{ij}(t) + \eta_\theta \Delta \theta_{ij}
\]

(37)

where \(\eta_m\), \(\eta_\sigma\), \(\eta_\sigma\), and \(\eta_\theta\) are the learning rate of the mean, the negative deviation, the positive deviation for the PG membership functions and weight of feedback, respectively.

**IV. ILLUSTRATIVE EXAMPLES**

To certify the performance of the PG-RCFNN for temporal problems, three typical examples are presented in this section. The first example is to predict a time sequence (Santini et al., 1995), the second example is to identify a nonlinear dynamic system (Lee and Teng, 2000), and the third example is to identify a chaotic system (Chen et al., 1997).

**Example 1:** Prediction of the Time Sequence

To clearly verify if the proposed PG-RCFNN can learn the temporal relationship, a simple time sequence prediction problem found in (Santini et al., 1995) is used for a test in the following example.

The test bed used is shown in Fig. 2(a). This is an “8” shape made up of a series with 12 points which are to be presented to the network in the order shown. The PG-RCFNN is asked to predict the succeeding point for every presented point. Obviously, a static network cannot accomplish this task, since the point at coordinate (0,0) has two successors: point 5 and point 11. The PG-RCFNN must decide the successor of (0,0) based on its predecessor; if the predecessor is 3, then the successor is 5, whereas if the predecessor is 9, the successor is 11.

In this example, the PG-RCFNN contains only two input nodes, which are activated with the two dimensional coordinate of the current point, and two output nodes, which represent the two dimensional coordinate of the predicted point. The learning rate \(\eta = 0.15\), the initial deviation \(\sigma_{init} = 0.25\) and the prespecified threshold \(\mathcal{P} = 0.01\) are chosen. The training process is continued for 1000 epochs. Starting at zero, the numbers of fuzzy rules grow dynamically for incoming training data. The final root-mean-square (rms) error of prediction output approximates 0.000581. There are seven fuzzy rules generated for incoming training data. Fig. 2(b) shows the prediction results by the trained PG-RCFNN model. Simulation results show that we can obtain perfect prediction capability.

We also apply the RFNN (Lee and Teng, 2000) model and a traditional (non-recurrent) fuzzy neural network (FNN)(Lin et al., 2001) for this time prediction problem. Fig. 2(c) shows the prediction results using the RFNN model. In this figure, the RFNN also obtains prediction capability, but some time prediction points cannot be matched exactly. Fig. 2(d) shows that a feedforward FNN model cannot predict successfully, because of its static mapping.
and needs less computation time than RFNN and FNN under the same training epochs.

**Example 2:** Identification of a Nonlinear Dynamic System

In this example, the nonlinear plant with multiple time delay is guided by the following difference equation:

\[
y_p(t+1) = f(y_p(t), y_p(t-1), y_p(t-2), u_p(t), u_p(t-1))
\]  

(38)
where

\[ f(x_1, x_2, x_3, x_4, x_5) = \frac{x_1 x_2 x_3 x_5 (x_3 - 1) + x_4}{1 + x_2 + x_3^2} \]  (39)

Here, the current output of the plant depends on three previous outputs and two previous inputs. In (Narendra and Parthasarathy, 1990), 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 \( u(t) \), are fed to the PG-RCFNN to determine the output \( y_p(t+1) \). The training inputs are independent and 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 is no repetition on these 900 training data, i.e., we have different training sets for each epoch. The checking input signal \( u(t) \) as the following equation is used to determine the identification results

\[ u(t) = \begin{cases} \sin(\pi t/25) & 0 < t < 250 \\ 1.0 & 250 \leq t < 500 \\ -1.0 & 500 \leq t < 750 \\ 0.3 \sin(\pi t/25) + 0.1 \sin(\pi t/32) + 0.6 \sin(\pi t/10) & 750 \leq t < 1000 \end{cases} \]  (40)

In applying the PG-RCFNN to this example, only 10 epochs are used, where each epoch is 900 time steps. The learning rate \( \eta=0.15 \), the initial deviation \( \sigma_{\text{init}}=0.3 \), and the prespecified threshold \( P=0.01 \) are chosen. After training, three fuzzy logic rules are generated, and the final RMS error is 0.000728 after the training 10 epochs. The final designed fuzzy logic rules are shown as follows:

**Rule-1:** IF \( u(t) \) is \((0.0189; 0.2800, 0.2782) \) and \( y(t) \) is \((0.2781; 0.2673, 0.3578) \) THEN \( y(t+1) \) is 0.2705

**Rule-2:** IF \( u(t) \) is \((-0.6720; 0.2294, 0.3319) \) and \( y(t) \) is \((0.9088; 0.4093, 0.5454) \) THEN \( y(t+1) \) is 0.8799

**Rule-3:** IF \( u(t) \) is \((-0.8502; 0.6110, 0.5824) \) and \( y(t) \) is \((-0.6069; 0.3844, 0.6067) \) THEN \( y(t+1) \) is \(-1.0356 \)

Fig. 3(a) illustrates the distribution of some of the training patterns and the final assignment of the rules in the \([u(t), y(t)]\) plain. This is due to the parameter learning process, which tunes the mean and variance of each membership function at each time step to minimize the output cost function. The region covered by a Gaussian membership function is unbounded, and the boundary of each ellipse represents a rule with firing strength 0.5. The membership functions on the \( u(t) \) and \( y(t) \) dimension are shown in Figs. 3(b) and 3(c).

Fig. 4(a) shows the RMS error of the PG-RCFNN and RFNN. Fig. 4(b) shows the outputs of the plant and the PG-RCFNN model. The results show the perfect identification capability of the PG-RCFNN model. Fig. 4(c) presents the error with the desired output and PG-RCFNN output. A comparison analysis of the PG-RCFNN, the RSONFIN (Juang and Lin, 1999), the RFNN (Lee and Teng, 2000), and the TRFN-S (Juang, 2002) are presented in Table 2. As shown in Table 2, the rms error in our model is rather smaller than other recurrent methods under the same training epochs. Although the proposed model needs more adjustable parameters than RFNN (Lee and Teng, 2000), for initializing parameters of the RFNN model, the rule number should be given in advance. But, the users need not give it any a priori knowledge or even any initial information for our proposed model.

**Example 3: Identification of The Chaotic System**

The discrete time Henon system is repeatedly used in the study of chaotic dynamics and is not exceedingly simple in the sense that it is of second...
order with one delay and two parameters (Chen et al., 1997). This chaotic system is described by

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

(41)

which, with \( P = 1.4 \) and \( Q = 0.3 \), produces a chaotic strange attractor as shown in Fig. 5(a). For this training, the input of the PG-RCFNN is \( y(t-1) \) and the output is \( y(t) \). We first randomly take the training data (1000 pairs) from the system over the interval \([-1.5, 1.5]\).

In applying the PG-RCFNN to this example, 100 training epochs are used. Here the initial point is \( [y(1), y(0)]^T = [0.4, 0.4]^T \). The learning rate \( \eta = 0.15 \), the initial deviation \( \sigma_{\text{init}} = 0.36 \), and the prespecified threshold \( \theta^* = 0.01 \) are chosen. After 100 training epochs, the final RMS error is 0.0013, and three fuzzy

---

**Table 2** Comparisons of the PG-RCFNN with other existing recurrent network for dynamic system identification in Example 2

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>33</td>
<td>36</td>
<td>21</td>
<td>33</td>
</tr>
<tr>
<td>RMS error (train)</td>
<td>0.000728</td>
<td>0.0248</td>
<td>0.0013</td>
<td>0.0084</td>
</tr>
<tr>
<td>RMS error (test)</td>
<td>0.0013867</td>
<td>0.0780</td>
<td>0.0034</td>
<td>0.0346</td>
</tr>
<tr>
<td>Epochs</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>
logic rules are generated. The phase plane of this chaotic system after training for the FNN and the PG-RCFNN are shown in Fig. 5(b) and Fig. 5(c). From the simulation results shown in Fig. 5(b), we can see that the FNN is inappropriate for a chaotic dynamics system because of its static mapping. Comparisons of the PG-RCFNN with other recurrent networks for this chaotic system are shown in Table 3. From the results in Table 3, we can see that performance of PG-RCFNN surpass RFNN (Lee and Teng, 2000) and FNN (Lin et al., 2001).

### Table 3 Comparisons of the PG-RCFNN with other recurrent networks for this chaotic system in Example 3

<table>
<thead>
<tr>
<th></th>
<th>PG-RCFNN</th>
<th>RFNN (Lee and Teng, 2000)</th>
<th>FNN (Lin et al., 2001)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of rules</td>
<td>3</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>RMS error (train)</td>
<td>0.0013</td>
<td>0.0363</td>
<td>0.1238</td>
</tr>
<tr>
<td>RMS error (test)</td>
<td>0.0014</td>
<td>0.0453</td>
<td>0.1077</td>
</tr>
<tr>
<td>Epochs</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

Fig. 5. 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 PG-RCFNN for the chaotic system. (d) Learning curve of the PG-RCFNN and RFNN.

V. CONCLUSIONS

In this paper, a Pseudo-Gaussian-based Recurrent Compensatory Fuzzy Neural Network (PG-RCFNN) is proposed for temporal problems. The compensatory operations (in the third layer) with adaptive fuzzy reasoning are more effective and adaptive than the conventional fuzzy neural system with non-adaptive fuzzy reasoning. The asymmetric Gaussian membership function’s variability and malleability are higher than traditional ones. Therefore,
the PG membership functions can approach the true result more easily. An on-line learning algorithm is proposed to automatically construct the PG-RCFNN. It does not require the human expert’s assistance and its structure is obtained from the input data. Computer simulation results show that the proposed model only requires a small number of tuning parameters and obtains a smaller RMS error than other models for dynamic system identification.

ACKNOWLEDGMENTS

This research was supported by the National Science Council of R.O.C. under grant NSC 93-2213-E-324-008.

REFERENCES


Manuscript Received: Jul. 15, 2003
Revision Received: Apr. 15, 2004
and Accepted: May 10, 2004