Adaptive Fuzzy Command Acquisition with Reinforcement Learning

Chin-Teng Lin and Ming-Chih Kan

***Abstract*—This paper proposes a four-layered adaptive fuzzy command acquisition network (AFCAN) for adaptively acquiring fuzzy command via interactions with the user or environment. It can catch the intended information from a sentence (command) given in natural language with fuzzy predicates. The intended information includes a meaningful semantic action and the fuzzy linguistic information of that action (for example, the phrase "move forward" represents the meaningful semantic action and the phrase "very high speed" represents the linguistic information in the fuzzy command "move forward at a very high speed"). The proposed AFCAN has three important features. First, we can make no restrictions whatever on the fuzzy command input, which is used to specify the desired information, and the network requires no acoustic, prosodic, grammar, and syntactic structure. Second, the linguistic information of an action is learned adaptively and it is represented by fuzzy numbers based on α-level sets. Third, the network can learn during the course of performing the task. The AFCAN can perform off-line as well as on-line learning. For the off-line learning, the mutual-information (MI) supervised learning scheme and the fuzzy backpropagation (FBP) learning scheme are employed when the training data are available in advance. The former learning scheme is used to learn meaningful semantic actions and the latter learn linguistic information. The AFCAN can also perform on-line learning interactively when it is in use for fuzzy command acquisition. For the on-line learning, the MI-reinforcement learning scheme and the fuzzy reinforcement learning scheme are developed for the on-line learning of meaningful actions and linguistic information, respectively. An experimental system (fuzzy commands acquisition of a voice control system) is constructed to illustrate the performance and applicability of the proposed AFCAN.**

***Index Terms*—Fuzzy backpropagation, fuzzy number, fuzzy reinforcement, linguistic information, mutual information, semantic action, voice control.**

I. Introduction

MOST researchers in automatic speech recognition (ASR) concentrate their efforts on process of converting speech to ordinary text with the intention of later combining their results with others who are working on language acquisition. That is, they assume that transcription and acquisition are distinct processes connected at a simple orthographical interface and always focus on highly faithful models of acoustic, prosodic, and syntactic structure, especially to the exclusion of meaning. It is found that the techniques that attempt to separate transcription and acquisition will result in systems whose performance usually falls far short of human capacity and is hard to reach the ultimate goal—unrestricted free communication between man and machine in a changing and uncertain world [1]. Hence, developing a language acquisition system that involves gaining the capability of decoding the intended information in a message spoken in natural language is the concern of this work. As a start to the fuzzy language acquisition system, in this paper, we shall use the fuzzy neural network to deal with the fuzzy command acquisition problems due to the capability of the fuzzy neural network in processing and learning both numerical and linguistic information. The proposed network will equip the ability to acquire fuzzy commands, which is a nature language consisting of desired actions and fuzzy linguistic information (fuzzy predicates). Moreover, it can perform on-line learning to acquire fuzzy commands during the course of performing tasks.

Much research on language acquisition has focused on discovering syntax structure, often to the exclusion of meaning. The goal is to develop a theory that can predict the set of grammatical sentences in a language from a finite number of observations [2]–[5]. The final purpose of these systems is to obtain an applicable syntactic structure. There are a few researchers who focus their attentions on obtaining the mapping from message to meaning. In [6] and [7], the systems learn the mapping from sentences to symbolic representations; that is, the approach is to represent the meaning symbolically, attempting to make the representation isomorphic to some subset of reality. Recently, neural networks were also utilized on language acquisition. In 1991, Jain utilized a modular recurrent connectionist network to learn to parse sentences. In 1989 and 1991, Miikkulainen and Dyer [8] used a modular network to learn to paraphrase script-based stories. In 1990, St. John and McClelland [9] also used modular connectionist networks to learn the mapping from input sentences to an output event description, comprising a set of thematic roles and their filters. In the above systems, the networks are trained supervisedly using the backpropagation (BP) algorithm. During the procedure of network training, the input sentence and the desired output are provided for the network.

In the above work, the semantic and pragmatic domain is quite rich and, as such, comprising an important portion of the semantic structure of the entire language. For the purpose of performing at a desired level, those systems require great constrained input—a severely restricted vocabulary or a rigid syntax. In contrast, a different approach was proposed by Gorin *et al.* [10], [11] where the system's understanding of an input

Manuscript received December 4, 1995; revised January 18, 1997. This work was supported by the R.O.C. National Science Council under Grant NSC86-2212-E-009-019.

The authors are with the Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu, Taiwan, R.O.C.

Publisher Item Identifier S 1063-6706(98)00803-0.

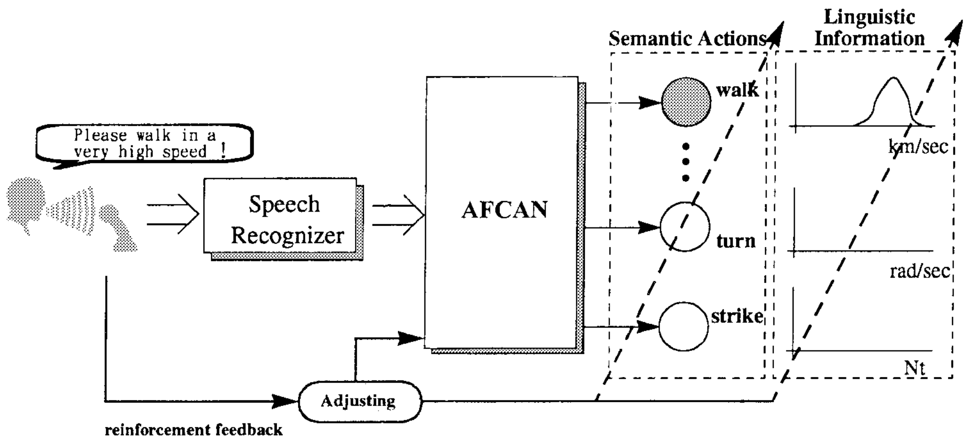


Fig. 1. The use of the proposed AFCAN in a voice control system.

message was evaluated on the basis of whether the system responded in an expected and appropriate way over a wide range of scripts. An information-theoretic-connectionist network was proposed that learned the mapping from the input message to a meaningful system response. The network learned through interactive feedback received from the environment or user as to the appropriateness of the system's response to an input message. The Gorin's system accepts input sentences without any restricted vocabulary, so it makes no restrictions on the language used to specify the desired action to the system. However, the action has a very restricted semantic domain; that is, it can perform a few actions.

In the above approaches, none of them can process or learn fuzzy linguistic information in natural language. Since such processing and learning ability is the length of the fuzzy neural network, we establish a fuzzy neural network called adaptive fuzzy command acquisition network (AFCAN) for acquiring fuzzy commands in this paper. The proposed network can adaptively acquire fuzzy commands via interactions with the user or environment. It can catch the intended information from a sentence (command) spoken or written in natural language with fuzzy predicates. The intended information includes a meaningful semantic action and the linguistic information of that action. Furthermore, since the AFCAN is developed based on Gorin's approach, it keeps the property that we can make no restrictions whatever on the fuzzy command input, which is used to specify the desired information; also, the network requires no acoustic, prosodic, grammar, and syntactic structure.

The AFCAN consists of four layers and can be regarded as a cascaded network comprising two subnetworks—the crisp connectionist architecture with numerical output (CCNO) net and the fuzzy connectionist architecture with linguistic output (FCLO) net. The former is a two-layered network with crisp mutual information weights and the latter is a three-layered network with fuzzy weights. The input to the AFCAN is unrestricted text in fuzzy language and the output of the AFCAN is the user's desired semantic action and the associated fuzzy linguistic information. More clearly, the CCNO processes the user's input command to acquire the desired semantic action and the FCLO maps a crisp input to a desired fuzzy linguistic output presented in the form of α-level sets [12].

The AFCAN learning includes two parts—off-line learning and on-line learning. In off-line learning, the training phase is finished before doing the performance phase, but in on-line learning, the training proceeds during the course of performing task. For these two kinds of learning, four learning algorithms are developed and employed: 1) mutual information (MI) supervised learning; 2) fuzzy backpropagation (FBP) learning; 3) MI-reinforcement learning; and 4) fuzzy reinforcement learning. Learning algorithms 1) and 3) are used in the CCNO to adjust its crisp MI weights and algorithms 2) and 4) are used in the FCLO to adjust its fuzzy weights. Learning algorithms 1) and 2) are used for off-line learning to build an initial network for real performance. These two learning algorithms are also used in the supervised mode of on-line learning and algorithms 3) and 4) are used in the reinforcement mode of on-line learning. The on-line learning is to rebuild or tune an off-line trained AFCAN according to the critics from the user/environment.

The proposed AFCAN can be applied in a voice control system, as shown in Fig. 1. The system combines a speech recognizer with the proposed AFCAN. The user can speak a fuzzy command to the microphone freely. The speech recognizer will recognize the user's speech signals and then the AFCAN will acquire the input command. In this system, the objective of acquisition is to produce the correct semantic action and proper linguistic information about that action. For example, if the user gives a fuzzy command, "Please walk at a very high speed," then the system will produce the results with the user's desired semantic action "walk" and its linguistic information in the form of α-level sets as shown in Fig. 1. Moreover, the system can do the on-line learning while functioning; that is, the user or teacher (supervisor) can observe the system's performance and give critic reinforcement feedback to the system. There are two kinds of reinforcement signals, one for semantic action (e.g., "No, I want to walk instead of run!") and the other for linguistic information (e.g., "The speed is too fast."). The system can utilize the reinforcement feedback to tune itself to become a more suitable system for users.

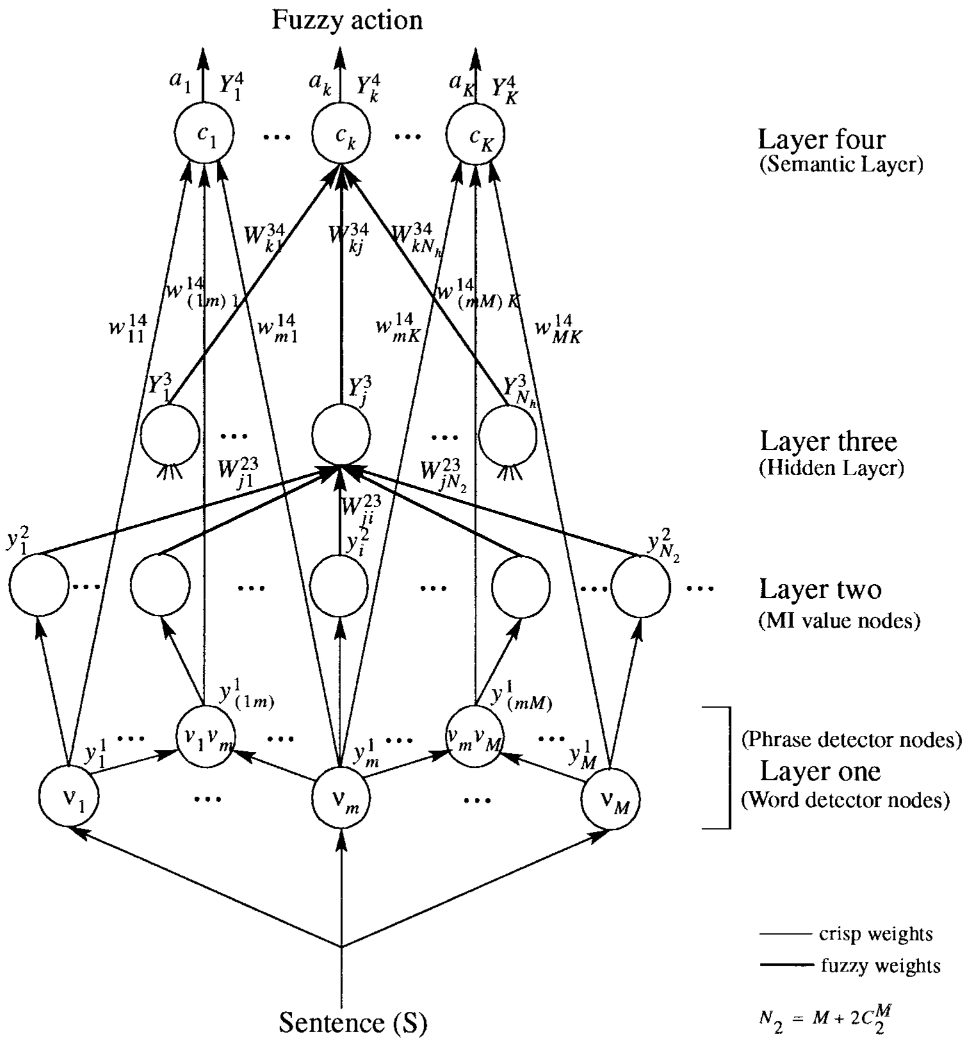


Fig. 2. Network structure of the AFCAN.

This paper is organized as follows. In Section II, we describe the structure of the proposed AFCAN and discuss the learning issues of the network. Section III presents the supervised learning schemes of the proposed network. In Section IV, the mutual information reinforcement learning scheme and fuzzy reinforcement learning scheme are developed. In Section V, an application of the AFCAN—*fuzzy commands acquisition of a prototype voice control system*—is illustrated. The computational efficiency and convergence property of the proposed learning algorithms are discussed in Section VI. Finally, conclusions are made in Section VII.

II. Adaptive Fuzzy Command Acquisition Network

In this section, we shall propose a network for our acquisition system whose input is the unrestricted text of fuzzy commands and output is one of a finite set of semantic fuzzy actions. This network is called AFCAN. The basic structure of the proposed network will be described first and then the learning of this network will be discussed briefly.

A. Basic Structure of the AFCAN

Fig. 2 shows the proposed network structure of the AFCAN, which has a total of four layers. The input is a sentence and output is a fuzzy action comprising the action and its linguistic information. Layer one consists of two kinds of nodes for different functions: the word detector nodes and the phrase detector nodes whose inputs are from the outputs of the word detector nodes directly. The word and phrase detector nodes are, respectively, used to detect the input isolated words and the input phrases (word pairs). The layer-one nodes are fully connected to layer-four nodes. Each node in layer two corresponds to one node in layer one. The input value of a layer-two node is produced by the output value of the corresponding layer-one node multiplied by the weight value between this layer-one node and the fired layer-four node. The links from layer two to layer three are fully connected and so are the links from layer three to layer four.

From another point of view, we can consider the AFCAN as the combination of "a multilayer neural network that maps a sentence to a semantic action" and "a multilayer fuzzy neural network that maps a numerical input vector to a fuzzy number." The former network is called CCNO and the latter is called FCLO. The CCNO is constituted of the layer one and layer four of the AFCAN, as shown in Fig. 3, and the FCLO is constituted of the layer two, layer three and layer four of the AFCAN, as shown in Fig. 4. The CCNO proceeds

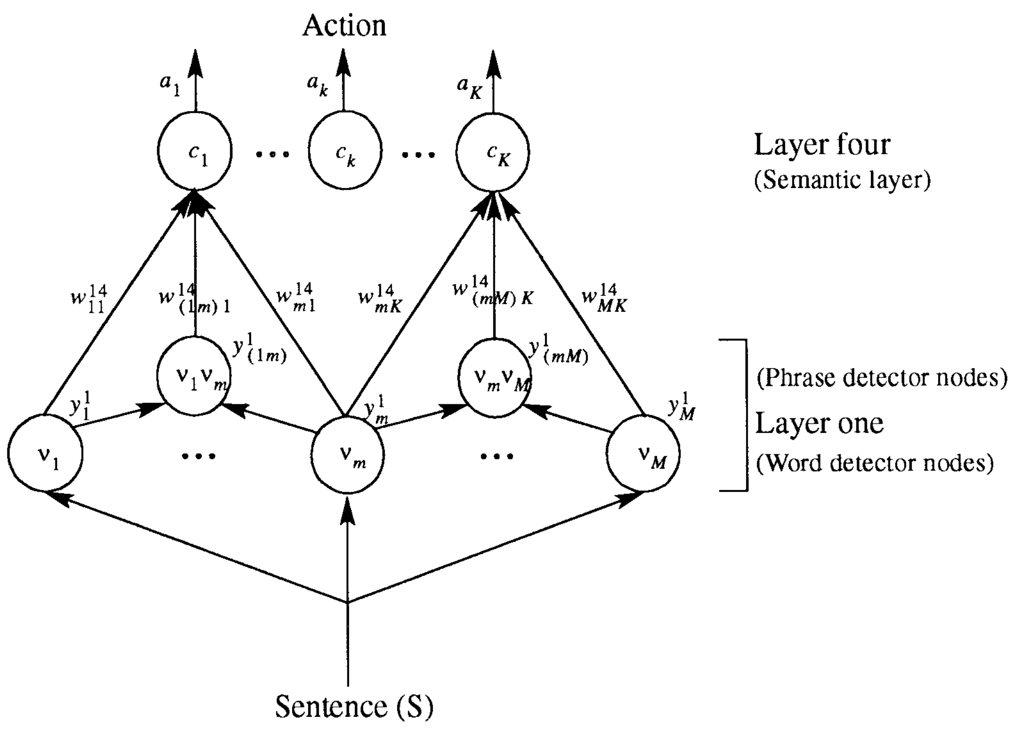


Fig. 3. Network structure of the CCNO.

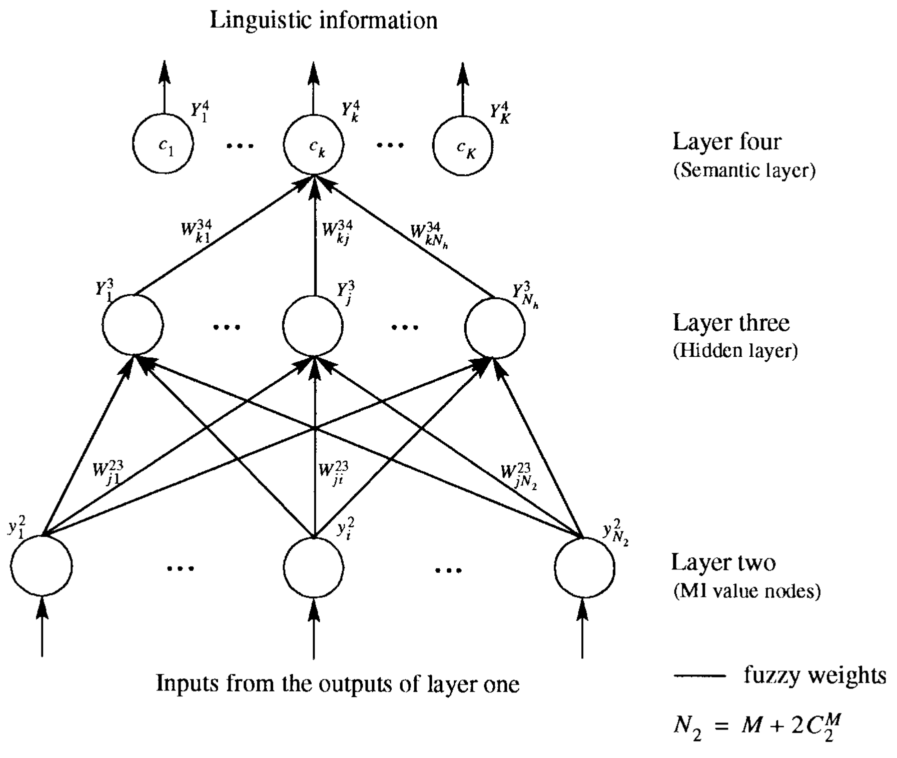


Fig. 4. Network structure of the FCLO.

before the FCLO; that is, for a given input sentence, the CCNO decides first the semantic action and then the FCLO decide the corresponding fuzzy linguistic information. Notice that although the input values of the FCLO are dependent of the weights from layer-one nodes to fired layer-four nodes, the AFCAN is a feedforward (instead of recurrent) network since its two constituent networks (CCNO and FCNO) function independently and sequentially.

In the AFCAN, we define the semantic actions c_k, $1 \leq k \leq K$, and the vocabulary words ν_m where the phrases $\nu_m\nu_n$ are also produced naturally, $1 \leq m, n \leq M$. The network maps an input sentence $S = \langle \nu_{m_1}, \nu_{m_2}, \cdots, \nu_{M_L} \rangle$ (L is the number of words in sentence S) to a semantic action $c_{k_0}(S)$. In layer one, the word detector nodes detect the presence of a vocabulary word in the input sentence and the phrase detector nodes, where we consider only phrases comprising two adjacent words, detect the presence of the adjacent word-pair in the sentence. Layer two of the AFCAN is employed to be the mutual-information (MI) value input layer. The MI values are calculated according to the weights between the nodes in layer one and the k_0th node in layer four after the semantic node c_{k_0} is decided (fired). The MI values are used as the inputs for training the FCLO to produce the desired fuzzy number in the k_0th semantic node in layer four. In this network, the "hidden (internal) layer," layer three, is added to increase the learning efficiency of the FCLO.

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 reverse direction in the learning process as we shall discuss in the following sections. 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. In the following, the notations y (Y) represent the output crisp (fuzzy) number of a node.

Layer 1—Detector Layer: The nodes in this layer are divided into two groups: word detector nodes and phrase detector nodes. The inputs to the phrase detector node $\nu_m\nu_n$ are the outputs of the word detector nodes ν_m and ν_n. For this reason, the input sentences have to pass the word detector nodes first and then pass through the phrase-detector nodes.

- Word detector nodes: the function of each word detector node is to detect the presence of a vocabulary word ν_m in the input sentence S and produce an output y_m^1 between zero and one. In this paper, words are counted only once, no matter how often they appear. The word detector nodes execute some function that can detect the presence of a word in the sentence. The simplest function is a matching function that produces the output 1 if a particular word is observed; otherwise produces the output 0. A more sophisticated function is one that produces an output y_m^1 equal to the probability that the word ν_m is in the sentence.
- Phrase detector nodes: the function of each phrase detector node is to detect the presence of a vocabulary phrase $\nu_m\nu_n$ in the sentence S and produce an output y_{mn}^1 between zero and one. The simplest case is for noise-free input in which the output is one if phrase $\nu_m\nu_n$ is observed; otherwise, the output is zero.

Note that the number of phrase detector nodes is $2 \times C_2^M$, depending on the number of word detector nodes (M) where, in addition to the phrase type $\nu_m\nu_n$, the inverse type $\nu_n\nu_m$ is also considered so that is why the number C_2^M should be multiplied by two.

As a summary, the input–output relation of each node in this layer is

$$\text{Input } (S) = \text{an unrestricted sentence} \tag{1}$$

$$\text{Output } y_m^1(y_{mn}^1) = \begin{cases} 1, & \text{if the word } (\nu_m) \text{ or phrase} \\ & \quad (\nu_m\nu_n) \text{ is observed} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Layer 2—MI-Value Layer: The input of each node in this layer is a numerical number coming from the output of the node $y_m^1(y_{mn}^1)$ in layer one multiplied by the weight $w_{mk_0}^{14}$, assuming that the semantic action $c_{k_0}(S)$ in layer four is recognized. That is, the input to a layer-two node is $y_m^1(y_{mn}^1) \times w_{mk_0}^{14}$. Each node in this layer only transmits input numerical number to the next layer directly. Hence, we have

$$y_i^2 = y_m^1(\text{or } y_{mn}^1) \times w_{mk_0}^{14} \tag{3}$$

where $1 \le i \le M + 2 \times C_2^M$, $1 \le m, n \le M$, $1 \le k_0 \le K$, and K is the number of nodes in layer four.

Layer 3—Hidden Layer: As described previously, layers two, three, and four of the AFCAN constitute the FCLO network that can map numerical input values to fuzzy output numbers. The input values fed into each node in this layer are the weighted output values of layer two, which are numerical numbers. In order to produce fuzzy outputs, there should exist fuzzy weights between layer two and layer three, so each node in this layer is fully connected to the nodes in layer two through fuzzy weights. More precisely, we have

$$\text{Output: } Y_j^3 = f(\text{Net}_j^3), \qquad j = 1, 2, \cdots, N_h \tag{4}$$

$$\text{Net}_j^3 = \sum_{i=1}^{M+2\times C_2^M} W_{ji}^{23} y_i^2 (+) \theta_j^3 \tag{5}$$

where Y_j^3 is computed by using the extension principle [13], "(+)" represents the addition of fuzzy numbers [13], [14], $f(x) = 1/(1+\exp^{-x})$ is the sigmoid function, N_h is the number of hidden nodes, and the net-inputs Net_j^3, and the biases θ_j^3 are fuzzy numbers. Here, the fuzzy weights W_{ji}^{23} and biases θ_j^3 will be updated in the learning process of the AFCAN.

Layer 4—Semantic Layer: The input values fed into each node in this layer have two sources; one is from layer one and the other from layer three. The outputs of layer one are combined by each of the semantic nodes in this layer to produce output activations a_k for a semantic action c_k as follows:

$$a_k = \sum_{m=1}^{M} y_m^1 w_{mk}^{14} + \sum_{m=1}^{M}\sum_{n=1}^{M} y_{mn}^1 w_{(mn)k}^{14} + w_k \tag{6}$$

where w_k are the biases and w_{mk}^{14} and $w_{(mn)k}^{14}$ are information-theoretic connection weights, which are crisp numbers and will be defined in the next section. The semantic node with the largest activation value a_k is considered to be "fired" and recognized as the acquired semantic action for the input sentence S, i.e., $c_{k_0}(S) = \arg\max_k a_k$.

The outputs of layer three are fed into each node in this layer too and each layer-four node is fully connected to the nodes in layer three through fuzzy weights. The fuzzy output of each layer-four node is described by

$$\text{Fuzzy Output: } Y_k^4 = f(\text{Net}_k^4), \qquad k = 1, 2, \cdots, K \tag{7}$$

$$\text{Net}_k^4 = \sum_{j=1}^{N_h} W_{kj}^{34} Y_j^3 (+) \theta_k^4 \tag{8}$$

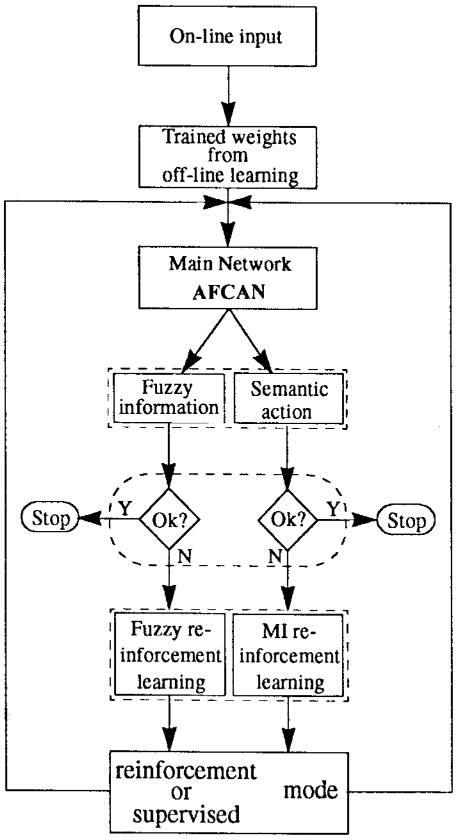

On-line input
Trained weights from off-line learning
Main Network AFCAN
Fuzzy information
Semantic action
Ok?
Y
N
Stop
Fuzzy re-inforcement learning
MI re-inforcement learning
reinforcement or supervised mode

Fig. 5. Block diagram of the on-line learning scheme of the AFCAN.

where Y_k^4 is computed by using the extension principle [13], "(+)" represents the addition of fuzzy numbers [13], [14], $f(x) = 1/(1 + \exp^{-x})$ is the sigmoid function, K is the number of nodes in layer four, and the net-inputs Net_k^4 and biases θ_k^4 are fuzzy numbers. Here, the fuzzy weights W_{kj}^{34} and biases θ_k^4 will also be updated in the learning process of the AFCAN.

Notice that the natural language contains several types of fuzziness. The types of fuzziness that can be treated with the AFCAN are *fuzzy predicates* and *fuzzy predicate modifiers* [15]. Words such as "high," "slowly," "lightly," "soon," and "much faster," are fuzzy predicates. As for the fuzzy predicate modifiers, in addition to the negation modifier there is a variety of fuzzy predicate modifiers which act as *hedges*, e.g., "very," "rather," "more or less," "slightly," "a little," "extremely." Hence, the AFCAN can acquire a fuzzy command like, "Let it turn extremely fast."

B. Learning of the AFCAN

In the AFCAN, we perform off-line learning to build an initial network and then use on-line learning to rebuild or tune a trained AFCAN according to the critics from the user/environment when the AFCAN is in use. The mutual information (MI) supervised learning and fuzzy backpropagation (FBP) learning are employed for the off-line learning of the AFCAN. For off-line learning, we need to prepare a set of training data for supervised learning. At first we used the MI-supervised learning (a well-known statistical method of measuring association) to obtain connection weights between the detector layer (layer one) and the semantic action layer (layer four) in the CCNO of the AFCAN (see Fig. 3). The training data $\langle$*input, output*$\rangle$ for the MI-supervised learning is a set of ordered pairs $\langle$*sentence, action*$\rangle$ consisting of a sentence and an associated semantic action. The MI-supervised learning algorithm will utilize the training data to calculate the association of each node in layer one and the semantic node in layer four. This association is recorded by numerical weights denoted as w_{mk}^{14} (described in Section II-A).

When the MI-supervised learning is completed, the FBP learning, which can be viewed as an extension of the backpropagation learning algorithm to the case of fuzzy data, is applied to the FCLO of the AFCAN (see Fig. 4). The training input data for the FCLO is the MI values computed from the trained CCNO weights as mentioned in Section II-A and the corresponding target output for the FCLO is a fuzzy number represented in the form of α-level sets. The FBP can learn the desired fuzzy input–output mapping, which is represented by fuzzy weights W_{ji}^{23} between layer two and layer three as well as W_{kj}^{34} between layer one and layer four. Hence, with the FBP algorithm we can learn the desired mapping between input fuzzy sentence and linguistic information of the semantic action.

As for on-line learning of the AFCAN, we propose two reinforcement learning schemes: *MI-reinforcement learning* and *fuzzy-reinforcement learning*. The on-line learning of the AFCAN includes the MI on-line learning phase and the fuzzy on-line learning phase. The MI supervised and reinforcement learning algorithms are for the MI on-line learning. The FBP and fuzzy reinforcement learning algorithms are for the fuzzy on-line learning. Like the off-line learning process, the MI on-line learning has to be performed first and, after we learn the desired semantic action, the fuzzy on-line learning is performed. The flowchart of the on-line learning is shown in Fig. 5. Based on the trained weights from the off-line learning, the AFCAN is in use for fuzzy command acquisition. For an input sentence, the AFCAN will produce a *semantic action* (by the CCNO in Fig. 3) and *fuzzy information* (by the FCLO in Fig. 4). Then the user/environment can provide a teaching signal to the AFCAN to indicate the appropriateness of the acquired semantic action and fuzzy information. When the AFCAN receives the teaching signal, it will perform on-line learning to improve itself. The MI on-line learning includes reinforcement mode and supervised mode according to the types of teaching signals, and so does the fuzzy on-line learning. The main difference between reinforcement mode and supervised mode is that the former's teaching signal only provides critic feedback (called *critic* or *reinforcement signal*), but the latter's indicates the desired output. In supervised mode, we use the MI-supervised learning algorithm or the FBP learning algorithm. In reinforcement mode, we use the MI-reinforcement learning algorithm or the fuzzy reinforcement learning algorithm. The details of these learning algorithms will be presented in the next two sections.

III. Supervised Learning of the AFCAN

In this section, we shall derive a supervised learning scheme for the proposed AFCAN. This scheme is suitable to the situations where pairs of input–output training data are available. For each training datum, the input is an unrestricted sentence and the outputs are a numerical number and a fuzzy number. Look at a simple example: if we consider a sentence $S = \langle$*Go in a very high speed*$\rangle$ as input, then the desired outputs consist of the semantic action "*go,*" which is indicated by a special numerical number c_k (the kth action) and the linguistic information "*very high speed,*" which is represented by a specific fuzzy number Y_k in the form of α-level sets where $k = 1, 2, \cdots, K$ and K is the total number of semantic actions. Hence, each training pair is in the form $\langle$*sentence; numerical number, fuzzy number*$\rangle$. Before the learning of the AFCAN is started, an initial network is first constructed. The initial structure of the AFCAN is constructed according to the number of words and number of semantic actions in the way described in Section II (see Fig. 2). The number of hidden nodes in layer three is guessed properly. The fuzzy weights W_{mk}^{23} (W_{kj}^{34}) between layer two and layer three (between layer three and layer four) are initialized randomly as fuzzy numbers and so are the biases θ_j (θ_k) of layer three (layer four). The other weights are randomly initialized as numerical numbers.

After the initialization process, the network is ready for learning. We shall next propose a two-phase supervised learning scheme for the AFCAN. In phase one, a mutual information (MI) learning algorithm is used to define and adjust the numerical weights and biases of the CCNO. In phase two, a fuzzy backpropagation (FBP) learning algorithm is used to adjust the fuzzy weights and biases of the FCLO. The flowchart of this two-phase supervised learning scheme is illustrated in Fig. 6.

A. MI-Supervised Learning

Mutual information (MI) is a famous statistical method of measuring association [16]. Methods based on MI for language understanding have been proposed [10], [11], [17], [18]. Given a system input A and observed system output B, the mutual information (MI) between A and B, denoted by $I(A, B)$ is defined by [19]

$$I(A, B) = \log \frac{P(A|B)}{P(A)} \tag{9}$$

where conditional probability $P(A|B)$ represents the amount of uncertainty remaining about the system input A after the system output B has been observed, and probability $P(A)$ represents our uncertainty about the system input before observing the system output. The $I(A, B)$ represents the uncertainty about the system input that is resolved by observing the system output. We shall use MI to define and adjust the crisp connection weights and biases of the CCNO in Fig. 3.

Rather than viewing the information weights and biases in (6) as abstract parameters, we give them explicit meaning by the following definitions proposed by Gorin *et al.* [10], [11]. As shown in the MI-supervised learning part of the flowchart in Fig. 6, the training database is preprocessed by word detectors and phrase detectors (in layer one of the AFCAN) and then the MI theory is applied to do the MI-supervised learning task. Finally, we can get the association weights w^{14} between detector layer (layer one) and semantic layer (layer four). The MI-supervised learning rule is specified as follows:

$$w_{mk}^{14} = I(\nu_m, c_k) = \log \frac{P(c_k|\nu_m)}{P(c_k)} \tag{10}$$

$$\begin{aligned} w_{(mn)k}^{14} &= I(\nu_m\nu_n, c_k) - I(\nu_m, c_k) - I(\nu_n, c_k) \\ &= \log \frac{P(c_k|\nu_m\nu_n)}{P(c_k)} - \log \frac{P(c_k|\nu_m)}{P(c_k)} \\ &\quad - \log \frac{P(c_k|\nu_n)}{P(c_k)} \end{aligned} \tag{11}$$

$$w_k = \log P(c_k) = \log P(c_k). \tag{12}$$

The notation $\nu_m\nu_n$, a word-pair, comprises the adjacent co-occurrence of words in the sentences. The connection weights from the phrase detector nodes in layer one to the semantic layer (layer four) [given in (11)] is the excess MI of the word pair over the individual words.

Given a semantically labeled sentence, i.e., an input–output pair $\langle$*sentence, action*$\rangle$, weight adaptation proceeds in two steps. First, each segmented token from the sentence is assigned to the best-matching input node. Second, the connection weights are updated in accordance with the mutual information theory. In particular, the mutual information between each word (or word pair) and action is computed from smoothed relative frequency estimates such that no gradient computations are required.

The MI weights are defined in terms of single and joint probabilities that are in turn estimated using the computation of relative frequencies. These are expressed as follows:

$$\begin{aligned} \hat{P}(c_k|\nu_m) &= \frac{N(\nu_m, c_k)}{N(\nu_m)} \\ \hat{P}(c_k|\nu_m\nu_n) &= \frac{N(\nu_m\nu_n, c_k)}{N(\nu_m\nu_n)} \\ \hat{P}(c_k) &= \frac{N(c_k)}{N_T} \end{aligned} \tag{13}$$

where $N(\nu_m)$ denotes the number of observations of the word ν_m in all classes, $N(c_k)$ denotes the number of observations of sentences in class c_k, $N(\nu_m\nu_n)$ denotes the number of observations of the word pair $\nu_m\nu_n$ in all classes, $N(\nu_m, c_k)$ denotes the number of observations of the word ν_m in all sentences of class c_k, $N(\nu_m\nu_n, c_k)$ denotes the number of observations of the word pair $\nu_m\nu_n$ in the class c_k, and $N_T = \sum_{k=1}^{K} N(c_k)$ denotes the total number of all sentences observed in all of the K classes. It is noted that the above estimate can converge asymptomatically, but if the number of observations is small, the estimate will produce quantization noise. To overcome this problem, smoothing the presence of a small number of observations by interpolating the measured relative frequencies with a prior belief is adopted in this paper. Moreover, for the purpose of accelerating the unlearning of false connections, we appropriately set a center clipping threshold so that a single counterexample drives the probability estimate back to the prior and the connection weight to zero. The details of the smoothing and center clipping scheme can be found in Gorin *et al.* [10], [11].

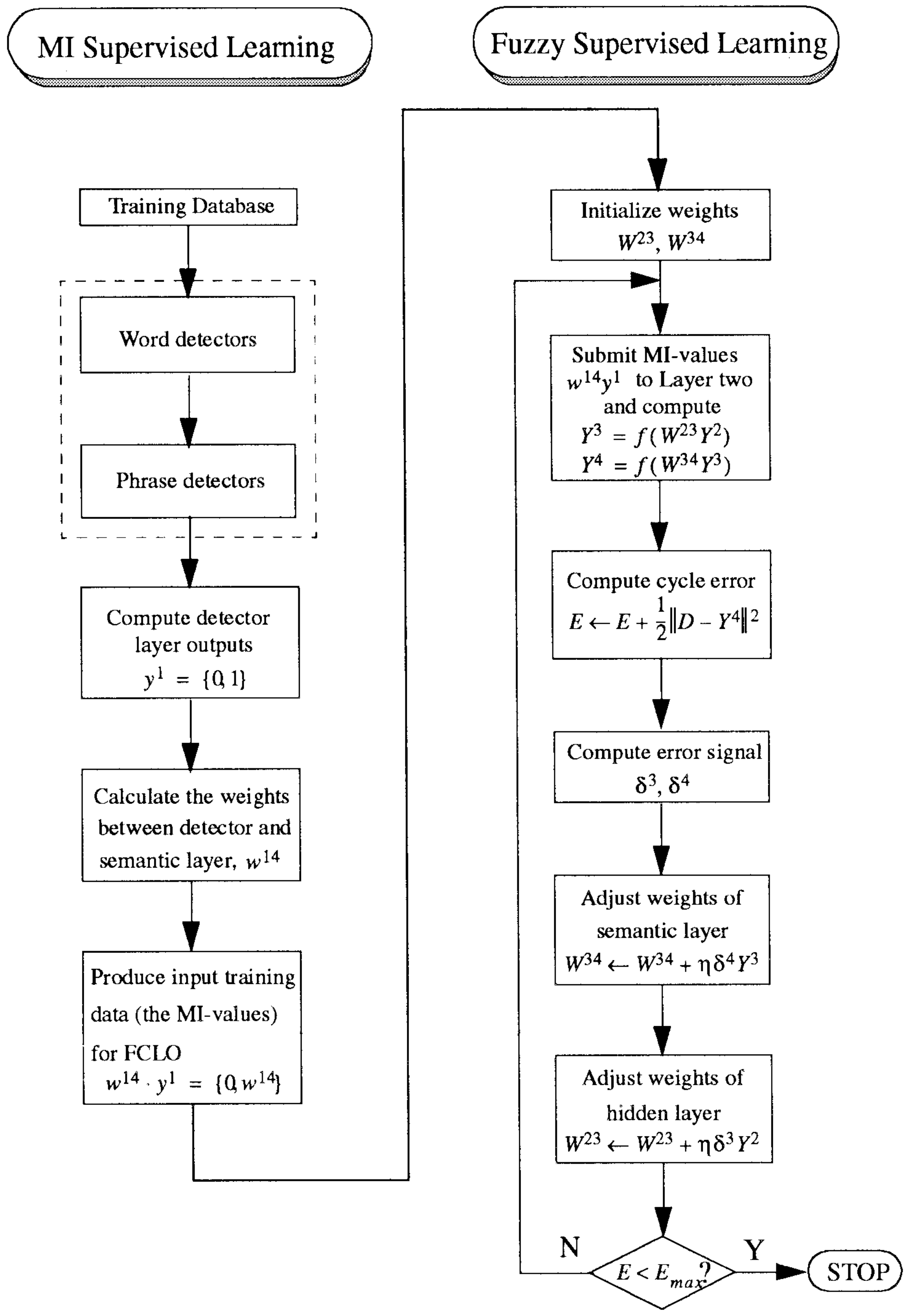

MI Supervised Learning
Fuzzy Supervised Learning
Training Database
Word detectors
Phrase detectors
Compute detector layer outputs $y^1 = \{0, 1\}$
Calculate the weights between detector and semantic layer, w^{14}
Produce input training data (the MI-values) for FCLO $w^{14} \cdot y^1 = \{0, w^{14}\}$
Initialize weights W^{23}, W^{34}
Submit MI-values $w^{14}y^1$ to Layer two and compute $Y^3 = f(W^{23}Y^2)$ $Y^4 = f(W^{34}Y^3)$
Compute cycle error $E \leftarrow E + \frac{1}{2}\|D - Y^4\|^2$
Compute error signal δ^3, δ^4
Adjust weights of semantic layer $W^{34} \leftarrow W^{34} + \eta\delta^4 Y^3$
Adjust weights of hidden layer $W^{23} \leftarrow W^{23} + \eta\delta^3 Y^2$
N
$E < E_{max}$?
Y
STOP

Fig. 6. Block diagram of the supervised learning scheme of the AFCAN.

B. Fuzzy Backpropagation Learning

The proposed fuzzy backpropagation (FBP) learning rule serves as the supervised learning algorithm of the FCLO subnetwork in the AFCAN (see Fig. 4). The FBP learning rule is derived by generalizing the traditional (crisp) backpropagation rule to its fuzzy counterpart [20]–[24]. The flowchart of this learning algorithm is shown in the fuzzy supervised learning part of Fig. 6. For the FBP learning, the weights to be updated include those between layers two and three (such as the weights $W_{j1}^{23}, \cdots, W_{ji}^{23}, \cdots, W_{jN_2}^{23}$ shown in Fig. 4) and those between layers three and four (such as the weights $W_{k1}^{34}, \cdots, W_{kj}^{34}, \cdots, W_{kN_h}^{34}$, shown in Fig. 4) of the AFCAN. The training input data of the FCLO are the mutual information values calculated from the multiplication of the outputs of layer one and the associated connection weights between layer one and layer four (i.e., the net inputs to layer-four nodes of the AFCAN). The desired output of the FCLO is the linguistic information associated with the selected semantic action by the CCNO. The linguistic information is represented by α-level sets. We denote the α-level sets of the current fuzzy output Y and the desired fuzzy output D as

$$Y = \bigcup_{\alpha} \alpha[y_L^{(\alpha)}, y_U^{(\alpha)}], \quad D = \bigcup_{\alpha} \alpha[d_L^{(\alpha)}, d_U^{(\alpha)}] \tag{14}$$

where $[\cdot]_L^{(\alpha)}$ and $[\cdot]_U^{(\alpha)}$ denote the lower and the upper limit of an α-level set, respectively. We train the FCLO with several values of α using the following error function:

$$\begin{aligned} e &= \text{diff}(Y, D) \\ &\equiv \tfrac{1}{2} \sum_\alpha \{[y_L^{(\alpha)} - d_L^{(\alpha)}]^2 + [y_U^{(\alpha)} - d_U^{(\alpha)}]^2\}. \end{aligned} \tag{15}$$

Assume that

$$W = \bigcup_\alpha \alpha[w_L^{(\alpha)}, w_U^{(\alpha)}] \tag{16}$$

is the adjustable fuzzy parameter in the FCLO subnetwork. To update fuzzy weights means to update the parameter $w_L^{(\alpha)}$ and $w_U^{(\alpha)}$. We shall next derive the update rules for these parameters layer by layer based on the general learning rule

$$w(t+1) = w(t) + \eta\left(-\frac{\partial e}{\partial w}\right) \tag{17}$$

where w represents $w_L^{(\alpha)}$ or $w_U^{(\alpha)}$ and η is the learning rate.

Layer 4: The output of a layer-four node is a fuzzy number $[Y_k^4]^{(\alpha)} = \bigcup_\alpha \alpha\{[y_k^4]_L^{(\alpha)}, [y_k^4]_U^{(\alpha)}\}$. The update rules of $[W_{kj}^{34}]^{(\alpha)} = \bigcup_\alpha \alpha\{[w_{kj}^{34}]_L^{(\alpha)}, [w_{kj}^{34}]_U^{(\alpha)}\}$ are derived as follows [see (7) and (8)] where only the derivation of $[w_{kj}^{34}]_L^{(\alpha)}$ is shown:

$$\begin{aligned} \frac{\partial e}{\partial [w_{kj}^{34}]_L^{(\alpha)}} &= \frac{\partial e}{\partial [y_k^4]_L^{(\alpha)}} \cdot \frac{\partial [y_k^4]_L^{(\alpha)}}{\partial [\text{Net}_k^4]_L^{(\alpha)}} \cdot \frac{\partial [\text{Net}_k^4]_L^{(\alpha)}}{\partial [w_{kj}^{34}]_L^{(\alpha)}} \\ &= \{[y_k^4]_L^{(\alpha)} - d_L^{(\alpha)}\}[y_k^4]_L^{(\alpha)}\{1 - [y_k^4]_L^{(\alpha)}\} \\ &\quad \cdot [y_j^3]_L^{(\alpha)}. \end{aligned} \tag{18}$$

The error term δ produced by the kth node in this layer is defined by

$$\begin{aligned} [\delta_k^4]_L^{(\alpha)} &= -\frac{\partial e}{\partial [\text{Net}_k^4]_L^{(\alpha)}} = -\frac{\partial e}{\partial [y_k^4]_L^{(\alpha)}} \frac{\partial [y_k^4]_L^{(\alpha)}}{\partial [\text{Net}_k^4]_L^{(\alpha)}} \\ &= -\{[y_k^4]_L^{(\alpha)} - d_L^{(\alpha)}\}[y_k^4]_L^{(\alpha)}\{1 - [y_k^4]_L^{(\alpha)}\} \end{aligned} \tag{19}$$

where

$$[y_k^4]_L^{(\alpha)} = f\{[\text{Net}_k^4]_L^{(\alpha)}\} = \frac{1}{1 + \exp^{-[\text{Net}_k^4]_L^{(\alpha)}}}. \tag{20}$$

Then, from (17) and by performing similar derivation for $[w_{kj}^{34}]_U^{(\alpha)}$, we have

$$[w_{kj}^{34}]_L^{(\alpha)}(t+1) = [w_{kj}^{34}]_L^{(\alpha)}(t) + \eta[\delta_k^4]_L^{(\alpha)}[y_j^3]_L^{(\alpha)} \tag{21}$$

$$[w_{kj}^{34}]_U^{(\alpha)}(t+1) = [w_{kj}^{34}]_U^{(\alpha)}(t) + \eta[\delta_k^4]_U^{(\alpha)}[y_j^3]_U^{(\alpha)} \tag{22}$$

where $[\delta_k^4]_U^{(\alpha)}$ is in the same form of (19) except that the subscript L is replaced by U.

Layer 3: In this layer, each node output is also a fuzzy number $[Y_k^3]^{(\alpha)} = \bigcup_\alpha \alpha\{[y_k^3]_L^{(\alpha)}, [y_k^3]_U^{(\alpha)}\}$. The update rules of $[W_{ji}^{23}]^{(\alpha)} = \bigcup_\alpha \alpha\{[w_{ji}^{23}]_L^{(\alpha)}, [w_{ji}^{23}]_U^{(\alpha)}\}$ can be derived using (4), (5), and (19) as follows, where only the derivation of $[w_{ji}^{23}]_L^{(\alpha)}$ is shown

$$\begin{aligned} \frac{\partial e}{\partial [w_{ji}^{23}]_L^{(\alpha)}} &= \frac{\partial e}{\partial [y_k^4]_L^{(\alpha)}} \cdot \frac{\partial [y_k^4]_L^{(\alpha)}}{\partial [\text{Net}_k^4]_L^{(\alpha)}} \cdot \frac{\partial [\text{Net}_k^4]_L^{(\alpha)}}{\partial [y_j^3]_L^{(\alpha)}} \\ &\quad \cdot \frac{\partial [y_j^3]_L^{(\alpha)}}{\partial [\text{Net}_j^3]_L^{(\alpha)}} \cdot \frac{\partial [\text{Net}_j^3]_L^{(\alpha)}}{\partial [w_{ji}^{23}]_L^{(\alpha)}} \\ &= -[\delta_k^4]_L^{(\alpha)}[w_{kj}^{34}]_L^{(\alpha)}[y_j^3]_L^{(\alpha)}\{1 - [y_j^3]_L^{(\alpha)}\} \\ &\quad \cdot [y_i^2]_L^{(\alpha)}. \end{aligned} \tag{23}$$

The error signal term δ produced by the jth node for this layer is defined by

$$\begin{aligned} [\delta_j^3]_L^{(\alpha)} &= -\frac{\partial e}{\partial [\text{Net}_j^3]_L^{(\alpha)}} = -\frac{\partial e}{\partial [y_j^3]_L^{(\alpha)}} \frac{\partial [y_j^3]_L^{(\alpha)}}{\partial [\text{Net}_j^3]_L^{(\alpha)}} \\ &= [y_j^3]_L^{(\alpha)}\{1 - [y_j^3]_L^{(\alpha)}\} \sum_{k=1}^{K} [\delta_k^4]_L^{(\alpha)}[w_{kj}^{34}]_L^{(\alpha)} \end{aligned} \tag{24}$$

where

$$[y_j^3]_L^{(\alpha)} = f\{[\text{Net}_j^3]_L^{(\alpha)}\} = \frac{1}{1 + \exp^{-[\text{Net}_j^3]_L^{(\alpha)}}}. \tag{25}$$

Then, from (17) and by performing similar derivation for $[w_{ji}^{23}]_U^{(\alpha)}$, we have

$$[w_{ji}^{23}]_L^{(\alpha)}(t+1) = [w_{ji}^{23}]_L^{(\alpha)}(t) + \eta[\delta_j^3]_L^{(\alpha)}[y_i^2]_L^{(\alpha)} \tag{26}$$

$$[w_{ji}^{23}]_U^{(\alpha)}(t+1) = [w_{ji}^{23}]_U^{(\alpha)}(t) + \eta[\delta_j^3]_U^{(\alpha)}[y_i^2]_U^{(\alpha)} \tag{27}$$

where $[\delta_j^3]_U^{(\alpha)}$ is in the same form of (24) except that the subscript L is replaced by U.

When fuzzy weights are adjusted by (18)–(27), an undesirable situation may occur. That is, the lower limits of the α-level sets of fuzzy weights may exceed the upper limits, and the updated fuzzy weights may thus become nonconvex. In order to deal with this situation, necessary modifications on the updated fuzzy weights to make sure that they are legal fuzzy numbers after updating are performed. This process is described as follows.

Procedure—Fuzzy Number Restoration:

Inputs: Fuzzy weights $W = \bigcup_\alpha \alpha[w_L^{(\alpha)}, w_U^{(\alpha)}]$ standing for W_{kj}^{34} and W_{ji}^{23}, which are updated by (18)–(27).

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

Step 1. $k = 1$, if $w_L^{(k)} > w_U^{(k)}$, then $\hat{w}_L^{(k)} = w_U^{(k)}$ and $\hat{w}_U^{(k)} = w_L^{(k)}$, else $\hat{w}_L^{(k)} = w_L^{(k)}$ and $\hat{w}_U^{(k)} = w_U^{(k)}$.

Step 2. For $k = h - 1$ down to zero, do if $w_L^{(k/h)} > \hat{w}_L^{[(k+1)/h]}$, then $\hat{w}_L^{(k/h)} = \hat{w}_L^{[(k+1)/h]}$, else $\hat{w}_L^{(k/h)} = w_L^{(k/h)}$ and if $w_U^{(k/h)} < \hat{w}_U^{[(k+1)/h]}$, then $\hat{w}_U^{(k/h)} = \hat{w}_U^{[(k+1)/h]}$, else $\hat{w}_U^{(k/h)} = w_U^{(k/h)}$.

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

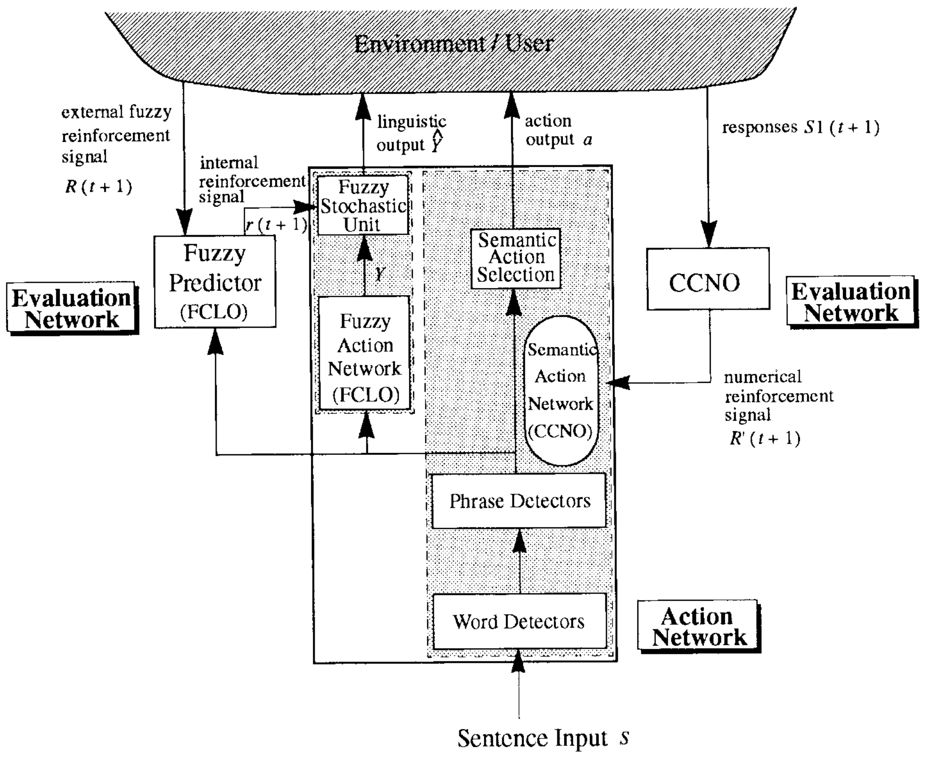


Fig. 7. The proposed RAFCAN system.

IV. Reinforcement Learning of the AFCAN

In this section, we shall derive reinforcement learning algorithms for the on-line learning of the AFCAN. Since the outputs of the AFCAN have two forms—numerical numbers and fuzzy numbers—standing for semantic actions and linguistic information, respectively, the proposed reinforcement learning scheme has two learning phases. In phase one, an MI-reinforcement learning algorithm is employed to tune the CCNO with the numerical reinforcement signal (numerical evaluative feedback). In phase two, a fuzzy reinforcement learning algorithm is utilized to tune the FCLO with the fuzzy reinforcement signal (fuzzy evaluative feedback). To solve the reinforcement learning problems, we need a more powerful network structure. We shall first set up such structure based on the original structure of the AFCAN in Fig. 2 in the following subsection and then develop a reinforcement learning scheme for the AFCAN in Sections IV-B and C.

A. Structure of the Reinforcement AFCAN

For reinforcement learning problems, almost all existing learning methods of neural networks focus their attention on *numerical evaluative* information [25]–[31]. In this paper, we shall attack the *fuzzy reinforcement learning* problem where only fuzzy critic signal (e.g., "faster," "slower," "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.

In the reinforcement learning problems, it is common to think of a network functioning in an environment. The environment supplies the inputs to the network, receives its output, and then provides the reinforcement signal. 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*. Furthermore, 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. With this concept, we propose a reinforcement learning model as shown in Fig. 7, which integrates two previously proposed four-layered networks (see Fig. 2) into a learning system. The new system is called reinforcement AFCAN (RAFCAN). As in the original AFCAN, each CCNO used in the RAFCAN maps a sentence input to a desired action (presented as a numerical number), and each FCLO maps a crisp MI-value input obtained from the CCNO to a desired linguistic information (presented in the form of α-level sets).

The original four-layered network in Fig. 2 serves as the *action network*, which comprises *semantic action network* and *fuzzy action network*. The semantic action network is built by a CCNO in Fig. 3 and includes the word detectors, phrase detectors, and the semantic action selection layer. The fuzzy action network is realized by a FCLO in Fig. 4 and a newly added *fuzzy stochastic unit*. The semantic action network and fuzzy action network are used respectively to choose the proper semantic action and linguistic information with respect to the current input sentence. For solving the reinforcement learning problem, we use another CCNO to serve as the *eval-*

uation network of the semantic action network. This evaluation network is to perform the acquisition of the external user's or environment's responses on the selected semantic action and produce numerical reinforcement signals. More clearly, the CCNO can acquire the user's or environment's responses to the selected semantic action and encode the negative feedback or affirmative feedback to a two-valued number $R'(t) \in \{-\infty, 0\}$, so the reinforcement signal is numerical. Similarly, we use another FCLO to serve as the *evaluation network* of the fuzzy action network. This evaluation network is called *fuzzy predictor*, which performs multistep prediction of the external fuzzy reinforcement signal. The evaluation network provides the action network with more informative and beforehand internal reinforcement signals for learning. Because the reinforcement signal for the fuzzy action network is a fuzzy number, a FCLO is used as the fuzzy predictor.

Like the supervised learning scheme introduced in Section III, we need to do network initialization before the reinforcement learning proceeds. The initialization process is exactly the same as that for the supervised learning (see Section III). 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 subnetworks. Next, we shall derive the reinforcement learning algorithms in the following subsections.

B. MI-Reinforcement Learning

As shown in Fig. 7, the CCNO-based evaluation network performs the acquisition of the external environment's response to the action selected by the semantic action network. In this section, we shall attack the problem of the semantic-level error feedback acquired by this evaluation network and consider the reinforcement signal $R'(t)$ as a crisp number, zero, or $-\infty$. Assume that $R'(t)$ is the crisp signal available at time step t and caused by the input and semantic actions chosen at time step $t-1$.

In the MI-reinforcement learning, there may need several steps for a user to induce the RAFCAN to perform his/her desired action. At first, the user gives a command (sentence) to the RAFCAN to execute his/her desired action. Let us denote this command as $s(1)$ (e.g., "*Please walk forward at a very high speed.*"). The output of the semantic action network indicates the RAFCAN's understanding of the user's command. Let us denote the current output as $y(1)$ (e.g., assume the selected semantic action is "*walk backward.*"). The user then responses with a message in accordance with the appropriateness of the network's performance to his/her command (e.g., "*My command is* '*walk forward,*' *instead of* '*walk backward.*' "). The critic response should be represented as another more clarifying message. With such critic responses, the RAFCAN will tune itself using the following MI-reinforcement learning rule. This process will continue till the user give positive response (e.g., "*Yes!,*" "*OK!,*" "*It is right!*") and we denote this response as an affirmative reinforcement signal.

We use $R'[s(t)]$ to denote the MI-reinforcement signal at time step t; if $R'[s(t)] = 0$, it represents that the user give affirmative response; otherwise $R'[s(t)] = -\infty$ if the user gives negative response. The MI-reinforcement learning rule is given as follows:

$$\begin{cases} A_k(1) = a_k[s(1)] \\ A_k(t) = (t-\alpha)A_k(t-1) + \alpha a_k[s(t)] + R'[s(t)] \end{cases} \quad (28)$$

$$y(t) = \arg\max_k A_k(t) \quad (29)$$

where $A_k(\cdot)$ is the total activation value of semantic node k in layer four of the semantic action network, $a_k[s(t)]$ is the activation value of semantic node k for $s(t)$ [i.e., (6)], t is the time step, α is a gain parameter which is assumed to be $1/t$, $s(t)$ denotes the tth user input, $R'[s(t)]$ (where $t \geq 2$) is the reinforcement signal, which is zero or $-\infty$, and $y(t)$ denotes the network's selected action after the tth user input.

The learning process stops when the reinforcement signal $R'[s(t)] = 0$ and at that step, the correct action $y(t-1)$ is executed. At the same time, since the RAFCAN has known the user's desired action, it will apply the MI-supervised learning algorithm developed in Section III-A to record (learn) the new mapping: $[s(1), y(t-1)]$, $[s(2), y(t-1)]$, $\cdots$, $[s(t), y(t-1)]$. The above whole learning procedure is called the *MI-reinforcement learning algorithm*.

It is noted that when the MI-reinforcement learning proceeds, the word detector and phrase detector nodes in layer one become adaptive too. That is, if new words are observed and necessary for the fuzzy command acquisition, the RAFCAN will regard them as the new received reference words and will add corresponding new word detector and phrase detector nodes. We have designed a word filter for the RAFCAN in performing the MI-reinforcement learning. The function of the word filter is to dismiss some usual use words such as "*we, he, she, they, is, are.*" This kind of words is not so important in the fuzzy command acquisition task and, if we accept them, they will result in time-consuming learning and memory-consuming for implementation.

C. Fuzzy Reinforcement Learning

In this subsection, we shall attack the fuzzy reinforcement learning problems 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

$$R(t) \in \{R_1, R_2, \cdots, R_n\} \quad (30)$$

and satisfies the following inequality relation

$$\begin{aligned} -1 &\leq \mathit{defuzzifier}(R_1) \leq \mathit{defuzzifier}(R_2) \\ &\leq \cdots \leq \mathit{defuzzifier}(R_n) \leq 0 \end{aligned} \quad (31)$$

where $\mathit{defuzzifier}[R(t)] \equiv [R_L^{(1)}(t) + R_U^{(1)}(t)]/2$ represents discrete degree of reward or penalty, where $[R_L^{(1)}(t), R_U^{(1)}(t)]$ is the α cut of $R(t)$ at $\alpha = 1$. For example, we may have $R(t) \in$ {very slow, slow, fast, very fast} with each fuzzy term defined by a proper membership function. Exemplary

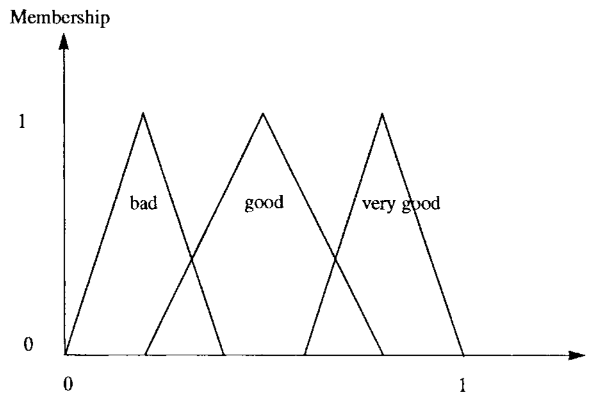


Fig. 8. The exemplary fuzzy reinforcement signals used in the RAFCAN.

fuzzy reinforcement signals used in the RAFCAN are shown in Fig. 8.

In the architecture of the proposed RAFCAN shown in Fig. 7, three key components take care of the fuzzy reinforcement learning problems: 1) the fuzzy action network maps a numerical vector (the MI values from the semantic-action network) into a linguistic information Y; 2) the evaluation network (fuzzy predictor) maps a numerical vector (the MI values) and an external fuzzy reinforcement signal into a predicted fuzzy reinforcement signal. This predicted signal is used to produce internal reinforcement signal for helping the learning of fuzzy action network; and 3) the fuzzy stochastic unit uses both Y and the predicted reinforcement signal from the fuzzy predictor to produce a fuzzy number $\hat{Y}$, which is sent out to the environment. We shall next describe the function of the stochastic unit and the reinforcement leaning of fuzzy predictor and fuzzy action network, respectively, in the following.

Fuzzy Stochastic Unit: In the fuzzy reinforcement learning mode, the output error gradient information of the fuzzy action network is not told, so it needs to be estimated. To estimate the gradient information, the output Y of the fuzzy action network is not directly sent out to the environment. Instead, the stochastic unit uses the predicted fuzzy reinforcement signal $P(t)$ from the fuzzy predictor and the fuzzy information Y recommended by the fuzzy action network to stochastically generate an actual fuzzy information $\hat{Y}$ sent out to the environment. The actual fuzzy information $\hat{Y}$ is a random fuzzy variable with fuzzy mean 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

$$\sigma(t) = \frac{2k}{1+e^{\lambda p(t)}} - k \tag{32}$$

where $p(t) = \textit{defuzzifier}[P(t)]$, λ is a search-range scaling constant which can be simply set to one, and $P(t)$ is the predicted fuzzy reinforcement signal used to predict $R(t+1)$ for the network input at time t. The magnitude of $\sigma(t)$ is large when $p(t)$ is small. Because we restrict the highest degree of reward to $p(t)=0$, the value of $\sigma(t)$ is zero when $p(t)=0$. The stochastic perturbation in the suggested approach leads to a better exploration of output space and better generalization ability.

Once the amount of exploration $\sigma(t)$ has been decided, the next problem is to generate the actual fuzzy output. Since the output is a fuzzy number $Y = \bigcup_\alpha \alpha[y_L^{(\alpha)}, y_U^{(\alpha)}]$, the fuzzy stochastic unit generates a fuzzy output $\hat{Y} = \bigcup_\alpha \alpha[\hat{y_L}^{(\alpha)}, \hat{y_U}^{(\alpha)}]$ based on the amount of exploration $\sigma(t)$. The parameter $\hat{y_L}^{(\alpha)}$ $[\hat{y_U}^{(\alpha)}]$ is set as a uniform random variable with mean $y_L^{(\alpha)}$ $[y_U^{(\alpha)}]$ and width $2\sigma(t)$. After the parameters $\hat{y_L}^{(\alpha)}$ and $\hat{y_U}^{(\alpha)}$ are decided, we must then maintain the convex property of the fuzzy output. We propose the following procedure to complete the fuzzy stochastic exploration. In this procedure, the notation h is the number of quantized membership grade.

Procedure. Fuzzy Stochastic Exploration:

Input: $Y = \bigcup_\alpha \alpha[y_L^{(\alpha)}, y_U^{(\alpha)}]$.

Output: $\hat{Y} = \bigcup_\alpha \alpha[\hat{y_L}^{(\alpha)}, \hat{y_U}^{(\alpha)}]$.

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

$$[y_L^{(k)} - \sigma(t)] \le \hat{y_L}^{(k)} \le [y_L^{(k)} + \sigma(t)]$$

and then find $\hat{y_U}^{(k)}$ randomly such that

$$\max[y_U^{(k)} - \sigma(t), \hat{y_U}^{(k)}] \le \hat{y_U}^{(k)} \le [y_U^{(k)} + \sigma(t)].$$

Step 2. For $k=h-1$ down to zero, find $\hat{y_L}^{(k/h)}$ randomly such that

$$[y_L^{(k/h)} - \sigma(t)] \le \hat{y_L}^{(k/h)} \le \min[\hat{y_L}^{(k+1/h)} \cdot y_L^{(k/h)} + \sigma(t)]$$

and find $\hat{y_U}^{(k/h)}$ randomly such that

$$\max[\hat{y_U}^{(k+1/h)}, y_U^{(k/h)} - \sigma(t)] \le \hat{y_U}^{(k/h)} \le [y_U^{(k/h)} + \sigma(t)].$$

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

Fuzzy Reinforcement Learning of Fuzzy Predictor: In a reinforcement learning environment, the learning system usually receives evaluation of its behavior only after a long sequence of outputs, called *delayed reinforcement*. We shall now discuss how the problem of learning with delayed reinforcement can be solved using the 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 output in a sequence for an eventual success or failure. The solution to the temporal credit assignment problem is to design a multistep fuzzy predictor that 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 [27]. 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 generalize the TD method to its fuzzy counterpart for training the fuzzy predictor in the RAFCAN. We shall discuss three different cases of reinforcement learning problems below.

Case 1—Prediction of Final Fuzzy Outcome: Assume we are given the numerical input sequences of the form $\mathbf{x}(1), \mathbf{x}(2), \cdots, \mathbf{x}(m)$ where each $\mathbf{x}(t)$ is an input vector of 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 a fuzzy number and an estimate of $R(m+1)$. Then based on the TD(λ) procedure in [27], the update rule for the fuzzy weights $W = \bigcup_\alpha \alpha[w_L^{(\alpha)}, w_U^{(\alpha)}]$ in the fuzzy predictor can be derived as

$$\frac{\partial e}{\partial w_L^{(\alpha)}(t)} = -[p_L^{(\alpha)}(t+1) - p_L^{(\alpha)}(t)] \sum_{k=1}^{t} \lambda^{t-k} \frac{\partial p_L^{(\alpha)}(k)}{\partial w_L^{(\alpha)}(k)} \tag{33}$$

$$\frac{\partial e}{\partial w_U^{(\alpha)}(t)} = -[p_U^{(\alpha)}(t+1) - p_U^{(\alpha)}(t)] \sum_{k=1}^{t} \lambda^{t-k} \frac{\partial p_U^{(\alpha)}(k)}{\partial w_U^{(\alpha)}(k)} \tag{34}$$

in which alterations to the predictions of input vectors occurring k steps in the past are weighted according to λ^{t-k} for $0 \leq \lambda \leq 1$.

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 $R(t+1) = \bigcup_\alpha \alpha[r_L^{(\alpha)}(t+1), r_U^{(\alpha)}(t+1)]$ 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 $Z(t) = \bigcup_\alpha \alpha[z_L^{(\alpha)}(t), z_U^{(\alpha)}(t)] = \sum_{k=t}^{m} R(k+1)$. Hence, we have

$$z_L^{(\alpha)}(t) = \sum_{k=t}^{m} r_L^{(\alpha)}(k+1), \quad z_U^{(\alpha)}(t) = \sum_{k=t}^{m} r_U^{(\alpha)}(k+1). \tag{35}$$

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

$$\begin{aligned} z_L^{(\alpha)}(t) - p_L^{(\alpha)}(t) &= \sum_{k=t}^{m} r_L^{(\alpha)}(k+1) - p_L^{(\alpha)}(t) \\ &= \sum_{k=t}^{m} [r_L^{(\alpha)}(k) + p_L^{(\alpha)}(k+1) - p_L^{(\alpha)}(k)] \end{aligned} \tag{36}$$

$$\begin{aligned} z_U^{(\alpha)}(t) - p_U^{(\alpha)}(t) &= \sum_{k=t}^{m} r_U^{(\alpha)}(k+1) - p_U^{(\alpha)}(t) \\ &= \sum_{k=t}^{m} [r_U^{(\alpha)}(k) + p_U^{(\alpha)}(k+1) - p_U^{(\alpha)}(k)] \end{aligned} \tag{37}$$

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

$$\frac{\partial e}{\partial w_L^{(\alpha)}(t)} = -[r_L^{(\alpha)}(t+1) + p_L^{(\alpha)}(t+1) - p_L^{(\alpha)}(t)] \cdot \sum_{k=1}^{t} \lambda^{t-k} \frac{\partial p_L^{(\alpha)}(k)}{\partial w_L^{(\alpha)}(t)} \tag{38}$$

$$\frac{\partial e}{\partial w_U^{(\alpha)}(t)} = -[r_U^{(\alpha)}(t+1) + p_U^{(\alpha)}(t+1) - p_U^{(\alpha)}(t)] \cdot \sum_{k=1}^{t} \lambda^{t-k} \frac{\partial p_U^{(\alpha)}(k)}{\partial w_U^{(\alpha)}(t)}. \tag{39}$$

Case 3—Prediction of Infinite Discounted Cumulative Fuzzy Outcomes: In this case, $P(t)$ predict the discounted sum: $Z(t) = \sum_{k=0}^{\infty} \gamma^k R(t+k+1)$, i.e., $z_L^{(\alpha)}(t) = \sum_{k=0}^{\infty} \gamma_k r_L^{(\alpha)}(t+k+1)$, $z_U^{(\alpha)}(t) = \sum_{k=0}^{\infty} \gamma_k r_U^{(\alpha)}(t+k+1)$, where γ, $0 \leq \gamma \leq 1$, is the discounted rate parameter. If the prediction is accurate, we can write

$$\begin{aligned} p_L^{(\alpha)}(t) &= \sum_{k=0}^{\infty} \gamma^k r_L^{(\alpha)}(t+k+1) \\ &= r_L^{(\alpha)}(t+1) + \gamma \sum_{k=0}^{\infty} \gamma^k r_L^{(\alpha)}(t+k+2) \\ &= r_L^{(\alpha)}(t+1) + \gamma p_L^{(\alpha)}(t+1) \end{aligned} \tag{40}$$

and

$$p_U^{(\alpha)}(t) = r_U^{(\alpha)}(t+1) + \gamma p_U^{(\alpha)}(t+1). \tag{41}$$

The mismatch or TD error is the difference between the right-hand and left-hand sides of these equations $r_L^{(\alpha)}(t+1) + \gamma p_L^{(\alpha)}(t+1)$, $r_U^{(\alpha)}(t+1) + \gamma p_U^{(\alpha)}(t+1)$ and, thus, the update rules are

$$\frac{\partial e}{\partial w_L^{(\alpha)}(t)} = -[r_L^{(\alpha)}(t+1) + \gamma p_L^{(\alpha)}(t+1) - p_L^{(\alpha)}(t)] \cdot \sum_{k=1}^{t} \lambda^{t-k} \frac{\partial p_L^{(\alpha)}(k)}{\partial w_L^{(\alpha)}(k)} \tag{42}$$

$$\frac{\partial e}{\partial w_U^{(\alpha)}(t)} = -[r_U^{(\alpha)}(t+1) + \gamma p_U^{(\alpha)}(t+1) - p_U^{(\alpha)}(t)] \cdot \sum_{k=1}^{t} \lambda^{t-k} \frac{\partial p_U^{(\alpha)}(k)}{\partial w_U^{(\alpha)}(k)}. \tag{43}$$

Once the output-error gradient information of the fuzzy predictor is obtained using the methods discussed in the above three cases, its learning becomes a supervised learning problem. Hence, (18)–(27) can be used here directly to train the fuzzy predictor if we replace the associated gradient terms in those equations by the gradient terms derived in the above three cases properly. Notice 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, $R(t) = \bigcup_\alpha \alpha[0, 0]$, for $2 \leq t \leq m$.

TABLE I
THE ACTIONS AND FUZZY TERMS USED IN THE ILLUSTRATED VOICE CONTROL SYSTEM

Action No.	1	2	3	4	5	6	7	8
Action	flail, knap, strike, etc.	hold, catch, seize, etc.	turn, whirl, rotate, etc.	leap, jump, bound, etc	move forward, shift forward, pass forward, etc	move backward, shift backward, pass backward, etc	move right, shift right, pass right, etc	move left, shift right, pass left, etc
Fuzzy term	very heavy, highly heavy, slightly heavy, heavy	very tight, tight, loose, very loose	very fast, fast slow, very slow	very high, high low, very low	very fast, fast, slow, very slow	very fast, fast, slow, very slow	very fast, fast, slow, very slow	very fast, fast, slow, very slow

Fuzzy Reinforcement Learning of Fuzzy Action Network: We next develop a reinforcement learning algorithm for the fuzzy action network. The goal of the reinforcement learning is to adjust the parameters $w_L^{(\alpha)}$ and $w_U^{(\alpha)}$ of the fuzzy action network such that the fuzzy reinforcement signal R is maximum; that is

$$\Delta w \propto \frac{\partial r}{\partial w} \tag{44}$$

where $w = w_L^{(\alpha)}$ or $w_U^{(\alpha)}$ and $r = \textit{defuzzifier}(R)$.

We first derive the reinforcement learning algorithm for the *crisp* action network with numerical output instead of fuzzy output. This will help us to derive the reinforcement learning algorithm for the fuzzy action network. To determine $\partial r/\partial w$, we need to know $\partial r/\partial y$ where y is the output of the crisp 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. According to [26], the gradient information for the crisp action network can be estimated by

$$\begin{aligned} -\frac{\partial e}{\partial y} \propto \frac{\partial r}{\partial y} &= [r(t+1) + \gamma p(t+1) - p(t)][\hat{y}(t) - y(t)] \\ &= \hat{r}(t+1)[\hat{y}(t) - y(t)] \end{aligned} \tag{45}$$

where

$$\hat{r}(t+1) \equiv r(t+1) + \gamma p(t+1) - p(t). \tag{46}$$

When the output is linguistic (for fuzzy action network), we can generalize the above update rule to its fuzzy counterpart and obtain

$$-\frac{\partial e}{\partial y_L^{(\alpha)}} \propto \frac{\partial r}{\partial y_L^{(\alpha)}} = \hat{r}(t+1)[\hat{y_L}^{(\alpha)}(t) - y_L^{(\alpha)}(t)] \tag{47}$$

$$-\frac{\partial e}{\partial y_U^{(\alpha)}} \propto \frac{\partial r}{\partial y_U^{(\alpha)}} = \hat{r}(t+1)[\hat{y_U}^{(\alpha)}(t) - y_U^{(\alpha)}(t)]. \tag{48}$$

Again, with the error gradient information available, the fuzzy supervised learning algorithm developed in Section III-B can be applied here directly to train the fuzzy action network.

In the proposed system, the fuzzy action network and the fuzzy predictor are trained together. However, since the fuzzy action network relies on accurate prediction of the fuzzy predictor, 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 fuzzy action network.

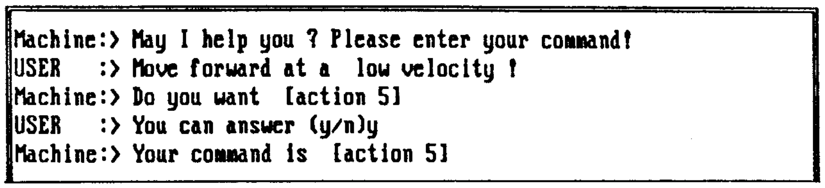
V. AN ILLUSTRATIVE EXAMPLE—FUZZY COMMAND ACQUISITION OF A VOICE CONTROL SYSTEM

In this section, we shall establish a system based on the proposed RAFCAN that can acquire fuzzy commands given by users in voice or typed input form. The system can acquire only one semantic action at a time, so if it acquires several semantic actions at the same time, it will list them along with their uncertainty factors, and the user should do a judgement (maybe a positive answer or negative answer) from the listed actions. The actions and associated linguistic information (fuzzy predicates) that this system can acquire are listed in Table I. After a command is acquired, the system will show the selected action and linguistic information in the form of α-level sets. We can make use of such output information to do the *fuzzy control* task directly.

The voice control system uses the architecture of the RAFCAN in Fig. 7. Initially, we set up the detector nodes in layer one of the AFCAN according to the given reference words and put random weights in the FCLO. The initial AFCAN has 41 word detector nodes (layer one), 1640 phrase detector nodes (layer one), 1681 MI-value nodes (layer two), 10 hidden nodes (layer three), and 8 semantic nodes (layer four). The 41 reference words for the word detector nodes are listed in Table II. We also design a word filter containing 36 words such as "the," "is," "are," "you," "mine," "hers," etc. We train the system using the off-line learning scheme developed in Section III on some input-output training pairs [⟨*sentence, fuzzy action*⟩] where the fuzzy action is represented by an action number (1–8) and a α-level set ($h = 0.2, 0.4, 0.6, 0.8, 1.0$). The whole off-line training data can be found in [32]. When the system is set up and in use, the on-line learning scheme developed in Section IV is performed all the time. We next do some simu-

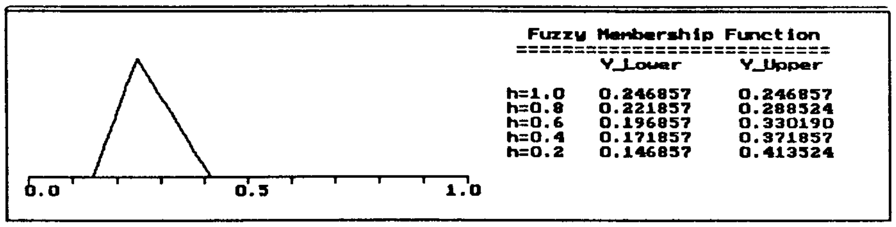
TABLE II
THE REFERENCE WORDS FOR THE WORD DETECTOR NODES IN THE ILLUSTRATED VOICE CONTROL SYSTEM

node 1	node 2	node 3	node 4	node 5	node 6	node 7	node 8
flail	heavy	light	knap	hit	strike	knock	hold
node 9	node 10	node 11	node 12	node 13	node 14	node 15	node 16
catch	gab	grasp	seize	tight	lux	loose	turn
node 17	node 18	node 19	node 20	node 21	node 22	node 23	node 24
wheel	rotate	whirl	speed	low	high	fast	slow
node 25	node 26	node 27	node 28	node 29	node 30	node 31	node 32
leap	jump	hop	skip	bound	transfer	pass	move
node 33	node 34	node 35	node 36	node 37	node 38	node 39	node 40
ambulate	shift	forward	backward	right	left	very	highly
node 41							
slightly							



```
Machine:> May I help you ? Please enter your command!
USER    :> Move forward at a  low velocity !
Machine:> Do you want  [action 5]
USER    :> You can answer (y/n)y
Machine:> Your command is  [action 5]
```

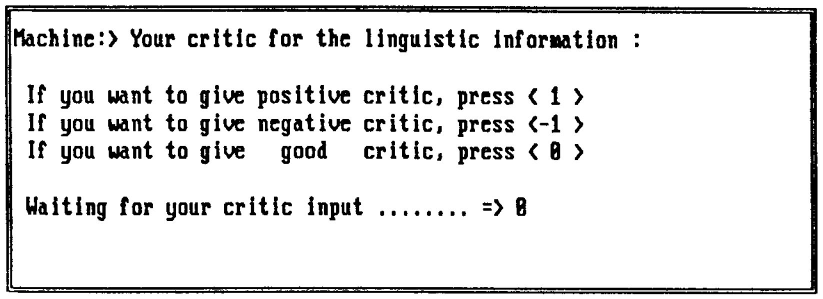
(a)



```
Fuzzy Membership Function
==========================
        Y_Lower     Y_Upper

h=1.0   0.246857    0.246857
h=0.8   0.221857    0.288524
h=0.6   0.196857    0.330190
h=0.4   0.171857    0.371857
h=0.2   0.146857    0.413524
```

(b)



```
Machine:> Your critic for the linguistic information :

 If you want to give positive critic, press < 1 >
 If you want to give negative critic, press <-1 >
 If you want to give  good   critic, press < 0 >

 Waiting for your critic input ........ => 0
```

(c)

Fig. 9. Screen copy of the voice control system—the first case.

lations to illustrate the power and specialist of the command acquisition system. We illustrate four cases in the following.

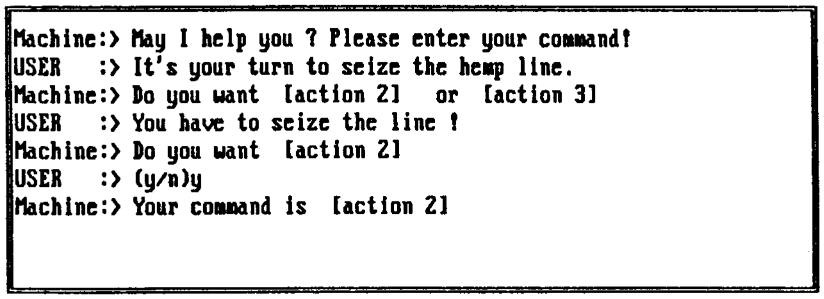
The First Case: This case is shown in Fig. 9, which is a screen copy of the system interface. According to Fig. 9(a), the system acquires the command correctly without further iteration after the user gives the command. Hence, the user replies "(y)es" as the next input to accept the selected action of the system. After the system recognizes the user's desired action, it then continues to acquire the linguistic information as shown in Fig. 9(b). The user then makes a critic on the shown membership function where we use "1" to stand for positive critic, "-1" for negative critic, and "0" for good critic (i.e., agree the system's output) [Fig. 9(c)]. Each of these three critic signals corresponds to one proper fuzzy term whose membership function is similar to that shown in Fig. 8. In the current case, the linguistic information matches the user's desire, so he/she chooses the "0" input.

Some other sentences that can be acquired correctly by the system are as follows.

"I have no more time. Please move forward in a very high speed!"

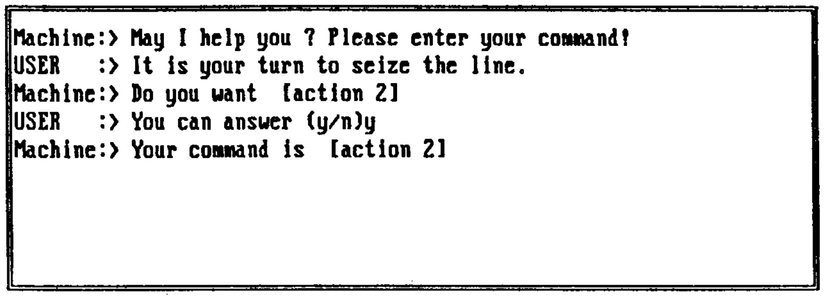
"My mother will come to my graduation. Would you please move forward at a very high velocity?"

"This vase is easily broken. You have to hold it very tightly."



```
Machine:> May I help you ? Please enter your command!
USER   :> It's your turn to seize the hemp line.
Machine:> Do you want  [action 2]   or  [action 3]
USER   :> You have to seize the line !
Machine:> Do you want  [action 2]
USER   :> (y/n)y
Machine:> Your command is  [action 2]
```

(a)



```
Machine:> May I help you ? Please enter your command!
USER   :> It is your turn to seize the line.
Machine:> Do you want  [action 2]
USER   :> You can answer (y/n)y
Machine:> Your command is  [action 2]
```

(b)

Fig. 10. Screen copy of the voice control system—the second case.

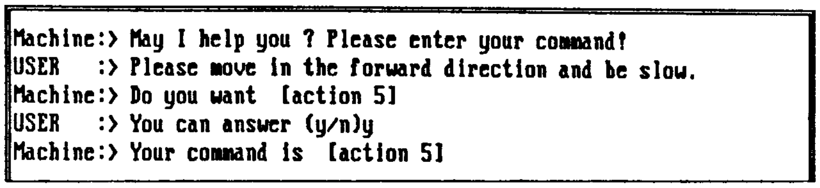
"The ground is sliding. You should whirl in a very low speed."

The Second Case: This case, as shown in Fig. 10(a), illustrates that the system cannot catch the user's intention exactly, so it shows all the promising actions that it acquires. After the user gives another command containing clarifying information, the system reduces its initial uncertainty and appropriately recognize the command. This case shows the power of the system's on-line learning ability using the MI-reinforcement learning algorithm. As shown in Fig. 10(b), the system can acquire the user's meaning correctly after it receives the second command via on-line learning. Another example belonging to this case is shown in the following:

```
Machine : May I help you? Please enter
            your command!
 User   : Seize the pinwheel very tightly
            opposing wind and let it whirl
            very fast.
Machine : Do you want [action 2] or
            [action 3]?
 User   : I mean to hold the pinwheel very
            tightly when opposing the wind!
Machine : Do you want [action 2]?
 User   : You can answer (y/n) y
Machine : Your command is [action 2].
```

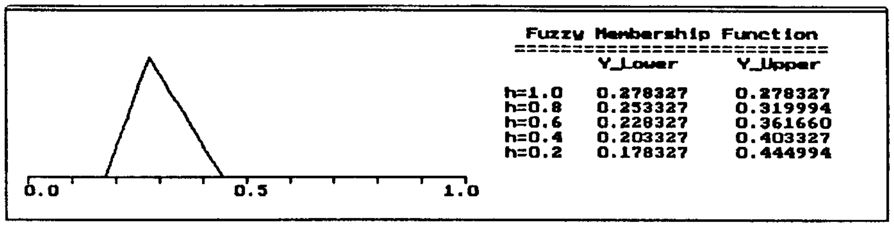
The Third Case: In this case (see Fig. 11), we aim at the on-line learning of the acquired linguistic information. After the action is acquired correctly, the system will show the membership function of the acquired linguistic information. The user then has three kinds of critic signals for use to express his/her judgement on the acquired linguistic information: *positive* (1), *negative* (−1) and *good* (0). In the current case, the user feel the linguistic information in Fig. 11(b) is "*too slow*," so he/she gives a negative critic (−1) [see Fig. 11(c)]. At this time, the system will perform on-line learning according to the user's critic [Fig. 11(d)]. If the user changes the mind and has different thinking on the linguistic information, he/she can again continue to give critics to the system. It is noted that the user can give the critics any time during the fuzzy reinforcement learning since we perform multistep prediction in the fuzzy predictor of the RAFCAN.

The Fourth Case: In the above cases, all the input commands can be acquired to some extend, since the input words and phrases have existed in the initial detector nodes. In the fourth case, we shall illustrate that the system can learn new words (phrases) and their semantic associations from interactive command inputs. From Fig. 12(a), we observe that the command, "*Go ahead very fast*," is not understood by the system because the words "*go*" and "*ahead*" are not included in the reference words originally (see Table II). Hence, the system replies with, "*The system cannot recognize*," and ask for the user to enter the command in other expression. When the user gives the second command, "*I mean to go forward very fast*," the system understands this command, and response with, "*Do you want [action 5].*" Since action 5 is the user's desired action, the user presses the "*y*" key. When the above process is finished, the system will add new detector nodes. Before doing this, the new words should pass the word filter to dismiss some usual use words (e.g., *he, she, and*). After adds new detector nodes, the system learns the weights (MI weights and fuzzy weights) of the new network. When the relearning process is finished, we give the original command to test the effect of the processes of new word adding and weight relearning. Fig. 12(b) shows that the system after learning can acquire correctly the command, "*Go ahead very fast.*" Fig. 12(c) shows the acquired linguistic information of this command.



```
Machine:> May I help you ? Please enter your command!
USER    :> Please move in the forward direction and be slow.
Machine:> Do you want  [action 5]
USER    :> You can answer (y/n)y
Machine:> Your command is  [action 5]
```

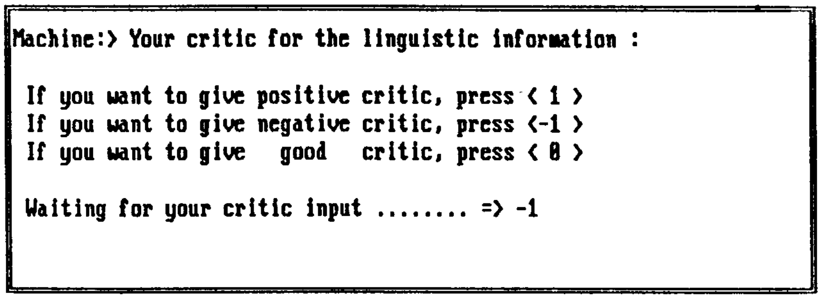
(a)



```
             Fuzzy Membership Function
             ==========================
              Y_Lower        Y_Upper

    h=1.0    0.278327       0.278327
    h=0.8    0.253327       0.319994
    h=0.6    0.228327       0.361660
    h=0.4    0.203327       0.403327
    h=0.2    0.178327       0.444994
```

(b)

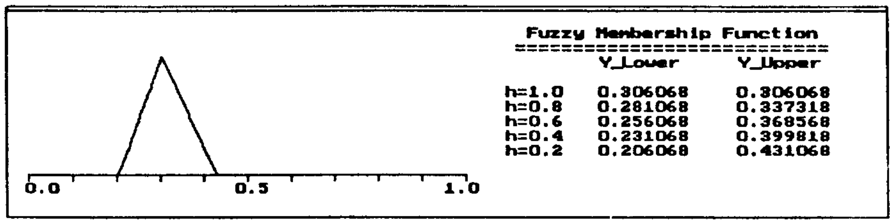


```
Machine:> Your critic for the linguistic information :

 If you want to give positive critic, press < 1 >
 If you want to give negative critic, press <-1 >
 If you want to give   good   critic, press < 0 >

 Waiting for your critic input ........ => -1
```

(c)



```
             Fuzzy Membership Function
             ==========================
              Y_Lower        Y_Upper

    h=1.0    0.306068       0.306068
    h=0.8    0.281068       0.337318
    h=0.6    0.256068       0.368568
    h=0.4    0.231068       0.399818
    h=0.2    0.206068       0.431068
```

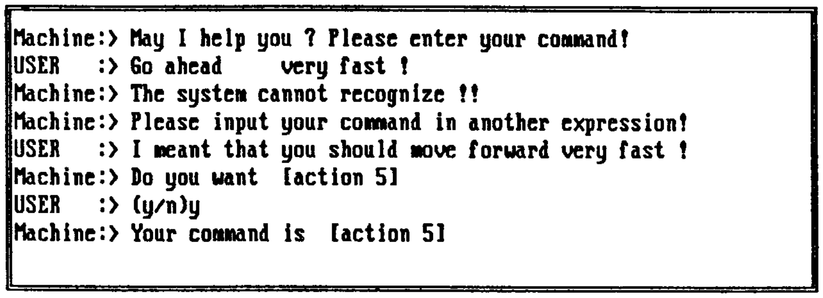
(d)

Fig. 11. Screen copy of the voice control system—the third case.

VI. Issues on Computational Efficiency and Convergence Property

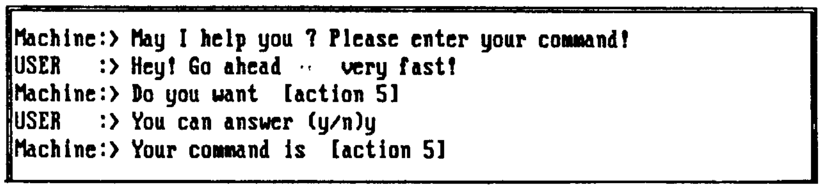
For the MI-supervised learning algorithm proposed in Section III-A, we observe from (13) that the weights and biases are totally determined by the count measurements $N(\nu_m)$, $N(\nu_n)$ and $N(\nu_m, c_k)$. Because counts can be accumulated sequentially, the estimates of the weights can be adaptively and sequentially updated with each new input. Since the definition of training procedure is sequential, the network only requires a single pass through the data and, thus, it provides fast learning. This is in contrast to those of stochastic gradient algorithms that may need many interactions through the data to converge only to some local minimum of the error function. As for the convergence of the MI-supervised learning algorithm, it is guaranteed to decrease the single step-error function for nonzero input vectors and converges asymptomatically [10]. Furthermore, it has been shown that if the words are unambiguous such that all layer-one outputs of the CCNO are zero or one and if the language generating the commands is first-order Markovian, then the CCNO is equivalent to a *maximum a posteriori* decision rule [10]. Although the hypotheses are rarely true in real cases, this property provides insight into the network's characteristics.

During the MI-reinforcement learning discussed in Section IV-B, the connection weights of the CCNO are held constant during the dialogue, and adapted using the MI-supervised learning algorithm after the dialogue converges and before commencing the next dialogue. Hence, as its supervised version, the MI-reinforcement learning algorithm is fast and needs only a single pass through each new input. From (28), if $R'[s(t)] = -\infty$ we have $A_k(t) = -\infty$, meaning that action k is not the desired action and, thus, the RAFCAN will never let action k be selected again in the current dialogue [see (29)]. The MI-reinforcement learning (i.e., dialogue) converges when $R'[s(t)] = 0$. The dialogue can be terminated and the user connected to a supervisor if it continues too long without convergence. If the dialogue fails to converge, then the negative examples could be exploited



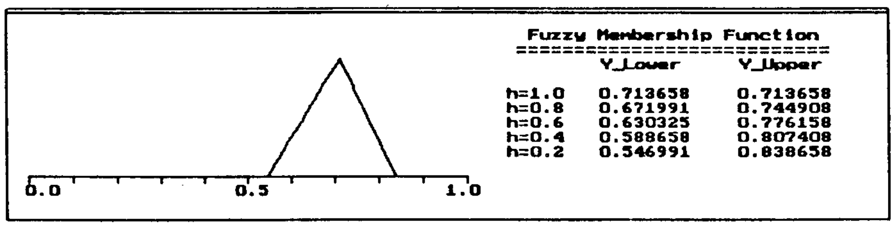
```
Machine:> May I help you ? Please enter your command!
USER    :> Go ahead      very fast !
Machine:> The system cannot recognize !!
Machine:> Please input your command in another expression!
USER    :> I meant that you should move forward very fast !
Machine:> Do you want  [action 5]
USER    :> (y/n)y
Machine:> Your command is  [action 5]
```

(a)



```
Machine:> May I help you ? Please enter your command!
USER    :> Hey! Go ahead      very fast!
Machine:> Do you want  [action 5]
USER    :> You can answer (y/n)y
Machine:> Your command is  [action 5]
```

(b)



```
        Fuzzy Membership Function
        ===========================
              Y_Lower       Y_Upper

h=1.0      0.713658      0.713658
h=0.8      0.671991      0.744908
h=0.6      0.630325      0.776158
h=0.4      0.588658      0.807408
h=0.2      0.546991      0.838658
```

(c)

Fig. 12. Screen copy of the voice control system—the fourth case.

for adaptation. In fact, it can be shown that the probability, $P(k)$, that a dialogue converges in k steps (sentences) is $P(k) = p^{k-1}(k - p)$ where p is the probability of error on each user's input command.

The proposed FBP algorithm in Section III-B is a gradient descent procedure in which the (fuzzy) weights are modified along the negative direction of the gradient of e in (15) with respective to the weights. Hence, it can be expected that the weights will eventually converge to the values that minimize e to within some small fluctuations if there is adequate learning, adequate number of hiddens nodes, and a deterministic rather than a stochastic relation between input and desired output. The FBP algorithm is bounded by all of the problems of any hill climbing procedure such as the problems of local minima and slow convergence. The average iteration number for the FCNO to learn a desired fuzzy output in our simulations is 6000. However, the use of resolution principle to express a fuzzy set in terms of its α-level sets has sped up the convergence of the FBP algorithm already. Also, empirical studies have shown that the poor local minima are rarely encountered if the FCNO contains a few more hidden nodes than required for a learning task. Even though, since the FBP algorithm is a generalization of the BP algorithm, several existent techniques for improving the local minima and slow convergence problems [13], [16] could be adopted.

The fuzzy reinforcement learning algorithms presented in Section IV-C are derived based on the FBP algorithm by incorporating the TD method on the fuzzy predictor, and the stochastic exploration technique on the fuzzy action network. Hence, their convergence property basically inherits that of the FBP algorithm mentioned in the above. On the side of fuzzy predictor, the TD(λ) method has been shown to converge in expected value to the idea predictions for general $\lambda \in [0, 1]$ based on the concept of dynamic programming [33]. In the two extreme cases, TD(0) converges in the mean for observations of an absorbing Markov chain and TD(1) reduces to the normal least-mean square (LMS) estimator as shown in [27]. The TD(λ) method is an efficient prediction procedure since the learning in TD does not have to wait until the actual outcome is known and can update the connection weights within a trial period. On the side of fuzzy action network, being instructed by the fuzzy predictor through an internal reinforcement signal, the fuzzy stochastic unit can perform an efficient exploration on the output space to seek for better outputs. This shortens the learning time of the fuzzy action network a lot as compared to normal reinforcement learning schemes [26].

The convergence property of the fuzzy reinforcement learning algorithm for the fuzzy action network is discussed as follows. According to (45), before the external reinforcement signal occurs [i.e., $r(t) = 0$], the reinforcement $\hat{r}$ sent to the crisp action network is the difference between the predicted reinforcement signal at the current time step (discounted by γ) and the predicted reinforcement signal at the previous time step [i.e., $\gamma p(t + 1) - p(t)$]. That is, (45) 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 crisp action network, and $p(t+1)$ is the predicted reinforcement signal for output $y(t+1)$. Because 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 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 therefore 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, (45) 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 (47) and (48) are appropriate for the fuzzy action network to estimate its output-error gradient correctly. Since the error gradient information can be estimated correctly in this way, the reinforcement learning of the fuzzy action network can converge like the FBP algorithm analyzed in the above.

VII. Conclusion

In this paper, the fuzzy command acquisition network, AFCAN, which consists of command acquisition and fuzzy information acquisition, is proposed. Unlike the general language acquisition systems, the proposed system has the following characteristics: 1) the system is built as a neural network trained by users' given data, so the system equips the ability to tune its parameters and structure to match the application environment; 2) the system has the ability to acquire fuzzy command, which is a nature language comprising the desired actions and fuzzy linguistic information; 3) the input sentences (commands) of this system are unrestricted, but the kinds of output semantic actions are quite restricted. Hence, the proposed AFCAN is suitable for constrained-action tasks. That is, one can ask the machine to perform one of a small number of actions, but is allowed total freedom in making such requests; 4) the proposed system needs not any acoustic, prosodic, syntactic, and grammatical structure. It is the network (connectionist) structure that enables it to decode the intended information from a natural language message and this structure makes the system be able to perform more human-like command acquisition and learning; and 5) the system can acquire fuzzy command during the course of performing task. That is, it has the capacity of on-line learning for seeking a more suitable network at any time by accepting the critics from the users. There are many promising applications for the proposed command acquisition system such as automated call routing in a telecommunications network, on-line information retrieval system, human voice-control robot, voice-control automatic car, etc. The proposed AFCAN can make the man-machine interface in the above applications more considerate and friendly. One deficiency of the proposed AFCAN is that the number of nodes in layers one and two increases exponentially as the number of words increases. This restricts the generalization of the proposed fuzzy *command* acquisition network to a general fuzzy *language* acquisition network directly. This problem is also due to the semantic complexity and context-sensitive structure of human languages. One promising approach to solving this problem partially is to use the distributed representation concept of neural networks in layers one and two of the proposed AFCAN. This will be our future research.

Acknowledgment

The authors would like to thank the reviewers for their helpful suggestions in improving the quality of the final manuscript.

References

[1] A. Waibel and K. F. Lee, "Speech recognition by machine: A review," in *Reading in Speech Recognition.* San Mateo, CA: Morgan Kaufmann, 1990, pp. 8–39.

[2] K. Fu and T. Booth, "Grammatical inference: Introduction and survey—Parts 1 and 2," *IEEE Trans. Syst. Man, Cybern.*, vol. SMC-5, no. 1, pp. 95–111, Jan./Feb. 1975; vol. 5, no. 4, pp. 409–423, July/Aug. 1975.

[3] J. K. Baker, "Trainable grammars for speech recognition," in *Speech Commun. Papers 97th Meet. Amer. Speech Assoc.*, 1979, pp. 547–550.

[4] P. Langley, "Language acquisition through error recover," in *Cognition Brain Theory.* Cambridge, MA: MIT Press, 1982, vol. 5, pp. 211–255.

[5] K. Lari and S. J. Young, "The estimation of stochastic context-free grammars using the inside–outside algorithm," *Comput. Speech, Language*, vol. 4, pp. 35–36, 1990.

[6] J. R. Anderson, "Induction of augmented transition networks," *Cognition Sci.*, vol. 1, pp. 125–157, 1977.

[7] E. Vidal, P. Garcia, and E. Segarra, "Inductive learning of finite-state transducers," in *Structural Pattern Analysis*, R. Mohr, T. Pavlidis, and A. Sanfeliu, Eds. San Diego, CA: World Scientific, 1989.

[8] R. Miikkulainen and M. Dyer, "A modular neural network architecture for sequential paraphrasing of script-based stories," in *Proc. IEEE Int. Joint Conf. Neural Networks*, pp. II.49–II.56, June 1989.

[9] M. St. John and J. McClelland, "Learning and applying contextual constraints in sentence comprehension," *Artificial Intell.*, vol. 46, pp. 217–257, 1990.

[10] A. L. Gorin, S. Levinson, A. Gertner, and E. Goldman, "Adaptive acquisition of language," *Comput. Speech, Language*, vol. 5, no. 2, pp. 101–132, Apr. 1991.

[11] A. L. Gorin and A. Gertner, "Visual focus of attention in adaptive language acquisition," in *Artificial Neural Networks for Speech and Vision.* London, U.K.: Chapman & Hall, 1993, pp. 324–356.

[12] 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, May 1993.

[13] C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neural-Fuzzy Synergism to Intelligent Systems.* Englewood Cliffs, NJ: Prentice-Hall, May, 1996.

[14] A. Kaufmann and M. M. Gupta, *Introduction to Fuzzy Arithmetic.* New York: Van Nostrand Reinhold, 1985.

[15] L. A. Zadeh, "Knowledge representation in fuzzy logic," *IEEE Trans. Knowl. Data Eng.*, vol. 1, no. 1, pp. 89–100, Mar. 1989.

[16] S. Haykin, *Neural Networks.* New York: Macmillan, 1994, ch. 11, pp. 444–471.

[17] K. W. Church and P. Hanks, "Word association norms, mutual information and lexicography," in *Proc. 27th Meet. Assoc. Computat. Linguistics*, 1989, pp. 76–83.

[18] R. Sproat and C. Shih, "A statistical method for finding word boundaries in Chinese text," in *Comput. Proc. Chinese Oriental Lang.*, 1990.

[19] R. M. Gray, *Entropy and Information Theory.* New York: Springer-Verlag, 1990.

[20] S. K. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets and classification," *IEEE Trans. Neural Networks*, vol. 3, pp. 683–696, May 1992.

[21] J. M. Keller and H. Tahani, "Backpropagation neural networks for fuzzy logic," *Inform. Sci.*, vol. 62, pp. 205–221, 1992.

[22] S. Horikawa, T. Furuhashi, and Y. Uchikawa, "On fuzzy modeling using fuzzy neural networks with the backpropagation algorithm," *IEEE Trans. Neural Networks*, vol. 3, pp. 801–806, May 1992.

[23] H. Ishibuchi, R. Fujioka, and H. Tanaka, "Neural networks that learn from fuzzy if–then rules," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 85–97, May 1993.

[24] Y. Hayashi, J. J. Buckley, and E. Czogula, "Fuzzy neural network," *Int. J. Intell. Syst.*, vol. 8, pp. 527–537, 1993.

[25] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation.* New York: Addison-Wesley, 1991.

[26] 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, pp. 46–63, Feb. 1995.

[27] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learning*, vol. 3, pp. 9–44, 1988.

[28] 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, Sept./Oct. 1983.

[29] H. R. Berenji and P. Khedkar, "Learning and tuning fuzzy logic controllers through reinforcements," *IEEE Trans. Neural Networks*, vol. 3, pp. 724–740, May 1992.

[30] C. W. Anderson, "Strategy learning with multilayer connectionist representations," in *Proc. 4th Int. Workshop Mach. Learning*, Irvine, CA, June 1987, pp. 103–114.

[31] 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, 1987, vol. II, pp. 601–608.

[32] M. C. Kan, "Adaptive fuzzy command acquisition with reinforcement learning," M.S. thesis, Dept. Contr. Eng., Nat. Chiao-Tung Univ., Taiwan, 1995.

[33] P. Dayan, "The convergence of TD(λ) for general λ," *Mach. Learning*, vol. 8, pp. 341–362, 1992.

[34] K. Tanaka, M. Sano, and H. Watanabe, "Modeling and control of carbon monoxide concentration using a neuro-fuzzy technique," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 271–279, Aug. 1995.

[35] Y. Lin and G. A. Cunningham, "A new approach to fuzzy-neural system modeling," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 190–197, May 1995.

[36] N. K. Kasabov, "Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems," *Fuzzy Sets Syst.*, vol. 82, no. 2, pp. 135–149, 1996.

[37] S. Tano, T. Oyama, and T. Arnould, "Deep combination of fuzzy inference and neural network in fuzzy inference software—FINEST," *Fuzzy Sets Syst.*, vol. 82, no. 2, pp. 151–160, 1996.

Chin-Teng Lin 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 currently a Professor of electrical and control engineering. His current research interests are fuzzy systems, neural networks, intelligent control, speech coding, human-machine interface, and video and audio processing. He is the co-author of *Neural Fuzzy Systems—A Neuro-Fuzzy Synergism to Intelligent Systems* (Englewood Cliffs, NJ: Prentice-Hall, 1996) and the author of *Neural Fuzzy Control Systems with Structure and Parameter Learning* (River Edge, NJ: World Scientific, 1994).

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.

Ming-Chih Kan received the B.S. and M.S. degrees in control engineering from the National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 1993 and 1995, respectively.

Since October 1995, he has been with the Institute for Information Industry, Taipei, Taiwan, R.O.C., where he is currently a System Engineer in the Special System Division. His current research interests include fuzzy systems, neural networks, design pattern, object-oriented methodology, and integrated circuit (IC) factory computer-integrated manufacturing.