Fifth IFSA World Congress (1993), 993-996

ON THE STRUCTURE AND LEARNING OF NEURAL-NETWORK-BASED FUZZY LOGIC CONTROL SYSTEMS

C. T. Lin
Department of Computer and Information Science
National Chiao-Tung University
Hsinchu, Taiwan, R.O.C.

C. S. George Lee
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907, USA

ABSTRACT

This paper addresses the structure and its associated learning algorithms of a feedforward multi-layered connectionist network, which has distributed learning abilities, for realizing the basic elements and functions of a traditional fuzzy logic controller. The proposed neural-network-based fuzzy logic control system (NN-FLCS) can be contrasted with the traditional fuzzy logic control system in their network structure and learning ability. An on-line supervised structure/parameter learning algorithm is proposed for constructing the NN-FLCS dynamically. The proposed dynamic learning algorithm can find proper fuzzy logic rules, membership functions, and the size of output fuzzy partitions simultaneously. Next, a Reinforcement Neural-Network-Based Fuzzy Logic Control System (RNN-FLCS) is proposed which consists of two closely integrated Neural-Network-Based Fuzzy Logic Controllers (NN-FLCs) for solving various reinforcement learning problems in fuzzy logic systems. One NN-FLC functions as a fuzzy predictor and the other as a fuzzy controller. Associated with the proposed RNN-FLCS is the reinforcement structure/parameter learning algorithm which dynamically determines the proper network size, connections, and parameters of the RNN-FLCS through an external reinforcement signal. Furthermore, learning can proceed even in the period without any external reinforcement feedback.

1. Neural-Network-Based Fuzzy Logic Control System

During the past decade, fuzzy logic has found fruitful applications in various fields [1-3]. However, most control engineers are still frustrated with this technique due to a lack of systematic procedures for the design of fuzzy logic systems. The choice of membership functions and/or fuzzy logic rules remains heuristic and subjective, and a trial-and-error procedure is commonly used for the design of fuzzy logic systems. Recent direction of exploration is to design fuzzy logic systems that have the capability of learning from experience by itself [4,5].

Distributed representation and learning capabilities are two major features of neural networks [6,7]. In distributed representation, a value is represented by a pattern of activity distributed over many computing elements (CEs), and each CE is involved in representing many different values. So each CE has a receptive field, which is the set of all values that include all the patterns it represents. Therefore, each CE corresponds to a fuzzy set, and its receptive field corresponds to the membership function. Among the three classes of learning schemes, the unsupervised procedures [8] are suitable to find clusters of data indicating the presence of fuzzy rules. The supervised procedures and the reinforcement procedures are good to adapt the fuzzy rules or membership functions for the desired output in fuzzy logic systems. Hence, bringing the learning abilities of neural networks to fuzzy logic systems will provide a promising approach.

This paper presents a general Neural-Network-Based Fuzzy Logic Control System (NN-FLCS) for realizing the basic elements and functions of a traditional fuzzy logic control and decision system [1-3]. In this connectionist structure [9-11], the input and output nodes represent the input states and output control/decision signals, respectively, and in the hidden layers, there are nodes functioning as membership functions and rules. An on-line supervised structure/parameter learning algorithm is proposed to construct NN-FLCS dynamically. This algorithm blends *fuzzy similarity measure* with supervised gradient-descent learning to perform structure and parameter learning simultaneously. The fuzzy similarity measure is a tool to determine the degree to which two fuzzy sets are equal. Using this measure, a new output membership function may be added, and the rule-node connections (the consequence links of rule nodes) can be changed properly. In some learning environments, obtaining exact training data may be expensive. This motivates the desire of integrating two Neural-Network-Based Fuzzy Logic Controllers (NN-FLCs) into a Reinforcement Neural-Network-Based Fuzzy Logic Control System (RNN-FLCS) for solving various reinforcement learning problems. One NN-FLC functions as a fuzzy predictor and the other as a fuzzy controller. Structurally, these two NN-FLCs share the first two layers of the proposed NN-FLCS; that is, they use the same distributed representation of input patterns. This representation is the overlapping type and is dynamically adjustable through the learning process. Associated with the proposed RNN-FLCS is the reinforcement structure/parameter learning algorithm which dynamically determines the proper network size, connections, and parameters of the RNN-FLCS through an external reinforcement signal. Furthermore, learning can proceed even in the period without any external reinforcement feedback. The proposed RNN-FLCS makes the design of fuzzy logic controllers more practical for real-world applications since it greatly lessens the quality and quantity requirements of the feedback training signals.

This work was supported in part by the National Science Foundation under Grant CDR 8803017 to the Engineering Research Center for Intelligent Manufacturing Systems and a grant from the Ford Foundation.

Figure 1 shows the structure of our NN-FLCS which has five layers. Nodes at layer one are input nodes which represent input linguistic variables. Layer five is the output layer. Nodes at layers two and four are *term nodes* and act as membership functions to represent the terms of the respective linguistic variable. Each node at layer three is a rule node which represents one fuzzy logic rule. Thus, all layer-three nodes form a fuzzy rule base. Layer-three links define the preconditions of the rule nodes, and layer-four links define the consequences of the rule nodes. The links at layers two and five are fully connected between linguistic nodes and their corresponding term nodes. We shall next describe the functions of the nodes in each of the five layers of the proposed connectionist model. In the following, f is an integration function of a node, which combines activation from other nodes to provide net input for this node. a is an activation function of a node, which outputs an activation value as a function of net input. In the following equations, superscript is used to indicate the layer number.

- **Layer 1:** The nodes in this layer transmit input values directly to the next layer. That is,

$$f = u_i^1 \quad \text{and} \quad a = f. \tag{1}$$

From Eq. (1), the link weight at layer one (w_i^1) is unity.

- **Layer 2:** If we use a single node to perform a simple membership function, then the output function of this node should be this membership function. For example, for a bell-shaped function,

$$f = M_{x_i}^j(m_{ij}, \sigma_{ij}) = -\frac{(u_i^2 - m_{ij})^2}{\sigma_{ij}^2} \quad \text{and} \quad a = e^f, \tag{2}$$

where m_{ij} and σ_{ij} are, respectively, the center (or mean) and the width (or variance) of the bell-shaped function of the jth term of the ith input linguistic variable x_i. Hence, the link weight at layer two (w_{ij}^2) can be interpreted as m_{ij}.

- **Layer 3:** The links in this layer are used to perform precondition matching of fuzzy logic rules. Hence, the rule nodes should perform the fuzzy AND operation,

$$f = \min(u_1^3, u_2^3, \cdots, u_p^3) \quad \text{and} \quad a = f. \tag{3}$$

The link weight in layer three (w_i^3) is then unity.

- **Layer 4:** The links at layer four perform the fuzzy OR operation to integrate the fired rules which have the same consequence,

$$f = \sum_{i=1}^{p} u_i^4 \quad \text{and} \quad a = \min(1, f). \tag{4}$$

Hence the link weight $w_i^4 = 1$.

- **Layer 5:** The nodes in this layer transmit the decision signal out of the network. These nodes and the layer-five links attached to them act as the defuzzifier. If m_{ij}^5's and σ_{ij}^5's are the centers and the widths of the membership functions, respectively, then the following functions can be used to simulate the *center of area* defuzzification method [3]:

$$f = \sum w_{ij}^5 u_i^5 = \sum (m_{ij}\sigma_{ij}) u_i^5 \quad \text{and} \quad a = \frac{f}{\sum \sigma_{ij} u_i^5}. \tag{5}$$

Here the link weight at layer five (w_{ij}^5) is $m_{ij}\sigma_{ij}$.

2. On-line Structure/Parameter Learning Algorithm

We first propose an on-line learning algorithm that can dynamically learn the network structure and parameters simultaneously. The proposed structure/parameter learning algorithm uses the fuzzy similarity measure [10] to perform the structure learning and the back-propagation algorithm to perform the parameter learning. Given the supervised training data, the proposed learning algorithm first decides whether or not to perform the structure learning based on the fuzzy similarity measure of the output membership functions. If the structure learning is necessary, then it will further decide whether or not to add a new output term node (a new membership function), and it will also change the consequences of some fuzzy logic rules properly. After the structure learning process, the parameter learning will be performed to adjust the parameters of current membership functions. This structure/parameter learning will be repeated for each on-line incoming training input/output data pair. After the structure/parameter training loop has been performed, rule combination is then initiated to find the minimum node representation of fuzzy logic rules as in [9].

An initial form of the network is first constructed before this network is trained. Then, during the learning process, new output term nodes may be added and some connections may be changed. Finally, after the learning process, some nodes and links of the network will be deleted or combined to form the final structure of the network. The initial form of the network is same as that described in [9], except that there is only one link between a rule node and an output linguistic variable. This link is connected to some term node of the output linguistic variable. The initial candidate (term node) of the consequence of a rule node can be assigned by an expert (if possible) or be chosen randomly. A suitable term in each output linguistic variable's term set will be chosen for each rule node after the learning process.

After the initialization process, the learning algorithm enters the training loop in which each loop corresponds to a set of training input data $x_i(t), i = 1,...,n$, and the desired output value $y_i(t), i = 1,...,m$, at a specific time t. Basically, the idea of back-propagation [7] is used for this supervised learning to find the errors of node outputs in each layer. Then, these errors are analyzed by the fuzzy similarity measure to perform structure adjustments or parameter adjustments. The goal is to minimize the error function

$$E = \frac{1}{2}[y(t) - \hat{y}(t)]^2 \tag{6}$$

where $y(t)$ is the desired output, and $\hat{y}(t)$ is the current output. For each training data set, starting at the input nodes, a forward pass is used to compute the activity levels of all the nodes in the network. Then starting at the output nodes, a backward pass is used to compute $\partial E/\partial y$ for all the hidden nodes. Assuming that w is the adjustable parameter in a node (e.g., center of a membership function), the general learning rule used is

$$w(t+1) = w(t) + \eta(-\frac{\partial E}{\partial w}) \quad \text{and} \quad \frac{\partial E}{\partial w} = \frac{\partial E}{\partial f}\frac{\partial f}{\partial w} = \frac{\partial E}{\partial a}\frac{\partial a}{\partial f}\frac{\partial f}{\partial w} \tag{7}$$

where η is the learning rate. To show the learning rules, we derive the rules layer by layer using the bell-shaped membership functions with centers m_i's and widths σ_i's as the adjustable parameters for these computations.

- **Layer 5:** Using Eqs. (5) and (6), the expected updated amount of the center parameter m_i and the width parameter σ_i are, respectively,

$$\Delta m_i(t) \triangleq m_i(t+1) - m_i(t) = \eta[y(t) - \hat{y}(t)]\frac{\sigma_i u_i}{\sum \sigma_i u_i}. \tag{8}$$

$$\Delta\sigma_i(t) \triangleq \sigma_i(t+1) - \sigma_i(t) = \eta[y(t) - \hat{y}(t)]\frac{m_i u_i(\sum \sigma_i u_i) - (\sum m_i \sigma_i u_i) u_i}{(\sum \sigma_i u_i)^2} \tag{9}$$

The error to be propagated to the previous layer is

$$\delta^5 = \frac{-\partial E}{\partial f} = \frac{-\partial E}{\partial a}\frac{\partial a}{\partial f} = y(t) - \hat{y}(t). \tag{10}$$

- **Fuzzy Similarity Measure:** In this step, the system will decide if the current structure should be changed or not according to the expected updated amount of the center and width parameters. To do this, the expected center and width are, respectively, computed as

$$m_{i-new} = m_i(t) + \Delta m_i(t) \quad \text{and} \quad \sigma_{i-new} = \sigma_i(t) + \Delta\sigma_i(t). \tag{11}$$

From the current membership functions of output linguistic variables, we want to find the one which is the most similar to the expected membership function by measuring their fuzzy similarity. Let $M(m_i, \sigma_i)$ represent the bell-shaped membership function with center m_i and width σ_i. Let

$$degree(i, t) = E[M(m_{i-new}, \sigma_{i-new}), M(m_{i-closest}, \sigma_{i-closest})] \tag{12}$$
$$= \max_{1 \le j \le k} E[M(m_{i-new}, \sigma_{i-new}), M(m_j, \sigma_j)],$$

where $k = |T(y)|$, $E(\cdot,\cdot)$ is the fuzzy similarity. If A and B are two fuzzy sets with bell-shaped membership functions, The approximate fuzzy similarity measure of A and B, $E(A, B)$, can be computed as follow: Assuming $m_1 \ge m_2$,

$$E(A, B) = \frac{M(A \cap B)}{M(A \cup B)} = \frac{M(A \cap B)}{\sigma_1\sqrt{\pi} + \sigma_2\sqrt{\pi} - M(A \cap B)}. \tag{13}$$

Here $$M(A \cap B) = \frac{1}{2}\frac{h^2(m_2 - m_1 + \sqrt{\pi}(\sigma_1 + \sigma_2))}{\sqrt{\pi}(\sigma_1 + \sigma_2)} + \tag{14}$$
$$\frac{1}{2}\frac{h^2(m_2 - m_1 + \sqrt{\pi}(\sigma_1 - \sigma_2))}{\sqrt{\pi}(\sigma_2 - \sigma_1)} + \frac{1}{2}\frac{h^2(m_2 - m_1 - \sqrt{\pi}(\sigma_1 - \sigma_2))}{\sqrt{\pi}(\sigma_1 - \sigma_2)},$$

where $h(x) = \max\{0, x\}$. After the most similar membership function $M(m_{i-closest}, \sigma_{i-closest})$ to the expected membership function $M(m_{i-new}, \sigma_{i-new})$ has been found, the following adjustment is made:

```
IF $degree(i, t) < \alpha(t)$,
   THEN
         create a new node $M(m_{i-new}, \sigma_{i-new})$ in layer 4
           and denote this node as the $i-closest$ node,
         do the structure learning process,
   ELSE IF $M(m_{i-closest}, \sigma_{i-closest}) \neq M(m_i, \sigma_i)$
         THEN
              do the structure learning process,
         ELSE
              do the following parameter adjustments:
              $m_i(t+1) = m_{i-new}$ and $\sigma_i(t+1) = \sigma_{i-new}$
              skip the structure learning process.
```

$\alpha(t)$ is a monotonically increasing scalar similarity criterion.

- **Structure Learning:** To find the rules whose consequences should be changed, we set a *firing strength threshold*, β. Only the rules whose firing strengths are higher than this threshold are treated as really firing rules. Only the really firing rules are considered to be changing their consequences, since only these rules are fired strongly enough to contribute to the above results of judgement. Assuming that the term node $M(m_i, \sigma_i)$ in layer 4 has inputs from rule nodes $1, \cdots, l$ in layer 3, whose corresponding firing strength are a_i^3's, $i = 1, \cdots, l$, then
  IF $a_i^3(t) \ge \beta$, THEN change the consequence of the ith rule
  node from $M(m_i, \sigma_i)$ to $M(m_{i-new}, \sigma_{i-new})$.

- **Layer 4:** There is no parameter to be adjusted in this layer. Only the error signals (δ_i^4's) need to be computed and propagated. From Eqs. (4) and (5), the error signal δ_i^4 is derived as:

$$\delta_i^4(t) = [y(t) - \hat{y}(t)]\frac{m_i\sigma_i(\sum \sigma_i u_i) - (\sum m_i \sigma_i u_i)\sigma_i}{(\sum \sigma_i u_i)^2}. \tag{15}$$

- **Layer 3:** As in layer four, only the error signals need to be computed. According to Eq. (3), this error signal can be derived as: $\delta_i^3 = \delta_i^4$. If there are multiple outputs, the error signal becomes $\delta_i^3 = \sum_k \delta_k^4$. Here the summation is performed over the consequences of a rule node.

- **Layer 2:** Using Eq. (7) and Eqs. (2) and (3), the adaptive rule of m_{ij} and σ_{ij} are, respectively,

$$m_{ij}(t+1) = m_{ij}(t) - \eta\frac{\partial E}{\partial a_i} e^{f_i}\frac{2(u_i - m_{ij})}{\sigma_{ij}^2}, \tag{16}$$

$$\sigma_{ij}(t+1) = \sigma_{ij}(t) - \eta\frac{\partial E}{\partial a_i} e^{f_i}\frac{2(u_i - m_{ij})^2}{\sigma_{ij}^3}, \tag{17}$$

where $\frac{\partial E}{\partial a_i} = \sum_k q_k$. The summation here is performed over the rule nodes that a_i feeds into, and

$$q_k = \begin{cases} \delta_k^3 & \text{if } a_i \text{ is minimum in } kth \text{ rule node's inputs} \\ 0 & \text{otherwise.} \end{cases} \tag{18}$$

3. Structure/Parameter Learning Algorithm for RNN-FLCS

Unlike the supervised learning problem, the reinforcement learning problem has only very simple "evaluative" information called reinforcement signal available for learning. In this paper, the reinforcement signal $r(t)$ is defined as a value between -1 and 1 corresponding to various degrees of failure or success. We also assume that $r(t)$ is the reinforcement signal available at time step t and is caused by the input and actions chosen at time step $t-1$ or even affected by earlier inputs and actions. The objective of learning is to maximize the reinforcement signal. The proposed RNN-FLCS, as shown in Fig. 2, integrates two NN-FLCs into a learning system: one NN-FLC for the fuzzy controller and the other for the fuzzy predictor. These two NN-FLCs share the same layers 1 and 2 and have individual layer 3 to layer 5. In this section, a reinforcement learning algorithm is proposed for the RNN-FLCS with a single-step fuzzy predictor to solve simpler reinforcement learning problems in which a reinforcement signal is only one time step behind its corresponding action. For the case that there is a long time delay between an action and the resulting reinforcement signal, a more powerful multi-step fuzzy predictor is necessary for the RNN-FLCS.

3.1. Stochastic Exploration

In this subsection, we first develop the learning algorithm for the action network. The goal of the reinforcement structure/parameter learning algorithm is to adjust the parameters (e.g., m_i's) of the action network or to change the connectionist structure or even to add new nodes, if necessary, such that the reinforcement signal is maximum. That is, $\Delta m_i \propto \frac{\partial r}{\partial m_i}$. To know $\frac{\partial r}{\partial m_i}$, we need to know $\frac{\partial r}{\partial y}$, where y is the output of the action network. In our learning algorithm, the gradient information, $\frac{\partial r}{\partial y}$, is estimated by the stochastic exploratory method [12]. In estimating the gradient information, the output y of the action network does not act on the environment directly. Instead, it is treated as a mean (expected) action. The actual action, $\hat{y}$, is chosen by exploring a range around this mean point. This range of exploration corresponds to the variance of a probability function, which is the normal distribution in our design. This amount of exploration, $\sigma(t)$, is chosen as

$$\sigma(t) = \frac{k}{2}[1 - \tanh(p(t))] = \frac{k}{1 + e^{2p(t)}}, \tag{19}$$

where k is a search-range scaling constant which can be simply set to 1, and $p(t)$ is the predicted (expected) reinforcement signal used to predict $r(t)$. Once the variance has been decided, the actual output of the stochastic node can be set as $\hat{y}(t) = N(y(t), \sigma(t))$. The gradient information is estimated as

$$\frac{\partial r}{\partial y} \approx [r(t) - p(t)]\left[\frac{\hat{y}(t-1) - y(t-1)}{\sigma(t-1)}\right] \equiv [r(t) - p(t)]\left[\frac{\hat{y} - y}{\sigma}\right]_{t-1} \tag{20}$$

where the subscript, $t-1$, represents the time displacement. Assuming that w is an adjustable parameter in a node (e.g., the center of a membership function), the general parameter learning rule used is (as in Eq. (7))

$$w(t+1) = w(t) + \eta(\frac{\partial r}{\partial w}) \text{ and } \frac{\partial r}{\partial w} = \frac{\partial r}{\partial a}\frac{\partial a}{\partial f}\frac{\partial f}{\partial w}. \tag{21}$$

- **Layer 5:** Using Eqs. (5), (20), and (21), the expected updated amount of the center parameter and the width parameter are respectively

$$\Delta m_i(t) = \eta[r(t) - p(t)]\left[\frac{\hat{y} - y}{\sigma}\right]_{t-1}\left[\frac{\sigma_i u_i}{\sum \sigma_i u_i}\right]_{t-1}. \tag{22}$$

$$\Delta \sigma_i(t) = \eta[r(t) - p(t)]\left[\frac{\hat{y} - y}{\sigma}\right]_{t-1}\left[\frac{m_i u_i(\sum \sigma_i u_i) - (\sum m_i \sigma_i u_i)u_i}{(\sum \sigma_i u_i)^2}\right]_{t-1} \tag{23}$$

The error to be propagated to the preceding layer is

$$\delta^5(t) = \frac{\partial r}{\partial f^5} = \frac{\partial r}{\partial a}\frac{\partial a}{\partial f^5} = [r(t) - p(t)]\left[\frac{\hat{y} - y}{\sigma}\right]_{t-1}. \tag{24}$$

Fuzzy Similarity Measure: In this step, the system will decide whether the current structure should be changed or not according to the expected updated amount of the center and width parameters (in Eqs. (22) and (23)). This procedure of using the fuzzy similarity measure is the same as for the on-line learning algorithm.

- **Layer 4:** There is no parameter to be adjusted in this layer. Only the error signals (δ_i^4's) need to be computed and propagated. From Eqs. (5) and (21), the error signal δ_i^4 is derived as

$$\delta_i^4(t) = [r(t) - p(t)]\left[\frac{\hat{y} - y}{\sigma}\right]_{t-1}\left[\frac{m_i \sigma_i(\sum \sigma_i u_i) - (\sum m_i \sigma_i u_i)\sigma_i}{(\sum \sigma_i u_i)^2}\right]_{t-1} \tag{25}$$

In the multi-output case, the computations in layers five and four are exactly the same as the above using the same internal reinforcement signals and proceeding independently for each output linguistic variable.

- **Layer 3:** As in layer four, only the error signals need to be computed. According to Eqs. (4) and (21), this error signal can be derived as $\delta_i^3(t) = \delta_i^4(t)$. If there are multiple outputs, then the error signal becomes $\delta_i^3(t) = \sum_k \delta_k^4(t)$, where the summation is performed over the consequences of a rule node; that is, the error of a rule node is the summation of the errors of its consequences.

- **Layer 2:** Using Eqs. (2) and (21), the adaptive rule of m_{ij} and σ_{ij} are respectively

$$m_{ij}(t+1) = m_{ij}(t) - \eta\left[\frac{\partial r}{\partial a_i}\right]_t\left[e^{f_i}\frac{2(u_i - m_{ij})}{\sigma_{ij}^2}\right]_{t-1}, \tag{26}$$

$$\sigma_{ij}(t+1) = \sigma_{ij}(t) - \eta\left[\frac{\partial r}{\partial a_i}\right]_t\left[e^{f_i}\frac{2(u_i - m_{ij})^2}{\sigma_{ij}^3}\right]_{t-1},$$

where $\frac{\partial r}{\partial a_i} = \sum_k q_k(t)$ as in Eq. (18).

3.2. Single-Step Fuzzy Predictor

We shall use an NN-FLC to develop a single-step fuzzy predictor (evaluation network) as shown in Fig. 2. The function of the single-step fuzzy predictor is to predict the external reinforcement signal, $r(t)$, one time step ahead, that is, at time $t-1$. Here, $r(t)$ is the real reinforcement signal resulting from the inputs and actions chosen at time step $t-1$, but it can only be known at time step t. If the fuzzy predictor can produce a signal, $p(t)$, which is the prediction of $r(t)$ but is available at time step $t-1$, then the time delay problem can be solved. With a correct predicted signal, $p(t)$, a better action can be chosen by the action network at time step $t-1$, and the corresponding learning can be performed on the action network at time step t upon receiving the external reinforcement signal $r(t)$. As indicated in the last subsection, $p(t)$ is necessary for the stochastic exploration with a multi-parameter probability distribution (in Eq. (6)). The other internal reinforcement signal, $\hat{r}(t)$, in Fig. 2 is set as $\hat{r}(t) = r(t) - p(t)$, which is the prediction error for computing Eq. (7) by the action network. The single-step prediction is the extreme case of the multi-step prediction which will be presented in the next section. The goal to train the single-step fuzzy predictor is to minimize the squared error prediction:

$$E = \frac{1}{2}[r(t) - p(t)]^2, \tag{27}$$

where $r(t)$ represents the desired output (real external reinforcement signal), and $p(t)$ is the current output (predicted reinforcement signal). Then the gradient information can be easily derived as $\frac{\partial E}{\partial p} = p(t) - r(t)$. Similar to the learning rule developed in the last section, we can derive the structure/parameter learning algorithm for the single-step fuzzy predictor using the general parameter learning rule: $w(t+1) = w(t) + \eta(-\frac{\partial E}{\partial w})$, where w is the adjustable parameters in the fuzzy predictor. The learning equations are the same as Eqs. (21)-(26) if $\frac{\partial r}{\partial y}$ is replaced by $(-\frac{\partial E}{\partial p})$ and the effects caused by this replacement are properly updated, that is, all the terms $[r(t) - p(t)]\left[\frac{\hat{y} - y}{\sigma}\right]_{t-1}$ in Eqs. (21)-(26) are replaced with the term $[r(t) - p(t)]$.

3.3. Multi-Step Fuzzy Predictor

When both the reinforcement signal and input patterns from the environment may depend arbitrarily on the past history of the network output and the network may only receive a reinforcement signal after a long sequence of outputs, the credit assignment problem becomes severe. This *temporal credit assignment* problem results because we need to assign credit or blame to each step individually in such a long sequence for an eventual success or failure. To solve the temporal credit assignment problem, the technique based on the temporal-difference methods [13,14], which are often closely related to the dynamic programming techniques [15], is used. Unlike the single-step prediction or the supervised learning method which assigns credit according to the difference between the predicted and actual output, the temporal-difference methods assign credit according to the difference between temporally successive predictions. See [11] for more details on this multi-step fuzzy predictor.

4. Conclusion

A general connectionist model of a fuzzy logic control system called NN-FLCS was proposed. To incorporate the NN-FLCS with on-line learning ability, an on-line structure/parameter learning algorithm was proposed. This on-line learning algorithm utilized the fuzzy similarity measure and the back propagation to provide a novel scheme to combine the structure learning and the parameter learning such that the whole network structure with correct parameters can be set up on-line. Then two NN-FLCs are closely integrated into a Reinforcement Neural-Network-Based Fuzzy Logic Control System (RNN-FLCS) for solving various reinforcement learning problems. Furthermore, by combining the techniques of temporal difference, stochastic exploration, and the proposed on-line supervised structure/parameter learning algorithm, a reinforcement structure/parameter learning algorithm was derived for the RNN-FLCS. Using the proposed reinforcement learning algorithm, a fuzzy logic controller to control a plant and a fuzzy predictor to model the plant can be set up dynamically through simultaneous structure/parameter learning for various classes of reinforcement learning problems. The proposed RNN-FLCS makes the design of fuzzy logic controllers more practical for real-world applications since it greatly lessens the quality and quantity requirements of the feedback training signals.

5. References

[1] L. A. Zadeh, "Fuzzy logic," *IEEE Computer*, Apr. 1988, pp. 83-93.

[2] M. Sugeno, Ed. *Industrial Applications of Fuzzy Control*, Amsterdam: North-Holland, 1985.

[3] C. C. Lee, "Fuzzy logic in control systems: fuzzy logic controller — part I & II," *IEEE Trans. Syst. Man Cybern.*, Vol. SMC-20, No. 2, 1990, pp. 404-435.

[4] R. Tanscheit and E. M. Scharf, "Experiments with the use of a rule-based self-organizing controller for robotics applications," *Fuzzy Sets Syst.*, Vol. 26, 1988, pp. 195-214.

[5] M. Sugeno and M. Nishida, "Fuzzy control of model car," *Fuzzy sets Syst.*, Vol. 16, 1985, pp. 103-113.

[6] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart, "Distributed representations," in *Parallel Distributed Processing*, Vol. 1, MIT Press, Cambridge, 1986, pp. 77-109.

[7] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing*, Vol. 1, MIT Press, Cambridge, 1986, pp. 318-362.

[8] B. Kosko, "Unsupervised learning in noise," *IEEE Trans. on Neural Networks*, Vol. 1, No. 1, 1990, pp. 44-57.

[9] C. T. Lin and C. S. G. Lee, "Neural-network-based fuzzy logic control and decision system," *IEEE Trans. on Computers*, Vol. C-40, No. 12, pp. 1320-1336, Dec. 1991.

[10] C. T. Lin and C. S. G. Lee, "Real-Time Supervised Structure/parameter Learning for Fuzzy Neural Network," *Proc. of 1992 IEEE Int'l Conf. on Fuzzy Systems*, San Diego, CA, pp. 1283-1290, March 8-12, 1992.

[11] C. T. Lin and C. S. G. Lee, "Reinforcement Structure/Parameter Learning for an Integrated Fuzzy Neural Network," to appear in *IEEE Transactions on Fuzzy Systems*.

[12] J. A. Franklin, "Input space representation for reinforcement learning control," *Proc. of IEEE Int'l Conf. Intelligent Machine*, pp. 115-122, 1989.

[13] 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. 13, No. 5, pp. 834-847, 1983.

[14] R. S. Sutton, "Learning to predict by the methods of temporal difference," *Machine Learning*, Vol. 3, pp. 9-44, 1988.

[15] P. J. Werbos, "A menu of design for reinforcement learning over time," in *Neural Networks for Control*, W. T. Miller, III, R. S. Sutton, and P. J. Werbos, eds, Cambridge: MIT Press, 1990.

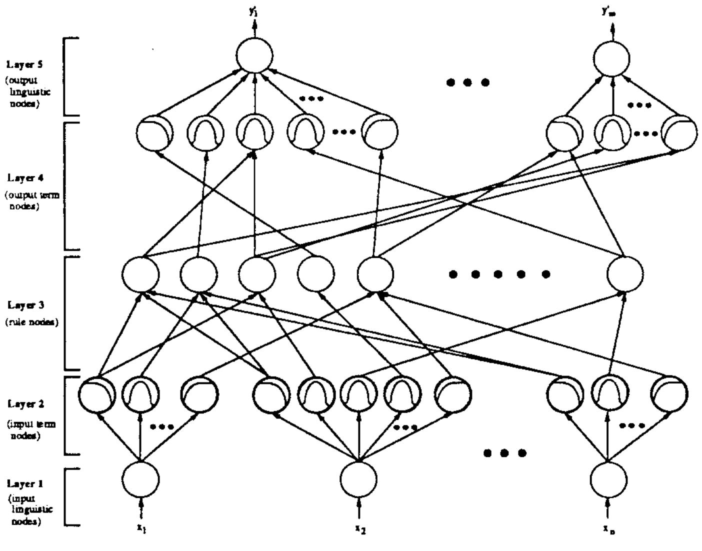
y_1
y_m
Layer 5 (output linguistic nodes)
Layer 4 (output term nodes)
Layer 3 (rule nodes)
Layer 2 (input term nodes)
Layer 1 (input linguistic nodes)
x_1
x_2
x_n

Figure 1. Proposed Neural-Network-Based Fuzzy Logic Control System (NN-FLCS).

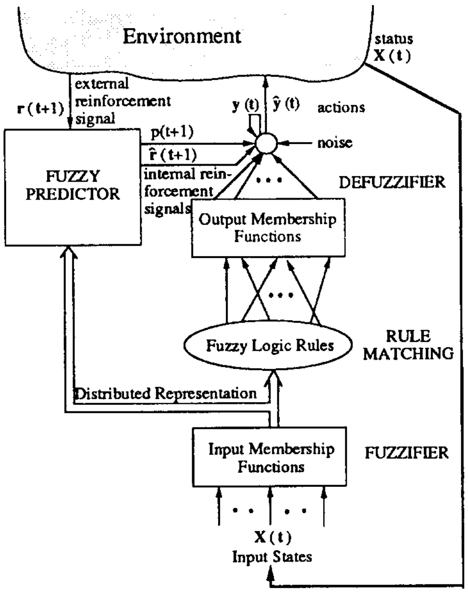

Figure 2. Proposed Reinforcement Neural-Network-Based Fuzzy Logic Control System (RNN-FLCS).