A Neural Fuzzy System with Linguistic Teaching Signals

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

***Abstract*—A neural fuzzy system learning with linguistic teaching signals is proposed in this paper. This system is able to process and learn numerical information as well as linguistic information. It can be used either as an adaptive fuzzy expert system or as an adaptive fuzzy controller. 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, two kinds of learning schemes are developed for the proposed system: fuzzy supervised learning and fuzzy reinforcement learning. They extend the normal supervised and reinforcement 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. In the fuzzy reinforcement learning problem that we consider, the reinforcement signal from the environment is linguistic information (fuzzy critic signal) such as "good," "very good," or "bad," instead of the normal numerical critic values such as "0" (success) or "-1" (failure). With the fuzzy critic signal from the environment, the proposed system can learn proper fuzzy control rules and membership functions. We discuss two kinds of fuzzy reinforcement learning problems: single-step prediction problems and multistep prediction problems. 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. Some 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.

In general, the learning methods can be distinguished into three classes: supervised learning, reinforcement learning, and unsupervised learning. In supervised learning, a teacher provides the desired objective at each time step to the learning system. In reinforcement learning, the teacher's response is not as direct, immediate, and informative as that in supervised learning and serves more to evaluate the state of system. The presence of a teacher or a supervisor to provide the correct response is not assumed in unsupervised learning, which is called "learning by observation." Unsupervised learning does not require any feedback, but the disadvantage is that the learner cannot receive any external guidance and thus is inefficient, especially for the applications in control and decisionmaking. Hence, in this paper we are interested in the supervised and reinforcement learning capabilities of the neural fuzzy system.

Most of the supervised and reinforcement learning methods of neural networks, for example the perception [1], the BP (backpropagation) algorithm [2] and [3], and the A_{R-P} algorithm [4], process only numerical data. For supervised learning problems, some approaches have been proposed to process linguistic information with fuzzy inputs, fuzzy outputs, or fuzzy weights. Ishibuchi and his colleagues have proposed a series of approaches and applications with the capacity of processing linguistic input or/and linguistic output [5]–[7]. In their methods, the weights, inputs, and outputs of the neural network are fuzzified using fuzzy numbers represented by α-level sets. They derived learning algorithms from a cost function defined by the α-level sets of actual fuzzy outputs and target fuzzy outputs. Hayashi *et al.* [8] presented a similar method with fuzzy signals and fuzzy weights by using triangular fuzzy numbers. A learning algorithm was derived from a nonfuzzy cost function. Hayashi *et al.* [9] also proposed a similar architecture of neural network with fuzzy weights and fuzzy signals, but the learning algorithm was complete different from the proposed methods of Ishibuchi. In their method, the BP learning algorithm is directly fuzzified based

Manuscript received July 8, 1994; revised December 9, 1994. This work was supported by the National Science Council, Republic of China, Contract NSC 82-0408-E-009-429.

The authors are with the Department of Control Engineering, National Chiao-Tung University, Hsinchu, Taiwan, Republic of China.

IEEE Log Number 9410129.

on a fuzzy-valued cost function; i.e., the rule for changing fuzzy weights is defined by fuzzy numbers.

The common points of the above approaches are summarized as follows: 1) The α-level sets of fuzzy numbers are used to represent linguistic inputs, linguistic outputs, fuzzy weights, or fuzzy biases. 2) The operations in neural networks 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 fuzzy numbers are propagated through the whole neural network, 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 quantized membership grade. 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.

For reinforcement learning problems, almost all existing learning methods of neural networks focus their attention on numerical evaluative information [4], [10]–[22]. Inspired by Klopf's [22] work and earlier simulation results [19], Barto and his colleagues used neuron-like adaptive elements to solve difficult learning control problems with only scalar reinforcement signal feedback [14]. They also proposed the associative reward-penalty (A_{R-P}) algorithm for adaptive elements called A_{R-P} elements [12]. Several generalizations of A_{R-P} algorithm have been proposed [20]. Williams formulated the reinforcement learning problem as a gradient-following procedure [18], and he identified a class of algorithms, called REINFORCE algorithms, that possess the gradient ascent property; however, these algorithms still do not include the full A_{R-P} algorithms. Recently, Berenji and Khedkar [15] proposed a fuzzy logic controller and its associated learning algorithm. Their architecture extends Anderson's method [16] by including *a priori* control knowledge of expert operators in terms of fuzzy control rules. Lin and Lee [10] also proposed a connectionist architecture, called RNN-FLCS, for solving various reinforcement learning problems. The RNN-FLCS can find proper network structure and parameters simultaneously and dynamically. All the above reinforcement learning schemes assume scalar critic feedback (scalar reinforcement signal) from the environment. In this paper, we shall attack the fuzzy reinforcement learning problem where only fuzzy critic signal (e.g., "good," "very good," "bad.") is available. This problem is much closer to the expert-instructing learning system in real world than the original one with scalar critic signal.

The objective of this paper is to explore the approaches to supervised learning and reinforcement 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, two kinds of learning schemes are developed for the proposed system. They are fuzzy supervised learning and fuzzy reinforcement learning. They generalize the normal supervised and reinforcement 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.

In the fuzzy reinforcement learning problem that we consider, the reinforcement signal from the environment is linguistic information (a fuzzy critic signal) such as "good," "very good," or "bad," instead of the normal numerical critic values such as "0" (success) or "-1" (failure). There are two major problems embedded in the reinforcement learning problems [10]: 1) there is no instructive feedback from the environment to tell the network how to adapt itself, and 2) (fuzzy) reinforcement signal may only be available at a time long after a sequence of actions has occurred (the credit assignment problem). To solve reinforcement learning problems in neural fuzzy systems, we integrate two of the proposed five-layered networks into a function unit. One network (action network) acts as a fuzzy controller that performs fuzzy stochastic exploration to find out its output errors. The other network (evaluation network) acts as a fuzzy predictor that uses the fuzzy temporal difference technique to predict the output errors for either single or multistep prediction. It also provides a more informative and in-time internal fuzzy reinforcement signal to the action network for its learning. After finding the output errors, the developed supervised learning scheme can be applied directly to train both the action and evaluation networks. Hence, with fuzzy critic signal from the environment, the proposed fuzzy reinforcement learning system can learn proper fuzzy control rules and membership functions.

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, a fuzzy reinforcement learning scheme is developed. We consider the learning methods in two situations: single-step prediction problems and multistep prediction problems. In Section VI, two applications are simulated to illustrate the practical effect of the proposed neural fuzzy system. One is rule base concentration for knowledge-based evaluator (KBE) with fuzzy supervised learning. The other is the cart-pole balancing problem with fuzzy reinforcement learning. Finally, conclusions are summarized in Section VII.

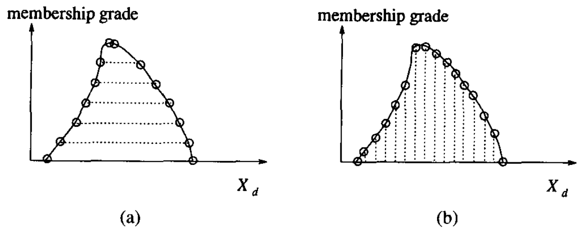


Fig. 1. Representations of fuzzy number. (a) α-level sets of fuzzy number. (b) discretized (pointwise) membership function.

II. Representation of Linguistic Information

When constructing information processing systems such as classifiers 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 (see Fig. 1(b)). Fuzzy sets, however, can be defined by the families of their α-level sets according to the resolution identity theorem (see Fig. 1(a)). In this paper, we use α-level sets of fuzzy numbers to represent linguistic information because of their advantages in both theoretical and practical considerations [23]–[25]. From theoretical point of view, they effectively study the effects of the fuzziness and the position of a fuzzy number in a universe of discourse. From practical point of view, they provide fast inference computations by using hardware construction in parallel and require less memory capacity for fuzzy numbers defined in universes of discourse with a large number of elements; they easily interface with two-valued logic; and they allow good matching with systems that include fuzzy number operations based on the extension principle.

Let us first review some notations and basic definitions of fuzzy sets. 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 μ_P: $X \to [0, 1]$. When X is continuum rather than a countable or finite set, the fuzzy set P is represented as $P = \int_X \mu_P(x)/x$, where $x \in X$. When X is a countable or finite set, $P = \Sigma_i \mu_P(x_i)/x_i$, where $x_i \in X$. We call the latter form as a discretized or pointwise membership function. A fuzzy set, P, is normal when its membership function, $\mu_P(x)$, satisfies the condition $\max_x \mu_P(x) = 1$. A fuzzy set P is convex if and only if $\mu_P(\lambda x_1 + (1-\lambda)x_2) \geq \min[\mu_P(x_1), \mu_P(x_2)]$, 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

$$P_\alpha = \{x | \mu_P(x) \geq \alpha\} \tag{1}$$

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 $\mathcal{R}$. It can be represented by

$$P_\alpha = [P_1^{(\alpha)}, P_2^{(\alpha)}] \tag{2}$$

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 set, P_α, according to the resolution identity theorem [26] as follows

$$\begin{aligned} P &= \bigcup_\alpha \alpha P_\alpha = \bigcup_\alpha \alpha[p_1^{(\alpha)}, p_2^{(\alpha)}] \\ &= \int_x \sup_\alpha \alpha \mu_{P_\alpha}(x)/x. \end{aligned} \tag{3}$$

We shall next 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 sections. More detailed operations of fuzzy numbers can be found in [27].

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

$$\begin{aligned} A_\alpha(+)B_\alpha &= [a_1^{(\alpha)}, a_2^{(\alpha)}](+)[b_1^{(\alpha)}, b_2^{(\alpha)}] \\ &= [a_1^{(\alpha)} + b_1^{(\alpha)}, a_2^{(\alpha)} + b_2^{(\alpha)}]. \end{aligned} \tag{4}$$

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

$$\begin{aligned} A_\alpha(-)B_\alpha &= [a_1^{(\alpha)}, a_2^{(\alpha)}](-)[b_1^{(\alpha)}, b_2^{(\alpha)}] \\ &= [a_1^{(\alpha)} - b_2^{(\alpha)}, a_2^{(\alpha)} - b_1^{(\alpha)}]. \end{aligned} \tag{5}$$

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

$$k \cdot A_\alpha = \begin{cases} [ka_1^{(\alpha)}, ka_2^{(\alpha)}], & \text{if } k \geq 0, \\ [ka_2^{(\alpha)}, ka_1^{(\alpha)}], & \text{if } k < 0. \end{cases} \tag{6}$$

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

$$\begin{aligned} A_\alpha(\cdot)B_\alpha &= [a_1^{(\alpha)}, a_2^{(\alpha)}](\cdot)[b_1^{(\alpha)}, b_2^{(\alpha)}] \\ &= [a_1^{(\alpha)} \cdot b_1^{(\alpha)}, a_2^{(\alpha)} \cdot b_2^{(\alpha)}]. \end{aligned} \tag{7}$$

The reader is referred to [27] for the general case that A and B are fuzzy numbers in $\mathcal{R}$.

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

$$\text{diff}(A, B) = \frac{1}{2} \sum_\alpha [(a_1^{(\alpha)} - b_1^{(\alpha)})^2 + (a_2^{(\alpha)} - b_2^{(\alpha)})^2]. \tag{8}$$

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 of mapping a fuzzy number into a nonfuzzy number is called "defuzzification." Various defuzzification strategies have been suggested in [28], [29]. In this section, 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 the membership function by

$$z_0 = \frac{\sum_{j=1}^{l} z_j}{l} \tag{9}$$

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 (9) as

$$\text{defuzzifier}(Z) = z_0 = \frac{(z_1^{(1)} + z_2^{(1)})}{2} \tag{10}$$

where defuzzifier represents a defuzzification operation.

- Center of Area Method (COA)

Assuming that a fuzzy number with a pointwise membership function 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

$$z_0 = \frac{\sum_{j=1}^{n} x_j \mu_Z(x_j)}{\sum_{j=1}^{n} \mu_Z(x_j)}. \tag{11}$$

For a fuzzy number Z in the form of α-level sets, it can be expressed according to (11) as

$$\text{defuzzifier}(Z) = z_0 = \frac{\sum_{\alpha} \alpha(z_1^{(\alpha)} + z_2^{(\alpha)})}{2\sum_{\alpha} \alpha}. \tag{12}$$

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. 2 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 [10], [30], and [31]. Nodes at layer one are input nodes whose imports are fuzzy numbers or crisp numbers. Each input node corresponds to one input linguistic variable. Layer five consists of output nodes whose export 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 four have fuzzy weights. 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

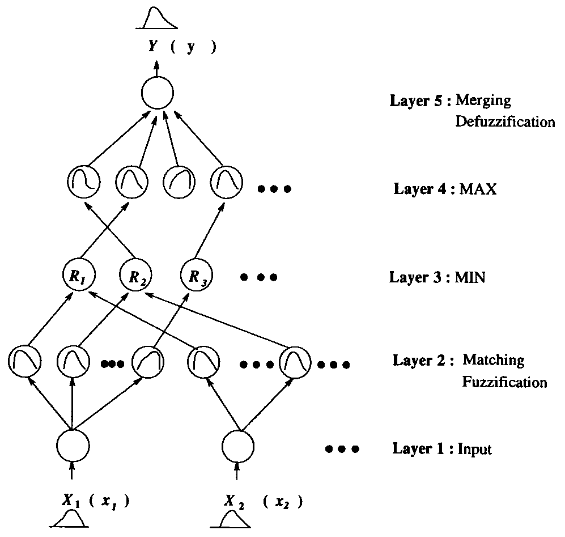


Fig. 2. The five-layered architecture of the proposed neural fuzzy system.

represents one fuzzy rule. Hence, all the layer-3 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 executes a "defuzzification" operation to obtain a crisp decision.

We shall next describe the signal propagation in the proposed network layer by layer following the arrow directions shown in Fig. 2. This is done by defining the transfer function of a node in each layer. Signal may flow in the reserve 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. 3). The notations u and U represent the input crisp and fuzzy numbers of a node, respectively. The notations o and O represent, respectively, the output crisp and fuzzy numbers of a node. 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

$$O_i^1 = \bigcup_{\alpha} \alpha[o_{i1}^{1(\alpha)}, o_{i2}^{1(\alpha)}] = X_i = \bigcup_{\alpha} \alpha[x_{i1}^{(\alpha)}, x_{i2}^{(\alpha)}]. \tag{13}$$

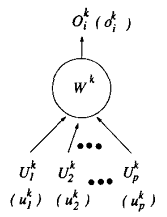


Fig. 3. Basic structure of a node in the proposed neural fuzzy system.

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

$$O_i^1 = \bigcup_\alpha \alpha[o_{i1}^{1(\alpha)}, o_{i2}^{1(\alpha)}] = \bigcup_\alpha \alpha[x_i, x_i]. \tag{14}$$

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

Layer 2 (Matching): 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,

$$\begin{aligned} f_{ij}^2 &= \text{diff}(WX_{ij}, U_i) = \tfrac{1}{2}\sum_\alpha [(wx_{ij1}^{(\alpha)} - u_{i1}^{2(\alpha)})^2 \\ &\quad + (wx_{ij2}^{(\alpha)} - u_{i2}^{2(\alpha)})^2], \qquad (15) \\ o_{ij}^2 &= a(f_{ij}^2) = e^{-(f_{ij}^2)^2/2\sigma^2} \qquad (16) \end{aligned}$$

where σ 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

$$o_{ij}^2 = a(f_{ij}^2) = r^{f_{ij}^2} \tag{17}$$

where $0 < r < 1$, or

$$o_{ij}^2 = a(f_{ij}^2) = \frac{2}{1 + e^{-\lambda f_{ij}^2}} \tag{18}$$

where λ is a nonnegative constant. The output of a layer-2 node indicates the matching degree of input and fuzzy weight. It is noted that the matching process in this layer is different from that in the conventional fuzzy control systems [29]. In the conventional fuzzy control systems, we usually consider numerical input data, and thus the matching process is simply the calculation of a membership function value. If we view a numerical value as a fuzzy singleton, then our formula in the above will achieve the same result as the conventional approach dose.

Layer 3 (MIN): The input and output of a 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 [29]. If intersection is used, we have

$$o_i^3 = \min(u_1^3, u_2^3, \cdots, u_k^3). \tag{19}$$

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

$$o_i^3 = u_1^3 u_2^3 \cdots u_k^3. \tag{20}$$

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 [29]. If the union operation is used, we have

$$o_i^4 = \max(u_1^4, u_2^4, \cdots, u_k^4). \tag{21}$$

If the bounded sum is used, we have

$$o_i^4 = \min(1, u_1^4 + u_2^4 + \cdots + u_k^4). \tag{22}$$

The input and output 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

$$O^5 = \bigcup_\alpha \alpha[o_1^{5(\alpha)}, o_2^{5(\alpha)}] = Y = \frac{\sum_i u_i^5 \cdot WY_i}{\sum_i u_i^5}. \tag{23}$$

Namely

$$Y = \bigcup_\alpha \alpha[y_1^{(\alpha)}, y_2^{(\alpha)}], \quad WY_i = \bigcup_\alpha \alpha[wy_{i1}^{(\alpha)}, wy_{i2}^{(\alpha)}] \tag{24}$$

where

$$y_1^{(\alpha)} = \frac{\sum_i u_i^5 wy_{i1}^{(\alpha)}}{\sum_i u_i^5}, \tag{25}$$

$$y_2^{(\alpha)} = \frac{\sum_i u_i^5 wy_{i2}^{(\alpha)}}{\sum_i u_i^5}. \tag{26}$$

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 Tsukumoto's defuzzification method [32]

$$f_i = \text{defuzzifier}(WY_i) = \frac{\sum_\alpha \alpha(wy_{i1}^{(\alpha)} + wy_{i2}^{(\alpha)})}{2\sum_\alpha \alpha}, \tag{27}$$

$$y = o^5 = \frac{\sum_i u_i^5 \sigma_i f_i}{\sum_i u_i^5 \sigma_i} \tag{28}$$

where

$$\sigma_i = \sum_\alpha (wy_{i2}^{(\alpha)} - wy_{i1}^{(\alpha)}). \tag{29}$$

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 them.

IV. Supervised Learning Algorithm

In this section, we shall derive supervised learning algorithms for the proposed neural fuzzy system (FCLO and FCNO). These algorithms are 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 training are

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

where $p = 1, 2, \cdots, m$, and m is the total number of training rules. These fuzzy if-then rules can be viewed as the fuzzy input-output pairs, $(X_{p1}, X_{p2}, \cdots, X_{pn}; Y_p)$, where $p = 1, 2, \cdots, m$. 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. We thus 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 $\Pi_i|T(x_i)|$ 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. 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 four are assigned randomly. The connections from layer-3 nodes to layer-4 nodes is one-to-one initially. That is, each layer-3 node is connected 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 presented in [33] and [34], is employed to adjust fuzzy weights in layer two and layer four of the proposed network. If the FCLO is used, the error function to be minimized is

$$e = \operatorname{diff}(Y, D) = \frac{1}{2}\sum_{\alpha}[(y_1^{(\alpha)} - d_1^{(\alpha)})^2 + (y_2^{(\alpha)} - d_2^{(\alpha)})^2] \quad (30)$$

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

$$e = \frac{1}{2}(d - y)^2 \quad (31)$$

where y is the current output and d is the desired output. We assume that $W = \cup_\alpha \alpha[w_1^{(\alpha)}, w_2^{(\alpha)}]$ is the adjustable fuzzy parameter in layer two and layer four. 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

$$w(t+1) = w(t) + \eta\left(-\frac{\partial e}{\partial w}\right) \tag{32}$$

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 (25) and (26) as follows

$$\frac{\partial e}{\partial wy_{i1}^{(\alpha)}} = \frac{\partial e}{\partial y_1^{(\alpha)}}\frac{\partial y_1^{(\alpha)}}{\partial wy_{i1}^{(\alpha)}} = (y_1^{(\alpha)} - d_1^{(\alpha)})\frac{u_i^5}{\sum_i u_i^5}, \tag{33}$$

$$\frac{\partial e}{\partial wy_{i2}^{(\alpha)}} = \frac{\partial e}{\partial y_2^{(\alpha)}}\frac{\partial y_2^{(\alpha)}}{\partial wy_{i2}^{(\alpha)}} = (y_2^{(\alpha)} - d_2^{(\alpha)})\frac{u_i^5}{\sum_i u_i^5}. \tag{34}$$

The error signals to be propagated to the preceding layer are

$$\delta_1^{5(\alpha)} = \frac{\partial e}{\partial o_1^{5(\alpha)}} = \frac{\partial e_1^{(\alpha)}}{\partial y_1^{(\alpha)}} = (y_1^{(\alpha)} - d_1^{(\alpha)}), \tag{35}$$

$$\delta_2^{5(\alpha)} = \frac{\partial e}{\partial o_2^{5(\alpha)}} = \frac{\partial e_2^{(\alpha)}}{\partial y_2^{(\alpha)}} = (y_2^{(\alpha)} - d_2^{(\alpha)}) \tag{36}$$

where

$$e_1^{(\alpha)} = \tfrac{1}{2}(y_1^{(\alpha)} - d_1^{(\alpha)})^2, \tag{37}$$

$$e_2^{(\alpha)} = \tfrac{1}{2}(y_2^{(\alpha)} - d_2^{(\alpha)})^2. \tag{38}$$

If an FCNO is used and the desired output is numerical value y, the update rules of the parameters are derived form (27) and (28) as follows

$$\frac{\partial e}{\partial wy_{i1}^{(\alpha)}} = \frac{\partial e}{\partial y}\frac{\partial y}{\partial wy_{i1}^{(\alpha)}} = (y-d)\frac{\partial y}{\partial wy_{i1}^{(\alpha)}} \tag{39}$$

where

$$\begin{aligned}\frac{\partial y}{\partial wy_{i1}^{(\alpha)}} &= \frac{\partial y}{\partial f_i}\frac{\partial f_i}{\partial wy_{i1}^{(\alpha)}} + \frac{\partial y}{\partial \sigma_i}\frac{\partial \sigma_i}{\partial wy_{i1}^{(\alpha)}} \\ &= \frac{u_i^5\sigma_i}{\sum_i u_i^5\sigma_i}\frac{\alpha}{2\sum_\alpha \alpha} \\ &\quad - \frac{\left(\sum_i u_i^5\sigma_i\right)u_i^5 f_i - \left(\sum_i u_i^5\sigma_i f_i\right)u_i^5}{\left(\sum_i u_i^5\sigma_i\right)^2}\end{aligned} \tag{40}$$

and

$$\frac{\partial e}{\partial wy_{i2}^{(\alpha)}} = \frac{\partial e}{\partial y}\frac{\partial y}{\partial wy_{i2}^{(\alpha)}} = (y-d)\frac{\partial y}{\partial wy_{i2}^{(\alpha)}} \tag{41}$$

where

$$\begin{aligned}\frac{\partial y}{\partial wy_{i2}^{(\alpha)}} &= \frac{\partial y}{\partial f_i}\frac{\partial f_i}{\partial wy_{i2}^{(\alpha)}} + \frac{\partial y}{\partial \sigma_i}\frac{\partial \sigma_i}{\partial wy_{i2}^{(\alpha)}} \\ &= \frac{u_i^5\sigma_i}{\sum_i u_i^5\sigma_i}\frac{\alpha}{2\sum_\alpha \alpha} \\ &\quad + \frac{\left(\sum_i u_i^5\sigma_i\right)u_i^5 f_i - \left(\sum_i u_i^5\sigma_i f_i\right)u_i^5}{\left(\sum_i u_i^5\sigma_i\right)^2}.\end{aligned} \tag{42}$$

The error signal to be propagated to the preceding layer is

$$\delta^5 = \frac{\partial e}{\partial y} = (y-d). \tag{43}$$

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 form (21) as follows

$$\delta_i^4 = \frac{\partial e}{\partial o_i^4} = \frac{\partial \sum_\alpha (e_1^{(\alpha)} + e_2^{(\alpha)})}{\partial o_i^4} = \sum_\alpha (\delta_{i1}^{4(\alpha)} + \delta_{i2}^{4(\alpha)}) \tag{44}$$

where

$$\begin{aligned}\delta_{i1}^{4(\alpha)} &= \frac{\partial e_1^{(\alpha)}}{\partial o_i^4} = \frac{\partial e_1^{(\alpha)}}{\partial o_1^{5(\alpha)}}\frac{\partial o_1^{5(\alpha)}}{\partial o_i^4} \\ &= \delta_1^{5(\alpha)}\frac{\partial y_1^{(\alpha)}}{\partial o_i^4} = \delta_1^{5(\alpha)}\frac{wy_{i1}^{(\alpha)}\left(\sum_i u_i^5\right) - \sum_i u_i^5 wy_{i1}^{(\alpha)}}{\left(\sum_i u_i^5\right)^2},\end{aligned} \tag{45}$$

$$\begin{aligned}\delta_{i2}^{4(\alpha)} &= \frac{\partial e_2^{(\alpha)}}{\partial o_i^4} = \frac{\partial e_2^{(\alpha)}}{\partial o_2^{5(\alpha)}}\frac{\partial o_2^{5(\alpha)}}{\partial o_i^4} \\ &= \delta_2^{5(\alpha)}\frac{\partial y_1^{(\alpha)}}{\partial o_i^4} = \delta_1^{5(\alpha)}\frac{wy_{i2}^{(\alpha)}\left(\sum_i u_i^5\right) - \sum_i u_i^5 wy_{i2}^{(\alpha)}}{\left(\sum_i u_i^5\right)^2}.\end{aligned} \tag{46}$$

If FCNO is used, the error signal δ_i^4 is derived from (21) as follows

$$\begin{aligned}\delta_i^4 &= \frac{\partial e}{\partial o_i^4} = \frac{\partial e}{\partial o^5}\frac{\partial o^5}{\partial o_i^4} \\ &= \delta^5\frac{\left(\sum_i u_i^5\sigma_i\right)f_i\sigma_i - \left(\sum_i u_i^5\sigma_i f_i\right)\sigma_i}{\left(\sum_i u_i^5\sigma_i\right)^2}.\end{aligned} \tag{47}$$

Layer 3: As in layer four, only the error signals need to be computed. According to (19), this error signal δ_i^3 can be derived as

$$\begin{aligned}\delta_i^3 &= \frac{\partial e}{\partial o_i^3} = \frac{\partial e}{\partial o_i^4}\frac{\partial o_i^4}{\partial o_i^3} \\ &= \begin{cases}\delta_i^4, & \text{if } o_i^4 = \max(u_1^4, \cdots, u_k^4), \\ 0, & \text{otherwise.}\end{cases}\end{aligned} \tag{48}$$

Layer 2: In this layer, there are fuzzy weights WX to be adjusted. The update rules can be derived from (15) and (16) as follows

$$\frac{\partial e}{\partial wx_{ij1}^{(\alpha)}} = \frac{\partial e}{\partial o_i^3}\frac{\partial o_i^3}{\partial o_{ij}^2}\frac{\partial o_{ij}^2}{\partial wx_{ij1}^{(\alpha)}} = \delta_i^3 \frac{\partial o_i^3}{\partial o_{ij}^2}\frac{\partial o_{ij}^2}{\partial wx_{ij1}^{(\alpha)}}, \tag{49}$$

$$\frac{\partial e}{\partial wx_{ij2}^{(\alpha)}} = \frac{\partial e}{\partial o_i^3}\frac{\partial o_i^3}{\partial o_{ij}^2}\frac{\partial o_{ij}^2}{\partial wx_{ij2}^{(\alpha)}} = \delta_i^3 \frac{\partial o_i^3}{\partial o_{ij}^2}\frac{\partial o_{ij}^2}{\partial wx_{ij2}^{(\alpha)}} \tag{50}$$

where

$$\frac{\partial o_i^3}{\partial o_{ij}^2} = \begin{cases} 1 & \text{if } o_{ij}^3 = \min(u_1^3, \cdots, u_k^3) \\ 0 & \text{otherwise,} \end{cases} \tag{51}$$

and

$$\begin{aligned} \frac{\partial o_{ij}^2}{\partial wx_{ij1}^{(\alpha)}} &= o_{ij}^2 \ln e\left(-\frac{2f_{ij}^2}{2\sigma^2}\right)\frac{\partial f_{ij}^2}{\partial wx_{ij1}^{(\alpha)}} \\ &= -o_{ij}^2 \frac{f_{ij}^2}{\sigma^2}(wx_{ij1}^{(\alpha)} - u_{i1}^{2(\alpha)}), \end{aligned} \tag{52}$$

$$\frac{\partial o_{ij}^2}{\partial wx_{ij2}^{(\alpha)}} = -o_{ij}^2 \frac{f_{ij}^2}{\sigma^2}(wx_{ij2}^{(\alpha)} - u_{i2}^{2(\alpha)}). \tag{53}$$

When fuzzy weights are adjusted by (33)–(53), 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. To cope with these undesirable situations, we perform 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 $W = \cup_\alpha \alpha[w_1^{(\alpha)}, w_2^{(\alpha)}]$ adjusted by (33)–(53).

Outputs: The modified fuzzy weights $\hat{W} = \cup_\alpha \alpha[\hat{w}_1^{(\alpha)}, \hat{w}_2^{(\alpha)}]$ 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 $\hat{w}_2^{(k)} = w_1^{(k)}$,
         else $\hat{w}_1^{(k)} = w_1^{(k)}$ and $\hat{w}_2^{(k)} = w_2^{(k)}$.
Step 2.  For $k = h - 1$ downto 0, do
         if $w_1^{(k/h)} > \hat{w}_1^{(k+1/h)}$, then $\hat{w}_1^{(k/h)} = \hat{w}_1^{(k+1/h)}$,
         else $\hat{w}_1^{(k/h)} = w_1^{(k/h)}$,
         and
         if $w_2^{(k/h)} < \hat{w}_2^{(k+1/h)}$, then $\hat{w}_2^{(k/h)} = \hat{w}_2^{(k+1/h)}$,
         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 in the following.

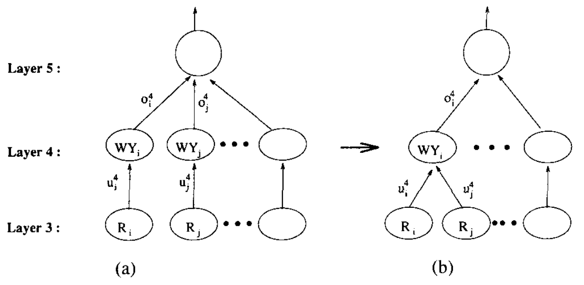


Fig. 4. Illustration of consequent combination.

A. Term-node combination scheme: Term-node combination is to combine similar fuzzy terms in the term sets of input and output linguistic variables. We shall present this technique on the term set of output linguistic variables. It is applied to the term set of input linguistic variables in exactly the same way. The whole learning procedure 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. If 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. 4).

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

The term-set combination scheme in Step 2 can automatically find the number of fuzzy partitions of output linguistic variables.

The operations in Step 2 can be equally applied to the term set of input linguistic variables.

B. 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 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 [30] and are given as follows.

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.

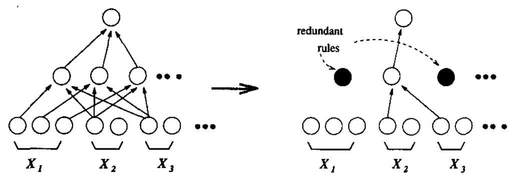


Fig. 5. Illustration of rule combination.

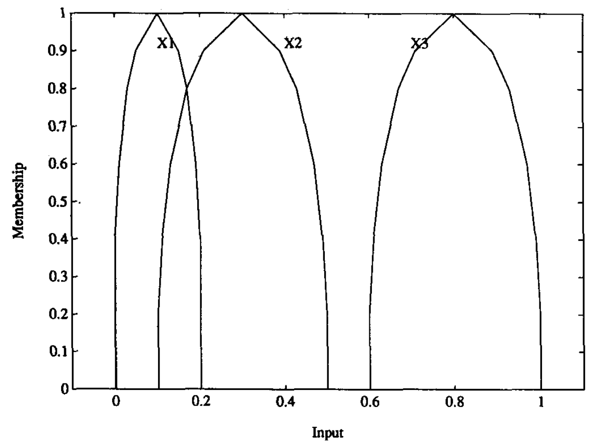


Fig. 6. The membership functions of the input linguistic value "very small" ($X1$), "small" ($X2$), "large" ($X3$) in Example 1.

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. Another example of rule combination is shown in Fig. 5.

The following simple examples illustrates the performance of the proposed supervised learning algorithm on the FCLO. One practical application of this technique is to do rule base concentration in fuzzy rule base. It can effectively find a small set of representative rules from a bunch of fuzzy if-then rules by removing redundancy and finding similarity. One practical example of rule base concentration will be given in Section VI.

Example 1: 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. 6. Because input and desired output are linguistic, an FCLO is used in this example. According to the initialization process, we set up a FCLO with two layer-2 nodes and two layer-4 nodes (and thus two layer-3 (rule) nodes). Fig. 7 shows the learning curves. The error tolerance is 0.0001 and the number of α-cuts is 6. After supervised learning, the fuzzy outputs of the learned FCLO and the corresponding desired outputs are shown in Fig. 8. The figure shows that they match closely. The two learned (representative) fuzzy rules after learning (condensing) are

$$\text{IF } x \text{ is WX1, THEN } y \text{ is WY1,} \quad \text{and}$$
$$\text{IF } x \text{ is WX2, THEN } y \text{ is WY2}$$

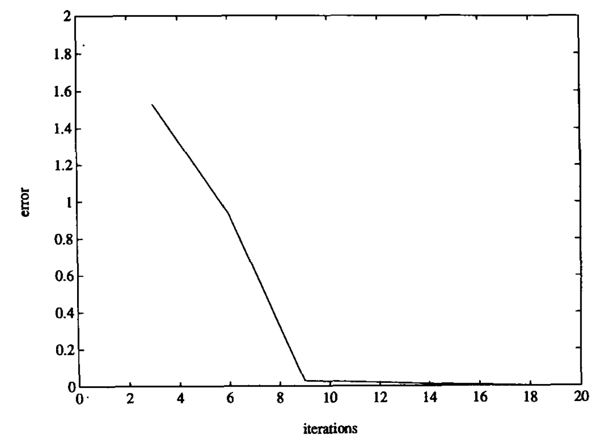


Fig. 7. The learning curve in Example 1.

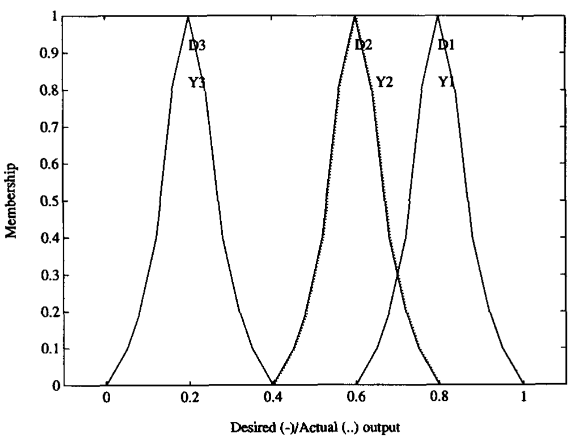


Fig. 8. The actual fuzzy outputs, $Y1, Y2, Y3$ of the learned neural fuzzy system and the corresponding desired fuzzy outputs, $D1, D2, D3$ in Example 1.

where the fuzzy weights after learning are shown in Fig. 9. For illustration, Figs. 10 and 11 show the change of fuzzy weights in the learning process. Hence the original three fuzzy rules have been condensed to two rules, and these two sets of fuzzy rules represent equivalent knowledge.

Example 2: In this example, we train an FCLO with five training fuzzy number pairs shown in Fig. 12. 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). Fig. 13 shows the learned fuzzy weights. To examine the generalization capability of the trained FCLO, we present three novel fuzzy numbers to its

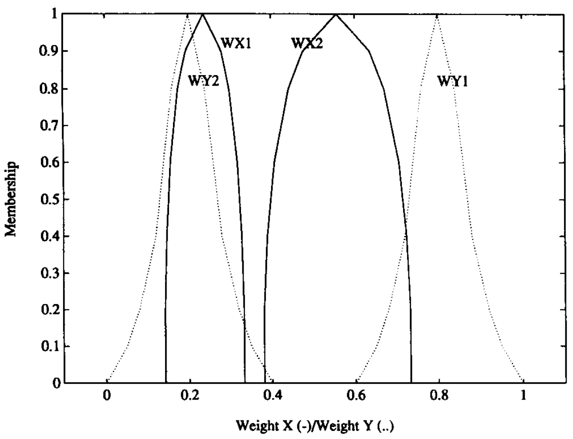


Fig. 9. The learned fuzzy weights of the FCLO in Example 1.

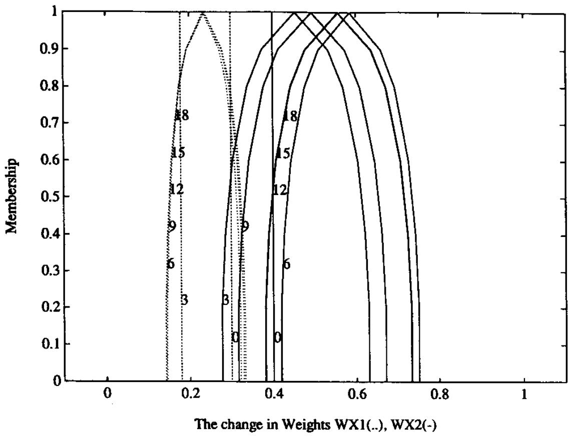


Fig. 10. Time evolving graph of fuzzy weights $WX1, WX2$ during the learning process in Example 1.

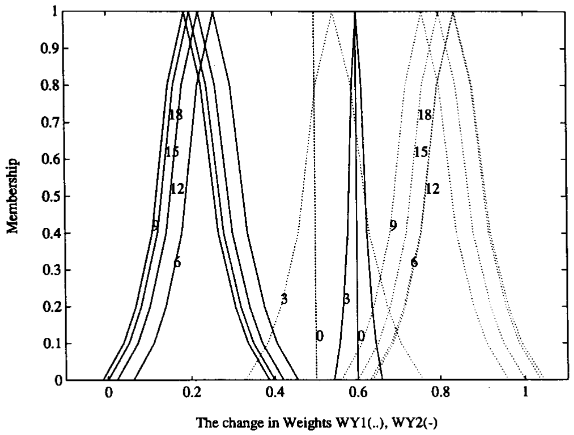


Fig. 11. Time evolving graph of fuzzy weights $WY1, WY2$ during the learning process in Example 1.

input nodes for testing. The results shown in Fig. 14 (the dashed rectangles) indicate the good generalization capability of the learned FCLO.

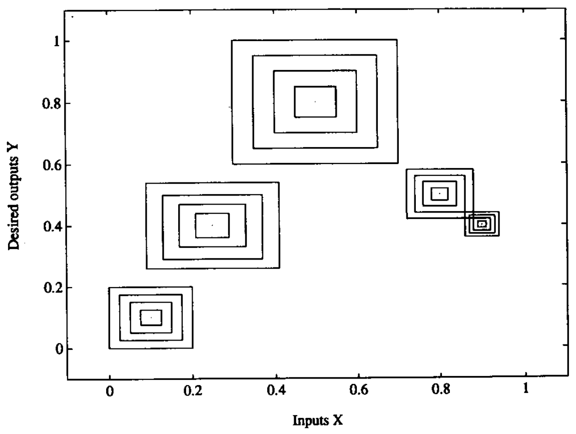


Fig. 12. The training fuzzy pairs (X, Y) in the form of α-level sets in Example 2.

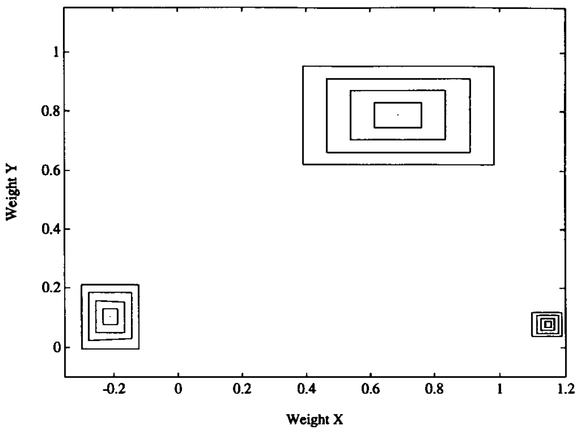


Fig. 13. The learned fuzzy weights in Example 2.

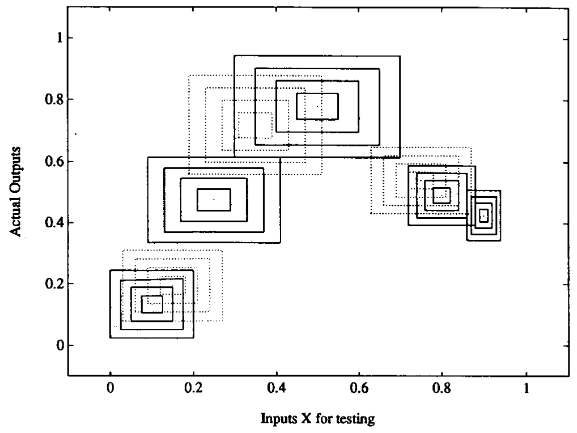


Fig. 14. Generalization test of the learned neural fuzzy system in Example 2.

V. Reinforcement Learning with Fuzzy Critic Signal

In the previous section, we considered the supervised learning of the proposed neural fuzzy system and assumed that

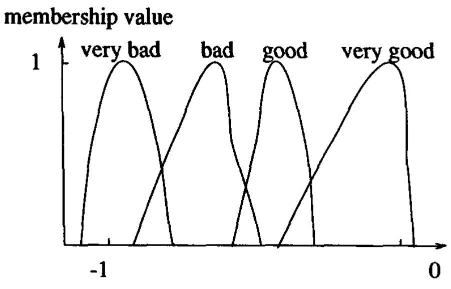


Fig. 15. An example of fuzzy reinforcement signals.

the precise "target" output for each input pattern was always available; however, in some real-world applications, precise training data are usually difficult and expensive to obtain. In this section, we extend the supervised learning of the proposed system to reinforcement learning. In the reinforcement learning problem, we get only evaluative feedback (called the reinforcement signal) from the environment. Because the reinforcement signal is only evaluative and not instructive, reinforcement learning is sometimes called "learning with a critic" as opposed to "learning with a teacher" in supervised learning.

Conventionally, the reinforcement signal is regarded as a real number. For example, the reinforcement signal, $r(t)$, can be one of the following forms: 1) a two-valued number, $r(t) \in \{-1, 0\}$, such that $r(t) = 0$ means "a reward" and $r(t) = -1$ means "a penalty"; 2) a multivalued discrete number in the range $[-1, 0]$, for example, $r(t) \in \{-1, -0.25, -0.5, -0.25, 0\}$ which corresponds to different degrees of reward or penalty; 3) a real number, $r(t) \in [-1, 0]$, which represents a more detailed and continuous degrees of reward or penalty.

The reinforcement signal given by the external environment (e.g., an expert), however, may be fuzzy feedback information such as "good," "very good," "bad," "too bad." This is true especially in the human-iterative learning environment, where human instructor is available. We call the reinforcement learning problem with fuzzy critic feedback as the fuzzy reinforcement learning problem. In this section, we shall attack this problem by considering the reinforcement signal $R(t)$ as a fuzzy number in the form of α-level sets. We also assume that $R(t)$ is the fuzzy signal available at time step t and caused by the input and action chosen at time step $t-1$ or even affected by earlier inputs and actions. Namely, the reinforcement signal is a fuzzy number such that

$$R(t) \in \{R_1, R_2, \cdots, R_n\} \tag{54}$$

where

$$\begin{aligned} -1 &\leq \text{defuzzifier}(R_1) \leq \text{defuzzifier}(R_2) \\ &\leq \cdots \leq \text{defuzzifier}(R_n) \leq 0 \end{aligned} \tag{55}$$

where defuzzifier$(R(t))$ represents discrete degree of reward or penalty. For example (see Fig. 15), we may have $R(t)$ $\in$(very bad, bad, good, very good).

In the reinforcement learning problems, it is common to think explicitly of a network functioning in an environment. The environment supplies the inputs to the network, receives its output, and then provides the reinforcement signal. There are several different reinforcement learning problems, depending on the nature of environment:

Class I: In the simplest case, the reinforcement signal is always the same for a given input-output pair. Thus there is a definite input-output mapping that the network must learn. Moreover, the reinforcement signals and input patterns do not depend on previous network outputs.

Class II: In a stochastic environment, a particular input-output pair determines only the probability of positive reinforcement. This probability is fixed for each input-output pair and again the input sequence does not depend on past history.

Class III: In the most general case, the environment may itself be governed by a complicated dynamical process. Both reinforcement signals and input patterns may depend on the past history of the network outputs.

If a reinforcement signal indicates that a particular output is wrong, it gives no hint as to what the right answer should be; in terms of a cost function, there is no gradient information. It is therefore important in a reinforcement learning network for there to be some source of randomness in this network, so that the space of possible outputs can be explored until a correct value is found. This is usually done by using stochastic units. Several approaches have been proposed for the above three different classes of reinforcement learning problems. Barto and Anandan [12] proposed the associative reward-penalty algorithm A_{R-P}, which is applicable to Class I and II problems. Its essential ingredient is the stochastic output unit. Another approach to reinforcement learning involves modeling the environment with an auxiliary network, which can then be used to produce a target for each output of the main network [17], [35], [36]. This scheme reduces the reinforcement learning problem to a two-stage supervised learning problem with known targets. This approach can be used to resolve reinforcement learning problems of Class I and II, and the general idea of this separate modeling network can be also applied to Class III problems. Another approach aiming at solving Class III reinforcement learning problems is "learning with predictor." In Class III problems, a reinforcement signal may only be available at a time long after a sequence of actions has occurred. To solve the long time-delay problem, prediction capabilities are necessary in a reinforcement learning system. In this scheme, a predictor (critic) receives the raw reinforcement signal r from the environment and feeds a processed signal $\hat{r}$ on to the main network. The $\hat{r}$ signal represents an evaluation of the current behavior of the main network, whereas r typically involves the past history. Recently, more and more researchers devote the reinforcement learning problems using this method [10], [12]–[15]. In this paper, we also use this scheme in our reinforcement learning model.

A. Architecture of Reinforcement Learning Model

The proposed reinforcement learning model, as shown in Fig. 16, integrates two previously proposed five-layered networks (FCLOs or FCNOs developed in Section III) into a

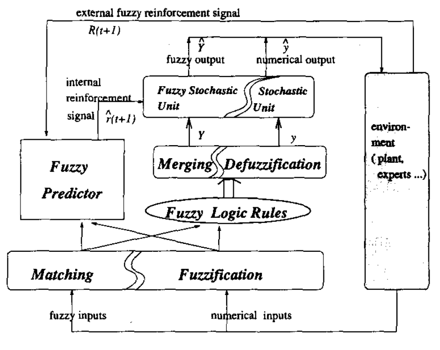


Fig. 16. The proposed fuzzy reinforcement learning system.

learning system. One (FCLO or FCNO) serving as the action network can choose a proper action or decision according to the current input. The action network acts as a fuzzy controller. The other (FCLO) serving as the evaluation network performs single or multistep prediction of the external fuzzy reinforcement signal. The evaluation network acts as a fuzzy predictor. The fuzzy predictor provides the action network with more informative and beforehand internal reinforcement signal for learning. Because the reinforcement signal is a fuzzy number, an FCLO is used for fuzzy predictor. The action network can be FCLO or FCNO depending on the actual requirement. Structurally, these two networks share the first two layers of the original FCNO or FCLO (see Fig. 16). This means that they partition the input space in the same way.

We distinguish two kinds of prediction-learning problems for the fuzzy predictor. In the single-step prediction problems, all information about the correctness of each prediction is revealed at once. In the multistep prediction problems, correctness is not revealed until more than one time step after the prediction is made, but partial information relevant to its correctness may be revealed at each time step. In the single-step prediction problems, data naturally comes in observation-output pairs; these problems are ideally suited to the pairwise supervised learning approach. We shall discuss these problems in Subsection V-B. For the multistep prediction problems, we use the temporal difference (TD) prediction technique, which is a class of incremental learning procedures introduced by Sutton [11]. The main characteristic of this technique is that they learn from successive predictions, whereas in supervised learning methods, learning occurs only when the difference between the predicted outcome and actual outcome is revealed. Hence the learning in TD does not have to wait until the actual outcome is known and can update its parameters at each time step. We shall explore these problems in our proposed model in Subsection V-C.

For the action network, the reinforcement learning algorithm allows its output nodes to perform stochastic exploration. With the internal fuzzy reinforcement signals from the fuzzy predictor, the output nodes of the action network can perform more effective stochastic searches with a higher probability of choosing a good action as well as discovering its output error accurately. The detailed learning procedure will be discussed in the following subsections.

In a word, the architecture of the proposed reinforcement learning model schematically shown in Fig. 16 has three components:

- The action network maps a state vector into a recommended actions, Y or y, using FCLO or FCNO.
- The evaluation network (predictor) maps a state vector and an external fuzzy reinforcement signal into a predicted fuzzy reinforcement signal which indicates state goodness. This is also used to produce internal reinforcement signal.
- The stochastic unit using both Y (or y) and the internal reinforcement signal to produce an action $\hat{Y}$ (or $\hat{y}$), which is applied to the environment.

Since the action network and the evaluation network are in fact the FCLO or FCNO introduced in Section III, their node operations in five layers are the same as those in the original structure. Let us now describe the operations in the stochastic unit in Fig. 16.

To estimate the gradient information of error function in a reinforcement learning network, there needs to be some source of randomness such that the space of possible output can be explored to find a correct value. Thus, the stochastic unit is necessary for the action network. In estimating the gradient information, the output $Y(y)$ of the action network does not directly act on the environment. Instead, the stochastic unit uses the predicted fuzzy reinforcement signal $P(t)$ of the evaluation network and the action $Y(y)$ recommended by the action network to stochastically generate an actual action $\hat{Y}(\hat{y})$ acting on the environment. The actual action $\hat{Y}(\hat{y})$ is a random variable with mean $\hat{Y}(\hat{y})$ and variance $\sigma(t)$. The variance (or width) $\sigma(t)$ representing the amount of exploration is some nonnegative, monotonically decreasing function of $P(t)$. In our model, $\sigma(t)$ is chosen as

$$\sigma(t) = \frac{2k}{1+e^{\lambda p(t)}} - k, \tag{56}$$

$$p(t) = \text{defuzzifier}(P(t)) \tag{57}$$

where λ is a search-range scaling constant which can be simply set to 1, and $P(t)$ is the predicted fuzzy reinforcement signal used to predict $R(t+1)$ when the environment state is $X(t)$ or $x(t)$. The magnitude of $\sigma(t)$ is large when $p(t)$ is low. Because we restrict the highest degree of reward to $p(t) = 0$, the value of $\sigma(t)$ is 0 when $p(t) = 0$. The action $\hat{Y}$, or $\hat{y}$, is what actually applied to the environment. The stochastic perturbation in the suggested approach leads to a better exploration of action space and better generalization ability.

Once the amount of exploration, $\sigma(t)$, has been decided, the next problem is how to generate the actual output. Because the output of the learning system can be fuzzy number or numerical number, we discuss these two situations separately in the followings.

1) Numerical Stochastic Unit: When the action is a numerical value y (i.e., a FCNO is used as the action network), the actual output $\hat{y}$ of the stochastic unit can be set as

$$\hat{y}(t) = N(y(t), \sigma(t)). \tag{58}$$

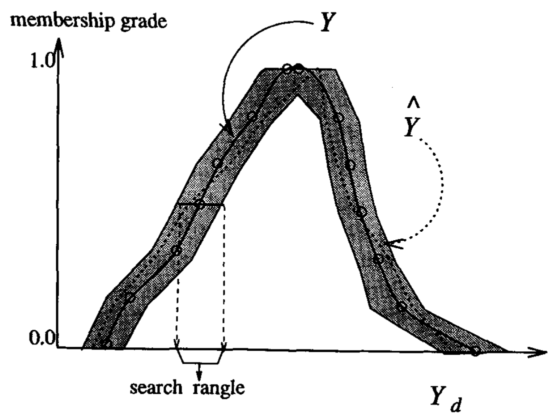


Fig. 17. Illustration of fuzzy stochastic exploration.

That is, $\hat{y}(t)$ is a normal or Gaussian random variable with variance $\sigma(t)$, mean $y(t)$, and the density function

$$f(\hat{y}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(\hat{y}-y)^2/2\sigma^2}. \tag{59}$$

The actual output $\hat{y}$ can be also set simply as a uniform random variable with width 2σ, mean y, and the density function

$$f(\hat{y}) = \begin{cases} \frac{1}{2\sigma}, & \text{if } (y-\sigma) \le \hat{y} \le (y+\sigma), \\ 0, & \text{otherwise.} \end{cases} \tag{60}$$

2) Fuzzy Stochastic Unit: When the action is a fuzzy number $Y = \cup_\alpha\ \alpha[y_1^{(\alpha)}, y_2^{(\alpha)}]$ (i.e., a FCLO is used as the action network), the fuzzy stochastic unit generates a fuzzy action, $\hat{Y} = \cup_\alpha\ \alpha[\hat{y}_1^{(\alpha)}, \hat{y}_2^{(\alpha)}]$, based on the amount of exploration σ. The parameter $\hat{y}_1^{(\alpha)}(\hat{y}_2^{(\alpha)})$ is set as a uniform random variable with mean $y_1^{(\alpha)}(y_2^{(\alpha)})$, width 2σ, and the density function as the same as above. After having decided these parameters, $\hat{y}_1^{(\alpha)}$ and $\hat{y}_2^{(\alpha)}$, we must then maintain the convex property of the fuzzy action. We propose the following procedure to complete the fuzzy stochastic exploration. In this procedure, the notation h is the number of quantized membership grade. As shown in Fig. 17, the actual fuzzy action $\hat{Y}$ must falls in the shadow region randomly in the fuzzy stochastic exploration.

Procedure: Fuzzy Stochastic Exploration:

Input: $Y = \cup_\alpha \alpha[y_1^{(\alpha)}, y_2^{(\alpha)}]$.

Output: $\hat{Y} = \cup_\alpha \alpha[\hat{y}_1^{(\alpha)}, \hat{y}_2^{(\alpha)}]$.

Step 1. For $k = 1$ to h, find $\hat{y}_1^{(k)}$ such that

$$(y_1^{(k)} - \sigma(t)) \le \hat{y}_1^{(k)} \le (y_1^{(k)} + \sigma(t)),$$

and then find $\hat{y}_2^{(k)}$ such that

$$\max(y_2^{(k)} - \sigma(t), \hat{y}_1^{(k)}) \le \hat{y}_2^{(k)} \le (y_2^{(k)} + \sigma(t)).$$

Step 2. For $k = h-1$ downto 0, find $\hat{y}_1^{(k/h)}$ such that

$$(y_1^{(k/h)} - \sigma(t)) \le \hat{y}_1^{(k/h)} \le \min(\hat{y}_1^{(k+1/h)}, y_1^{(k/h)} + \sigma(t)),$$

and find $\hat{y}_2^{(k/h)}$ such that

$$\max(\hat{y}_2^{(k+1/h)}, y_2^{(k/h)} - \sigma(t)) \le \hat{y}_2^{(k/h)} \le (y_2^{(k/h)} + \sigma(t)).$$

Step 3. Output $\hat{Y}$ and stop.

Like the supervised learning process introduced in Section IV, we need to perform two kinds of initialization: structure initialization and parameter initialization before performing the reinforcement learning algorithm. The initialization process is exactly the same as that for the supervised learning (see Section IV). It should be done on both the action network and the evaluation network. After the initialization process, the reinforcement learning algorithms are performed on both networks. These learning algorithms for both the action network and the evaluation network are derived below. Again, we discuss the single step and multistep prediction problems separately in the following subsections.

B. Learning Algorithm for Single-Step Prediction Problems

In this subsection, a reinforcement learning algorithm is proposed to solve the Class I and Class II reinforcement learning problems described previously using a single-step fuzzy predictor. The function of the single-step fuzzy predictor is to predict the external fuzzy reinforcement signal, $R(t+1)$, one time step ahead, that is, at time t. Here, $R(t+1)$ is the external fuzzy reinforcement signal resulting from the inputs and actions chosen at time step t, but it can only be known at time step $t+1$. If the fuzzy predictor can produce a fuzzy signal $P(t)$ at time step t, which is the prediction of $R(t+1)$, a better action can be chosen by the action network at time step t, and the corresponding learning can be performed in the action network at time step $t+1$ upon receiving the external reinforcement signal $R(t+1)$.

Basically, the reinforcement learning of a single-step fuzzy predictor is simply a supervised learning problem. The goal is to minimize the squared error

$$e = \tfrac{1}{2}\sum_\alpha [(p_1^{(\alpha)}(t) - r_1^{(\alpha)}(t+1))^2 + (p_2^{(\alpha)}(t) - r_2^{(\alpha)}(t+1))^2] \tag{61}$$

where $R(t+1) = \cup_\alpha \alpha[r_1^{(\alpha)}(t+1), r_2^{(\alpha)}(t+1)]$ represents the desired fuzzy output, and $P(t) = \cup_\alpha \alpha[p_1^{(\alpha)}(t), p_2^{(\alpha)}(t)]$ is the current predictor output. Similar to the supervised learning algorithm developed for FCLO in Subsection IV-A, we can derive the learning algorithm for the single-step fuzzy predictor. The update rules are the same as (33)–(53) if Y is replaced by $P(t)$ and D is replaced by $R(t+1)$.

We next develop the learning algorithm for the action network. The goal of the reinforcement learning algorithm is to adjust the parameters $w_1^{(\alpha)}$ and $w_2^{(\alpha)}$ of the action network such that the fuzzy reinforcement signal R is maximum; that is

$$\Delta w \propto \frac{\partial r}{\partial w} \tag{62}$$

where $w = w_1^{(\alpha)}$ or $w_2^{(\alpha)}$ and $r = \text{defuzzifier}(R)$. There are two different reinforcement learning algorithms for the action network depending on either FCNO or FCLO being used as the action network. We describe these two reinforcement learning algorithms for the action network in the followings.

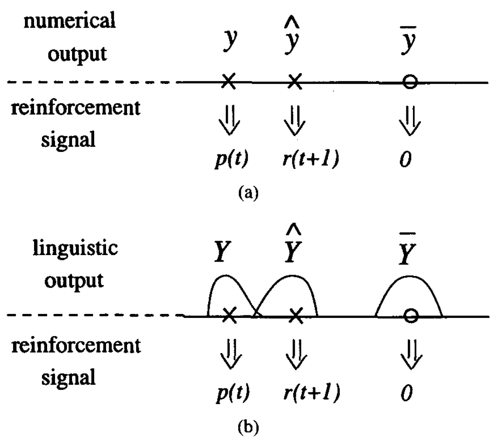


Fig. 18. The concept of deciding desired outputs in the stochastic unit.

We first derive the reinforcement learning algorithm for the action network with numerical output (FCNO). In this situation, to determine $\partial r/\partial w$, we need to know $\partial r/\partial y$, where y is the output of the action network. Since the fuzzy reinforcement signal does not provide any hint as to what the right answer should be in terms of a cost function, the gradient, $\partial r/\partial y$, can only be estimated using the stochastic unit.

The output of the single-step predictor, $P(t)$, is a predicted fuzzy reinforcement signal for the output of the action network, $y(t)$, and the external fuzzy signal $R(t+1)$ is a critic score from environment for actual output $\hat{y}(t)$ of the stochastic unit. From these values, we can construct a desired target output $\bar{y}(t)$. Note that we restrict the values of defuzzified reinforcement signals, $P(t)$ and $R(t+1)$, in the range $[-1,0]$, that is, $-1 \le p(t) \le 0$ and $-1 \le r(t+1) \le 0$, where $p(t) = \text{defuzzifier}(P(t))$ and $r(t+1) = \text{defuzzifier}(R(t+1))$. From Fig. 18, we find that the expected target output $\bar{y}$ should fall on the position representing that the value of reinforcement signal is 0. Hence, if $p(t) \ne r(t+1)$, we let

$$\bar{y} = \frac{(-r(t+1))y - (-p(t))\hat{y}}{(-r(t+1)) - (-p(t))} = \frac{p(t)\hat{y} - r(t+1)y}{p(t) - r(t+1)}. \tag{63}$$

Then, if $p(t) \ne r(t+1)$, we can define

$$e = \tfrac{1}{2}(y - \bar{y})^2. \tag{64}$$

When $p(t) = r(t+1)$, the weights are not changed, that is, $\partial e/\partial y = 0$. With this error function, the learning rules of the FCNO-based action network can be derived. They are the same as (33)–(53) if d is replaced by $\bar{y}$.

When the output of the action network is a fuzzy number Y, we can derive the learning algorithm in a similar way as we did in the above. We decide an expected fuzzy target output $\bar{Y} = \cup_\alpha \alpha[\bar{y}_1^{(\alpha)}, \bar{y}_2^{(\alpha)}]$ as

$$\bar{y}_1^{(\alpha)} = \frac{p(t)\hat{y}_1^{(\alpha)} - r(t+1)y_1^{(\alpha)}}{p(t) - r(t+1)}, \tag{65}$$

$$\bar{y}_2^{(\alpha)} = \frac{p(t)\hat{y}_2^{(\alpha)} - r(t+1)y_2^{(\alpha)}}{p(t) - r(t+1)} \tag{66}$$

when $p(t) \ne r(t+1)$. Then we can define the error function as

$$e = \tfrac{1}{2}\sum_\alpha [(y_1^{(\alpha)} - \bar{y}_1^{(\alpha)})^2 + (y_2^{(\alpha)} - \bar{y}_2^{(\alpha)})^2]. \tag{67}$$

When $p(t) = r(t+1)$, the weights are not changed, that is, $\partial e/\partial y_1^{(\alpha)} = 0$ and $\partial e/\partial y_2^{(\alpha)} = 0$. With this error function, the learning equations of the FCLO-based action network are the same as (33)–(53) if D is replaced by $\bar{Y}$.

There is another method to estimate the gradient information [10]. This method does not construct the expected target output, but finds the gradient direction of the error function. With this method, the output error gradient in (33)–(53) for FCNO can be replaced by

$$-\frac{\partial e}{\partial y} \propto \frac{\partial r}{\partial y} = (r(t+1) - p(t))(\hat{y}(t) - y(t)) \\ = \hat{r}(t+1)(\hat{y}(t) - y(t)), \tag{68}$$

and for FCLO, the gradient information is estimated by

$$-\frac{\partial e}{\partial y_1^{(\alpha)}} = (r(t+1) - p(t))(\hat{y}_1^{(\alpha)} - y_1^{(\alpha)}), \tag{69}$$

$$-\frac{\partial e}{\partial y_2^{(\alpha)}} = (r(t+1) - p(t))(\hat{y}_2^{(\alpha)} - y_2^{(\alpha)}) \tag{70}$$

where $\hat{r}$ is the internal reinforcement signal sent to the action network. In (68), $r(t+1)$ is the actual fuzzy reinforcement feedback for the actual action, $\hat{y}(t)$, and $p(t)$ is the predicted fuzzy reinforcement signal for the expected action, $y(t)$. The ratiole behind the above equations is described as follows. If $r(t+1) > p(t)$, then $\hat{y}(t)$ is a better action than the expected one, $y(t)$, and $y(t)$ should be moved closer to $\hat{y}(t)$. That is, this is a rewarding event when $\hat{r}(t+1) > 0$. If $r(t+1) < p(t)$, then $\hat{y}(t)$ is a worse action than the expected one, and $y(t)$ should be moved farther away from $\hat{y}(t)$. That is, this is a penalizing event when $\hat{r}(t+1) < 0$.

In the proposed system, the action network and the evaluation network are trained together, however, since the action network relies on accurate prediction of the evaluation network, it seems practical to train the fuzzy predictor first, at least partially, or to let the fuzzy predictor have a higher learning rate than the action network. Besides the above parameter learning phase, the structure learning phase described in Section IV is executed to complete the whole learning process.

C. Learning Algorithm for Multistep Prediction Problems

The algorithms described in the last subsection work under the assumption that the environment returns a fuzzy reinforcement signal in response to every single action acting on it. There are many applications in which the learning system receives evaluation of its behavior only after a long sequence of actions; that is, both reinforcement signals and environment states may depend arbitrarily on the past history of the network output. This kind of reinforcement scheme is called delayed reinforcement. In this section we shall discuss how the problem of learning with delayed reinforcement can be solved using the multistep fuzzy predictor.

In the delayed reinforcement learning problem, the temporal credit assignment problem becomes severe because we have to assign credit or blame individually to each action in a sequence for an eventual success or failure. The solution to the temporal credit assignment problem is to design a multistep fuzzy predictor which can predict the reinforcement signal at each time step. To achieve this purpose, the technique based on the temporal difference (TD) method is used. The TD method is a class of incremental learning procedures introduced by Sutton [11]. The main characteristic of the TD method is that they learn from successive predictions, whereas in the case of supervised learning, learning occurs only when the difference between the predicted outcome and the actual outcome is revealed. Hence the learning in TD does not have to wait until the actual outcome is known, and can update its parameters within a trial period. In the proposed reinforcement learning system, we use TD methods in the evaluation network to make it function as a multistep fuzzy predictor. We shall discuss three different cases of reinforcement learning problems below. Note again that because the reinforcement signal is linguistic, we use an FCLO as the multistep fuzzy predictor.

Case 1: Prediction of final fuzzy outcome. Assume we are given the fuzzy/numerical input sequences of the form, $X(1), X(2), \cdots, X(m)$, where each $X(t)$ is an input vector of fuzzy numbers or real numbers available at time step t from the environment and the fuzzy reinforcement signal is $R(m+1)$ at time step $m+1$. For each input sequence, the fuzzy predictor produces a corresponding sequence of predictions $P(1), P(2), \cdots, P(m)$, each of which is an estimate of $R(m+1)$. We assume that the fuzzy weights $W = \cup_\alpha \alpha[w_1^{(\alpha)}, w_2^{(\alpha)}]$ in the evaluation network are updated only once for each complete input sequence and does not change during a sequence. After a complete sequence has been processed, $w(w_1^{(\alpha)}$ or $w_2^{(\alpha)})$ is changed by the sum of all the sequence's increments

$$w \leftarrow w + \sum_{t=1}^{m} \Delta w(t). \tag{71}$$

Because each $P(t)$ is an estimate of $R(m+1)$, the error function based on supervised learning approach in each time step t is

$$e = \tfrac{1}{2}\sum_{\alpha}[(p_1^{(\alpha)}(t) - r_1^{(\alpha)}(m+1))^2 + (p_2^{(\alpha)}(t) - r_2^{(\alpha)}(m+1))^2] \tag{72}$$

and

$$\Delta w(t) \propto -\frac{\partial e}{\partial w(t)}. \tag{73}$$

Thus, the update rules of the $\Delta w(t)$ are derived as

$$\frac{\partial e}{\partial w_1^{(\alpha)}(t)} = \frac{\partial e}{\partial p_1^{(\alpha)}(t)}\frac{\partial p_1^{(\alpha)}(t)}{\partial w_1^{(\alpha)}(t)} = (p_1^{(\alpha)}(t) - r_1^{(\alpha)}(m+1))\frac{\partial p_1^{(\alpha)}(t)}{\partial w_1^{(\alpha)}(t)}, \tag{74}$$

$$\frac{\partial e}{\partial w_2^{(\alpha)}(t)} = (p_2^{(\alpha)}(t) - r_2^{(\alpha)}(m+1))\frac{\partial p_2^{(\alpha)}(t)}{\partial w_2^{(\alpha)}(t)}. \tag{75}$$

In either case, note that all $\Delta w(t)$ in (74) and (75) depend critically on $R(m+1)$, and thus cannot be determined until the end of the sequence when $R(m+1)$ becomes known. Thus, all observations (inputs) and predictions made during a sequence must be remembered until the end, and then all the $\Delta w(t)$'s can be computed.

In fact, (74) and (75) can be computed incrementally as shown below. First, consider the following facts

$$r_1^{(\alpha)}(m+1) - p_1^{(\alpha)}(t) = \sum_{k=t}^{m}(p_1^{(\alpha)}(k+1) - p_1^{(\alpha)}(k)), \tag{76}$$

$$r_2^{(\alpha)}(m+1) - p_2^{(\alpha)}(t) = \sum_{k=t}^{m}(p_2^{(\alpha)}(k+1) - p_2^{(\alpha)}(k)) \tag{77}$$

where $p_1^{(\alpha)}(m+1) = r_1^{(\alpha)}(m+1)$ and $p_2^{(\alpha)}(m+1) = r_2^{(\alpha)}(m+1)$. By replacing $r_1^{(\alpha)}(m+1) - p_1^{(\alpha)}(t)$ and $r_2^{(\alpha)}(m+1) - p_2^{(\alpha)}(t)$ by (76) and (77), (74) and (75) are transformed to

$$\frac{\partial e}{\partial w_1^{(\alpha)}(t)} = -(p_1^{(\alpha)}(t+1) - p_1^{(\alpha)}(t))\sum_{k=1}^{t}\frac{\partial p_1^{(\alpha)}(k)}{\partial w_1^{(\alpha)}(k)}, \tag{78}$$

$$\frac{\partial e}{\partial w_2^{(\alpha)}(t)} = -(p_2^{(\alpha)}(t+1) - p_2^{(\alpha)}(t))\sum_{k=1}^{t}\frac{\partial p_2^{(\alpha)}(k)}{\partial w_2^{(\alpha)}(k)} \tag{79}$$

which can be computed incrementally at each time step. The procedure given by (78) and (79) is the special case of the TD procedure, called TD(1), in which all of the predictions are altered to an equal extent. It can be extended to the following general form

$$\frac{\partial e}{\partial w_1^{(\alpha)}(t)} = -(p_1^{(\alpha)}(t+1) - p_1^{(\alpha)}(t))\sum_{k=1}^{t}\lambda^{t-k}\frac{\partial p_1^{(\alpha)}(k)}{\partial w_1^{(\alpha)}(k)}, \tag{80}$$

$$\frac{\partial e}{\partial w_2^{(\alpha)}(t)} = -(p_2^{(\alpha)}(t+1) - p_2^{(\alpha)}(t))\sum_{k=1}^{t}\lambda^{t-k}\frac{\partial p_2^{(\alpha)}(k)}{\partial w_2^{(\alpha)}(k)} \tag{81}$$

in which alterations to the predictions of input vectors occuring k steps in the past are weighted according to λ^{t-k} for $0 \le \lambda \le 1$. The formulas in (80) and (81) are called the TD(λ) procedure proposed by Sutton in [11]. In the extreme case that $\lambda = 0$, called TD(0), the weight increment is determined only by its effect on the prediction associated with the most recent observation

$$\frac{\partial e}{\partial w_1^{(\alpha)}(t)} = -(p_1^{(\alpha)}(t+1) - p_1^{(\alpha)}(t))\frac{\partial p_1^{(\alpha)}(k)}{\partial w_1^{(\alpha)}(k)}, \tag{82}$$

$$\frac{\partial e}{\partial w_2^{(\alpha)}(t)} = -(p_2^{(\alpha)}(t+1) - p_2^{(\alpha)}(t))\frac{\partial p_2^{(\alpha)}(k)}{\partial w_2^{(\alpha)}(k)}. \tag{83}$$

Case 2: Prediction of finite cumulative fuzzy outcomes. The TD method can be also used to predict a quantity that accumulates over a sequence. That is, each step of a sequence may incur a cost, and we wish to predict the expected total cost over the sequence. In this problem, the predictor output $P(t)$ is to predict the remaining cumulative fuzzy cost given the tth observation rather than the overall fuzzy cost for the sequence. In our system, we consider the cost to be the value of the reinforcement signal. Let $R(t+1) = \cup_\alpha \alpha[r_1^{(\alpha)}(t+1), r_2^{(\alpha)}(t+1)]$ denote the actual fuzzy cost incurred between time steps t and $t+1$. We would like $P(t)$ to be equal to the expected value of $Z(t) = \cup_\alpha \alpha[z_1^{(\alpha)}(t), z_2^{(\alpha)}(t)] = \Sigma_{k=t}^m R(k+1)$. Thus,

$$z_1^{(\alpha)}(t) = \sum_{k=t}^{m} r_1^{(\alpha)}(k+1), \tag{84}$$

$$z_2^{(\alpha)}(t) = \sum_{k=t}^{m} r_2^{(\alpha)}(k+1). \tag{85}$$

The prediction error can be represented in terms of temporal difference as

$$\begin{aligned} z_1^{(\alpha)}(t) - p_1^{(\alpha)}(t) &= \sum_{k=t}^{m} r_1^{(\alpha)}(k+1) - p_1^{(\alpha)}(t) \\ &= \sum_{k=t}^{m} (r_1^{(\alpha)}(k) + p_1^{(\alpha)}(k+1) - p_1^{(\alpha)}(k)), \end{aligned} \tag{86}$$

$$\begin{aligned} z_2^{(\alpha)}(t) - p_2^{(\alpha)}(t) &= \sum_{k=t}^{m} r_2^{(\alpha)}(k+1) - p_2^{(\alpha)}(t) \\ &= \sum_{k=t}^{m} (r_2^{(\alpha)}(k) + p_2^{(\alpha)}(k+1) - p_2^{(\alpha)}(k)) \end{aligned} \tag{87}$$

where $P(m+1) = \cup_\alpha \alpha[0, 0]$. Thus, the update rules are

$$\begin{aligned} \frac{\partial e}{\partial w_1^{(\alpha)}(t)} &= -(r_1^{(\alpha)}(t+1) + p_1^{(\alpha)}(t+1) \\ &\quad - p_1^{(\alpha)}(t)) \sum_{k=1}^{t} \lambda^{t-k} \frac{\partial p_1^{(\alpha)}(k)}{\partial w_1^{(\alpha)}(t)}, \end{aligned} \tag{88}$$

$$\begin{aligned} \frac{\partial e}{\partial w_2^{(\alpha)}(t)} &= -(r_2^{(\alpha)}(t+1) + p_2^{(\alpha)}(t+1) \\ &\quad - p_2^{(\alpha)}(t)) \sum_{k=1}^{t} \lambda^{t-k} \frac{\partial p_2^{(\alpha)}(k)}{\partial w_2^{(\alpha)}(t)}. \end{aligned} \tag{89}$$

Case 3: Prediction of infinite discounted cumulative fuzzy outcomes. In this case, $P(t)$ predicts the discounted sum $Z(t) = \Sigma_{k=0}^{\infty} \gamma^k R(t+k+1)$; i.e., $z_1^{(\alpha)}(t) = \Sigma_{k=0}^{\infty} \gamma_k r_1^{(\alpha)}(t+k+1), z_2^{(\alpha)}(t) = \Sigma_{k=0}^{\infty} \gamma_k r_2^{(\alpha)}(t+k+1)$, where $\gamma, 0 \le \gamma \le 1$, is the discounted rate parameter. If the prediction is accurate, we can write

$$\begin{aligned} p_1^{(\alpha)}(t) &= \sum_{k=0}^{\infty} \gamma^k r_1^{(\alpha)}(t+k+1) \\ &= r_1^{(\alpha)}(t+1) + \gamma \sum_{k=0}^{\infty} \gamma^k r_1^{(\alpha)}(t+k+2) \\ &= r_1^{(\alpha)}(t+1) + \gamma p_1^{(\alpha)}(t+1) \end{aligned} \tag{90}$$

and

$$p_2^{(\alpha)}(t) = r_2^{(\alpha)}(t+1) + \gamma p_2^{(\alpha)}(t+1). \tag{91}$$

The mismatch or TD error is the difference between the right hand and left hand sides of these equations $r_1^{(\alpha)}(t+1) + \gamma p_1^{(\alpha)}(t+1), r_2^{(\alpha)}(t+1) + \gamma p_2^{(\alpha)}(t+1)$ and thus the update rules are

$$\begin{aligned} \frac{\partial e}{\partial w_1^{(\alpha)}(t)} &= -(r_1^{(\alpha)}(t+1) + \gamma p_1^{(\alpha)}(t+1) \\ &\quad - p_1^{(\alpha)}(t)) \sum_{k=1}^{t} \lambda^{t-k} \frac{\partial p_1^{(\alpha)}(k)}{\partial w_1^{(\alpha)}(k)}, \end{aligned} \tag{92}$$

$$\begin{aligned} \frac{\partial e}{\partial w_2^{(\alpha)}(t)} &= -(r_2^{(\alpha)}(t+1) + \gamma p_2^{(\alpha)}(t+1) \\ &\quad - p_2^{(\alpha)}(t)) \sum_{k=1}^{t} \lambda^{t-k} \frac{\partial p_2^{(\alpha)}(k)}{\partial w_2^{(\alpha)}(k)}. \end{aligned} \tag{93}$$

Once the output error gradient information of the multistep fuzzy predictor is obtained using the methods discussed in the above three cases, its learning becomes a supervised learning problem and thus (33)–(53) in Section IV can be used here directly. Note that, the system only receives an external reinforcement signal $R(m+1)$ after a sequence of inputs at the time step $m+1$. Hence, we can assume that the external reinforcement signal $R(t)$ is zero (nonexisting) at the other time steps, that is, $R(t) = \cup_\alpha \alpha[0, 0]$, for $2 \le t \le m$.

As for the action network, the learning algorithm of the action network is similar to that derived in Subsection V-B. When the output is numerical, we have

$$\hat{r}(t+1) = r(t+1) + \gamma p(t+1) - p(t) \tag{94}$$

and

$$\begin{aligned} -\frac{\partial e}{\partial y} \propto \frac{\partial r}{\partial y} &= (r(t+1) + \gamma p(t+1) - p(t))(\hat{y}(t) - y(t)) \\ &= \hat{r}(t+1)(\hat{y}(t) - y(t)). \end{aligned} \tag{95}$$

When the output is linguistic, we have

$$-\frac{\partial e}{\partial y_1^{(\alpha)}} \propto \frac{\partial r}{\partial y_1^{(\alpha)}} = \hat{r}(t+1)(\hat{y}_1^{(\alpha)}(t) - y_1^{(\alpha)}(t)), \tag{96}$$

$$-\frac{\partial e}{\partial y_2^{(\alpha)}} \propto \frac{\partial r}{\partial y_2^{(\alpha)}} = \hat{r}(t+1)(\hat{y}_2^{(\alpha)}(t) - y_2^{(\alpha)}(t)). \tag{97}$$

According to (95), before the external reinforcement signal occurs (i.e., $r(t) = 0$), the reinforcement $\hat{r}$ sent to the action network is the difference between the predicted reinforcement signal of the current time step (discounted by γ) and the predicted reinforcement signal of the previous time step (i.e., $\gamma p(t+1) - p(t)$). That is, (95) becomes $\partial r/\partial y = (\gamma p(t+1) - p(t))(\hat{y}(t) - y(t))$, where $p(t)$ is the predicted reinforcement signal for the output $y(t)$ of the action network, and $p(t+1)$ is the predicted reinforcement signal for output $y(t+1)$. Because both external reinforcement signals and input patterns depend on the past history, $\hat{y}(t)$ will influence the predicted reinforcement signal $p(t+1)$ for output $y(t+1)$; that is, the output $\hat{y}(t)$ at time step t will influence the output $y(t+1)$ at time step $t+1$. Thus, $p(t+1)$ can be viewed as the predicted

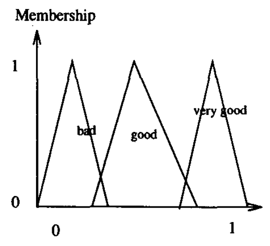


Fig. 19. The fuzzy reinforcement signals used in Example 3.

reinforcement signal for the actual output $\hat{y}(t)$ at time step t. From the above description, we know that the value of $\partial r/\partial y$ is always positive. Namely, if $\gamma = 1$, increases in reinforcement prediction become rewarding events (i.e., $\hat{r} > 0$), and decreases become penalizing events (i.e., $\hat{r} < 0$).

When the external reinforcement signal occurs, the situation is slightly different. When the external reinforcement signal comes at time step $t+1$, we let the corresponding predicted reinforcement signal, $p(t+1)$, be zero. In this situation, (95) becomes $\partial r/\partial y = (r(t+1) - p(t))(\hat{y}(t) - y(t))$. The external reinforcement signal $r(t+1)$ is the actual critic score for $\hat{y}$, and $p(t)$ is the predicted reinforcement signal of $y(t)$. Thus, the value of $\partial r/\partial y$ will be positive. From the above observation, we understand that (95)–(97) are appropriate for the action network.

Until now, we have developed the parameter learning algorithms for the action network and the evaluation network for multistep prediction problems. The structure learning in these two networks is the same as that described in Section IV for supervised learning. The following simple example illustrates the proposed fuzzy reinforcement learning model.

Example 3: In this example, we transform the supervised learning problem in Example 1 (in Section IV) to a reinforcement learning problem. We use the same training data as in Example 1 except that we assume the desired outputs are not known to the learning system. In this example, we have three fuzzy input data as follows

X_1: x is very small, (the desired output: y is very large (D_1)),

X_2: x is small, (the desired output: y is large (D_2)),

X_3: x is large, (the desired output: y is small (D_3))

where the fuzzy numbers "small," "large," and "very large" in inputs are shown in Fig. 6, and the fuzzy numbers "very large," " large," and "small" in desired outputs $(D1, D2, D3)$ are shown in Fig. 8. The actual output $E1(E2, E3)$ is supposed to equal $D1(D2, D3)$ in Fig. 8. These desired outputs are not known to the neural network. Only the reinforcement signal defined below is imported to the learning system at each time step t

$$R(t) = \begin{cases} \text{very good}, & \text{if Error} < 0.05, \\ \text{good}, & \text{if } 0.05 \le \text{Error} \le 0.5, \\ \text{bad}, & \text{if Error} > 0.5 \end{cases}$$

where Error $= 1/2\Sigma_\alpha\,[(x_{i1}^{(\alpha)} - d_{i1}^{(\alpha)})^2 + (x_{i2}^{(\alpha)} - d_{i2}^{(\alpha)})^2], i = 1, 2, 3$. The reinforcement signals are shown in Fig. 19.

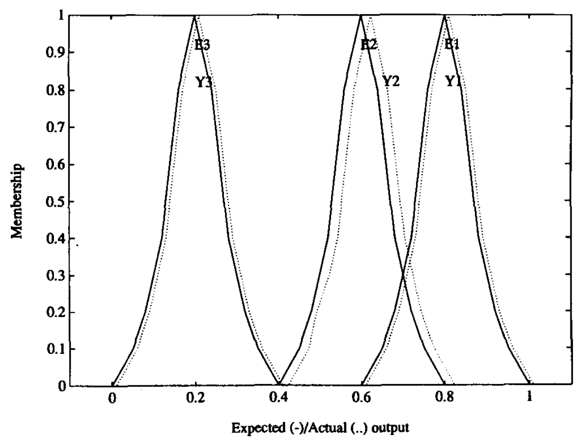


Fig. 20. The desired outputs (E) and actual outputs (Y) after reinforcement learning in Example 3.

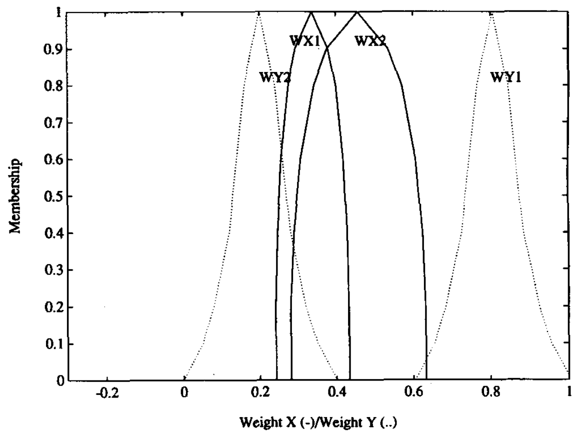


Fig. 21. The learned fuzzy weights in Example 3.

This problem belongs to the single-step prediction problem, thus a single-step predictor is required for our proposed system. Moreover, owing to the desired linguistic output, the procedure of fuzzy stochastic exploration in Subsection V-B is used for the system. The simulation results are shown in Figs. 20 and 21. Fig. 20 indicates that the actual outputs coincide the desired outputs quite well.

VI. Illustrative Examples

Two typical examples are presented in this section to show the fundamental applications of the proposed neural fuzzy system. First, a fuzzified cart-pole balancing problem is used to demonstrate the proposed fuzzy reinforcement learning model developed in Section V. Second, we apply our fuzzy supervised learning model derived in Section IV to a practical fuzzy expert system for rule concentration.

Example 4: Cart-Pole Balancing Problem

In this example, we apply the proposed fuzzy reinforcement learning model to the fuzzified cart-pole balancing system. In this system, a pole is hinged to a moter-driven cart that moves on rail tracks to its right or its left as shown in Fig. 22. The

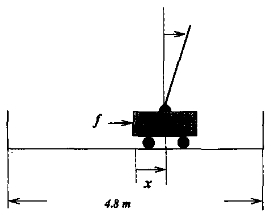


Fig. 22. The cart-pole balancing system in Example 4.

pole has only one degree of freedom (rotation about the hinge point). The primary control tasks are to keep the pole vertically balanced and keep the cart within the rail track boundaries.

Four state variables are used to describe the system status, and one control variable represents the force applied to the cart. They are

θ angle of the pole from an upright position (in radian);

$\dot{\theta}$ angular velocity of the pole (in radian/s);

x horizontal position of the cart's center (in meters);

$\dot{x}$ velocity of the cart (in meters/s);

f the amount of force (N) applied to the cart to move it toward left or right.

The goal of this learning problem is to train the proposed reinforcement learning model such that it can determine a sequence of forces with proper magnitudes to apply to the cart to balance the pole for as long as possible without failure. The dynamics of the cart-pole balancing system are modeled by the following equations [14] as shown in (98) at the bottom of the page and where

g—$9.8\,\mathrm{m/s}^2$, acceleration due to the gravity,

m_c 1.0 kg, the mass of the cart,

m 0.1 kg, the mass of the pole,

l 0.5 m, the half-pole length,

μ_c 0.0005, the coefficient of friction of cart on track,

μ_p 0.000002, the coefficient of friction of pole on cart,

$\triangle$ 0.02, sampling interval.

The constraints on the variables are $-12\,\text{degrees} \le \theta \le 12\,\text{degrees}, -2.4\text{ m} \le x \le 2.4\text{ m}$, and $-10N \le f \le 10N$. A more challenging part of this problem is that the only available feedback is a reinforcement signal that notifies the controller when a failure occurs; that is, either $|\theta| > 12\,\text{degrees}$ or $|x| > 2.4\,\text{m}$. Since a reinforcement signal may only be available after a long sequence of time steps in this failure avoidance task, this cart-pole balancing problem belongs to the multistep prediction problem discussed in Subsection V-C. The fuzzy reinforcement signal is defined as

$$R(t) = \begin{cases} \text{bad}, & \text{if } |\theta(t)| > 12\,\text{degrees or } |x(t)| > 2.4m, \\ \text{good}, & \text{otherwise.} \end{cases}$$

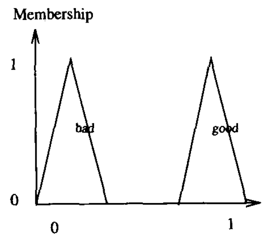


Fig. 23. The membership functions of fuzzy reinforcement signals in Example 4.

The membership function of $R(t)$ are shown in Fig. 23. Because the control action required for the cart-pole balancing system is numerical, we use an FCNO as the action network. The simulation results shown in Fig. 24 indicate that the proposed reinforcement learning model can learn the control task in less than 12 trials, where a trial is ended by a failure signal.

This example successfully demonstrates the applicability of the proposed technique to the learning problems with semantic-level error signal. Such learning problems are usually found in the human-machine interactive systems. Currently, we are applying the proposed fuzzy reinforcement learning technique to the adaptive spoken language acquisition system with semantic-level error feedback for automated call routining.

Example 5: Rule Base Concentration for KBE.

In this example, we apply an FCLO with supervised learning to a practical application, the KBE, for rule concentration. The KBE [37] is an expert system that evaluates expert system application. In KBE, an expert system decides whether or not an application with certain characteristics is a good expert system application. Selecting good applications for expert system is crucial not only to the success of a particular project,

$$\begin{aligned}
\theta(t+1) &= \theta(t) + \triangle\dot{\theta}(t), \\
\dot{\theta}(t+1) &= \dot{\theta}(t) + \triangle\ddot{\theta}(t) \\
&= \dot{\theta}(t) + \triangle\frac{g\sin\theta(t) + \cos\theta(t)\left[\dfrac{-f(t) - ml\dot{\theta}^2(t)\sin\theta(t) + \mu_c\mathrm{sgn}(\dot{x}(t))}{m_c + m}\right]}{l\left[\dfrac{4}{3} - \dfrac{m\cos^2\theta(t)}{m_c + m}\right]}, \\
x(t+1) &= x(t) + \triangle\dot{x}(t), \\
\dot{x}(t+1) &= \dot{x}(t) + \triangle\ddot{x}(t) \\
&= \dot{x}(t) + \triangle\frac{f(t) + ml[\dot{\theta}^2(t)\sin\theta(t) - \ddot{\theta}(t)\cos\theta(t)] - \mu_c\mathrm{sgn}[\dot{x}(t)]}{m_c + m}
\end{aligned} \tag{98}$$

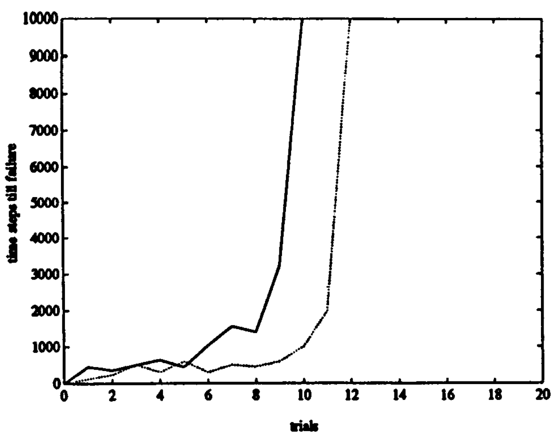


Fig. 24. Learning curves of the proposed fuzzy reinforcement learning system on the cart-pole balancing problem in Example 4.

but to the long term view of knowledge-based system work which develops in your company.

In KBE, inputs of each feature are linguistic terms describing the characteristics of the application, and the output of KBE is suitability which indicates whether the application of the expert system on a domain is good or bad. These features are described as follows:

- Worth: The worth for doing any software project based on a payoff of some kinds.
- Employee Acceptance: How will employees react to the system? That is, the effects of the system on corporate culture.
- Solution Available: How good is an existing solution?
- Easier Solution: Is there an easier way to solve the problem?
- Teachability: How easily is the skill taught?
- Risk: The likelihood of not being able to complete a project.

Because all the inputs and output are linguistic, we use FCLO and supervised learning scheme in Section IV for rule base concentration for KBE. We are given 486 fuzzy if-then rules from experts in advance. These fuzzy if-then rules and the membership functions used for each attribute are illustrated in Table I, Fig. 25, and Fig. 26. In Table I, the star (*) means "don't care." Hence, for example, the first rule in the table in fact includes 54 ($3 \times 3 \times 3 \times 2$) rules. The learned fuzzy logic rules and fuzzy weights are shown in Table II, Figs. 27 and 28. We use only 13 fuzzy if-then rules to represent the 486 fuzzy if-then rules in the original knowledge base.

VII. 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

TABLE I
The KBE Training Rules in Example 5

Feat-ures	Input Vector						Output
	Worth	Employee-Acceptance	Solution-Available	Easier-Solution	Teachability	Risk	Suitability
1	Negative	*	*	*	*	*	Poor
2	Low	*	*	*	*	High	Poor
3	Low	Positive	*	*	*	Low	Good
4	Low	Neutral	*	*	*	Low	Fair
5	Low	*	Adequate	*	*	Low	Good
6	Low	*	Partial	*	*	Low	Fair
7	Low	*	*	Complete	*	Low	Good
8	Low	*	*	Partial	*	Low	Fair
9	Low	*	*	*	Frequent	Low	Good
10	Low	*	*	*	Possible	Low	Fair
11	Low	Negative	None	None	Difficult	Low	Poor
12	Low	Negative	None	None	Possible	Low	Fair
13	Low	Negative	None	Partial	Difficult	Low	Fair
14	Low	Negative	Partial	None	Difficult	Low	Fair
15	Low	Neutral	None	None	Difficult	Low	Fair
16	High	*	*	*	*	Low	Good
17	High	Positive	*	*	*	High	Good
18	High	Neutral	*	*	*	High	Fair
19	High	*	Adequate	*	*	High	Good
20	High	*	Partial	*	*	High	Fair
21	High	*	*	Complete	*	High	Good
22	High	*	*	Partial	*	High	Fair
23	High	*	*	*	Frequent	High	Good
24	High	*	*	*	Possible	High	Fair
25	High	Negative	None	None	Difficult	High	Poor
26	High	Neutral	None	None	Difficult	High	Fair
27	High	Negative	Partial	None	Difficult	High	Fair
28	High	Negative	None	Partial	Difficult	High	Fair
29	High	Negative	None	None	Possible	High	Fair

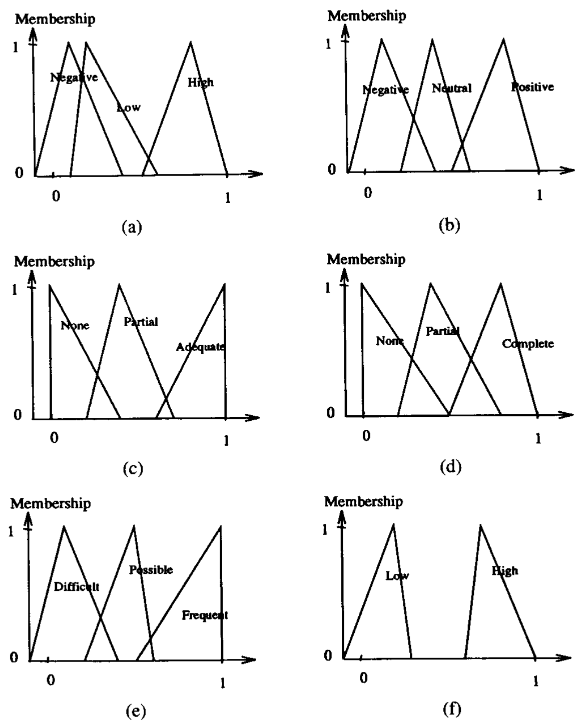


Fig. 25. The input membership functions for training in Example 5.

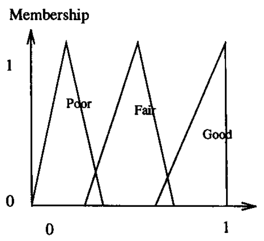


Fig. 26. The output membership functions for training in Example 5.

TABLE II
THE LEARNED FUZZY RULES IN EXAMPLE 5

Feat-ures	Input Vector						Output
	Worth	Employee-Acceptance	Solution-Available	Easier-Solution	Teachability	Risk	Suitability
1	Negative	–	–	–	–	–	Poor
2	Low	–	–	–	–	High	Poor
3	Low	Positive	–	–	–	Low	Good
4	Low	–	Adequate	–	–	Low	Good
5	Low	–	–	Complete	–	Low	Good
6	Low	–	–	–	Frequent	Low	Good
7	Low	Negative	None	None	Difficult	Low	Poor
8	High	–	–	–	–	Low	Good
9	High	Positive	–	–	–	High	Good
10	High	–	Adequate	–	–	High	Good
11	High	–	–	Complete	–	High	Good
12	High	–	–	–	Frequent	High	Good
13	High	Negative	None	None	Difficult	High	Poor

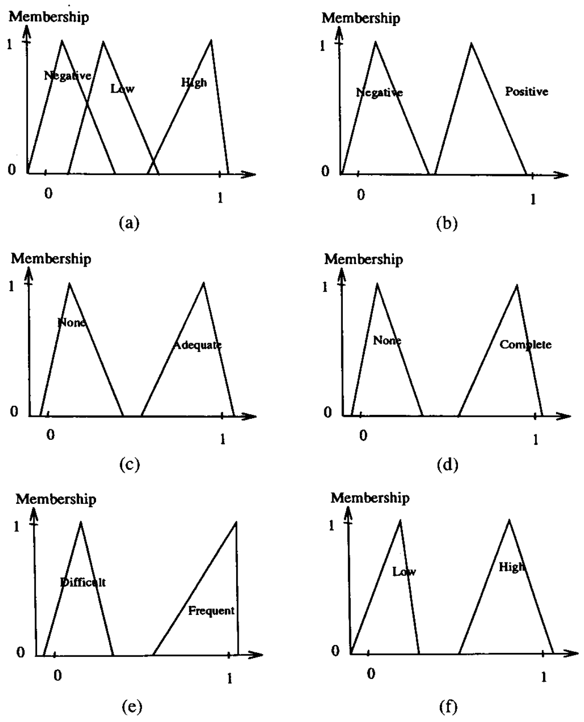


Fig. 27. The learned input fuzzy weights in Example 5.

has both supervised and reinforcement learning capabilities. With supervised learning, the proposed system can be used for a fuzzy expert system, fuzzy system modeling, or rule base concentration. With reinforcement learning, the proposed system can be used as an adaptive fuzzy controller. It can learn proper fuzzy control rules and membership functions through experts' linguistic instructions. Computer simulations satisfactorily verified the performance of the proposed neural fuzzy system.

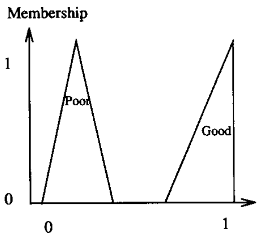
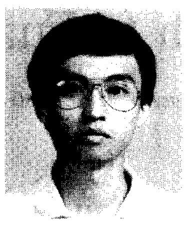


Fig. 28. The learned output fuzzy weights in Example 5.

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," *Inform. 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] J. Hertz, A. Krogh and R. G. Palmer, *Introduction to the Theory of Neural Computation*. New York: Addison-Wesley, 1991.
[5] 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.
[6] H. Ishibuchi and H. Tanaka, "Fuzzy regression analysis using neural networks," *Fuzzy Sets and Systems*, vol. 50, pp. 257–265, 1992.
[7] 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, 1993, pp. 1650–1655.
[8] Y. Hayashi, J. J. Buckley, and E. Czogula, "Fuzzy neural network," *Int. J. Intelligent Syst.*, vol. 8, pp. 527–537, 1993.
[9] Y. Hayashi, J. J. Buckley, and E. Czogula, "Systems engineering application of fuzzy neural networks," in *Proc. Int. Joint Conf. on Neural Networks*, Baltimore, 1992, pp. 413–418.
[10] C. T. Lin and C. S. G. Lee, "Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems," *IEEE Trans. Fuzzy Syst.*, vol. 2, no. 1, pp. 46–63, 1995.
[11] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, pp. 9–44, 1988.
[12] A. Barto and P. Anandan, "Pattern-recognizing stochastic learning automata," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, no. 3, pp. 360–375, 1985.
[13] H. R. Berenji, "A reinforcement learning-based architecture for fuzzy control," *Int. J. Approximate Reasoning*, vol. 6, pp. 267–292, 1992.
[14] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, no. 5, pp. 834–847, 1993.
[15] H. R. Berenji and P. Khedkar, "Learning and tuning fuzzy logic controllers through reinforcements," *IEEE Trans. Neural Networks*, vol. 3, no. 5, pp. 724–740, 1992.
[16] C. W. Anderson, "Strategy learning with multilayer connectionist representations," in *Proc. Fourth Int. Workshop Mach. Learn.*, Irvine, CA, June 1987, pp. 103–114.
[17] P. Munro, "A dual backpropagation scheme for scalar reward Learning," in *Ninth Annual Conf. Cognitive Sci. Soc.*, Erlbaum, Seattle, 1987, pp. 165–176.
[18] R. J. Williams, "A class of gradient-estimating algorithms for reinforcement learning in neural networks," in *Proc. Int. Joint Conf. Neural Networks*, San Diego, CA, vol. II, 1987, pp. 601–608.
[19] A. G. Barto and R. S. Sutton, "Landmark learning: An illustration of association search," *Biol. Cerbern.*, vol. 42, pp. 1–8, 1981.
[20] A. G. Barto and M. I. Jordan, "Gradient following without backpropagation in layered networks," in *Proc. 1987 Int. Joint Conf. Neural Networks*, San Diego, 1987, pp. 629–636.
[21] K. S. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction*. Englewood Cliffs, NJ: Prentice Hall, 1989.

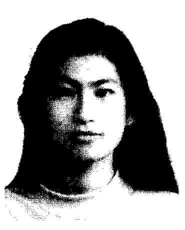
[22] A. H. Klopf, *The Headonistic Neuron: A Theory of Memory, Learning and Intelligence*. Washington, DC: Hemisphere, 1982.
[23] K. Uehara and M. Fujise, "Fuzzy inference based on families of α-level sets," *IEEE Trans. Fuzzy Syst.*, vol. 1, no. 2, pp. 111–124, 1993.
[24] K. Uehara, "Computational efficiency of fuzzy inference based on level sets," in *Proc. 1989 Spring Nat. Conv. Rec., IEICE*, Japan, p. D-400.
[25] ——, "Fast operation of fuzzy inference based on level sets," in *Proc. 38th Ann. Conv. Rec. IPS Japan Rec.*, 1989, p. 3G-3.
[26] L. A. Zadeh, "The concept of a linguistic truth variable and its application to approximate reasoning—I and II," *Inform. Sci.*, vol. 8, pp. 199–249, 301–357, 1975.
[27] A. Kaufmann and M. M. Gupta, *Introduction to Fuzzy Arithmetic*. New York: Van Nostrand, 1985.
[28] M. Braae and D. A. Rutherford, "Fuzzy relations in a control setting," *Kyberbetes*, vol. 7, no. 3, pp. 185–188, 1978.
[29] C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller—Parts I and II," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-20, no. 2, pp. 404–435, 1990.
[30] 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.
[31] Jyh-Shing Jang, "Self-learning fuzzy controllers based on temporal backpropagation," *IEEE Trans. Neural Networks*, vol. 3, no. 5, pp. 714–723, 1992.
[32] 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. New York: North-Holland, 1979.
[33] G. E. Hinton, "Connectionist learning procedure," *Art. Intelligence*, vol. 40, no. 1, pp. 143–150, 1989.
[34] J. M. Zurada, *Introduction to Artificial Neural Systems*. New York: West, 1992.
[35] P. J. Werbos, "Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research," *IEEE Trans. Syst., Man, Cybern.*, vol. 17, pp. 7–20, 1987.
[36] ——, "Generalization of backpropagation with application to a recurrent gas market model," *Neural Networks*, vol. 1, pp. 339–356, 1988.
[37] R. Keller, *Expert System Technology—Development and Application*. Englewood, NJ: Prentice-Hall, 1987.
[38] B. Kosko, *Neural Networks and Fuzzy Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1992.



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, National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., where he is 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).

Dr. Lin is a member of Tau Beta Pi and Eta Kappa Nu. He is also a member of the IEEE Computer Society, the IEEE Robotics and Automation Society, and the IEEE Systems, Man, Cybernetics Society.



Ya-Ching Lu 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, Taipei, Taiwan, R.O.C., where she is currently a Research Assistant at the Institute of Information Science. Her current research interests include fuzzy systems, neural networks, and pattern recognition.