ELSEVIER

Fuzzy Sets and Systems 71 (1995) 25–45

FUZZY sets and systems

Fuzzy adaptive learning control network with on-line neural learning☆

Chin-Teng Lin[a,*], Cheng-Jian Lin[a], C.S. George Lee[b]

[a] *Department of Control Engineering, National Chiao-Tung University, Hsinchu, Taiwan, ROC*
[b] *School of Electrical Engineering, Purdue University, West Lafayette, IN 47907, USA*

Received April 1994; revised June 1994

Abstract

This paper addresses the structure and the associated on-line learning algorithms of a feedforward multilayered connectionist network for realizing the basic elements and functions of a traditional fuzzy logic controller. The proposed Fuzzy Adaptive Learning COntrol Network (FALCON) can be contrasted with the traditional fuzzy logic control systems in their network structure and learning ability. The connectionist structure of the proposed FALCON can be constructed from training examples by neural learning techniques to find proper fuzzy partitions, membership functions, and fuzzy logic rules. Two complementary on-line structure/parameter learning algorithms, FALCON-FSM and FALCON-ART, are proposed for constructing the FALCON dynamically. The FALCON-FSM combines the backpropagation learning scheme for parameter learning and a fuzzy similarity measure for structure learning. The FALCON-FSM can find proper fuzzy logic rules, membership functions, and the size of output partitions simultaneously. In the FALCON-FSM algorithm, the input and output spaces are partitioned into "grids". The grid-typed space partitioning certainly makes both the fuzzy logic controller software emulation and fuzzy chip implementation convenient. However, as the number of input/output variables increases, the number of partitioned grids will grow combinatorially. To avoid the problem of combinatorial growth of partitioned grids in some complex systems, the FALCON-ART algorithm is developed, which can partition the input and output spaces in a more flexible way based on the distribution of the training data. The FALCON-ART combines the backpropagation learning scheme for parameter learning and a fuzzy ART algorithm for structure learning. The FALCON-ART can on-line partition the input and output spaces, tune membership functions and find proper fuzzy logic rules dynamically. Computer simulations were conducted to illustrate the performance and applicability of both FALCON-FSM and FALCON-ART learning algorithms.

Keywords: Fuzzy similarity measure; Fuzzy ART; Fuzzy ARTMAP; Fuzzy logic rule; Fuzzy partition; Membership function; Supervised learning

☆ This work was supported by the ROC National Science Council under grant NSC 82-0408-E-009-429.
* Corresponding author. Tel: (886) 35-712121 ext. 3315. Fax: (886) 715-998. E-mail: ctlin@fnn.cn.nctu.edu.tw. Tel: (317) 494-1384. Fax: (317) 494-6440. E-mail: csglee@ecn.purdue.edu

SSDI 0165-0114(94)00195-2

1. Introduction

Bringing the learning abilities of neural networks to automate and realize the design of fuzzy logic control systems has recently become a very active research area [1–3, 6, 8, 9, 11, 12, 14–17, 18, 19, 21, 25, 27–31]. This integration brings the low-level learning and the computational power of neural networks into fuzzy logic systems, and provides the high-level, human-like thinking and reasoning of fuzzy logic systems into neural networks. Such synergism of integrating neural networks and fuzzy logic systems into a functional system provides a new direction toward the realization of intelligent systems for various applications.

In general, there are three methods that neural networks can be used to facilitate and automate the realization of fuzzy logic controllers [1]. First, one can use a neural network with its neural learning ability as a separate system from the fuzzy system for automatic determination and adjustment of fuzzy rules and/or membership functions. In this case, the fuzzy system is not really affected by the neural architecture. Takagi and Hayashi [25] designated this method as NN-driven fuzzy reasoning. In the second method, the adjustment of the membership functions is simply a gradient method. This gradient method plays the similar role as the one in neural networks and also in any other kind of parametric systems seeking to optimize their parameters. An example is the gradient-based algorithm developed by Nomura et al. [19] for the optimization of the parameters of Sugeno's-type fuzzy systems. The third method is the most popular one. In this method, the fuzzy system can be realized in an architecture isomorphic to neural networks, i.e. a multilayered network, where each node performs a function such as to make the entire network perfectly equivalent to the fuzzy system. In this approach, the gradient-descent method that is akin to the backpropagation algorithm is usually used to train the network. Examples of this method include Lin and Lee's neural-network-based fuzzy logic control system (NN-FLCS) [14–17], Jang's ANFIS (adaptive-network-based fuzzy inference system) [8, 21], Berenji and Khedkar's GARIC (generalized approximate reasoning-based intelligent control) [12, 3] for reinforcement learning problems, Yager's implementation of fuzzy controllers using a neural-network framework [31], Nauck and Kruse's fuzzy backpropagation approach [18], Wang and Mendel's orthogonal least-squares learning [29], and many others [6, 9, 28]. Most of these models belong to off-line training systems; i.e., they require a set of training data available at hand to set up the structure first and then to tune the parameters appropriately. Some of them even require experts to determine the structure in advance. Moreover, they all require the input and output spaces to be partitioned into (fuzzy) grids empirically.

In this paper, we are extending our previous work on neural-network-based fuzzy logic control systems [14] to on-line supervised learning problems. The proposed Fuzzy Adaptive Learning COntrol Network (FALCON) can be constructed automatically by learning from training examples. It can be contrasted with the traditional fuzzy logic control systems in their network structure and learning ability. The "standard" FALCON is a five-layer connectionist structure as shown in Fig. 1. Nodes in layer one are input nodes (linguistic nodes) which represent input linguistic variables. Layer five is the output layer. We have two linguistic nodes for each output variable. One is for training data (desired output) to feed into this net, and the other is for decision signal (actual output) to be pumped out of the net. Nodes in layers two and four are term nodes which act as membership functions to represent the terms of the respective linguistic variable. Each node in 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. The mathematical description of this network will be presented formally in the following sections. This connectionist model maintains the spirit of human-like thinking and reasoning in its network structure. In [14], we have developed a two-phase hybrid learning algorithm which can off-line construct the FALCON automatically through training data. The hybrid learning algorithm combines unsupervised learning and supervised gradient-descent learning procedures to build the rule nodes and to train the membership functions.

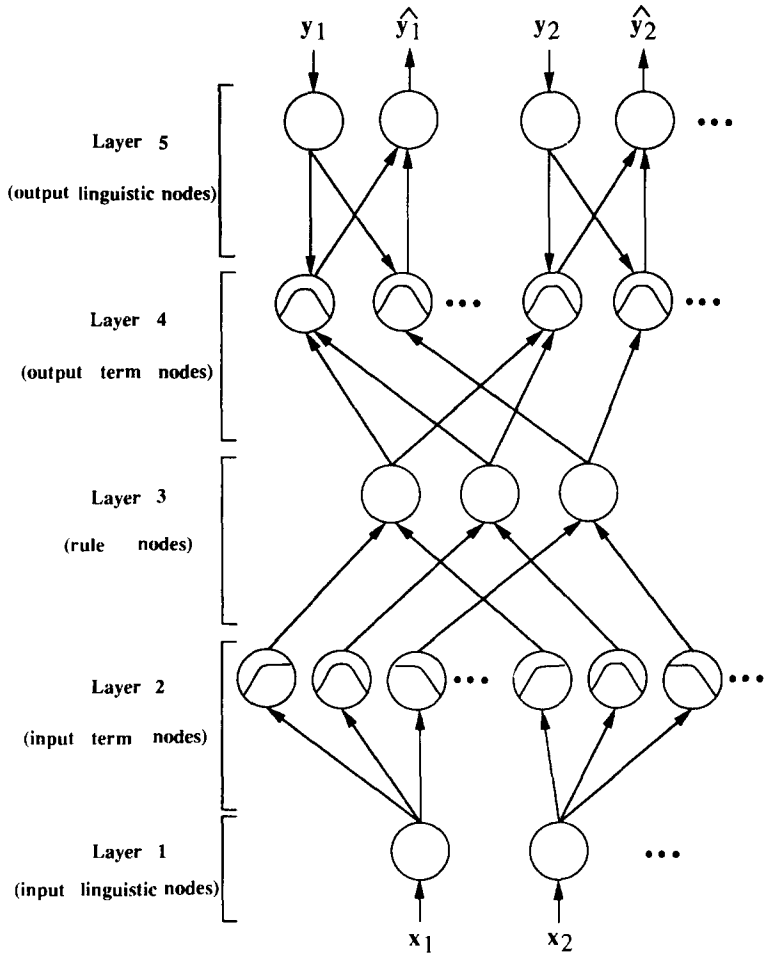

Fig. 1. Proposed fuzzy adaptive learning control network (FALCON).

The hybrid learning algorithm in [14] for the FALCON performs well if sets of training data are available at hand; however, its off-line learning nature makes its application to real-time environment inconvenient. Furthermore, it does not have the ability to increase nodes or change network structure dynamically. In this paper, two complementary on-line structure and parameter learning algorithms are proposed for the FALCON. The first on-line learning algorithm is based on the development of a fuzzy similarity measure (FSM) [15, 16] which is a tool to determine the degree to which two fuzzy sets are equal (or how much they are similar) in contradiction to the traditional "crisp" definition of "equality" in the fuzzy logic theory. This algorithm, called FALCON-FSM, combines the backpropagation learning scheme in [14] for parameter learning and the FSM for structure learning. Based on the FSM, two most similar output membership functions (output term nodes) in the FALCON can be found. Then this FSM will determine whether a new output membership function (output term node) should be added or not, and the rule-node connections are then changed properly. The backpropagation in the original hybrid learning algorithm in [14] is used here again to decide the node output errors in each layer. These errors are used both for the fuzzy similarity

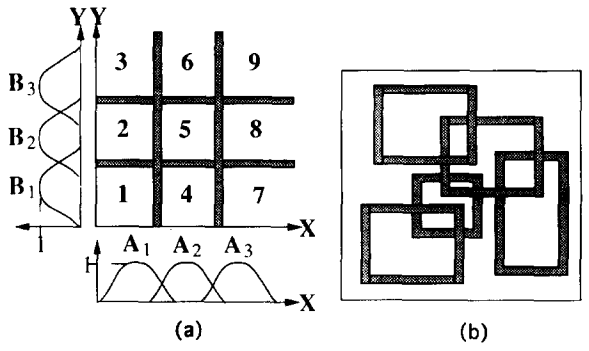

Fig. 2. (a) Grid-type partitioning. (b) Proposed partitioning method.

measure in structure learning and the membership functions adjustment in parameter learning. The proposed FALCON-FSM can find proper fuzzy logic rules, membership functions, and the size of output fuzzy partitions (i.e., the number of output term nodes) simultaneously.

In the FALCON-FSM algorithm, the input and output spaces are partitioned into "grids". Each grid defines a fuzzy region, and the overlapping region between the grids provides a smooth and continuous membership output surface. For example, consider a fuzzy logic controller with two input variables. If each of them is partitioned into three fuzzy terms (e.g., "small", "medium", and "large"), then the corresponding input space partition is as shown in Fig. 2(a). Although during the learning process, the position and shape of membership functions will be changed, they remain grid-typed partitions inherently. The grid-typed space partitioning of input and output spaces has been widely used in many existing fuzzy systems. It certainly makes both the fuzzy logic controller software emulation and fuzzy chip implementation convenient. However, as the number of input and output variables increases, the number of the partitioned grids will grow combinatorially. This creates more difficulty in learning because finer space partitioning needs more training samples (i.e., samples from every fuzzy region); otherwise insufficient learning will occur. Furthermore, the required size of memory and hardware may pose a problem to the designer.

The problem of space partitioning from numerical training data is basically a clustering problem. To avoid the problem of combinatorial growth of partitioned grids in complex systems, a flexible and irregular space partitioning method, based on Carpenter and Grossberg's adaptive resonance theory (ART) [4, 5], is proposed. This flexible and irregular space partitioning method, called FALCON-ART, is an on-line learning algorithm which applies the fuzzy adaptive resonance theory (fuzzy ART) to perform fuzzy clustering in the input and output spaces and determine proper fuzzy logic rules dynamically by associating input clusters with output clusters. The backpropagation learning scheme in [14] is then used again for tuning the input and output membership functions. Hence, the FALCON-ART combines the backpropagation algorithm for parameter learning and the fuzzy ART for structure learning. The FALCON-ART can thus on-line partition the input and output spaces, tune membership functions and find proper fuzzy logic rules dynamically. Fig. 2(b) shows the proposed on-line partitioning method as compared with a fixed-grid space partitioning method.

This paper is organized as follows: Section 2 describes the fuzzy-similarity-measure-based on-line structure/parameter learning algorithm of the FALCON-FSM model. The model car example as suggested by Sugeno is simulated to demonstrate the capabilities and applicability of the FALCON-FSM algorithm. The fuzzy-ART-based on-line structure/parameter learning algorithm (FALCON-ART) which combines fuzzy ART with backpropagation is presented in Section 3. The FALCON-ART was used to predict the

Mackey–Glass chaotic time series to demonstrate its on-line learning capability. Conclusions are then summarized in the last section.

2. A FSM-based fuzzy adaptive learning control network

We shall first describe the five-layer connectionist structure of FALCON (see Fig. 1) and then its associated FSM-based on-line structure/parameter learning algorithm. The proposed FALCON-FSM algorithm blends the fuzzy similarity measure with a supervised gradient-descent learning scheme to perform on-line 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. After establishing the input and output membership functions and the rule-node connectivity, the backpropagation algorithm is used to fine tune the parameters of the input and output membership functions.

2.1. Structure of FALCON-FSM model

The FALCON-FSM is a five-layer connectionist structure. In this five-layered connectionist structure, 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 input and output membership functions (layers two and four, respectively) and fuzzy logic rules (layer 3). The proposed connectionist model maintains the spirit of human thinking and reasoning in its fuzzy logic rules. This connectionist structure also saves the rule- matching time of the inference engine in the traditional fuzzy logic system.

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, and z represents the input signal to a node. In the following equations, superscript is used to indicate the layer number.

Layer 1: The nodes in this layer just transmit input values to the next layer directly, i.e.

$$f = z_i^1, \qquad a = f. \tag{1}$$

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

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

$$f = -\frac{(z_i^2 - m_{ij})^2}{\sigma_{ij}^2}, \qquad a = e^f, \tag{2}$$

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}. If we use a set of nodes to simulate a more complex arbitrary shaped membership function, then the whole subnet is trained on-line to perform the desired membership function by a backpropagation algorithm.

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

$$f = \min(z_1^3, z_2^3, \ldots, z_p^3), \qquad a = f. \tag{3}$$

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

Layer 4: The links at layer four should perform a fuzzy OR operation for the fired rules which have the same consequence,

$$f = \sum_{i=1}^{p} z_i^4, \qquad a = \min(1, f). \tag{4}$$

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 membership functions, respectively, then the following functions can be used to simulate the *center of area* defuzzification method:

$$f = \sum w_{ij}^5 z_i^5 = \sum (m_{ij}^5 \sigma_{ij}^5) z_i^5, \qquad a = \frac{f}{\sum \sigma_{ij}^5 z_i^5}. \tag{5}$$

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

2.2. The FSM-based on-line learning algorithm – FALCON-FSM

One important characteristic of the FALCON-FSM algorithm is that it can learn the network structure and parameters simultaneously. The learning of network structure includes deciding the proper number of output term nodes in layer four and the proper link connections between the nodes in layers three and four. This structure learning also decides the coarse of the output fuzzy partitions and the finding of correct fuzzy logic rules. The learning of network parameters includes the adjustment of the node parameters in layers two and four. This corresponds to the learning of input and output membership functions.

The proposed on-line structure/parameter learning algorithm uses the fuzzy similarity measure to perform the structure learning and uses the backpropagation 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 in layer four), and it will also change the consequences of some fuzzy logic rules properly. After the process of structure learning, the parameter learning will be performed to adjust the parameters of the current input/output membership functions. This structure/parameter learning will be repeated for each on-line incoming training input/output data pair.

To initiate the learning scheme, the desired coarse of input fuzzy partitions (i.e., the size of the term set of each input linguistic variable) and the initial guess of output fuzzy partitions must be provided from the outside world. Based on this information, an initial structure of the network is constructed to initiate the learning process. Then, during the learning process, new nodes may be added and link 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. In its initial structure, there are $\prod_i |T(x_i)|$ rule nodes with the inputs of each rule node coming from one possible combination of the terms of input linguistic variables under the constraint that only one term in a term set can be a rule node's input. If $|T(x_i)|$ denotes the number of terms of x_i (i.e., the number of fuzzy partitions of input state linguistic variable x_i), then the state space is initially divided into $|T(x_1)| \times |T(x_2)| \times \cdots \times |T(x_n)|$ linguistically defined nodes (or fuzzy cells) which represent the preconditions of fuzzy rules. Also, 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 can 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.

Before initiating the proposed learning algorithm, we need the structure initialization and parameter initialization before entering the training loop. The structure initialization is to provide the initial link connection between the nodes in layers three and four, i.e. to decide the initial consequence of each fuzzy logic rule. The parameter initialization is to decide the initial membership functions of input/output linguistic variables. A more efficient way is to use identical membership functions such that their domains can cover the region of corresponding input and output spaces evenly according to the given initial coarses of fuzzy partitions.

After the initialization process, the learning algorithm enters the training loop in which each loop corresponds to a set of training input data $x_i(t)$, $i = 1, \ldots, n$, and the desired output values $y_i(t)$, $i = 1, \ldots, m$, at a specific time t. Basically, the idea of backpropagation [20] is used to find the errors of node outputs in each layer. Then, these errors are analyzed by the FSM to perform structure adjustments or parameter adjustments. The detailed learning rules are derived as follows. Without any loss of generality, we shall consider the case of single output for clarity. The derivation can be easily extended to the multiple-output case. The goal of the proposed learning algorithm is to minimize a least-squares error function

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

where $y(t)$ is the desired output and $\hat{y}(t)$ is the current actual 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 FALCON. Then, starting at the output nodes, a backward pass is used to compute $\partial E/\partial w$ for all the nodes in the hidden layers. 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

$$\begin{aligned} w(t+1) &= w(t) + \eta\left(-\frac{\partial E}{\partial w}\right), \\ \frac{\partial E}{\partial w} &= \frac{\partial E}{\partial a}\frac{\partial a}{\partial w}, \end{aligned} \tag{7}$$

where η is the learning rate. To show the learning rule, we shall show the computations of $\partial E/\partial w$, starting from the output nodes, and layer by layer, to the input layer, and we shall use 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 (7), the adaptive rules of the center m_i and width σ_i can be, respectively, derived as

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

and

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

The error to be propagated to the preceding layer is

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

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

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

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 (Eq. (11)) by measuring their similarity. Let $M(m_i, \sigma_i)$ represent the bell-shaped membership function with center m_i and width σ_i. Let

$$\begin{aligned} \textit{degree}(i,t) &= E[M(m_{i-\text{new}}, \sigma_{i-\text{new}}),\ M(m_{i-\text{closest}}, \sigma_{i-\text{closest}})] \\ &= \max_{1 \leqslant j \leqslant k} E[M(m_{i-\text{new}}, \sigma_{i-\text{new}}),\ M(m_j, \sigma_j)], \end{aligned} \tag{12}$$

where $k = |T(y)|$ is the size of the fuzzy partition of the output linguistic variable $y(t)$ and $E(A, B)$ is the fuzzy similarity measure of fuzzy sets A and B. 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 follows [15]: Assuming $m_1 \geqslant m_2$,

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

Here

$$\begin{aligned} |(A \cap B)| = {} & \frac{1}{2}\frac{h^2(m_2 - m_1 + \sqrt{\pi}(\sigma_1 + \sigma_2))}{\sqrt{\pi}(\sigma_1 + \sigma_2)} + \frac{1}{2}\frac{h^2(m_2 - m_1 + \sqrt{\pi}(\sigma_1 - \sigma_2))}{\sqrt{\pi}(\sigma_2 - \sigma_1)} \\ & + \frac{1}{2}\frac{h^2(m_2 - m_1 - \sqrt{\pi}(\sigma_1 - \sigma_2))}{\sqrt{\pi}(\sigma_1 - \sigma_2)}, \end{aligned} \tag{14}$$

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

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

where $\alpha(t)$ is a monotonically increasing scalar similarity criterion. The structure learning process in the above adjustment is described next.

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 for changing their consequences, since only these rules are fired strongly enough to contribute to the above results of judgment. Assuming that the term node $M(m_i, \sigma_i)$ in layer four has inputs from rule nodes 1, ... , l in layer three, whose corresponding firing strengths are a_i^3's, $i = 1, \ldots, l$, then

IF $a_i^3(t) > \beta$ THEN change the consequence of the ith rule node from $M(m_i, \sigma_i)$ to $M(m_{i-\text{closest}}, \sigma_{i-\text{closest}})$.

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 (10), the error signal δ_i^4 can be derived as

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

In the multiple-output case, the computations in layers four and five are exactly the same as the above and proceed independently for each output linguistic variable.

Layer 3: As in layer four, only the error signals need to be computed. According to Eq. (4), this error signal can be derived as

$$\delta_i^3 = -\frac{\partial E}{\partial a_i} = \delta_i^4 \frac{\partial f^4}{\partial z_i^4} = \delta_i^4. \tag{16}$$

Hence, the error signal is $\delta_i^3 = \delta_i^4$. If there are multiple outputs, then the error signal becomes $\delta_i^3 = \sum_k \delta_k^4$, where the summation is performed over the consequences of a rule node; i.e. the error of a rule node is the summation of the errors of its consequences.

Layer 2: Using Eqs. (2), (3), (7), and (16), the adaptive rules of m_{ij} and σ_{ij} can be, respectively, derived as

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

and

$$\sigma_{ij}(t+1) = \sigma_{ij}(t) - \eta \frac{\partial E}{\partial a_i} e^{f_i} \frac{2(z_i - m_{ij})^2}{\sigma_{ij}^3}, \tag{18}$$

where

$$\frac{\partial E}{\partial a_i} = \sum_k q_k, \tag{19}$$

where the summation is performed over the rule nodes that a_i feeds into, and

$$q_k = \begin{cases} -\delta_k^3 & \text{if } a_i \text{ is minimum in } k\text{th rule node's inputs,} \\ 0 & \text{otherwise.} \end{cases} \tag{20}$$

The proposed supervised learning algorithm provides a novel scheme to combine the structure learning and the parameter learning such that they can be performed simultaneously and on-line. Finally, it should be noted that this backpropagation algorithm can be easily extended to train the membership function which is implemented by a subneural net instead of a single term node in layer two since, from the above analysis, the error signal can be propagated to the output node of the subneural net. Then, by using a similar backpropagation rule in this subneural net, the parameters in this subneural net can be adjusted.

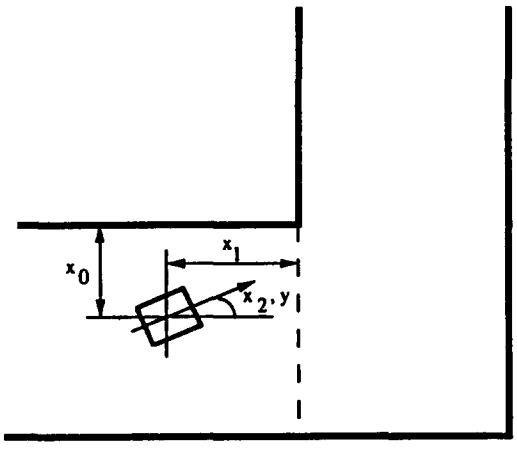

Fig. 3. The state variables of the fuzzy car.

2.3. An illustrative example

In this example, the FALCON-FSM model was used to simulate the control of the fuzzy car conceived by Sugeno [22]. The car has the ability to learn from examples to move automatically along a track with *rectangular turns*, where the path for the car to follow is known a priori and no other path planning is required in this simulation. The goal is to demonstrate that the car can run automatically as it is driven by a skilled driver from some past driving experiences used as training data. The input linguistic variables are x_0, x_1, and x_2, which represent the distance of the car from the side boundary of the track, the distance of the car from the turning point at a corner, and the current steering angle, respectively (see Fig. 3). The output linguistic variable y is the next steering angle. The training data are obtained in the process when an operator guides the fuzzy car along the track as shown in Fig. 6. In the simulation, we set the size of fuzzy partitions of x_0, x_1, and x_2 as 3, 5, and 5, respectively. That is, x_0 has three fuzzy sets ("close", "normal", and "far") in describing the distance of the car from the side of the track. These numbers are kept the same for all the other learning processes as discussed later. The initial guess of the size of fuzzy partitions of the output linguistic variable is set to 3. The initial fuzzy logic rules (the link connections between nodes in layer three and layer four) are set randomly. Also, in the beginning, we use identical membership functions such that their domains can cover the region of corresponding input and output spaces evenly.

After the FSM-based learning process, the number of output fuzzy partitions has finally increased to 10. That is, the learning algorithm adds seven extra term nodes to layer four of the FALCON-FSM. The solid curve in Fig. 4 is the learning curve of the mean error with respect to the number of epochs for this simulation. We can see a large learning error in the beginning of the learning process due to our random choice of initial structure and parameters. For the purpose of comparison, we show the learning curves of the originally proposed two-phase learning algorithm in [14]; these are the two dashed curves in Fig. 4. In the two-phase learning process, the number of output fuzzy partitions is set by the user. Here, two different values, 10 and 15, are used. The upper dashed curve in Fig. 4 is the learning curve when the number of output fuzzy partitions is set to 10, which is the final number we obtained in the FSM-based learning algorithm. The lower dashed curve in Fig. 4 is the learning curve when the number of output fuzzy partitions is set to 15, which is used in [14] and shows a more satisfactory result. The dotted curve in Fig. 4 is the learning curve when using the revised two-phase learning algorithm whose second-phase supervised learning is replaced by

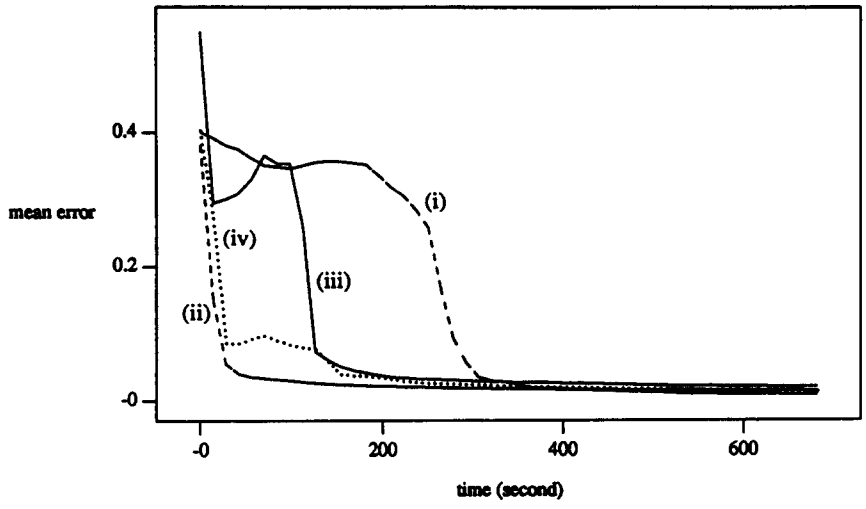

Fig. 4. Learning curves in various fuzzy car simulations.

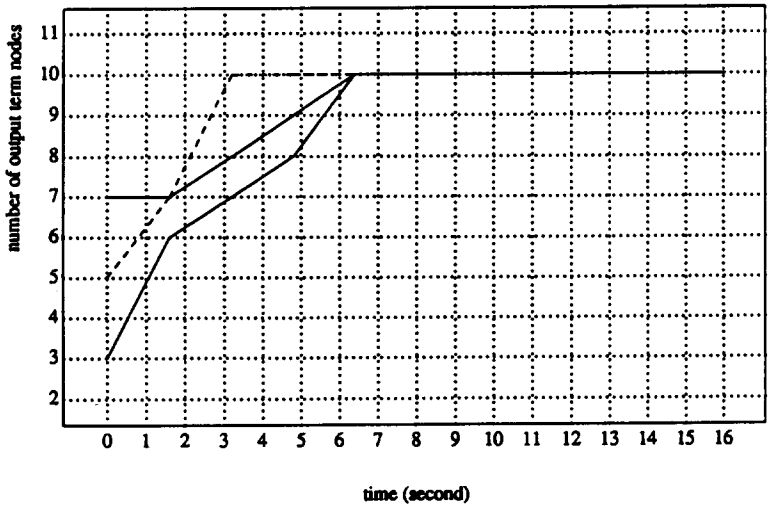

Fig. 5. Learning curves in the structure/parameter learning.

the proposed FALCON-FSM algorithm. In phase one, Kohonen's feature-maps algorithm and a competitive learning law are used to decide the proper initial fuzzy logic rules and membership functions for the phase-two learning [14]. Provided with this kind of a priori knowledge, the proposed structure/parameter learning algorithm produces fewer errors as shown by the dotted curve in Fig. 4. To demonstrate the dynamic increase of output term nodes, three different initial guesses of the number of output fuzzy partitions are used. Fig. 5 shows the growing curves of the number of output term nodes with respect to the number of

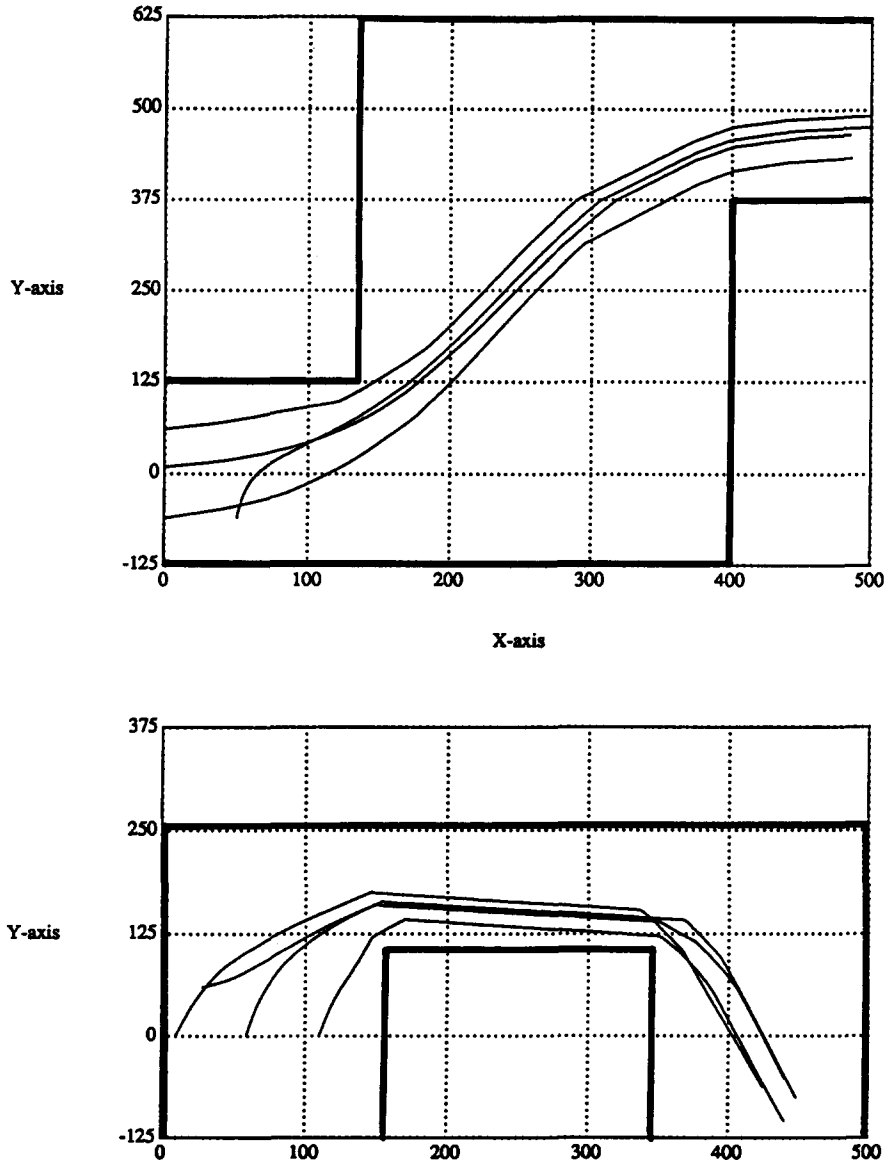

Fig. 6. Simulation results of the fuzzy car running under the control of the proposed FALCON-FSM.

epochs, and we can see that they all reach the final number of 10 term nodes. After the whole connectionist fuzzy logic controller is established, it is used as a fuzzy logic controller to control the fuzzy car. We keep the speed of the car constant and assume there are sensors on the car to measure the state variables x_0, x_1, and x_2 which are fed into the controller to derive the next steering angle. The simulated results are shown in Fig. 6 where different paths, different starting points, and different starting angles are used.

3. An ART-based fuzzy adaptive learning control network

The proposed FSM-based on-line structure/parameter learning algorithm works well when the number of input and output variables is small. For some applications, when the number of input and output variables is large, this will cause a large number of fuzzy regions which in turn will create learning problem, huge memory and hardware implementation problems. A more flexible space partitioning method, based on the fuzzy ART technique, is proposed to solve this fixed-grid-typed space partitioning technique in the FALCON-FSM algorithm. The proposed ART-based on-line learning algorithm, called FALCON-ART, combines the fast-learning fuzzy ART with supervised gradient-descent learning to perform structure and parameter learning simultaneously. The proposed FALCON-ART algorithm uses the fast-learning fuzzy ART to perform structure learning and the backpropagation algorithm to perform parameter learning. The proposed FALCON-ART algorithm is able to partition the input and output spaces into a flexible and irregular form as shown in Fig. 2(b) dynamically and, at the same time, find proper fuzzy rules and optimal membership functions. We shall first describe the structure and function of each layer of the proposed FALCON-ART model, and then its associated fuzzy-ART-based learning algorithm.

3.1. Structure of FALCON-ART model

In this subsection, we shall describe the functions of the nodes in each of the five layers of the FALCON-ART model. Before doing so, we first describe a preprocessing process performed in this model. In the FALCON-ART model, the technique of *complement coding* used in the fuzzy ART [4] is adopted here to normalize the input/output training vectors. Complement coding is a normalization process that rescales an n-dimensional vector in $\mathbb{R}^n, \boldsymbol{x} = (x_1, x_2, \ldots, x_n)^t$, to its $2n$-dimensional complement coding form $\boldsymbol{x}'$ in $[0,1]^{2n}$ such that

$$\boldsymbol{x}' = (x_1', (x_1')^c, x_2', (x_2')^c, \ldots, x_n', (x_n')^c)^t \equiv (\bar{x}_1, 1-\bar{x}_1, \bar{x}_2, 1-\bar{x}_2, \ldots, \bar{x}_n, 1-\bar{x}_n)^t, \tag{21}$$

where $\bar{\boldsymbol{x}} = (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)^t = \boldsymbol{x}/\|\boldsymbol{x}\|$, and the superscript t denotes the transpose operation on a vector/matrix. That is, $x_i' = \bar{x}_i$ and $(x_i')^c = 1 - \bar{x}_i$, for $i = 1, 2, \ldots, n$. As indicated in [4], the complement coding can avoid the problem of proliferation of categories in doing fuzzy clustering using fuzzy ART. It also preserves amplitude information of the training vectors. In applying the complement coding technique to the FALCON-ART, input and output training data are transformed to their complement coding forms in the preprocessing process. These transformed vectors are then used for training the FALCON-ART model.

The functions of the nodes in each layer of the FALCON-ART model are described as follows.

Layer 1: The nodes in this layer just transmit input signals to the next layer directly, i.e.

$$f = z_i^1 = (\bar{x}_i, \bar{x}_i^c) = (\bar{x}_i, 1-\bar{x}_i), \qquad a = f. \tag{22}$$

From the above equation, the link weight at layer one (w_i^1) is unity. Note that for each input node i, there are two input values, $\bar{x}_i$ and $\bar{x}_i^c = 1 - \bar{x}_i$, due to the complement coding process.

Layer 2: Each node in this layer acts as a one-dimensional membership function. The following trapezoidal membership function [24] is used:

$$f = \frac{1}{n}[1 - g(z_i^2 - v_{ij}, \gamma) - g(u_{ij} - z_i^2, \gamma)], \qquad a = f, \tag{23}$$

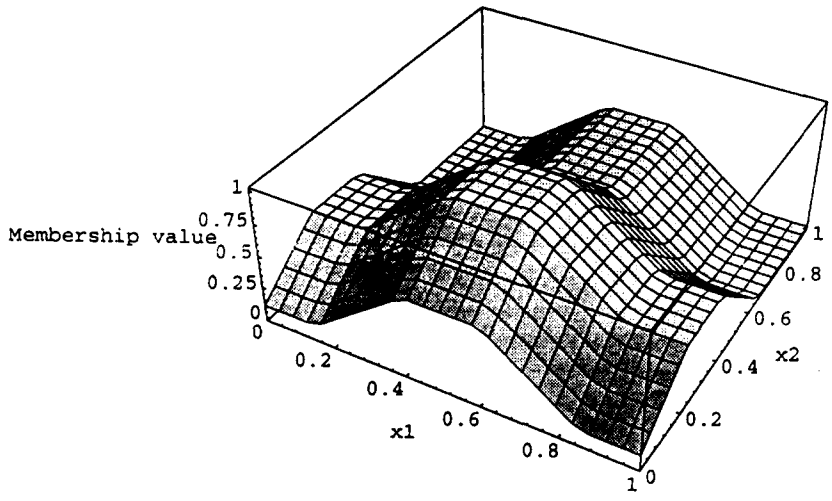

Fig. 7. Two-dimensional trapezoidal membership function.

where u_{ij} and v_{ij} are the left-flat point and the right-flat point, respectively, of the trapezoidal function with $u_{ij} \leqslant v_{ij}$, and

$$g(s,\gamma) = \begin{cases} 1 & \text{if } s\gamma > 1, \\ s\gamma & \text{if } 0 \leqslant s\gamma \leqslant 1, \\ 0 & \text{if } s\gamma < 0. \end{cases} \tag{24}$$

The parameter γ is the sensitivity parameter that regulates the fuzziness of the trapezoidal membership function. When γ is large, the fuzzy set becomes more crisp, and when γ is small the fuzzy set becomes less crisp. There are two weights on each layer-two link. We denote the two weights on the link from input node i (corresponding to the input linguistic variable x_i) to its jth term node as u_{ij} and v_{ij}^c. These two weights defines the membership function in Eq. (23), where $v_{ij} = 1 - v_{ij}^c$.

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

$$f = \sum_i z_i^3, \qquad a = f. \tag{25}$$

The link weight in layer three (w_i^3) is then unity. The summation used in the above equation is equivalent to defining a multi-dimensional membership function, which is the summation of the trapezoidal functions in Eq. (24) over i. This forms a multi-dimensional trapezoidal membership function as shown in Fig. 7. This function is called the *hyperbox membership function* [24] since it is defined on a hyperbox in the input space. The corners (or bounds) of the hyperbox are decided by the layer-two weights, u_{ij} and v_{ij}^c, for all i's. More clearly, the interval $[u_{ij}, v_{ij}] = [u_{ij}, 1 - v_{ij}^c]$ defines the edge of the hyperbox in the ith dimension.

Layer 4: The nodes in this layer have two operation modes: *down–up* transmission and *up–down* transmission modes. In the down–up transmission mode (for regular operation), the links at layer four perform a fuzzy OR operation for the fired rules which have the same consequence,

$$f = \max(z_1^4, z_2^4, \ldots, z_p^4), \qquad a = f. \tag{26}$$

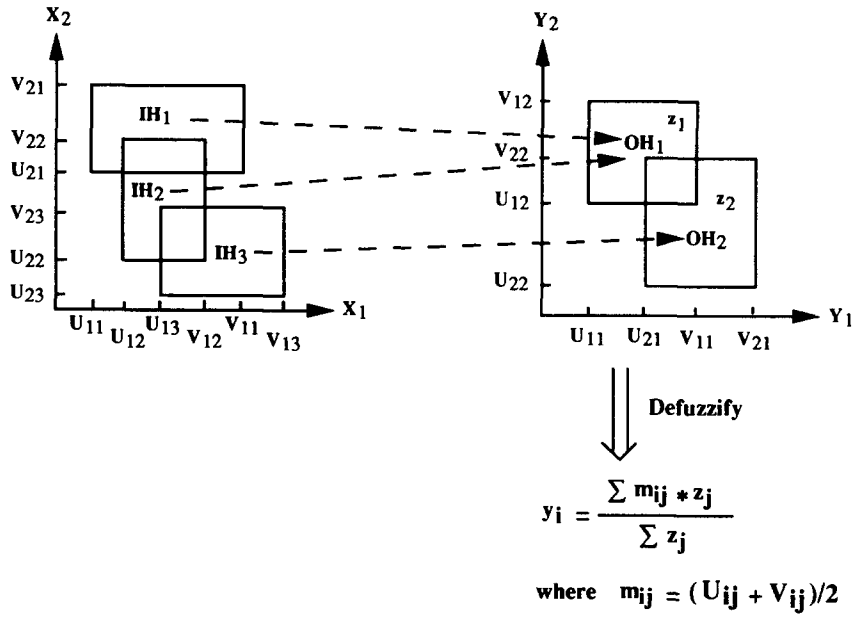

Fig. 8. The fuzzy reasoning process in the FALCON-ART model.

Hence, the link weight $w_i^4 = 1$. In the up–down transmission mode (for training the membership functions), the nodes in this layer and the links in layer-five (up–down transmission links) function exactly the same as those in layer two. Hence, there are also two weights on each of the up–down transmission links in layer five. The weights define hyperboxes on the normalized output space as well as the associated hyperbox membership functions.

Layer 5: There are two kinds of nodes in this layer. The first kind of node performs the up–down transmission for training data to feed into the network. It acts exactly like the input nodes. For this kind of node,

$$f = (\bar{y}_i, \bar{y}_i^c) = (\bar{y}_i, 1 - \bar{y}_i), \qquad a = f, \tag{27}$$

where $\bar{y}_i$ is the ith element of the normalized desired output vector. Note that complement coding is also performed on the desired output vectors. The second kind of node performs the down–up transmission for the decision signal output. These nodes and the layer-five down–up links attached to them act as the defuzzifier. If u_{ij}^5 and v_{ij}^5 are the corners of the hyperbox of the jth term of the ith output linguistic variable y_i, then the following functions can be used to simulate the *center of area* defuzzification method:

$$f = \sum w_{ij}^5 z_i^5 = \sum m_{ij}^5 z_i^5, \qquad a = \frac{f}{\sum z_i^5}, \tag{28}$$

where $m_{ij}^5 = (u_{ij}^5 + v_{ij}^5)/2$ denotes the center value of the output membership function. The center of a fuzzy region is defined as the point that has the smallest absolute value among all the points at which the membership function for this region has membership value equal to one. Here the link weight at layer five (w_{ij}^5) is m_{ij}^5.

The fuzzy reasoning process in the FALCON-ART model is illustrated in Fig. 8, which shows a graphic interpretation of the center of area defuzzification method. Here, we consider a two-input and two-output

case. As shown in the figure, three hyperboxes (IH_1, IH_2, and IH_3) are formed in the input space and two hyperboxes (OH_1 and OH_2) are formed in the output space. These hyperboxes are defined by the weights u_{ij} and v_{ij}. The three fuzzy rules indicated in the figure are "IF $\boldsymbol{x}$ is IH_1, THEN $\boldsymbol{y}$ is OH_1," "IF $\boldsymbol{x}$ is IH_2, THEN $\boldsymbol{y}$ is OH_1," and "IF $\boldsymbol{x}$ is IH_3, THEN $\boldsymbol{y}$ is OH_2," where $\boldsymbol{x} = (x_1, x_2)^t$ and $\boldsymbol{y} = (y_1, y_2)^t$. If an input pattern is located inside the hyperbox, the membership value is equal to one (see Eq. (24)). In this figure, z_1 is obtained by performing a fuzzy OR operation (max operation) on the inferred results of rules 1 and 2, which have the same consequence.

Based on the above structure, the on-line learning algorithm, FALCON-ART, is proposed to determine optimal corners of hyperbox (u_{ij}'s and v_{ij}'s) of term nodes in layers two and four. Also, it will learn fuzzy logic rules and connection types of the links at layers three and four; i.e. the precondition links and the consequence links of the rule nodes.

3.2. The ART-based on-line learning algorithm – FALCON-ART

In this section, an on-line two-phase learning scheme for the proposed FALCON-ART model will be discussed. For an on-line incoming training input/output pair, the following two steps are performed. First, a structure learning scheme is used to decide proper fuzzy partitions and to find the presence of rules. In the second step, a supervised learning scheme is used to optimally adjust the membership functions for the desired outputs. The proposed FALCON-ART algorithm uses the fast-learning fuzzy ART to perform structure learning and the backpropagation algorithm to perform parameter learning. This structure/parameter learning cycle is repeated for each on-line incoming input/output training pair. In this learning method, the users do not need to provide an initial fuzzy partitions of input and output spaces, membership functions and fuzzy logic rules. Hence, there is no input/output term nodes and no rule nodes at the beginning of the learning. The input/output term nodes and the rule nodes are created dynamically as the learning proceeds upon receiving on-line incoming training data. In other words, the initial structure of the network has only input and output linguistic nodes before the network is trained. Then, during the learning process, new input and output term nodes and rule nodes are added as required from the fuzzy ART learning algorithm.

The problem for the structure learning can be stated as: Given the training input data at time t, $x_i(t)$, $i = 1, \ldots, n$, and the desired output values $y_i(t)$, $i = 1, \ldots, m$, we want to decide proper fuzzy partitions of input and output spaces as well as membership functions and find the fuzzy logic rules. In this step, the network works in a two-sided manner; i.e. the nodes and links at layer four are in the up–down transmission mode so that the training input and output data can be fed into this network from both sides.

The *Fuzzy ARTMAP* algorithm proposed by Carpenter et al. [5] is adopted in the structure learning step directly. Given the input/output training pair, this algorithm first performs fuzzy clustering on the input space and the output space individually by using the *fuzzy ART fast learning* algorithm [4]. Each cluster corresponds to a hyperbox in the input or output space. Starting from zero cluster, the number of clusters will grow dynamically for incoming training data. Equivalently, the process of fuzzy clustering is to find the parameters $w_{ij} = (u_{ij}, v_{ij}^c)$ of the input as well as output membership functions. After the hyperboxes in the input and output spaces are tuned or created in the fuzzy clustering process for the current training data pair, the fuzzy ARTMAP algorithm then decides proper mapping between the input clusters and the output clusters. This is done by finding their causal relations. The mapping process directly decides the connections between layer-three nodes and layer-four nodes of the FALCON-ART structure (see Fig. 1). This is equivalent to deciding the consequents of fuzzy logic rules. The aforementioned structure learning step is illustrated in Fig. 8.

After the network structure has been adjusted according to the current training data pair, the network then enters the second learning step to adjust the parameters of the membership functions optimally for the same training data pair. The problem for the parameter learning can be stated as: Given the training input data

$(x_i(t), x_i^c(t))$, $i = 1, \ldots, n$, the desired output values $(y_i(t), y_i^c(t))$, $i = 1, \ldots, m$, the input and output hyperboxes and the fuzzy logic rules, we want to adjust the parameters of the membership functions optimally. These hyperboxes and fuzzy logic rules were learned in the structure learning step. In the parameter learning, the network works in the feedforward manner; that is, the nodes and links in layer four are in the down–up transmission mode. Basically, the idea of backpropagation is used for this parameter learning to find the errors of node outputs in each layer. Then, these errors are analyzed and utilized for parameter adjustments. The goal is again to minimize the least-squares error function in Eq. (6). For a training data pair, starting from the input nodes, a forward pass is used to compute the activity levels of all the nodes in the network. Then starting from the output nodes, a backward pass is used to compute $\partial E/\partial w$ for all the hidden nodes. It is noted that in the parameter learning, we only use the normalized training vectors, $\bar{x}$ and $\bar{y}$ rather than the complement coded ones x' and y'. Again assuming that w is an adjustable parameter in a node, the general learning rule in Eq. (7) is used to derive the learning rules layer by layer using the hyperbox membership functions with corners u_{ij}'s and v_{ij}'s as the adjustable parameters for these computations. For clarity, we derive the learning rules for the single-output case.

Layer 5: Using Eqs. (7) and (28), the updating rules of the corners of hyperbox membership function v_i and u_i can be, respectively, derived as

$$v_i(t+1) = v_i(t) + \eta[y(t) - \hat{y}(t)] \frac{z_i}{2\sum z_i} \tag{29}$$

and

$$u_i(t+1) = u_i(t) + \eta[y(t) - \hat{y}(t)] \frac{z_i}{2\sum z_i}. \tag{30}$$

The error to be propagated to the preceding layer is

$$\delta^5 = -\frac{\partial E}{\partial a^5} = y(t) - \hat{y}(t). \tag{31}$$

Layer 4: In the down–up operation mode, there are no parameter to be adjusted in this layer. Only the error signal (δ_i^4) needs to be computed and propagated. From Eqs. (7) and (28), the error signal δ_i^4 is derived as in the following:

$$\delta_i^4 = -\frac{\partial E}{\partial a^4} = \delta^5 \frac{m_i \sum z_i - \sum m_i z_i}{(\sum z_i)^2}. \tag{32}$$

In the multiple-output case, the computations in layers five and four are exactly the same as the above and proceed independently for each output linguistic variable.

Layer 3: As in layer four, only the error signals need to be computed. According to Eqs. (7), (26), and (32), this error signal can be derived as

$$\delta_i^3 = -\frac{\partial E}{\partial a^3} = \frac{z_i^4}{z_{\max}} \delta_i^4, \tag{33}$$

where $z_{\max} = \max(\text{inputs of output term node } j)$. This term, $z_i^4/z_{\max}$, normalizes the error propagation for the fired rule of the same consequence. If there are multiple outputs, then the error signal becomes $\delta_i^3 = \sum_k (z_k^4/z_{\max})\delta_k^4$, 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. (7), (23), (25), and (33), the updating rules of v_{ij} and u_{ij} can be, respectively, derived as

$$v_{ij}(t+1) = v_{ij}(t) + \eta \frac{\partial a^2}{\partial v_{ij}} \delta_i^3 \tag{34}$$

and

$$u_{ij}(t+1) = u_{ij}(t) + \eta \frac{\partial a^2}{\partial u_{ij}} \delta_i^3, \tag{35}$$

where

$$\frac{\partial a^2}{\partial v_{ij}} = \begin{cases} \gamma/n & \text{if } 0 \leqslant (x_i - v_{ij})\gamma \leqslant 1, \\ 0 & \text{otherwise} \end{cases} \tag{36}$$

and

$$\frac{\partial a^2}{\partial u_{ij}} = \begin{cases} -\gamma/n & \text{if } 0 \leqslant (u_{ij} - x_i)\gamma \leqslant 1, \\ 0 & \text{otherwise.} \end{cases} \tag{37}$$

3.3. An illustrative example

Time-series prediction [30] is an important practical problem. Applications of time-series prediction can be found in the areas of economic and business planning, weather forecasting, control, and other fields. Let $p(k)$, $k = 1, 2, \ldots$ be a time series. The problem of time-series prediction can be formulated as: Given $p(k-m+1), p(k-m+2), \ldots, p(k)$, determine $p(k+l)$, where m and l are fixed positive integers; i.e., determine a mapping from $[p(k-m+1), p(k-m+2), \ldots, p(k)] \in \mathbb{R}^m$ to $[p(k+l)] \in \mathbb{R}$. To illustrate its

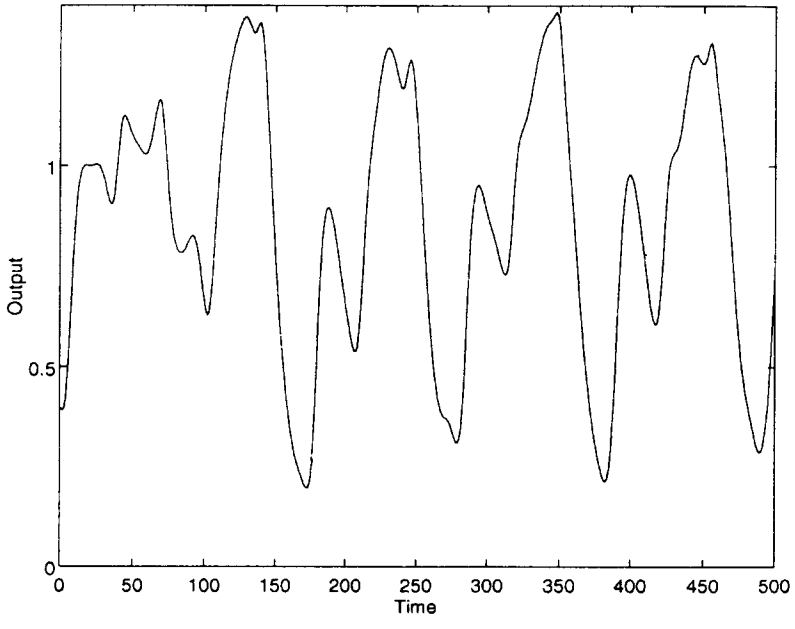

Fig. 9. The Mackey-Glass chaotic time series.

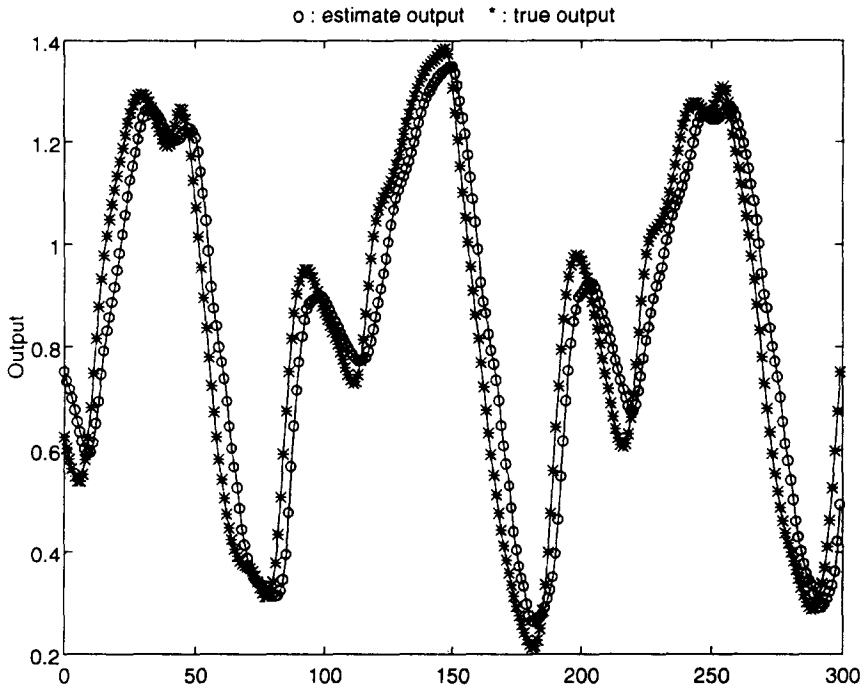

Fig. 10. Simulation results of the prediction time series under the proposed FALCON-ART.

on-line learning ability, the proposed FALCON-ART model is used to predict the Mackey–Glass chaotic time series. The Mackey–Glass chaotic time series is generated from the following delay differential equation:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t), \tag{38}$$

where $\tau > 17$. This equation shows a chaotic behavior. Higher values of τ yield higher dimensional chaos.

In our simulation, we choose the series with $\tau = 30$. Fig. 9 shows 500 points of this chaotic series that are used to test the proposed FALCON-ART model. We choose $m = 9$ and $l = 1$ in our simulation, i.e., nine point values in the series are used to predict the value of the next time point. The first 200 points of the series are used as training data, and the final 300 points are used as test data. After the structure/parameter learning, there are 22 fuzzy logic rules generated in our model. Fig. 10 shows the prediction of the chaotic time series from $x(201)$ to $x(500)$ when 200 training data (from $x(1)$ to $x(200)$) are used. In this figure, predictions of the time series by the FALCON-ART model are represented as O's while true values are represented as *'s. The results showed a near-perfect prediction capability of the FALCON-ART model trained by a small set of training data. The FALCON-ART model has been used to identify the nonlinear dynamic system, control the truck backer-upper, and control the ball and beam system successfully to demonstrate its on-line learning capability. We shall give a detailed explanation in next paper.

4. Conclusion

In this paper, we introduced a general five-layer connectionist model of a fuzzy logic control system called FALCON. Two complementary on-line structure/parameter learning algorithms, FALCON-FSM and

FALCON-ART, were proposed for constructing the FALCON dynamically. The proposed learning algorithms are able to partition the input and output spaces and then find proper fuzzy rules and optimal membership functions. The FALCON-FSM partitions the pattern space into grids and is thus easier for fuzzy chip implementation. The FALCON-ART partitions the pattern space into hyperboxes and thus can void the problem of combinatorial complexity of partitioned grids in some complex systems. Computer simulations demonstrated that the proposed on-line structure/parameter learning algorithms are quite effective in controlling a fuzzy car and predicting a chaotic time series. One problem observed in our simulations is that if we choose the vigilance parameter improperly in the FALCON-ART, redundant rules may be generated. Future research will focus on incorporating adaptive vigilance parameter into the FALCON-ART model for solving the redundant rules problem.

References

[1] H. Bersini, J.P. Nordvik and A. Bonarini, A simple direct adaptive fuzzy controller derived from its neural equivalent, *Proc. 1993 IEEE Internat. Conf. on Fuzzy Systems*, San Francisco (1993) 345–350.

[2] H.R. Berenji, An architecture for designing fuzzy controllers using neural networks, *Int. J. Approximate Reasoning* **6** (1992) 267–292.

[3] H.R. Berenji and P. Khedkar, Learning and tuning fuzzy logic controllers through reinforcements, *IEEE Trans. Neural Networks* **3** (1992) 724–740.

[4] G.A. Carpenter, S. Grossberg and D.B. Rosen, Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system, *Neural Networks* **4** (1991) 759–771.

[5] G.A. Carpenter, S. Grossberg, N. Markuzon, J.H. Reynolds and D.B. Rosen, Fuzzy ARTMAP: a neural network architecture for incremental supervised learning of analog multidimensional maps, *IEEE Trans. Neural Networks* **3** (1992) 698–712.

[6] I. Enbutsu, K. Baba and N. Hara, Fuzzy rule extraction from a multilayered neural network, *Proc. 1991 IEEE Internat. Joint Conf. on Neural Networks*, Seattle (1991) 461–465.

[7] G.E. Hinton, J.L. McClelland and D.E. Rumelhart, *Distributed Representations in Parallel Distributed Processing* 1 (MIT Press, Cambridge, 1986) 77–109.

[8] J.S. Jang, Self-learning fuzzy controllers based on temporal back propagation, *IEEE Trans. Neural Networks* **3** (1992) 741–723.

[9] E. Khan and P. Venkatapuram, Neufuz: neural network based fuzzy logic design algorithms, *Proc. 1993 IEEE Internat. Conf. on Fuzzy Systems*, San Francisco (1993) 647–654.

[10] B. Kosko, Unsupervised learning in noise, *IEEE Trans. Neural Networks* **1** (1990) 44–57.

[11] B. Kosko, *Neural Networks and Fuzzy Systems* (Prentice-Hall, Englewood Cliffs, NJ, 1992).

[12] C.C. Lee and H.R. Berenji, An intelligent controller based on approximate reasoning and reinforcement learning. *Proc. IEEE Internat. Symp. on Intelligent Control 1989*, Albany (1989), 200–205.

[13] C.C. Lee, Fuzzy logic in control systems: fuzzy logic controller – Parts I and II, *IEEE Trans. Systems Man Cybern.* **SMC-20** (1990) 404–435.

[14] C.T. Lin and C.S.G. Lee, Neural-network-based fuzzy logic control and decision system, *IEEE Trans. Comput.* **C-40** (1991) 1320–1336.

[15] C.T. Lin and C.S.G. Lee, Real-time supervised structure/parameter learning for fuzzy neural network, *Proc. 1992 IEEE Internat. Conf. on Fuzzy Systems*, San Diego (1992) 1283–1290.

[16] C.T. Lin and C.S.G. Lee, On the structure and learning of neural-network-based fuzzy logic control systems, *Proc. Fifth Internat. Fuzzy Systems Association World Congress* 2 (1993) 993–996.

[17] C.T. Lin and C.S.G. Lee, Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems, *IEEE Trans. Fuzzy Systems* **2** (1994) 46–63.

[18] D. Nauck and R. Kruse, A fuzzy neural network learning fuzzy control rules and membership functions by fuzzy error backpropagation, *Proc. 1993 IEEE Internat. Conf. on Neural Networks*, San Francisco (1993) 1022–1027.

[19] H. Nomura, I. Hayashi and N. Wakami, A learning method of fuzzy inference rules by descent method, *Proc. 1992 Internat. Joint Conf. on Neural Networks*, Baltimore (1992) 203–210.

[20] D.E. Rumelhart, G.E. Hinton and R.J. Williams, Learning internal representations by error propagation, in: McClelland et al. Eds., *Parallel Distributed Processing* 1 (MIT Press, Cambridge, 1986) 318–362.

[21] C.T. Sun and J.S. Jang, A neuro-fuzzy classifier and its applications, *Proc. 1993 IEEE Internat. Conf. on Fuzzy Systems*, San Francisco (1993) 94–98.

[22] M. Sugeno, Ed., *Industrial Applications of Fuzzy Control* (North-Holland, Amsterdam, 1985).

[23] M. Sugeno and M. Nishida, Fuzzy control of model car, *Fuzzy Sets and Systems* **16** (1985) 103–113.
[24] P.K. Simpson, Fuzzy min-max neural networks – Part 2: Clustering, *IEEE Trans. Fuzzy Systems* **1** (1993) 32–45.
[25] T. Tagaki and I. Hayashi, NN-driven fuzzy reasoning, *Internat. J. Approximate Reasoning* **5** (1991) 191–212.
[26] R. Tanscheit and E.M. Scharf, Experiments with the use of a rule-based self-organizing controller for robotics applications, *Fuzzy Sets and Systems* **26** (1988) 195–214.
[27] P.J. Werbos, Neural control and fuzzy logic: connections and designs, *Internat. J. Approximate Reasoning* **6** (1992) 185–219.
[28] L.X. Wang and J.M. Mendel, Backpropagation fuzzy system as nonlinear dynamic system identifiers, *Proc. 1992 IEEE Internat. Conf. on Fuzzy Systems*, San Diego (1992) 1409–1418.
[29] L.X. Wang and J.M. Mendel, Fuzzy basis functions, universal approximation, and orthogonal least-squares learning, *IEEE Trans. Neural Networks* **3** (1992) 807–814.
[30] L.X. Wang and J.M. Mendel, Generating fuzzy rules by learning from examples, *IEEE Trans. Systems Man Cybern.* **22** (1992) 1414–1427.
[31] R.R. Yager, Implementing fuzzy logic controllers using a neural network framework, *Fuzzy Sets and Systems* **48** (1992) 53–64.
[32] L.A. Zadeh, Fuzzy Logic, *IEEE Comput.* (1988) 83–93.