A Neural Fuzzy System with Fuzzy Supervised Learning

Chin-Teng Lin, *Member, IEEE,* and Ya-Ching Lu

Abstract—**A neural fuzzy system learning with fuzzy training data (fuzzy if-then rules) is proposed in this paper. This system is able to process and learn numerical information as well as linguistic information. At first, we propose a five-layered neural network for the connectionist realization of a fuzzy inference system. The connectionist structure can house fuzzy logic rules and membership functions for fuzzy inference. We use α-level sets of fuzzy numbers to represent linguistic information. The inputs, outputs, and weights of the proposed network can be fuzzy numbers of any shape. Furthermore, they can be hybrid of fuzzy numbers and numerical numbers through the use of fuzzy singletons. Based on interval arithmetics, a *fuzzy supervised learning* algorithm is developed for the proposed system. It extends the normal supervised learning techniques to the learning problems where only linguistic teaching signals are available. The fuzzy supervised learning scheme can train the proposed system with desired fuzzy input-output pairs which are fuzzy numbers instead of the normal numerical values. With fuzzy supervised learning, the proposed system can be used for *rule base concentration* to reduce the number of rules in a fuzzy rule base. Simulation results are presented to illustrate the performance and applicability of the proposed system.**

I. Introduction

SOME observations obtained from a system are precise, while some cannot be measured at all. Namely, two kinds of information are available. One is numerical information from measuring instruments, and the other is linguistic information from human experts. However, some of data obtained in this manner are hybrid; that is, their components are not homogeneous but a blend of precise and fuzzy information.

Neural networks adopt numerical computations with fault-tolerance, massively parallel computing, and trainable properties. However, numerical quantities evidently suffer from a lack of representation power. Therefore, it is useful for neural networks to be capable of symbolic processing. Most learning methods in neural networks are designed for real vectors. There are many applications that the information cannot be represented meaningfully or measured directly as real vectors. That is, we have to deal with fuzzy information in the learning process of neural networks. Fuzzy set is a good representation form for linguistic data. Therefore, combining neural networks with fuzzy set could combine the advantages of symbolic and numerical processing. In this paper, we propose a new model of neural fuzzy system that can process the hybrid of numerical and fuzzy information. The main task is to develop a fuzzy supervised learning algorithm for the proposed neural fuzzy system.

Most of the supervised learning methods of neural networks, e.g., the perceptron [1], the BP (backpropagation) algorithm [2], [3], process only numerical data. Some approaches have been proposed to process linguistic information with fuzzy inputs, fuzzy outputs, or fuzzy weights. Tanaka and his colleagues have proposed a series of approaches and applications with the capacity of processing linguistic input or/and linguistic output [4]–[6]. In their methods, the weights, inputs, and outputs of the neural network are fuzzified using fuzzy numbers represented by α-level sets. In [4], learning methods of neural networks were proposed to utilize not only numerical data but also expert knowledge represented by fuzzy if-then rules. In order to handle linguistic informations in neural networks, they proposed an architecture of multilayer feedforward neural network that can deal with fuzzy input vectors. The fuzzy data are viewed as convex and normal fuzzy sets represented by α-level sets. Since the α-level sets of fuzzy numbers are intervals, the operations of real number in the traditional neural networks are extended to the closed intervals in their proposed model. The characteristics of their proposed architecture are as follows:

1) fuzzy numbers are propagated through neural networks;
2) a single unit is employed for representing a fuzzy number;
3) the weights and biases in neural networks are still real numbers.

Tanaka *et al.* [5], [6], also proposed an architecture of fuzzy neural network with fuzzy weights and fuzzy biases, and derived its learning algorithm from the modeling of nonlinear fuzzy systems which map a real input vector to a fuzzy output. The architecture of fuzzy neural network is the same as that in [4], except that the numerical weights and biases have been replaced by the fuzzy weights and biases. The calculations in the fuzzy neural network are performed by using interval arithmetic operations for α-level sets. Also, the learning algorithm is derived from a cost function defined by the α-level sets of a actual fuzzy output and a target fuzzy output. Furthermore, they show that the learning algorithm can be applied to the case of fuzzy input vectors and fuzzy target outputs by slightly modifying it. They use symmetric triangular (or trapezoidal) fuzzy numbers for fuzzy weights and fuzzy biases and a learning algorithm is derived from a nonfuzzy cost function.

Manuscript received February 4, 1995; revised July 25, 1995. This work was supported by the National Science Council, R.O.C., under Grant NSC 84-2212-E-009-034.

The authors are with the Department of Control Engineering, National Chiao-Tung University, Hsinchu, Taiwan, R.O.C. (e-mail: chinteng @ecn.purdue.edu).

Publisher Item Identifier S 1083-4419(96)05364-2.

Hayashi *et al.* [7] presented a similar method with fuzzy signals and fuzzy weights by using triangular fuzzy numbers. A similar learning algorithm is derived from a nonfuzzy cost function. Hayashi *et al.* [8] also proposed a similar architecture of neural network with fuzzy weights and fuzzy signals, but the learning algorithm was completely different from the proposed methods of Tanaka. In [8], the BP learning algorithm is directly fuzzified based on a fuzzy-valued cost function; i.e., the rule for changing fuzzy weights is also defined by fuzzy numbers.

The common points of the above approaches are summarized as follows:

1) α-level sets of fuzzy numbers represent linguistic inputs, linguistic outputs, fuzzy weights, or fuzzy biases;
2) the operations in neural network are performed by using interval arithmetic operations for α-level sets;
3) fuzzy numbers are propagated through neural networks;
4) fuzzy weights are usually triangular or trapezoidal fuzzy numbers.

Because the real number arithmetic operations in the traditional neural networks are extended to interval arithmetic operations for α-level sets in the above fuzzified networks, the computations become complex (e.g., multiplication of interval) and time-consuming. Moreover, since the fuzzy numbers are propagated through the whole neural networks, the time of computations and the required memory capacities are $2h$ times of those in the traditional neural networks, where h represents the number of α-level sets. In this paper, we attack this problem by allowing numerical signals to flow in the proposed network internally and reach the same purpose of processing fuzzy numbers.

The objective of this paper is to explore the approach to supervised learning of neural fuzzy systems which receive only linguistic teaching signals. At first, we propose a five-layered feedforward network for the network realization of a fuzzy inference system. This connectionist structure can house fuzzy logic rules and membership functions, and perform fuzzy inference. We use α-level sets of fuzzy numbers to represent linguistic information. The inputs, outputs, and weights of the proposed network can be fuzzy numbers of any shape. Since numerical values can be represented by fuzzy singletons, the proposed system can in fact process and learn hybrid of fuzzy numbers and numerical numbers. Based on interval arithmetics, a *fuzzy supervised learning* scheme is developed for the proposed system. It generalizes the normal supervised learning techniques to the learning problems where only linguistic teaching signals are available. The fuzzy supervised learning scheme can train the proposed network with desired fuzzy input-output pairs (or, equivalently, desired fuzzy if-then rules) represented by fuzzy numbers instead of numerical values. With supervised learning, the proposed system can be used for *rule base concentration* to reduce the number of rules in a fuzzy rule base.

This paper is organized as follows: Section II describes the fundamental properties and operations of fuzzy numbers and their α-level sets. These operations and properties will be used in later derivation. In Section III, the structure of our neural fuzzy system is proposed. A fuzzy supervised learning algorithm for the proposed system is presented in Section IV. The learning algorithm contains structure and parameter learning phases. In Section V, simulations are performed to illustrate the performance of the proposed system. Finally, conclusions and future research are summarized in the last section.

II. Representation of Linguistic Information

When constructing information processing systems such as classifier and controllers, two kinds of information are available. One is numerical information from measuring instruments and the other is linguistic information from human experts. We can naturally indicate the numerical information using real numbers. But, how to represent the linguistic data? It has been popular for using fuzzy sets defined by discretized (pointwise) membership functions to represent linguistic information. Fuzzy sets, however, can be defined by the families of their α-level sets according to the resolution identity theorem. In our model, we use α-level sets of fuzzy numbers (i.e., convex and normal fuzzy sets) to represent linguistic information due to their several good properties such as closeness and good representation form. In using α-level sets, we consider a fuzzy number to be an extension of the concept of the interval of confidence [9]. Instead of considering the interval of confidence at one unique level, it is considered at several levels and more generally at all levels from 0 to 1. Namely, the level of presumption α, $\alpha \in [0,1]$ gives an interval of confidence $A_\alpha = [a_1^{(\alpha)}, a_2^{(\alpha)}]$, which is a monotonical decreasing function of α; that is,

$$(\alpha' > \alpha) \rightarrow (A_{\alpha'} \subset A_\alpha) \tag{1}$$

or

$$(\alpha' > \alpha) \rightarrow \left(\left[a_1^{(\alpha')}, a_2^{(\alpha')}\right] \subset \left[a_1^{(\alpha)}, a_2^{(\alpha)}\right]\right) \tag{2}$$

for every $\alpha, \alpha' \in [0,1]$.

The expressions level of presumption are well suited to the concept of fuzzy numbers. Moreover the intervals of confidence at all level from 0 to 1 are the same as the α-level sets of fuzzy number. In this paper, we use α-level sets of fuzzy number to indicate linguistic information because of their advantages in both theoretical and practical considerations [10]. These advantages will be explained in details in subsection II-C.

A. Basic Definitions of Fuzzy Numbers

Some notations and basic definitions are given in this subsection. We use the uppercase letter to represent a fuzzy set and the lowercase letter to represent a real number.

Let x be an element in a universe of discourse X. A fuzzy set, P, is defined by a membership function, $\mu_P(x)$, as

$$\mu_P : x \rightarrow [0,1]. \tag{3}$$

When X is continuum rather than a countable or finite set, the fuzzy set P is represented as

$$P = \int_X \mu_P(x)/x \tag{4}$$

where $x \in X$. When X is a countable or finite set

$$P = \sum_i \mu_P(x_i)/x_i \tag{5}$$

where $x_i \in X$. We call the form in the above equation as a *discretized* or *pointwise* membership function.

A fuzzy set, P, is *normal* when its membership function, $\mu_P(x)$, satisfies

$$\max_x \mu_P(x) = 1. \tag{6}$$

A fuzzy set is *convex* if and only if each of its α-level sets is a convex set. Equivalently, we may say that a fuzzy set P is convex if and only if

$$\mu_P(\lambda x_1 + (1-\lambda)x_2) \geq \min[\mu_P(x_1), \mu_P(x_2)] \tag{7}$$

where $0 \leq \lambda \leq 1, x_1 \in X, x_2 \in X$.

The *α-level set* of a fuzzy set P, P_α, is defined by

$$P_\alpha = \{x \mid \mu_P(x) \geq \alpha\} \tag{8}$$

where $0 \leq \alpha \leq 1, x \in X$.

A fuzzy set P is convex if and only if every P_α is convex; that is, P_α is a closed interval of $\Re$. It can be represented by

$$P_\alpha = [p_1^{(\alpha)}, p_2^{(\alpha)}] \tag{9}$$

where $\alpha \in [0,1]$. A convex and normalized fuzzy set whose membership function is piecewise continuous is called a fuzzy number. Thus, a fuzzy number can be considered as containing the real numbers within some interval to varying degrees. Namely, a fuzzy number P may be decomposed into its α-level sets, P_α, according to the resolution identity theorem [11] as follows:

$$P = \bigcup_\alpha \alpha P_\alpha = \bigcup_\alpha \alpha [p_1^{(\alpha)}, p_2^{(\alpha)}] = \int_x \sup_\alpha \alpha \mu_{P_\alpha}(x)/x. \tag{10}$$

B. Basic Operations of Fuzzy Numbers

In this subsection, we introduce some basic operations of α-level sets of fuzzy numbers. These operations will be used in the derivation of our model in the following section. More detailed operations of fuzzy numbers can be found in [9].

Addition: Let A and B be two fuzzy numbers and A_α and B_α their α-level sets, $A = \bigcup_\alpha \alpha A_\alpha = \bigcup_\alpha \alpha [a_1^{(\alpha)}, a_2^{(\alpha)}]$, $B = \bigcup_\alpha \alpha B_\alpha = \bigcup_\alpha \alpha [b_1^{(\alpha)}, b_2^{(\alpha)}]$. Then we can write

$$\begin{aligned} A_\alpha(+)B_\alpha &= [a_1^{(\alpha)}, a_2^{(\alpha)}](+)[b_1^{(\alpha)}, b_2^{(\alpha)}] \\ &= [a_1^{(\alpha)} + b_1^{(\alpha)}, a_2^{(\alpha)} + b_2^{(\alpha)}] \end{aligned} \tag{11}$$

where $\alpha \in [0,1]$.

Subtraction: The definition of addition can be extended to the definition of subtraction as follows:

$$\begin{aligned} A_\alpha(-)B_\alpha &= [a_1^{(\alpha)}, a_2^{(\alpha)}](-)[b_1^{(\alpha)}, b_2^{(\alpha)}] \\ &= [a_1^{(\alpha)} - b_2^{(\alpha)}, a_2^{(\alpha)} - b_1^{(\alpha)}] \end{aligned} \tag{12}$$

where $\alpha \in [0,1]$.

Multiplication by an Ordinary Number: Let A be a fuzzy number in $\Re$ and k an ordinary number $k \in \Re$. We have

$$k \cdot A_\alpha = \begin{cases} [ka_1^{(\alpha)}, ka_2^{(\alpha)}], & \text{if } k \geq 0 \\ [ka_2^{(\alpha)}, ka_1^{(\alpha)}], & \text{if } k < 0. \end{cases} \tag{13}$$

Multiplication: Here we consider multiplication of fuzzy numbers in $\Re^+$. Consider two fuzzy numbers A and B in $\Re^+$. For the level α of presumption, we have

$$\begin{aligned} A_\alpha(\cdot)B_\alpha &= [a_1^{(\alpha)}, a_2^{(\alpha)}](\cdot)[b_1^{(\alpha)}, b_2^{(\alpha)}] \\ &= [a_1^{(\alpha)} \cdot b_1^{(\alpha)}, a_2^{(\alpha)} \cdot b_2^{(\alpha)}]. \end{aligned} \tag{14}$$

The reader is referred to [9] for the general case that A and B are fuzzy numbers in $\Re$.

The operations on fuzzy numbers can be performed on the basis of the extension principle. Let $G(A,B)$ be an operation on fuzzy numbers A and B by giving a function $g(x,y)$. The membership function of $G(A,B)$, $\mu_{G(A,B)}(z)$, is obtained by extension principle as follows:

$$\mu_{G(A,B)}(z) = \begin{cases} \sup_{(x,y)\in g^{-1}(z)}(\mu_A(x) \wedge \mu_B(y)), & \text{if } g^{-1}(z) \neq \emptyset \\ 0, & \text{if } g^{-1}(z) = \emptyset \end{cases} \tag{15}$$

where $x \in A, y \in B, z \in Z$.

If $G(\cdot)$ is addition, subtraction, or multiplication as above, the result can be easily obtained by using the α-level sets of A and B as above operations. It can be proved easily that these operations based on α-level sets and the extension principle are equivalent [9]. When $G(\cdot)$ is one of the operations given above, if A and B are fuzzy numbers in $\Re$, then $G(A,B)$ is also a fuzzy number. Owing to this closed property and good representation form, we use fuzzy numbers rather than general fuzzy sets to represent linguistic information in our model.

Difference: We can compute the difference between fuzzy numbers A and B by

$$\text{diff}(A,B) = \frac{1}{2}\sum_\alpha \left[\left(a_1^{(\alpha)} - b_1^{(\alpha)}\right)^2 + \left(a_2^{(\alpha)} - b_2^{(\alpha)}\right)^2\right]. \tag{16}$$

Fuzzification: Fuzzification is a subjective valuation to transform a measurement crisp value into a valuation of a subjective value. Hence, it can be defined as a mapping from an observed input space to labels of fuzzy sets in a specified input universe of discourse. In this paper, we intuitively consider fuzzification to be an operation that calculates the membership grade of x, $\mu_P(x)$, to a given fuzzy number P. The values $\mu_P(x)$ can be easily obtained by using the following procedure. In this procedure, the notation h represents the number of quantized membership grade.

Procedure: Fuzzification Search

Inputs}: The order set $S = \{p_1^{(0)}, p_1^{(1/h)}, p_1^{(2/h)}, \ldots, p_1^{(1)}, p_2^{(1)}, \ldots, p_2^{(0)}\}$ and a numerical value x, where S represents the α-level sets of the interested fuzzy number P and x is the input numerical value to be fuzzified.

Output}: $\hat{\alpha}$ indicating the membership grade of x, $\mu_P(x)$, to fuzzy number P.

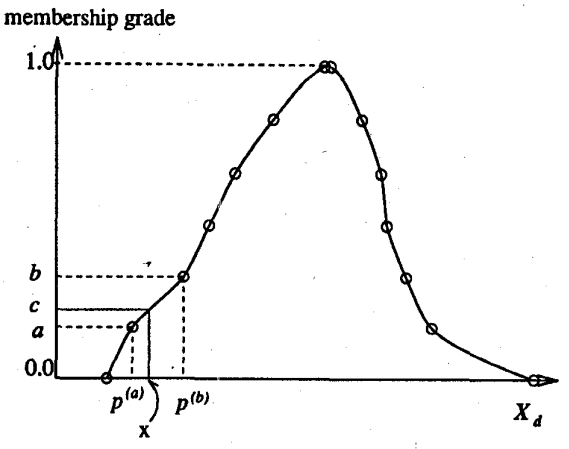


Fig. 1. Illustration of the fuzzification search procedure (the value of c is equal to $\hat{\alpha}$).

Step 1: Use *binary search* to find the correct position of x in S; that is, to find $p^{(a)}$ and $p^{(b)}$ such that $p^{(a)} \leq x \leq p^{(b)}, p^{(a)}, p^{(b)} \in S$. In the binary search, we compare first x to the entry in the middle of the list S. If x is larger, it is in the second half; with one comparison the entire first half of the list is eliminated from consideration. Conversely, if x is smaller than the entry in the middle of the list, the second half the list is eliminated from consideration. This process continues until the right interval that x locates is found.

Step 2: Compute the membership degree by $\hat{\alpha} = a + (x - p^{(a)})\frac{b-a}{p^{(b)}-p^{(a)}}$.

Step 3: Output $\hat{\alpha}$, and stop.

After this recursive calculation, the value $\hat{\alpha}$ is equal to the value of $\mu_P(x)$ as shown in Fig. 1. If we use the binary search method, the processing time with this procedure is proportional to $\log_2 h$. If we use the sequential search method, the processing time is $O(h)$. Therefore, performing this procedure is very easy and not time consuming. Another method to find $\mu_p(x)$ can be found in [10].

Defuzzification: In many practical applications such as control and classification, numerical (crisp) data are required. That is, it is essential to transform a fuzzy number to a numerical value. The process mapping a fuzzy number into a nonfuzzy number is called "defuzzification". Various defuzzification strategies have been suggested in [12], [13]. In this subsection, we describe two methods (MOM, COA) that transform a fuzzy number in the form of α-level sets into a crisp value.

- *Mean of Maximum Method (MOM)*
  The mean of maximum method (MOM) generates a crisp value by averaging the support values whose membership values reach the maximum. For a discrete universe of discourse, this is calculated based on membership function by

$$z_0 = \frac{\sum_{j=1}^{l} z_j}{l} \tag{17}$$

  where l is the number of quantized z values which reach their maximum membership value.

  For a fuzzy number Z in the form of α-level sets, the defuzzification method can be expressed according to (17) as

$$\text{defuzzifier}(Z) = z_0 = \frac{\left(z_1^{(1)} + z_2^{(1)}\right)}{2} \tag{18}$$

  where *defuzzifier* represents a defuzzification operation.
- *Center of Area Method (COA)*
  Assuming that a fuzzy number with a pointwise membership μ_Z has been produced, the center of area method calculates the center of gravity of the distribution for the nonfuzzy value. Assuming a discrete universe of discourse, we have

$$z_0 = \frac{\sum_{j=1}^{n} x_j \mu_Z(x_j)}{\sum_{j=1}^{n} \mu_Z(x_j)}. \tag{19}$$

For a fuzzy number Z with the representation form of α-level sets, it can be approximately expressed according to (19) as

$$\text{defuzzifier}(Z) = z_0 = \frac{\sum_{\alpha} \alpha\left(z_1^{(\alpha)} + z_2^{(\alpha)}\right)}{2\sum_{\alpha} \alpha}. \tag{20}$$

C. Advantages of Using α-Level Sets of Fuzzy Numbers

Fuzzy sets have been studied actively and applied in a wide variety of areas such as system control, modeling, signal processing, and expert systems. In this paper, we also use fuzzy numbers to represent linguistic data. Conventionally, fuzzy numbers are defined by membership functions, such as bell shaped functions, triangular functions, trapezoidal functions, etc. In these forms, we only need a few simple parameters (e.g., the center and the width) to define them, but they can only display some normal situations. In this paper, we use α-level sets to define fuzzy number owing to the following advantages [10]:

a) Theoretical

- It effectively studies the effects of the position in a universe of discourse and the fuzziness of a fuzzy number.

b) Practical

- It provides fast inference computations by using hardware construction in parallel and requires less memory capacity for fuzzy numbers defined in universes of discourse with a large number of elements.
- It easily interfaces with two-valued logic.
- It allows good matching with systems that include fuzzy number operations based on the extension principle.

These advantages are explained in details below. Fig. 2(b) illustrates a discretized membership function for a fuzzy number in a universe of discourse with a discrete space X_d. In this figure, the discretized membership function does not explicitly give the position of the fuzzy number in the universe of discourse and the fuzziness. In contrast, the α-level sets in Fig. 2(a) are more effective representation forms of fuzzy numbers. The left and right limits of one of the α-level sets directly give the position of the fuzzy number, and the width of

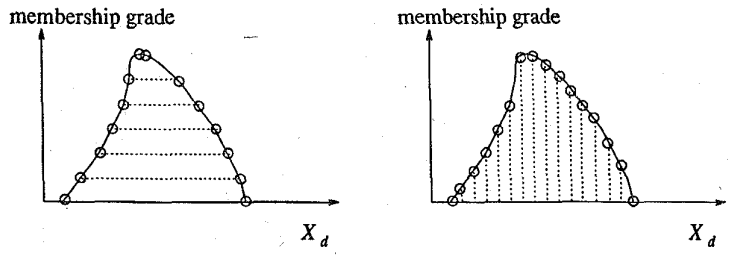


Fig. 2. Representations of fuzzy number: (a) α-level sets of fuzzy number, and (b) discretized (pointwise) membership function.

the α-level set is directly related to the fuzziness and converge of the fuzzy number in the universe of discourse. If fuzzy-inference operations can be represented using parameters that define α-level sets, namely, the left and right limits mentioned above, the relation between the given facts and the inferred consequent might easily be derived with direct information on the position and fuzziness of fuzzy numbers.

Providing hardware-related advantages is also important for α-level-set-based formulation. Hardware systems for fuzzy inference have been developed and studied [14] owing to the development of fast and cost-effective fuzzy-inference engines. In conventional fuzzy propositions based on regular membership functions such as triangular or trapezoidal forms, the fuzzy-inference consequent is not always triangular or trapezoidal if fuzzy relation is adopted. As a result, the hardware may have complex structure. Also, the required memory capacity for discretized membership functions must be proportional to the number of elements in the universe of discourse. This facts also means that fuzzy inference operations are time-consuming processes because calculations for inference have to be conducted for "all elements" to which membership grades are assigned. Due to the above reasons, the fuzzy-logic hardware based on discretized membership functions has a disadvantage in the required memory capacity and processing time.

In the scheme based on α-level sets, the required memory capacity and processing time depend on the number of membership-grade levels instead of the number of elements in the universe of discourse. It is clear that the number of elements in a universe of discourse is much larger than the required number of membership-grade levels. Hence hardware systems based on α-level sets have the advantage in required memory capacity and processing time. These advantages also exist in adaptive fuzzy inference systems where fuzzy numbers are adjusted in the learning process. These advantages are considered in greater details in [10].

Fuzzy inference based on α-level sets interfaces well with two-valued logic because α-level sets themselves are crisp sets. This interface is useful, for example, when two-valued decision making is needed for inference consequents. Moreover, the operations on fuzzy numbers based on the extension principle, such as the addition and multiplication of fuzzy numbers, can be performed efficiently using α-level sets. Consequently, efficient calculation may be possible for both fuzzy-inference operations.

Due to the above advantages, more and more authors represent fuzzy numbers using α-level sets rather than pointwise membership functions [15], [16]. In this paper, the α-level sets are also used to represent fuzzy numbers in the proposed neural fuzzy systems. It will be oberved that the use of α-level sets also simplifies the learning rules for updating fuzzy weights in the network.

III. Basic Structure of the Neural Fuzzy System

In this section, we construct an architecture of neural fuzzy system that can process fuzzy and crisp information. Fig. 3 shows the proposed network structure which has a total of five layers. This five-layered connectionist structure performs fuzzy inference effectively. Similar architectures have been proposed in [17], [18], and [19]. Nodes at layer one are input nodes whose inputs are fuzzy numbers or crisp numbers. Each input node corresponds to one input linguistic variable. Layer five consists of output nodes whose output are also fuzzy numbers or crisp numbers. Each output node corresponds to one output linguistic variable. Nodes at layers two and four are *term nodes* which define membership functions representing the fuzzy terms of the respective linguistic variable. Only the nodes at layer two and five have fuzzy weights (as shown by bold lines in Fig. 3). The other weights are numerical (as shown by plain lines in Fig. 3). Each node in layer two executes a "match" action to find the match degree between the input fuzzy number and the fuzzy weight if the input is linguistic information. If the input is a crisp number, they execute a "fuzzification" operation to map the input value from an observed input space to the fuzzy weights in nodes at layer two. Each node at layer three is a rule node which represents one fuzzy rule. Hence, all the layer-three nodes form a fuzzy rule base. Links from layers two to three define the preconditions of the fuzzy rules, and links from layer three to four define the consequents of the fuzzy rules. Therefore, for each rule node, there is at most one link (maybe none) from or to some term node of a linguistic node. This is true both for precondition links and consequent links. The links at layers two and five are fully connected between linguistic nodes and their corresponding term nodes. If linguistic outputs are expected, each node in layer five "merges" all fuzzy weights connected to it, scaled by the output values of layer four, to produce a new fuzzy number. If numerical output values are required, each layer-5 node execute a "defuzzification" operation to obtain a crisp decision.

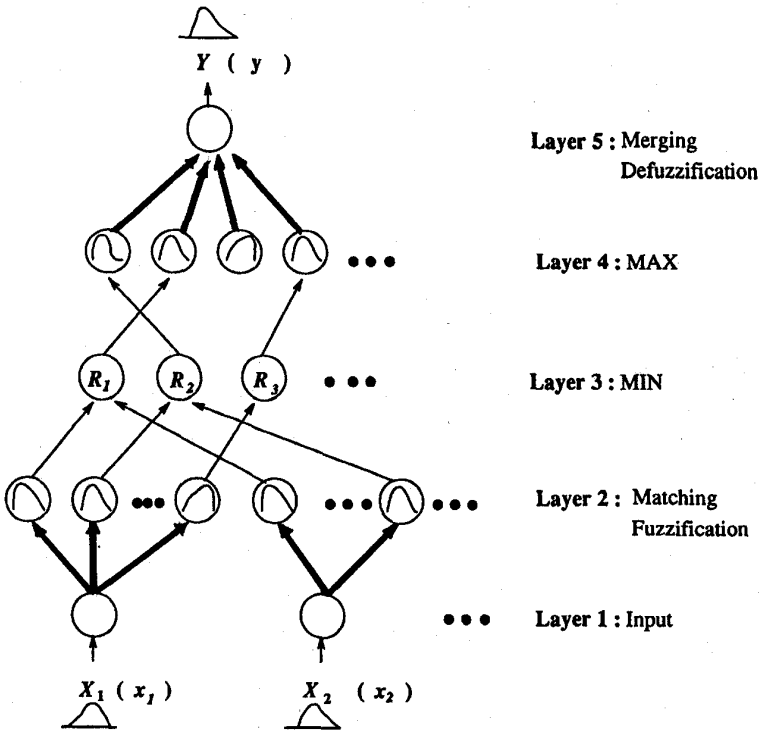


Fig. 3. The five-layered architecture of the proposed neural fuzzy system. The bold lines indicate the links with fuzzy weights and the plain lines indicate the links with numerical weights.

We shall next describe the signal propagation in the proposed network layer by layer following the arrow directions shown in Fig. 3. This is done by defining the transfer function of a node in each layer. Signal may flow in the reverse direction in the learning process as we shall discuss in the following sections. In the following description, we shall consider the case of single output node for clarity. It can be easily extended to the case of multiple output nodes. A typical neural network consists of nodes, each of which has some finite fan-in of connections represented by weight values from other nodes and fan-out of connections to other nodes (see Fig. 4). The notations u and U represent the input crisp and fuzzy numbers of the node, respectively. The notations o and O represent, respectively, the output crisp and fuzzy numbers. The superscript in the following formulas indicates the layer number.

Layer 1 (Input): If the input is a fuzzy number, each node in this layer only transmits input fuzzy number X_i to the next layer directly. No computation is done in this layer. That is,

$$O_i^1 = \bigcup_\alpha \alpha\left[o_{i1}^{1(\alpha)}, o_{i2}^{1(\alpha)}\right] = X_i = \bigcup_\alpha \alpha\left[x_{i1}^{(\alpha)}, x_{i2}^{(\alpha)}\right]. \quad (21)$$

If the input is a crisp number x_i, it can be viewed as a fuzzy *singleton*, i.e.,

$$O_i^1 = \bigcup_\alpha \alpha\left[o_{i1}^{1(\alpha)}, o_{i2}^{1(\alpha)}\right] = \bigcup_\alpha \alpha[x_i, x_i]. \quad (22)$$

Note that there is no weight to be adjusted in this layer.

Layer 2 (Matching/Fuzzification): Each node in this layer has exactly one input from some input linguistic node, and feeds its output to rule node(s). For each layer-2 node, the input is a fuzzy number and the output is a numerical number. The weight in this layer is a fuzzy number WX_{ij}. The index i, j means the jth term of the ith input linguistic variable X_i. The transfer function of each layer-2 node is

$$f_{ij}^2 = \text{diff}(WX_{ij}, U_i) = \frac{1}{2}\sum_\alpha \left[\left(wx_{ij1}^{(\alpha)} - u_{i1}^{2(\alpha)}\right)^2 + \left(wx_{ij2}^{(\alpha)} - u_{i2}^{2(\alpha)}\right)^2\right] \quad (23)$$

$$o_{ij}^2 = a\left(f_{ij}^2\right) = e^{-(f_{ij}^2)^2/2\sigma^2} \quad (24)$$

where σ^2 is the variance of the activation function $a(\cdot)$. It is a constant given in advance. The activation function $a(\cdot)$ is a nonnegative, monotonically decreasing function of $f_{ij}^2 \in [0, \infty]$, and $a(0)$ is equal to 1. For example, $a(\cdot)$ can also be given alternatively as

$$o_{ij}^2 = a\left(f_{ij}^2\right) = r^{f_{ij}^2} \quad (25)$$

where $0 < r < 1$, or

$$o_{ij}^2 = a\left(f_{ij}^2\right) = \frac{2}{1 + e^{-\lambda f_{ij}^2}} \quad (26)$$

where λ is a nonnegative constant.

Layer 3 (MIN): The input and output of the node in this layer are both numerical. The links in this layer perform precondition matching of fuzzy logic rules. Hence, the rule nodes should perform fuzzy AND operation. The most commonly

used fuzzy AND operations are *intersection* and *algebraic product* [13]. If intersection is used, we have

$$o_i^3 = \min\left(u_1^3, u_2^3, \ldots, u_k^3\right). \tag{27}$$

On the other hand, if algebraic product is used, we have

$$o_i^3 = u_1^3 u_2^3 \cdots u_k^3. \tag{28}$$

Similar to layer one, there is no weight to be adjusted in this layer.

Layer 4 (MAX): The nodes in this layer should perform fuzzy OR operation to integrate the fired rules which have the same consequent. The most commonly used fuzzy OR operations are *union* and *bounded sum* [13]. If the union operation is used in this model, we have

$$o_i^4 = \max\left(u_1^4, u_2^4, \ldots, u_k^4\right). \tag{29}$$

If the bounded sum is used, we have

$$o_i^4 = \min\left(1, u_1^4 + u_2^4 + \cdots + u_k^4\right). \tag{30}$$

The output and input of each layer-4 node are both numerical values.

Layer 5 (Merging/Defuzzification): In this layer, each node has a fuzzy weight WY_i. There are two kinds of operations in this layer. When we need a fuzzy output Y, the following formula is executed to perform a "merging" action:

$$O^5 = \bigcup_\alpha \alpha\left[o_1^{5(\alpha)}, o_2^{5(\alpha)}\right] = Y = \frac{\sum_i u_i^5 \cdot WY_i}{\sum_i u_i^5}. \tag{31}$$

Namely,

$$\begin{aligned} Y &= \bigcup_\alpha \alpha\left[y_1^{(\alpha)}, y_2^{(\alpha)}\right] \\ WY_i &= \bigcup_\alpha \alpha\left[wy_{i1}^{(\alpha)}, wy_{i2}^{(\alpha)}\right] \end{aligned} \tag{32}$$

where

$$y_1^{(\alpha)} = \frac{\sum_i u_i^5 wy_{i1}^{(\alpha)}}{\sum_i u_i^5} \tag{33}$$

$$y_2^{(\alpha)} = \frac{\sum_i u_i^5 wy_{i2}^{(\alpha)}}{\sum_i u_i^5}. \tag{34}$$

If the output of the neural fuzzy system is required to be a numerical value, the output node executes the defuzzification action. The following formulas simulate the Tsukamoto's defuzzification method [20]

$$f_i = \text{defuzzifier}(WY_i) = \frac{\sum_\alpha \alpha\left(wy_{i1}^{(\alpha)} + wy_{i2}^{(\alpha)}\right)}{2\sum_\alpha \alpha} \tag{35}$$

$$y = o^5 = \frac{\sum_i u_i^5 \sigma_i f_i}{\sum_i u_i^5 \sigma_i} \tag{36}$$

where

$$\sigma_i = \sum_\alpha \left(wy_{i2}^{(\alpha)} - wy_{i1}^{(\alpha)}\right). \tag{37}$$

According to the function processing of the proposed network described in the above, our network can process both fuzzy numbers and numerical numbers by viewing the latter as fuzzy singletons. It is noted that although the topology of the proposed network is almost the same as that of the conventional ones such as [17]–[19] (i.e., they are all five-layer fuzzy inference networks), their functions are quite different. Especially, the fuzzy weights used in our network allow the proposed system to process and adapt fuzzy data in addition to numerical data. However, the conventional fuzzy inference networks can only process numerical data. Although some normal multilayer neural networks have been proposed for processing and learning fuzzy and numerical data (e.g., [4]–[8]), they suffer the inherited problem of neural networks—the hidden layers are opaque to the user. Hence the outside observer cannot understand what the network has learned exactly. This makes the normal multilayer networks difficult to be used for rule extraction from fuzzy training data, and for fuzzy rule base concentration.

In the rest of this paper, we call the above proposed five-layered neural network with fuzzy output as "Fuzzy Connectionist Architecture with Linguistic Output" (FCLO), and that with numerical output as "Fuzzy Connectionist Architecture with Numerical Output" (FCNO). From the above description we observe that only layer-1 inputs and layer-5 outputs of the FCLO and FCNO are possibly fuzzy numbers (in the form of α-level sets). Real numbers are propagated internally from layer two to layer four in the FCLO and FCNO. This makes the operations in our proposed network less time-consuming as compared to the neural networks that can also process fuzzy input/output data but require fuzzy signals flowing in it. It is noted that (see subsection II-A) the linguistic output Y must be a fuzzy number; that is, it must be convex and normal. To guarantee this, we have to keep the fuzzy weights in our network convex and normal. This constraint should be satisfied in deriving weight update rules in the following section.

IV. Supervised Learning Algorithm

In this section, we shall derive a supervised learning algorithm for the proposed neural fuzzy system (FCLO and FCNO). This algorithm is applicable to the situations that pairs of input-output training data are available. We allow the training data (either input or output) to be numerical values (vectors), fuzzy numbers, or mixture of them. When the system is trained to be a fuzzy controller or fuzzy classifier, the input and desired output are usually numerical values. On the other hand, when the system is trained to be a fuzzy expert system, the input and desired output are usually fuzzy numbers (linguistic information). In this case, the fuzzy input-output training pairs can be regarded as fuzzy if-then rules and the trained neural fuzzy system is like a (*condensed*) fuzzy knowledge base. Consider for example the following two training fuzzy if-then rules:

R_1: IF x_1 is small and x_2 is large, THEN y is good

R_2: IF x_1 is large and x_2 is small, THEN y is bad.

Then the corresponding two input-output training pairs are "(small, large; good)" and "(large, small; bad)," where the

fuzzy terms are defined by given fuzzy numbers in the form of α-level sets. In general, the fuzzy rules for trainging are

R_p: IF x_1 is X_{p1} and $\cdots$ and x_p is X_{pn}, THEN y is Y_p

where $p = 1, 2, \ldots, m$, and m is the total number of training rules. These fuzzy if-then rules can be viewed as the fuzzy input-output pairs, $(X_{p1}, X_{p2}, \ldots, X_{pn}; Y_p)$, where $p = 1, 2, \ldots, m$. If the input or output are crisp data, the corresponding fuzzy elements in the training pairs become numerical elements.

With the supervised learning algorithm that we shall develop, the proposed system can learn fuzzy if-then rules from numerical data. Moreover, it can learn fuzzy if-then rules from experts' linguistic knowledge represented by fuzzy if-then rules. This means that it can learn to represent a set of training fuzzy if-then rules using another smaller set of fuzzy if-then rules. This is a novel and efficient approach to rule combination. The proposed neural fuzzy system can thus be used for *rule base concentration* to reduce the number of rules. This provides a useful tool for designing a fuzzy knowledge base. A knowledge base is usually contributed by several domain experts, so duplication of if-then rules is inevitable. Hence, we usually need to compress the rule base by combining similar rules into representative rules.

Before the learning of the neural fuzzy system, an initial network structure is first constructed. Then during the learning process, some nodes and links in the initial network are deleted or combined to form the final structure of the network. At first, the number of input (output) nodes is set equal to the number of input (output) linguistic variables. The number of nodes in the second layer is decided by the number of fuzzy partitions of each input linguistic variable x_i, $|T(x_i)|$, which must be assigned by the user. The fuzzy weights WX_{ij} in layer two are initialized randomly as fuzzy numbers. One better way is to distribute the initial fuzzy weights evenly on the interested domain of the corresponding input linguistic variable. As for layer three of the initial network, there are $\prod_i |T(x_i)|$ rule nodes with the inputs of each rule node coming from one possible combination of the terms of input linguistic variables under the constraint that only one term in a term set can be a rule node's input. This gives the preconditions of initial fuzzy rules.

Finally, let us consider the structure of layer four in the initial network. This is equivalent to determining the consequents of initial fuzzy rules. We propose two approaches to initializing the layer-4 nodes and the links from layer-3 nodes to layer-4 nodes.

Method 1: The first method uses the *self-organized learning* technique. To use this method, the number of fuzzy partitions of the output linguistic variable y, $|T(y)|$, must be given in advance. The initial fuzzy weights WY_i in layer five are distributed evenly on the output space. We let the layer-5 nodes to perform *up-down* transmission. In the up-down transmission, the desired outputs are pumped into the network from its output side and the operations in layer five are the same as those in layer two. Signals from both external sides of the network can thus reach the output points of term nodes at layer two and layer four. Since the outputs of term nodes at layer two can be transmitted to rule nodes through the initial architecture of layer-3 links, we can obtain the output (firing strength) of each rule node. Based on the firing strengths of rule nodes (o_i^3) and the outputs of term nodes at layer four (o_j^4), we can decide the correct consequent links of each rule node by unsupervised (self-organized) learning method [21]. The links at layer four are fully connected initially. We denote the weight on the link from the ith rule node to the jth output term node as w_{ij}. The Hebbian learning law is used to update these weights for each training data set. The learning rule is described as

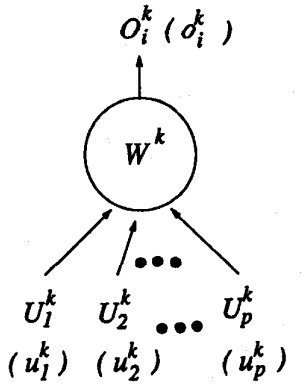


Fig. 4. Basic structure of a node in the proposed neural fuzzy system.

$$\triangle w_{ij} = c o_j^4 o_i^3 \tag{38}$$

where c is a positive constant. After the learning, only one link with the biggest weight in the fan-out of a layer-3 (rule) node is remained as its consequent.

Method 2: This method is simpler than Method 1, but it requires a larger memory capacity. First, let the number of nodes in layer four be the same as the number of rule nodes in layer three. Also, the fuzzy weights in layer five are assigned randomly. The mapping from layer-3 nodes to layer-4 nodes is one-to-one initially. That is, each layer-3 node connects to its respective layer-4 node. Some of layer-4 links and nodes will be eliminated properly in the structure learning process which will be described in subsections IV-B.

With the above initialization process, the network is ready for learning. We shall next propose a two-phase supervised learning algorithm for our five-layered neural fuzzy system. In phase one, a parameter learning scheme is used to adjust the fuzzy weights. In phase two, a structure learning scheme is used to delete or combine some nodes and links in the neural fuzzy system.

A. Parameter Learning Phase

A gradient-descent-based backpropagation algorithm [21], [22] is employed to adjust fuzzy weights in layer two and layer five of the proposed network. If the FCLO is used, the error function to be minimized is

$$e = \text{diff}(Y, D) = \frac{1}{2} \sum_{\alpha} \left[\left(y_1^{(\alpha)} - d_1^{(\alpha)} \right)^2 + \left(y_2^{(\alpha)} - d_2^{(\alpha)} \right)^2 \right] \tag{39}$$

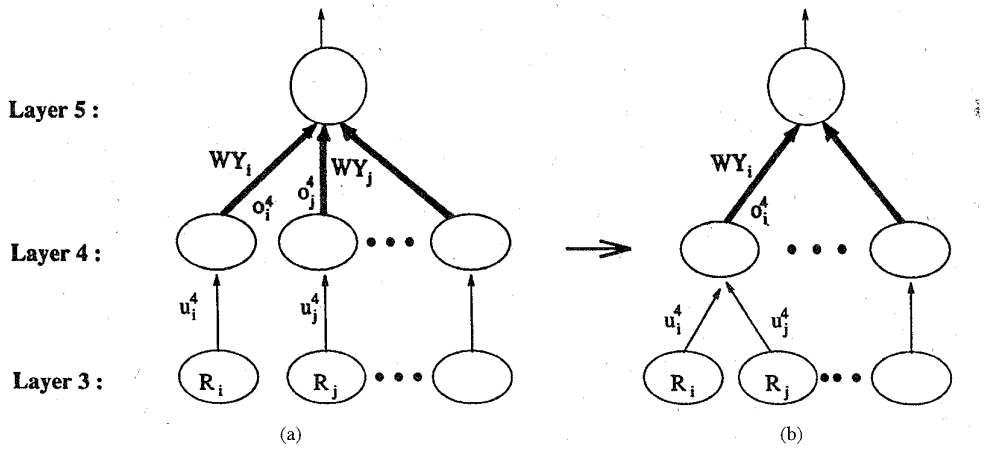


Fig. 5. Illustration of consequent combination.

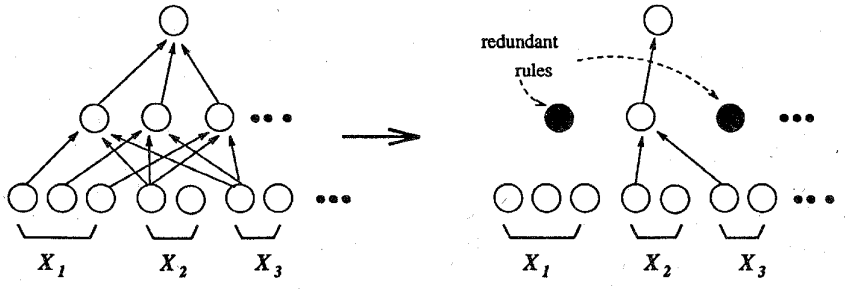


Fig. 6. Illustration of rule combination.

where $Y = \bigcup_\alpha \alpha[y_1^{(\alpha)}, y_2^{(\alpha)}]$ is the current fuzzy output and $D = \bigcup_\alpha \alpha[d_1^{(\alpha)}, d_2^{(\alpha)}]$ is the desired fuzzy output. If the FCNO is used, the error function to be minimized is

$$e = \frac{1}{2}(d - y)^2 \tag{40}$$

where y is the current output and d is the desired output. We assume that $W = \bigcup_\alpha \alpha[w_1^{(\alpha)}, w_2^{(\alpha)}]$ is the adjustable fuzzy parameter in layer two and layer five. Then to update fuzzy weights means to update the parameters $w_1^{(\alpha)}$ and $w_2^{(\alpha)}$. We shall next derive the update rules for these parameters layer by layer based on the general learning rule

$$w(t+1) = w(t) + \eta\left(-\frac{\partial e}{\partial w}\right) \tag{41}$$

where w represents $w_1^{(\alpha)}$ or $w_2^{(\alpha)}$, and η is the learning rate.

Layer 5: If an FCLO is used and the desired output is fuzzy number Y, the update rules of $wy_{i1}^{(\alpha)}$ and $wy_{i2}^{(\alpha)}$ are derived from (33) and (34) as follows:

$$\frac{\partial e}{\partial wy_{i1}^{(\alpha)}} = \frac{\partial e}{\partial y_1^{(\alpha)}}\frac{\partial y_1^{(\alpha)}}{\partial wy_{i1}^{(\alpha)}} = \left(y_1^{(\alpha)} - d_1^{(\alpha)}\right)\frac{u_i^5}{\sum_i u_i^5} \tag{42}$$

$$\frac{\partial e}{\partial wy_{i2}^{(\alpha)}} = \frac{\partial e}{\partial y_2^{(\alpha)}}\frac{\partial y_2^{(\alpha)}}{\partial wy_{i2}^{(\alpha)}} = \left(y_2^{(\alpha)} - d_2^{(\alpha)}\right)\frac{u_i^5}{\sum_i u_i^5}. \tag{43}$$

The error signals to be propagated to the preceding layer are

$$\delta_1^{5(\alpha)} = \frac{\partial e}{\partial o_1^{5(\alpha)}} = \frac{\partial e_1^{(\alpha)}}{\partial y_1^{(\alpha)}} = \left(y_1^{(\alpha)} - d_1^{(\alpha)}\right) \tag{44}$$

$$\delta_2^{5(\alpha)} = \frac{\partial e}{\partial o_2^{5(\alpha)}} = \frac{\partial e_2^{(\alpha)}}{\partial y_2^{(\alpha)}} = \left(y_2^{(\alpha)} - d_2^{(\alpha)}\right) \tag{45}$$

where

$$e_1^{(\alpha)} = \frac{1}{2}\left(y_1^{(\alpha)} - d_1^{(\alpha)}\right)^2 \tag{46}$$

$$e_2^{(\alpha)} = \frac{1}{2}\left(y_2^{(\alpha)} - d_2^{(\alpha)}\right)^2. \tag{47}$$

If an FCNO is used and the desired output is numerical value y, the update rules of the parameters are derived from (35) and (36) as follows:

$$\frac{\partial e}{\partial wy_{i1}^{(\alpha)}} = \frac{\partial e}{\partial y}\frac{\partial y}{\partial wy_{i1}^{(\alpha)}} = (y - d)\frac{\partial y}{\partial wy_{i1}^{(\alpha)}} \tag{48}$$

where

$$\begin{aligned}\frac{\partial y}{\partial wy_{i1}^{(\alpha)}} &= \frac{\partial y}{\partial f_i}\frac{\partial f_i}{\partial wy_{i1}^{(\alpha)}} + \frac{\partial y}{\partial \sigma_i}\frac{\partial \sigma_i}{\partial wy_{i1}^{(\alpha)}} \\ &= \frac{u_i^5\sigma_i}{\sum_i u_i^5\sigma_i}\frac{\alpha}{2\sum_\alpha \alpha} \\ &\quad - \frac{\left(\sum_i u_i^5\sigma_i\right)u_i^5 f_i - \left(\sum_i u_i^5\sigma_i f_i\right)u_i^5}{\left(\sum_i u_i^5\sigma_i\right)^2}\end{aligned} \tag{49}$$

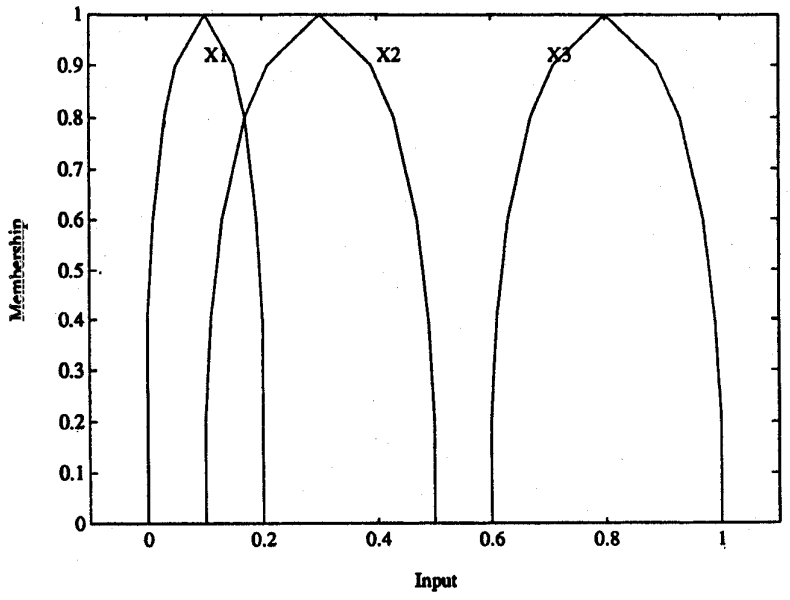


Fig. 7. The membership functions of the input linguistic value "very small" $(X1)$, "small" $(X2)$, "large" $(X3)$,in Example 1.

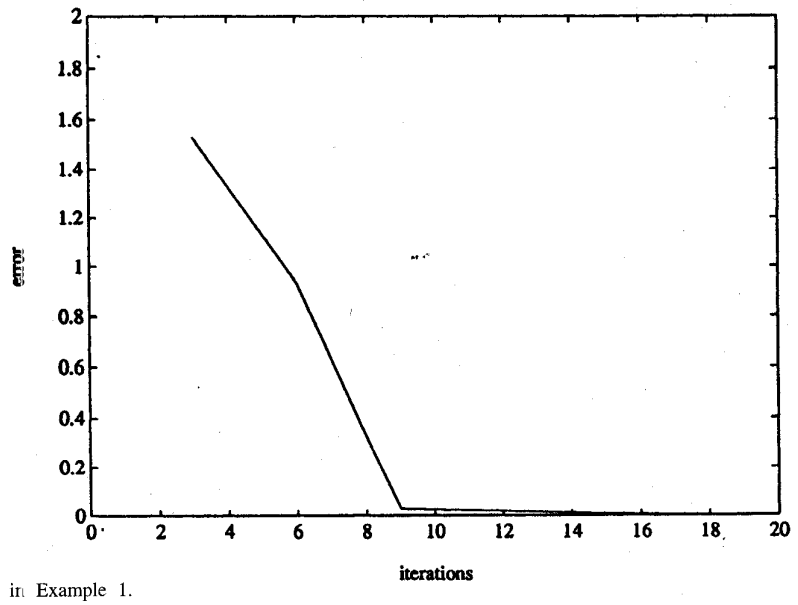


Fig. 8. The learning curve in Example 1.

and

$$\frac{\partial e}{\partial wy_{i2}^{(\alpha)}} = \frac{\partial e}{\partial y}\frac{\partial y}{\partial wy_{i2}^{(\alpha)}} = (y-d)\frac{\partial y}{\partial wy_{i2}^{(\alpha)}} \tag{50}$$

where

$$\begin{aligned}\frac{\partial y}{\partial wy_{i2}^{(\alpha)}} &= \frac{\partial y}{\partial f_i}\frac{\partial f_i}{\partial wy_{i2}^{(\alpha)}} + \frac{\partial y}{\partial \sigma_i}\frac{\partial \sigma_i}{\partial wy_{i2}^{(\alpha)}} \\ &= \frac{u_i^5\sigma_i}{\sum_i u_i^5\sigma_i}\frac{\alpha}{2\sum_\alpha \alpha} \\ &\quad + \frac{\left(\sum_i u_i^5\sigma_i\right)u_i^5 f_i - \left(\sum_i u_i^5\sigma_i f_i\right)u_i^5}{\left(\sum_i u_i^5\sigma_i\right)^2}. \end{aligned} \tag{51}$$

The error signal to be propagated to the preceding layer is

$$\delta^5 = \frac{\partial e}{\partial y} = (y-d). \tag{52}$$

Layer 4: In this layer, there is no weights to be adjusted. Only the error signals need to be computed and propagated. If an FCLO is used, the error signal δ_i^4 is derived from (29) as follows:

$$\delta_i^4 = \frac{\partial e}{\partial o_i^4} = \frac{\partial \sum_\alpha \left(e_1^{(\alpha)} + e_2^{(\alpha)}\right)}{\partial o_i^4} = \sum_\alpha \left(\delta_{i1}^{4(\alpha)} + \delta_{i2}^{4(\alpha)}\right) \tag{53}$$

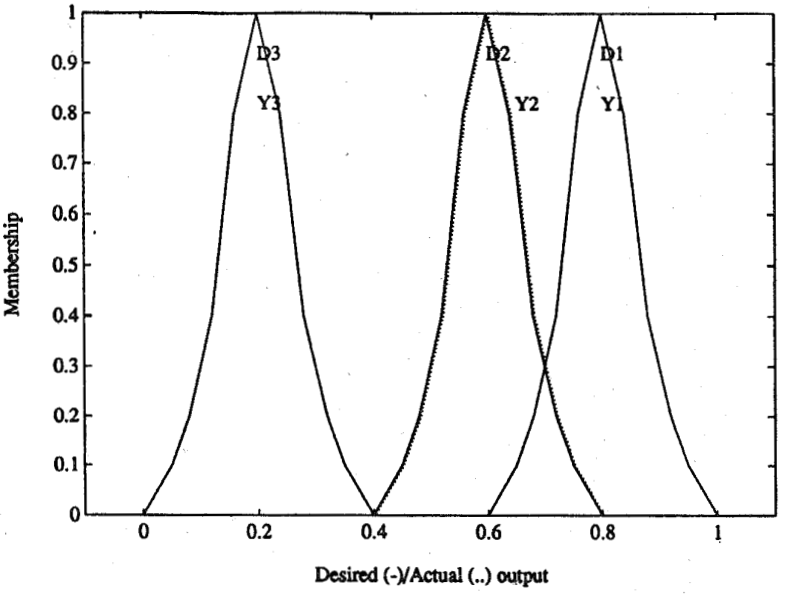


Fig. 9. The actual fuzzy outputs, $Y1, Y2, Y3$, of the learned neural fuzzy system and the corresponding desired fuzzy outputs, $D1, D2, D3$, in Example 1.

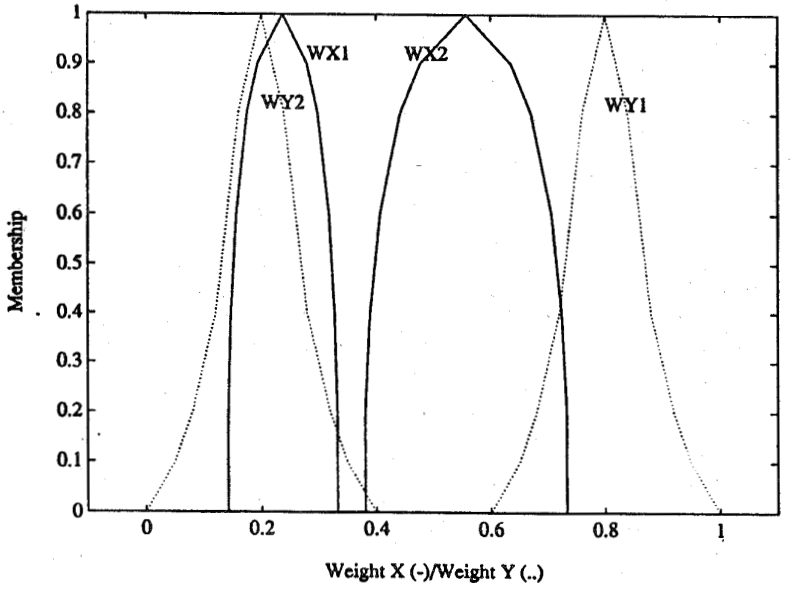


Fig. 10. The learned fuzzy weights of the FCLO in Example 1.

where

$$\delta_{i1}^{4(\alpha)} = \frac{\partial e_1^{(\alpha)}}{\partial o_i^4} = \frac{\partial e_1^{(\alpha)}}{\partial o_1^{5(\alpha)}} \frac{\partial o_1^{5(\alpha)}}{\partial o_i^4}$$
$$= \delta_1^{5(\alpha)} \frac{\partial y_1^{(\alpha)}}{\partial o_i^4} = \delta_1^{5(\alpha)} \frac{wy_{i1}^{(\alpha)} - \sum_i u_i^5 wy_{i1}^{(\alpha)}}{\left(\sum_i u_i^5\right)^2} \quad (54)$$

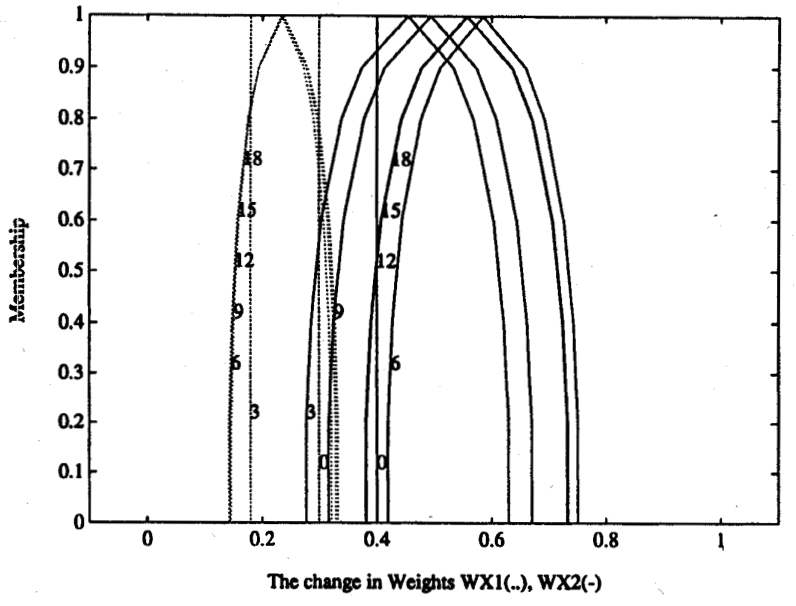
$$\delta_{i2}^{4(\alpha)} = \frac{\partial e_2^{(\alpha)}}{\partial o_i^4} = \frac{\partial e_2^{(\alpha)}}{\partial o_2^{5(\alpha)}} \frac{\partial o_2^{5(\alpha)}}{\partial o_i^4}$$
$$= \delta_2^{5(\alpha)} \frac{\partial y_1^{(\alpha)}}{\partial o_i^4} = \delta_1^{5(\alpha)} \frac{wy_{i2}^{(\alpha)} - \sum_i u_i^5 wy_{i2}^{(\alpha)}}{\left(\sum_i u_i^5\right)^2}. \quad (55)$$

If FCNO is used, the error signal δ_i^4 is derived from (29) as follows:

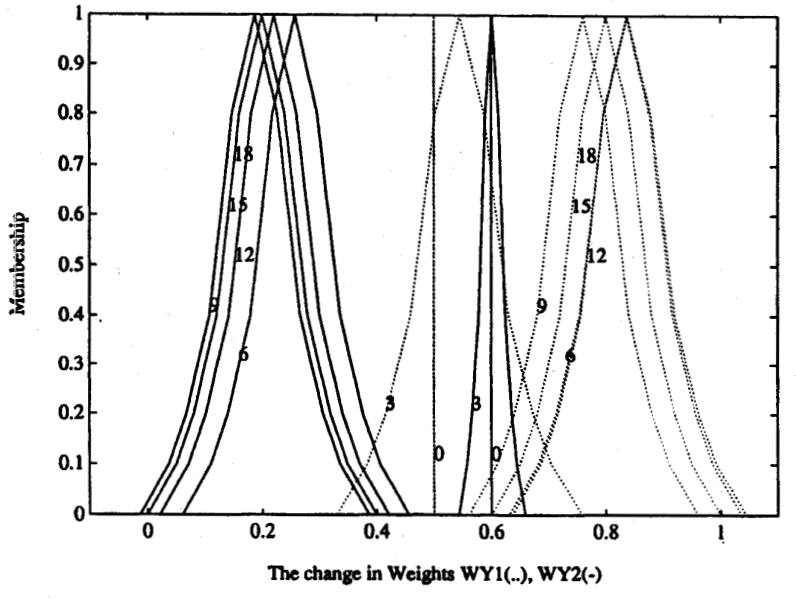
$$\delta_i^4 = \frac{\partial e}{\partial o_i^4} = \frac{\partial e}{\partial o^5} \frac{\partial o^5}{\partial o_i^4}$$
$$= \delta^5 \frac{\left(\sum_i u_i^5 \sigma_i\right) f_i \sigma_i - \left(\sum_i u_i^5 \sigma_i f_i\right) \sigma_i}{\left(\sum_i u_i^5 \sigma_i\right)^2}. \quad (56)$$

Layer 3: As in layer four, only the error signals need to be computed. According to (27), this error signal δ_i^3 can be derived as

$$\delta_i^3 = \frac{\partial e}{\partial o_i^3} = \frac{\partial e}{\partial o_i^4} \frac{\partial o_i^4}{\partial o_i^3}$$



(a)



(b)

Fig. 11. (a) Time evolving graph of fuzzy weights $WX1, WX2$, during the learning process in Example 1. (b) Time evolving graph of fuzzy weights $WY1, WY2$, during the learning process in Example 1.

$$= \begin{cases} \delta_i^4 & \text{if } o_i^4 = \max\left(u_1^4, \ldots, u_k^4\right) \\ 0 & \text{otherwise.} \end{cases} \tag{57}$$

Layer 2: In this layer, there are fuzzy weights WX to be adjusted. The update rules can be derived from (23) and (24) as follows:

$$\frac{\partial e}{\partial wx_{ij1}^{(\alpha)}} = \frac{\partial e}{\partial o_i^3}\frac{\partial o_i^3}{\partial o_{ij}^2}\frac{\partial o_{ij}^2}{\partial wx_{ij1}^{(\alpha)}} = \delta_i^3\frac{\partial o_i^3}{\partial o_{ij}^2}\frac{\partial o_{ij}^2}{\partial wx_{ij1}^{(\alpha)}} \tag{58}$$

$$\frac{\partial e}{\partial wx_{ij2}^{(\alpha)}} = \frac{\partial e}{\partial o_i^3}\frac{\partial o_i^3}{\partial o_{ij}^2}\frac{\partial o_{ij}^2}{\partial wx_{ij2}^{(\alpha)}} = \delta_i^3\frac{\partial o_i^3}{\partial o_{ij}^2}\frac{\partial o_{ij}^2}{\partial wx_{ij2}^{(\alpha)}} \tag{59}$$

where

$$\frac{\partial o_i^3}{\partial o_{ij}^2} = \begin{cases} 1 & \text{if } o_{ij}^3 = \min(u_1^3, \ldots, u_k^3) \\ 0 & \text{otherwise} \end{cases} \tag{60}$$

and

$$\frac{\partial o_{ij}^2}{\partial wx_{ij1}^{(\alpha)}} = o_{ij}^2\left(-\frac{2f_{ij}^2}{2\sigma}\right)\frac{\partial f_{ij}^2}{\partial wx_{ij1}^{(\alpha)}} = -o_{ij}^2\frac{f_{ij}^2}{\sigma^2}\left(wx_{ij1}^{(\alpha)} - u_{i1}^{2(\alpha)}\right), \tag{61}$$

$$\frac{\partial o_{ij}^2}{\partial wx_{ij2}^{(\alpha)}} = -o_{ij}^2\frac{f_{ij}^2}{\sigma^2}\left(wx_{ij2}^{(\alpha)} - u_{i2}^{2(\alpha)}\right). \tag{62}$$

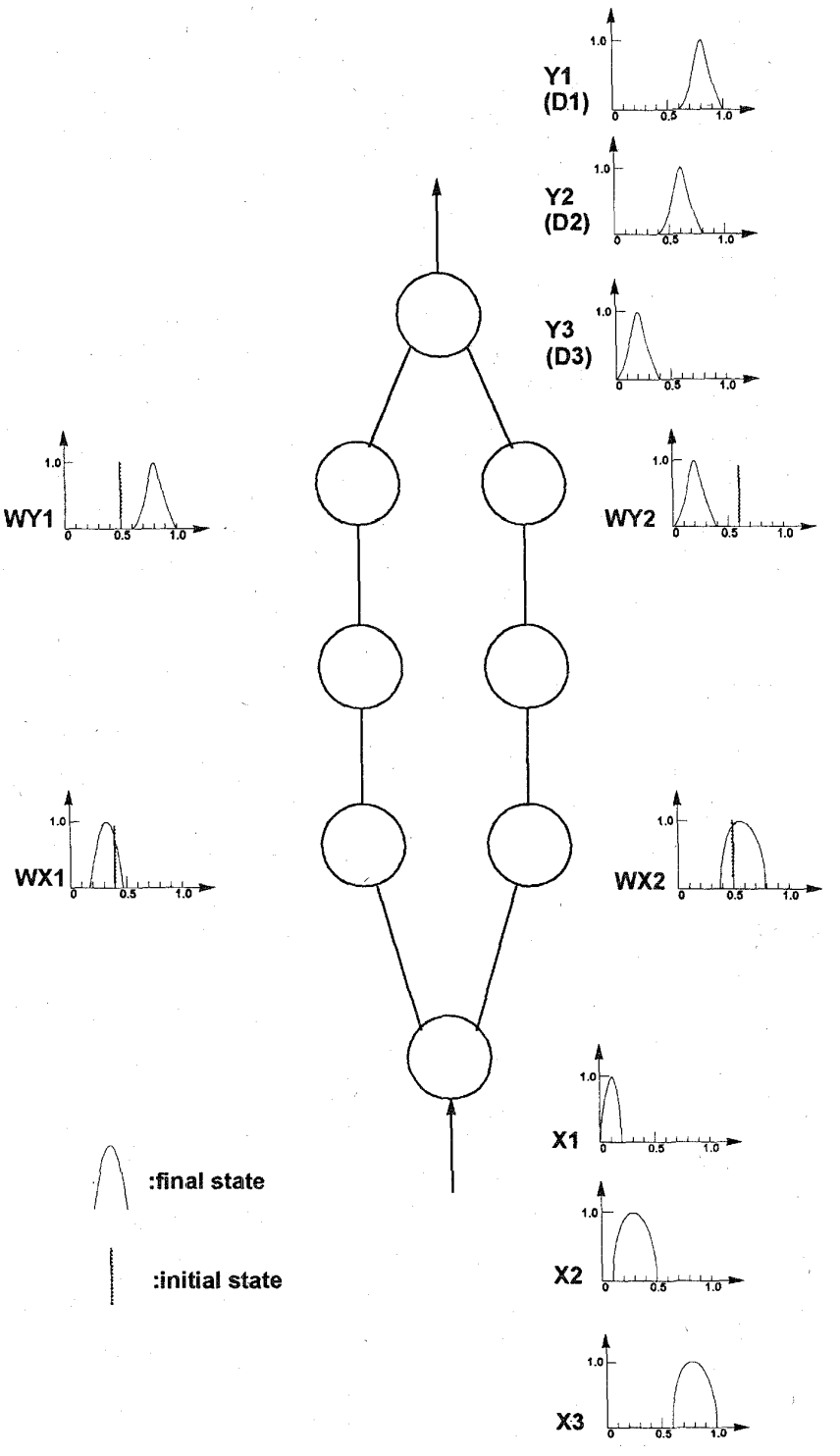


Fig. 12. The neural fuzzy system before and after learning in Example 1.

When fuzzy weights are adjusted by (42)–(62), two undesirable situations may happen. That is, the lower limits of the α-level sets of fuzzy weights may exceed the upper limits, and the updated fuzzy weighted may become nonconvex. In order to cope with these undesirable situations, we do necessary modifications on the updated fuzzy weights to make sure that they are legal fuzzy numbers after updating. This process is described as follows:

Procedure: Fuzzy Number Restoration

Inputs:} Fuzzy weights $W = \bigcup_\alpha \alpha[w_1^{(\alpha)}, w_2^{(\alpha)}]$ adjusted by (42)–(62).

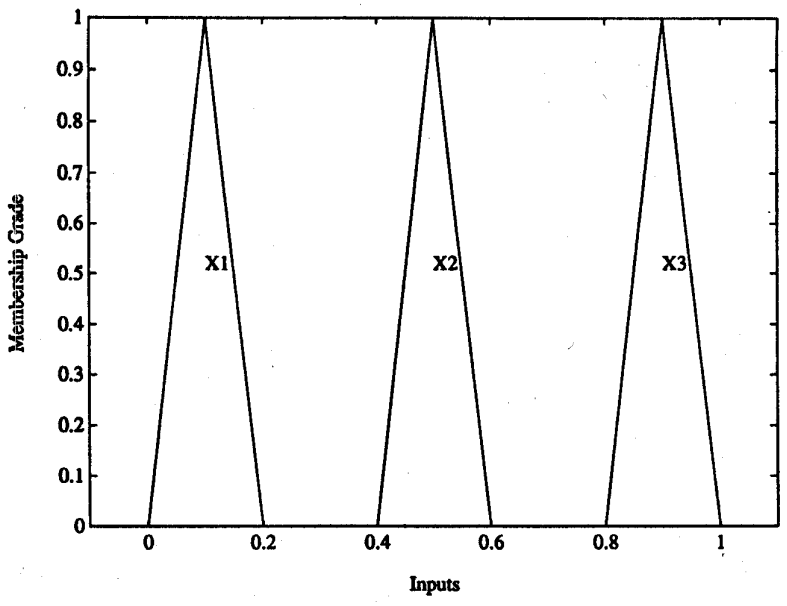


Fig. 13. The membership functions of the input linguistic values $X1, X2, X3$, in Example 2.

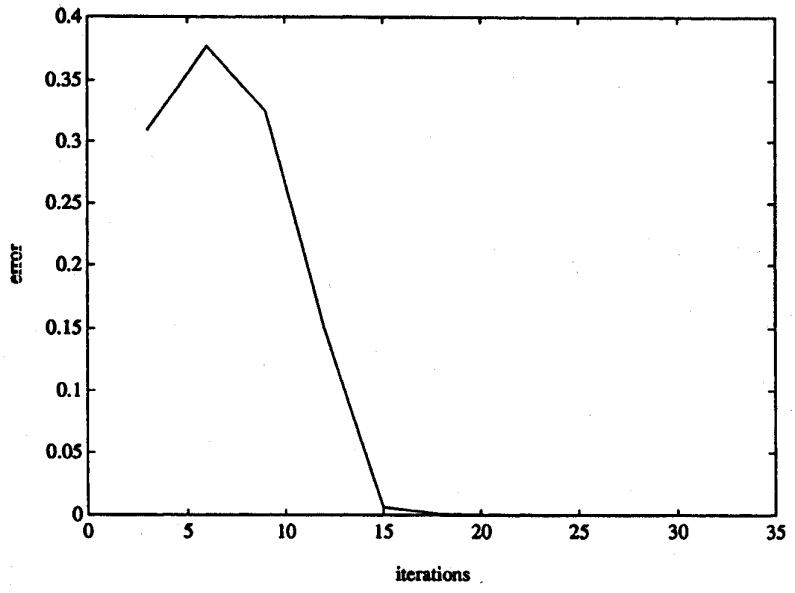


Fig. 14. The learning curve in Example 2.

Outputs:} The modified fuzzy weights $\hat{W} = \bigcup_\alpha \alpha[\hat{w}_1^{(\alpha)}, \hat{w}_2^{(\alpha)}]$ which are legal fuzzy numbers.

Step 1:

$k = 1$

if $w_1^{(k)} > w_2^{(k)}$ then $\hat{w}_1^{(k)} = w_2^{(k)}$ and $\hat{w}_2^{(k)} = w_2^{(k)}$,

else $\hat{w}_1^{(k)} = w_1^{(k)}$ and $\hat{w}_2^{(k)} = w_2^{(k)}$.

Step 2:

For $k = h - 1$ down to 0, do

if $w_1^{(k/h)} > \hat{w}_1^{(\frac{k+1}{h})}$, then $\hat{w}_1^{(k/h)} = \hat{w}_1^{(\frac{k+1}{h})}$

else $\hat{w}_1^{(k/h)} = w_1^{(k/h)}$

and

if $w_2^{(k/h)} < \hat{w}_2^{(\frac{k+1}{h})}$, then $\hat{w}_2^{(k/h)} = \hat{w}_2^{(\frac{k+1}{h})}$

else $\hat{w}_2^{(k/h)} = w_2^{(k/h)}$.

Step 3:

Output $\hat{W}$, and stop.

B. Structure Learning Phase

In this subsection, we propose a structure learning algorithm for the proposed neural fuzzy system to reduce its node and link number. This structure learning algorithm is divided into two parts: One is to merge the fuzzy terms of input and output linguistic variables (*term-node combination*). The other is to do *rule combination* to reduce the number of rules. We shall discuss these two parts separately below.

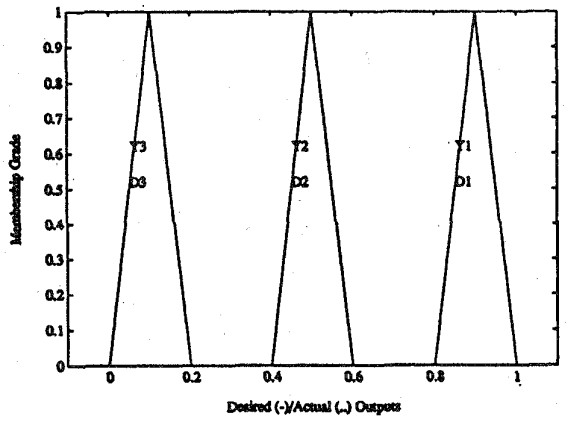


Fig. 15. The actual fuzzy outputs, $Y1, Y2, Y3$, of the learned neural fuzzy system and the corresponding desired fuzzy outputs, $D1, D2, D3$, in Example 2.

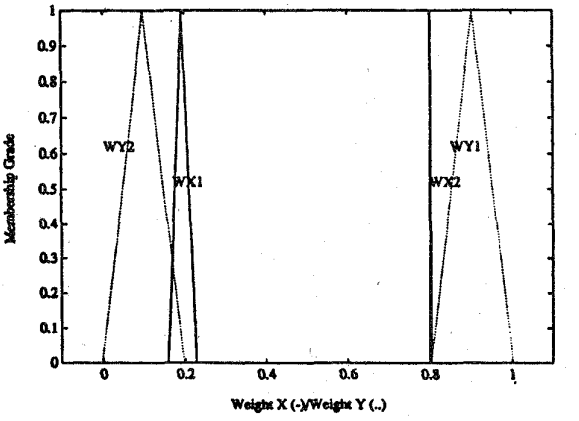


Fig. 16. The learned fuzzy weights of the FCLO in Example 2.

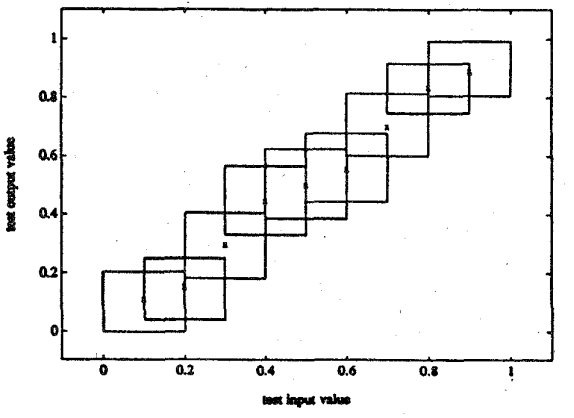


Fig. 17. Generalization test of the learned neural fuzzy system in Example 2.

Term-Node Combination Scheme: Term-node combination is to combine similar terms in the term sets of input and output linguistic variables. We shall present this technique for the term set of output linguistic variables. It is applied to the term set of input linguistic variables in exactly the same way. We have proposed two methods to initialize the nodes and links in layer four of FCLO and FCNO in the initialization process. If Method 1 is used, the consequents of the rule nodes have been established by self-organized learning in the initialization process. Hence no further structure learning is necessary in the consequent parts of the rule nodes if Method 1 of initialization process is used. On the other hand, if Method 2 is used, further actions should be taken to eliminate redundant nodes and links in layer four. The whole learning procedure using Method 2 of initialization is described as follows:

Step 1: Perform parameter learning until the output error is smaller than a given value; i.e., $e \leq$ *error_limit*, where *error_limit* is a small positive constant.

Step 2: $\mathrm{diff}(WY_i, WY_j) \leq$ *similar_limit* and *similar_limit* is a given positive constant, remove term node j with fuzzy weight WY_j and its fan-out links, and connect rule node j in layer 3 to term node i in layer four (see Fig. 5).

Step 3: Perform the parameter learning again to optimally adjust the network weights.

The term-node combination scheme in Step 2 can automatically find the number of fuzzy partitions of output linguistic variables. The rationale of this step is described mathematically in the followings. Let us consider the supervised learning of FCLO. If Step 2 is not performed (see Fig. 5(a)), FCLO does the following operations in layer 4

$$o_i^4 = u_i^4, o_j^4 = u_j^4 \tag{63}$$

$$Y_a = \frac{u_i^4 WY_i + u_j^4 WY_j + C}{u_i^4 + u_j^4 + c} \tag{64}$$

where c is the sum of other input values and C is the sum of the other products of the input values and fuzzy weights in layer four.

On the other hand, if Step 2 is performed (see Fig. 5(b)), FCLO does the following operations:

$$o_i^4 = \min\left(1, u_i^4 + u_j^4\right). \tag{65}$$

If $u_i^4 + u_j^4 \leq 1$, then

$$Y_b = \frac{\left(u_i^4 + u_j^4\right) WY_i + C}{u_i^4 + u_j^4 + c}. \tag{66}$$

Notice that in (65), the bounded sum is used for fuzzy OR operation. From (64) and (66), we find that if WY_i is very similar to WY_j (as the required condition in Step 2) and $(u_i^4 + u_j^4) \leq 1$, then $Y_a \approx Y_b$. This means that the networks before and after the rule combination process is nearly equivalent. We can obtain the same result for the supervised learning of FCNO. Hence, the proposed term-node combination scheme is appropriate and accurate. The operations in Step 2 can be equally applied to the term set of input linguistic variables.

Rule Combination Scheme: After the fuzzy parameters and the consequents of the rule nodes are determined, the rule combination scheme is performed to reduce the number of rules. The idea of rule combination is easily understood

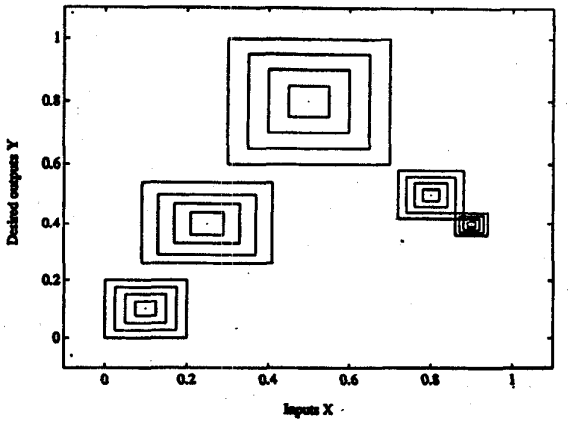


Fig. 18. The training fuzzy pairs (X, Y) in the form of α-level sets in Example 3.

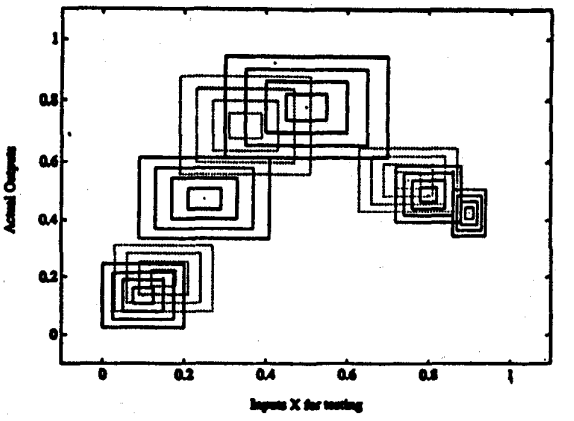


Fig. 20. Generalization test of the learned neural fuzzy system in Example 3.

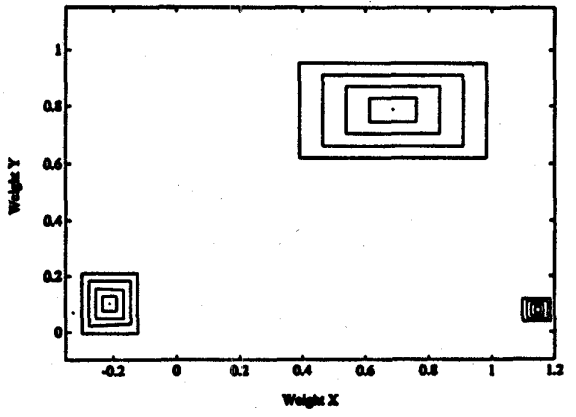


Fig. 19. The learned fuzzy weights in Example 3.

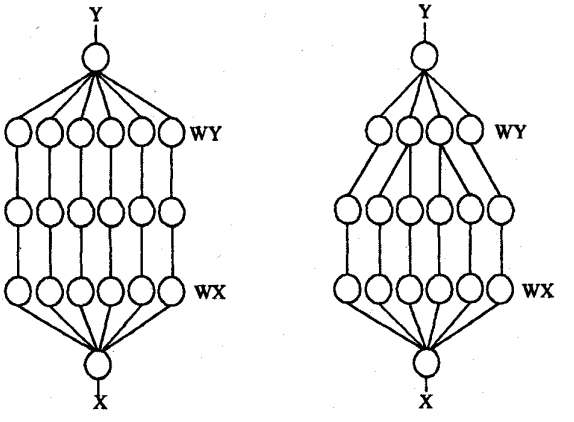


Fig. 21. The structures before and after learning of the FCLO in Example 4.

through the following example. Consider a system contains the following fuzzy if-then rules:

R_1: IF x_1 is small and x_2 is small, THEN y is good
R_2: IF x_1 is medium and x_2 is small, THEN y is good
R_3: IF x_1 is large and x_2 is small, THEN y is good

where the fuzzy partitions of input linguistic variable x_1 are "small," "medium," and "large". The three rules R_1, R_2 and R_3 can be merged to one rule as follows:

R: IF x_2 is small, THEN y is good.

That is, the input variable x_1 is not necessary in this situation. The conditions for applying rule combination has been explored in [17] and are given as below.

1) These rule nodes have exactly the same consequents.
2) Some preconditions are common to all the rule nodes, that is, the rule nodes are associated with the same term nodes.
3) The union of other preconditions of these rule nodes composes the whole terms set of some input linguistic variables.

If some rule nodes satisfy these three conditions, then these rules can be combined into a single rule. An illustration in shown in Fig. 6. It should be noted that the proposed rule combination scheme is exactly only for max-min composition.

V. Illustrative Examples

In this section, we shall use some examples to illustrate the performance of the proposed fuzzy supervised learning algorithm on FCLO and FCNO. All the following simulations are run on a Sun SPARCstation.

Example 1 - Fuzzy Input and Fuzzy Output: Consider the following three fuzzy if-then rules for training:

R_1: IF x is very small (X1), THEN y is very large (D1)
R_2: IF x is small (X2), THEN y is large (D2)
R_3: IF x is large (X3), THEN y is small (D3)

where the fuzzy numbers "small," "large," "very small" are given in Fig. 7. Because input and desired output are linguistic, an FCLO is used in this example. Using Method 2 of initialization process, we set up an FCLO with two layer-2 nodes and two layer-4 nodes (and thus, two layer-3 (rule) nodes). Fig. 8 shows the learning curve. The total learning time is 0.853 s. The error tolerance is 0.0001 and the number

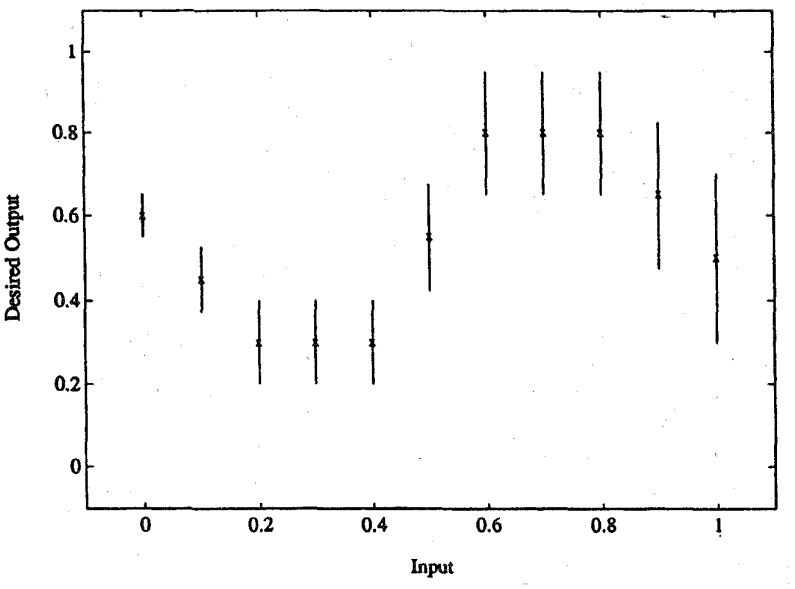


Fig. 22. The training data pairs in Example 4.

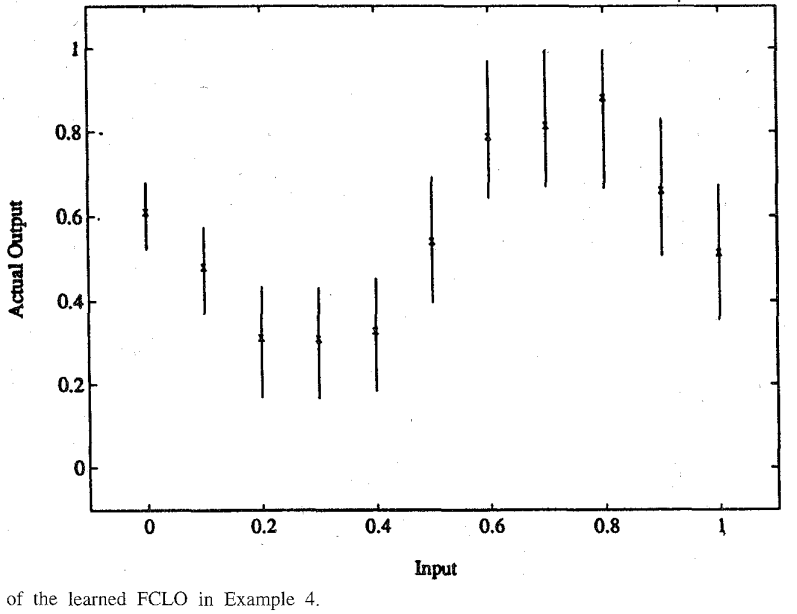


Fig. 23. The actual outputs of the learned FCLO in Example 4.

of α-cuts is 6. After supervised learning, the fuzzy outputs of the learned FCLO and the corresponding desired outputs are shown in Fig. 9. The figure shows that they match closely. The two learned (representative) fuzzy rules after learning (condensing) are

$$\text{IF } x \text{ is } WX1, \text{ THEN } y \text{ is } WY1$$

and

$$\text{IF } x \text{ is } WX2, \text{ THEN } y \text{ is } WY2$$

where the fuzzy weights after learning are shown in Fig. 10. For illustration, Fig. 11 shows the change of fuzzy weights in the learning process. The five-layered network structure of the neural fuzzy system used in this example is shown in Fig. 12. In the figure, the fuzzy weights before and after learning are shown, where the exact fuzzy weights after learning can be found in Fig. 10. It is observed that these weights grow to fuzzy numbers from the initial numerical numbers (fuzzy singletons) when the network is trained with fuzzy input/output data. Three fuzzy input numbers and the corresponding fuzzy output numbers are also shown in Fig. 12 for illustration. Hence the original three fuzzy rules have been condensed to two rules, and these two sets of rules represent equivalent knowledge. As a comparison, when we solved the problem in this example using the approach in [4], the learning time needed to obtain the same learning result is 2.455 s. There are two major reasons that the proposed system can learn faster than the compared one. First, the proposed system allows numerical signals to flow in the network. Second,

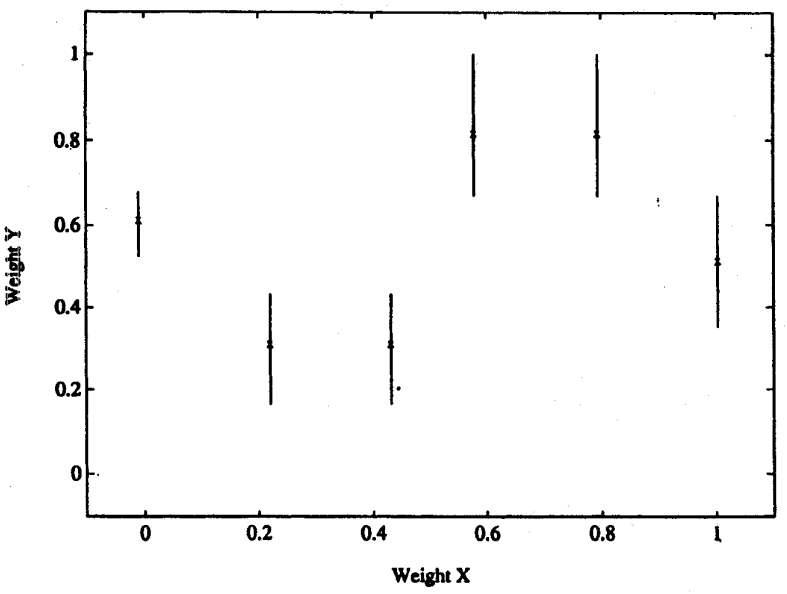


Fig. 24. The learned fuzzy weights in Example 4.

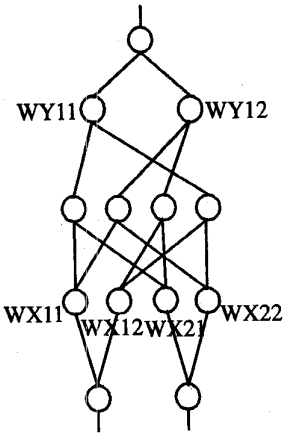


Fig. 25. The structure of the FCNO in Example 5.

the proposed system is a structured neural network (a fuzzy inference network), but the one in [4] is a normal multilayer neural network.

Example 2 - Fuzzy Input and Fuzzy Output: In this simulation, we train an FCLO having the same structure as that in Example 1 using fuzzy training data $\{(X1, D1), (X2, D2), (X3, D3)\}$ shown in Figs. 13 and 15. Here we set the number of α-level sets as 2; i.e., $\alpha = \{0, 1\}$ to emulate the triangular membership function that is used widely. The learning curve shown in Fig. 14 indicates that it takes a little bit more iterations to converge for smaller number of quantized membership grade. The total learning time is 1.572 s. (The learning time needed for the approach in [4] is 4.828 s.) The fuzzy outputs $(Y1, Y2, Y3)$ of the learned FCLO and the corresponding desired outputs are shown in Fig. 15. The figure shows that they match perfectly. The learned fuzzy weights are shown in Fig. 16. We also use novel fuzzy inputs to test the generalization capability of the learned FCLO. The results shown in Fig. 17 demonstrate its good interpolation and extrapolation capability, where the squares correspond to the $\alpha = 0$ level sets.

Example 3 - Fuzzy Input and Fuzzy Output: In this example, we train an FCLO with five training fuzzy number pairs shown in Fig. 18. In this figure, the stacked rectangles represent different α-level sets. Five level sets corresponding to $\alpha = 0, 0.25, 0.5, 0.75, 1$ are used for each fuzzy number. In the initial FCLO, there are four nodes in each of layers 2, 3, and 4. That is, there are four fuzzy rules initially. After the structure and parameter learning, we obtain an FCLO containing three fuzzy rules (i.e., there are three nodes in each of layers 2, 3, and 4). The total learning time is 2.627 s. (The learning time needed for the approach in [4] is 8.422 s.) Fig. 19 shows the learned fuzzy weights. In order to examine the generalization capability of the trained FCLO, we present three novel fuzzy numbers to its input nodes for testing. The results shown in Fig. 20 (the dashed rectangles) indicate the good generalization capability of the learned FCLO.

Example 4 - Crisp Input and Fuzzy Output: In this example, we train an FCLO with training pairs containing numerical inputs and fuzzy outputs. With Method 2 of initialization process, the initial structure of the FCLO is shown in Fig. 21(a). The eleven training data pairs are shown in Fig. 22, where each input is a point, and the desired output is represented by the interval corresponding to the level set of $\alpha = 0$. Five level sets corresponding to $\alpha = 0, 0.25, 0.5, 0.75, 1$ are used for each fuzzy number. After learning, the final FCLO is shown in Fig. 21(b), and its the actual outputs with respect to the training inputs are shown in Fig. 23. The total learning time is 2.392 s. (The learning time needed for the approach in [4] is 7.348 s.) The learned weights are shown in Fig. 24, where the layer-2 weights are crispy, and the layer-4 weights are fuzzy.

Example 5 - Crisp Input and Crisp Output: In this example, we train an FCNO with numerical training data pairs.

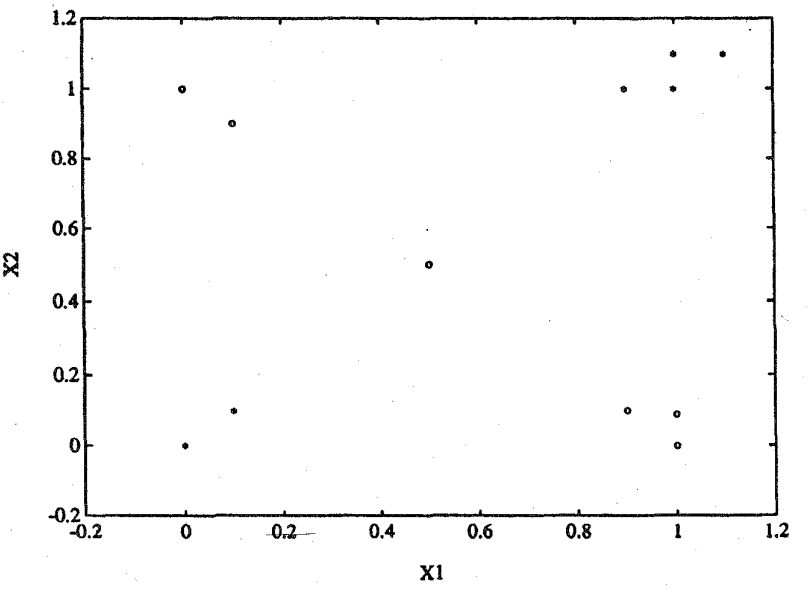


Fig. 26. The training data pairs in Example 5.

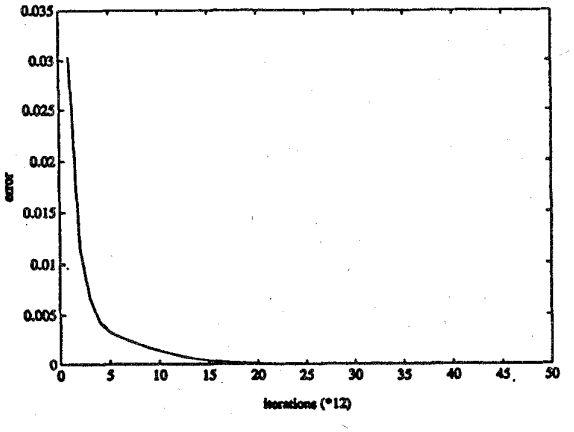


Fig. 27. The learning curve in Example 5.

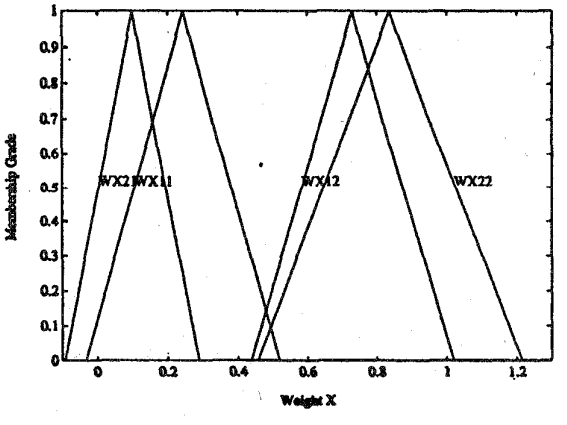


Fig. 28. The learned input fuzzy weights in Example 5.

With Method 1 of initialization process, the structure of the FCNO is shown in Fig. 25. The twelve training data pairs are shown in Fig. 26, where the circle has desired output of 0 and the star has desired output of 1. Eleven level sets are used in this example. The learning curve is shown in Fig. 27. The total learning time is 5.841 s. (The learning time needed for the approach in [4] is 16.2 s.) The learned weights are shown in Figs. 28 and 29. It is observed that the final learned weights could be fuzzy numbers even though the training data are all numerical.

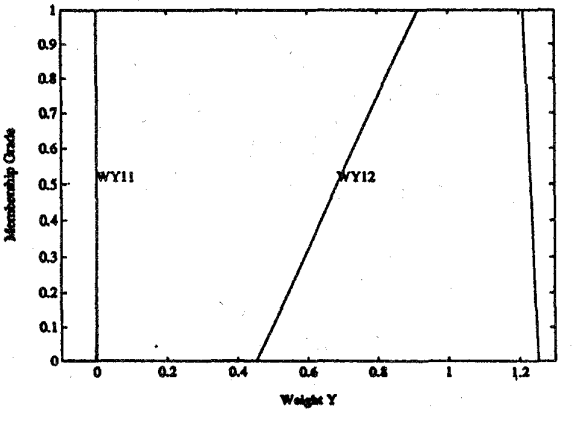


Fig. 29. The learned output fuzzy weights in Example 5.

VI. Conclusion

In this paper, we proposed a neural fuzzy system that can process both numerical and linguistic information. The proposed system has some characteristics and advantages:

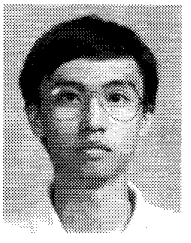
1) The inputs and outputs can be fuzzy numbers or numerical numbers.
2) The weights of the proposed neural fuzzy system are fuzzy weights.

3) Owing to the representation forms of the α-level sets, the fuzzy weights, fuzzy inputs, and fuzzy outputs can be fuzzy numbers of any shape.
4) Except the input and output layers, numerical numbers are propagated through the whole neural fuzzy system; thus the operations in the proposed neural fuzzy system are not time-consuming and the required memory capacity is small.

The proposed system has fuzzy supervised learning capability. With fuzzy supervised learning, the proposed system can be used for a fuzzy expert system, fuzzy system modeling, or rule base concentration. When learning with numerical values (real vectors), the proposed system can be used for an adaptive fuzzy controller. Computer simulations satisfactorily verified the performance of the proposed neural fuzzy system. We are currently extending the fuzzy supervised learning scheme proposed in this paper to a fuzzy reinforcement learning scheme with the expectation that a neural fuzzy system can learn teacher's oral (linguistic) instructions.

REFERENCES

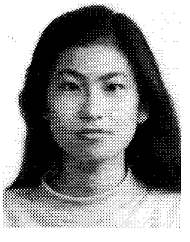
[1] S. K. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets and classification," *IEEE Trans. Neural Networks*, vol. 3, no. 5, pp. 683–696, 1992.
[2] J. M. Keller and H. Tahani, "Backpropagation neural networks for fuzzy logic," *Inf. Sci.*, vol. 62, pp. 205–221, 1992.
[3] S. Horikawa, T. Furuhashi, and Y. Uchikawa, "On fuzzy modeling using fuzzy neural networks with the backpropagation algorithm," *IEEE Trans. Neural Networks*, vol. 3, no. 5, pp. 801–806, 1992.
[4] H. Ishibuchi, R. Fujioka, and H. Tanaka, "Neural networks that learn from fuzzy if-then rules," *IEEE Trans. Fuzzy Syst.*, vol. 1, no. 2, pp. 85–97, 1993.
[5] H. Ishibuchi and H. Tanaka, "Fuzzy regression analysis using neural networks," *Fuzzy Sets and Systems*, vol. 50, pp. 257–265, 1992.
[6] H. Ishibuchi, H. Tanaka, and H. Okada, "Fuzzy neural networks with fuzzy weights and fuzzy biases," in *Proc. Int. Joint Conf. Neural Networks*, San Francisco, CA, 1993, pp. 1650–1655.
[7] Y. Hayashi, J. J. Buckley, and E. Czogula, "Fuzzy neural network," *International Journal of Intelligent Systems*, vol. 8, pp. 527–537, 1993.
[8] ——, "Systems engineering application of fuzzy neural networks," in *Proc. Int. Joint Conf. Neural Networks*, Baltimore, MD, 1992, pp. 413–418.
[9] A. Kaufmann and M. M. Gupta, *Introduction to Fuzzy Arithmetic*. New York: Van Nostrand, 1985.
[10] K. Uehara and M. Fujise, "Fuzzy inference based on families of α-level sets," *IEEE Trans. Fuzzy Syst.*, vol. 1, no. 2, pp. 111–124, 1993.
[11] L. A. Zadeh, "The concept of a linguistic truth variable and its application to approximate reasoning—I, II," *Inf.. Sci.*, vol. 8, pp. 199–249, 301–357, 1975.
[12] M. Braae and D. A. Rutherford, "Fuzzy relations in a control setting," *Kyberbetes*, vol. 7, no. 3, pp. 185–188, 1978.
[13] C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller—Part I & II," *IEEE Trans. Syst. Man, Cybern.*, vol. 20, no. 2, pp. 404–435, 1990.
[14] T. Yamakawa and T. Miki, "The current mode fuzzy logic integrated circuits fabricated by the standard CMOS process," *IEEE Trans. Comput.*, vol. C-35, no. 2, pp. 161–167, 1986.
[15] K. Uehara, "Computational efficiency of fuzzy inference based on level sets," in *Proc. 1989 Spring Nat. Conv. Rec., IEICE*, Japan, p. D-400.
[16] ——, "Fast operation of fuzzy inference based on level sets," in *Proc. 38th Ann. Conv. Rec. IPS Japan Rec.*, 1989, p. 3G-3.
[17] C. T. Lin and C. S. G. Lee, "Neural-network-based fuzzy logic control and decision system," *IEEE Trans. Comput.*, vol. 40, no. 12, pp. 1320–1336, 1991.
[18] ——, "Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems," *IEEE Trans. Fuzzy Syst.*, vol. 2, no. 1, pp. 46–63, 1995.
[19] J.-S. Jang, "Self-learning fuzzy controllers based on temporal back propagation," *IEEE Trans. Neural Networks*, vol. 3, no. 5, pp. 714–723, 1992.
[20] T. Tsukamoto, "An approach to fuzzy reasoning method," in *Advances in Fuzzy Set Theory and Applications*, M. M. Gupta, R. K. Regade, and R. R. Yager, Eds. North-Holland, Amsterdam, 1979.
[21] J. M. Zurada, *Introduction to Artificial Neural Systems*. New York: West, 1992.
[22] G. E. Hinton, "Connectionist learning procedure," *Artificial Intelligence*, vol. 40, no. 1, pp. 143–150, 1989.
[23] R. Keller, *Expert System Technology—Development and Application*. Englewood Cliffs, NJ: Prentice-Hall, 1987.



Chin-Teng Lin (S'88–M'91) received the B.S. degree in control engineering from the National Chiao-Tung University, Taiwan, R.O.C., in 1986 and the M.S.E.E. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, in 1989 and 1992, respectively.

Since August 1992, he has been with the College of Electrical Engineering and Computer Science, the National Chiao-Tung University, Hsinchu, Taiwan, where he is currently an Associate Professor of Control Engineering. His current research interests are fuzzy systems, neural networks, intelligent control, human-machine interface, and video and audio processing. He is the author of *Neural Fuzzy Control Systems with Structure and Parameter Learning* (World Scientific), and coauthor of *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems* (Englewood Cliffs, NJ: Prentice-Hall).

Dr. Lin is a member of Tau Beta Pi and Eta Kappa Nu.



Ya-Ching Lu was born in Tainan, Taiwan, R.O.C., in 1970. She received the B.S. and M.S. degrees in computer and information science from the National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 1992 and 1994, respectively.

Since July 1994, she has been with the Academia Sinica, Taiwan, where she is currently an Assistant Researcher in the Institute of Information Science. Her current research interests include fuzzy systems, neural networks, and pattern recognition.