ELSEVIER

Fuzzy Sets and Systems 71 (1995) 345–357

FUZZY
sets and systems

A fuzzy neural network for rule acquiring on fuzzy control systems☆

J.J. Shann*, H.C. Fu

Department of Computer Science and Information Engineering, National Chiao-Tung University, Hsinchu, Taiwan 300, ROC

Abstract

This paper presents a layer-structured *fuzzy neural network* (FNN) for learning rules of fuzzy-logic control systems. Initially, FNN is constructed to contain all the possible fuzzy rules. We propose a two-phase learning procedure for this network. The first phase is a *error-backprop* (EBP) training, and the second phase is a rule pruning. Since some functions of the nodes in the FNN have the competitive characteristics, the EBP training will converge quickly. After the training, a pruning process is performed to delete redundant rules for obtaining a concise fuzzy rule base. Simulation results show that for the *truck backer-upper* control problem, the training phase learns the knowledge of fuzzy rules in several dozen epochs with an error rate of less than 1%. Moreover, the fuzzy rule base generated by the pruning process contains only 14% of the initial fuzzy rules and is identical to the target fuzzy rule base.

Keywords: Fuzzy logic control; Neural networks; Learning algorithms

1. Introduction

During the past decade, a variety of applications of fuzzy set theory [16] have been implemented in various fields. One of the most important applications is fuzzy logic controllers [13, 7]. Meanwhile, interest in artificial neural networks has grown rapidly after two decades of eclipse. Many network topologies and learning methodologies have been explored. Among these learning methodologies, the backpropagation algorithm [15, 9], i.e., gradient descent supervised learning, has had an enormous influence in research on neural networks.

Recently, more and more research has been published concerning the integration of fuzzy systems and neural networks, with the goal of combining the human inference style and natural language description of fuzzy systems with the learning ability and parallel processing of neural networks [1, 3–5, 8, 14]. Most of this research has been proposed for or can be applied to the knowledge learning of a fuzzy logic controller.

In [8], a multilayer feedforward connectionist model designed for fuzzy logic controllers and decision-making systems was presented. A hybrid two-step learning scheme that combined self-organized and supervised learning algorithms for learning

☆ This work was supported in part by National Science Council of ROC, under grant No: NSC 84-2213-E009-039.

* Corresponding author.

SSDI 0165-0114(94)00277-0

fuzzy logic rules and membership functions was developed. Some heuristics for rule reduction and combination were provided.

In [3, 4], adaptive fuzzy associative memories (AFAM) were proposed to integrate a neural network and fuzzy logic. Unsupervised differential competitive learning (DCL) and product-space clustering adaptively generated fuzzy rules from training samples.

In [14], fuzzy systems were viewed as a three-layer feedforward dedicated network with heterogeneous neurons. The network was trained by backpropagation algorithm for membership function learning.

In [1], a fuzzy modeling method using fuzzy neural networks with the backpropagation algorithm was presented. Three types of fuzzy neural networks, 6, 7, and 10 layers, respectively, were proposed to realize three different types of fuzzy reasoning. These networks can acquire fuzzy inference rules and tune the membership functions of nonlinear systems.

In [5], a fuzzy-set-based hierarchical network for information fusion in computer vision was presented. The proposed scheme could be trained as a neural network in which parametrized families of operators were used as activation functions and the gradient descent and backpropagation learning procedure was performed to generate degrees of satisfaction of various decision criteria by adjusting the parameters of these operators. After training, the network could be interpreted as a set of rules for decision making. Some heuristics were described to eliminate redundant criteria.

In general, most of the methodologies for learning knowledge are in one of the following two categories: backpropagation type and competitive type. Roughly speaking, backpropagation-type learning algorithms learn more precisely than competitive-type algorithms because they are based on the gradient descent search, but they take a long time and numerous training epochs to converge. In contrast, competitive-type learning algorithms learn more rapidly than backpropagation-type algorithms because they are based on unsupervised clustering, but the knowledge learned may not be precise enough. Therefore, one of the goals in the field of knowledge learning is to learn knowledge both precisely and rapidly.

For a fuzzy logic controller, the principal design issues are fuzzification strategies, database, rule base, decision-making logic, and fuzzification strategies [7]. Some items of the design issues, such as the membership functions of the linguistic values of input and output linguistic variables, the fuzzy rules, and the fuzzy operators, etc., might be unknown or uncertain prior to the construction of a controller. Hence, we wish to acquire more knowledge of these items through learning. The proposed fuzzy neural network (FNN) is flexible and extendable for learning different combinations of these items, such as the membership functions of the input and output linguistic variables, the fuzzy rules, and the fuzzy operators [10–12]. However, in this paper, we concentrate on acquiring fuzzy rules of a fuzzy control system in order to observe and analyze the nature of rule learning more thoroughly and deeply. For the FNN presented in this paper, the working process of a fuzzy-logic control system is embedded in the layered structure of the network. The fuzzification strategies, decision-making logic, and defuzzification strategies chosen are formulated as the functions of the nodes in the network; and the fuzzy rules are represented by the learnable weights of the input links on one of the layers in the network. The learning strategy of the FNN is "start large and prune". For this strategy, the network is large initially and the training is performed with the large network. After the training is finished, the network will then be reduced based on some criterion. Therefore, for fuzzy rule learning, the FNN is constructed to contain all the possible fuzzy rules, each with a weak strength, i.e., a small weight, initially according to the related linguistic variables and values of the system.

The learning procedure proposed for the FNN consists of two phases. The first phase is an *error-backprop* (EBP) training phase, and the second phase is a rule-pruning phase. The EBP learning algorithm is based on a gradient descent search in the network, in which some of the node functions are formulated with competitive operations such as the *min* and *max* operators, i.e., some nodes in the network have competitive activation functions. The dominant terms, i.e., the winners, of these competitive operators are determined in the forward pass of the learning algorithm, while in the backward pass, the computation of the gradients for adjusting the learnable weights (see Section 3) are performed only on those links that are related to the dominant terms of the competitive operators. Therefore, the EBP algorithm applied on a network with competitive activation functions enables the network to learn both precisely and quickly, and can be viewed as a compromise between the advantages of backpropagation-type and competitive-type learning methodologies. After the EBP training phase, a rule-pruning process is executed to delete redundant fuzzy rules and obtain a rule base with much smaller size than the initial one. A *truck backer-upper* control problem for backing up a simulated truck to a loading dock in a parking lot was chosen as a benchmark [3]. Simulation results show that for this problem, the EBP training phase is completed in several dozen epochs with a training error of less than 1%. Moreover, the fuzzy rule base generated by the pruning process in the second phase contains only 14% of the initial fuzzy rules and reproduces the target fuzzy rule base with no error.

This paper is organized as follows. The structure of the FNN and the functions of the nodes in the network are described in Section 2. The EBP learning algorithm for the FNN with competitive activation functions is stated in Section 3. A pruning method for deleting redundant fuzzy rules and obtaining a precise (and sound) rule base is described in Section 4. In Section 5, the *truck backer-upper* problem is simulated and the simulation results are analyzed and compared with those for other adaptive controller systems. Finally, our conclusions and plans for future research are presented in Section 6.

2. The structure of the fuzzy neural network

In the fuzzy neural network presented, the fuzzy logic rules considered are the state evaluation fuzzy control rules in linguistic *IF–THEN* form for multiple inputs and single output [7]. The *IF*-part, i.e., the antecedent, of a rule is the conjunction of all input linguistic variables [17], each associated with one of its linguistic values. The *THEN*-part, i.e., the consequent, of a rule contains only one output linguistic variable associated with a linguistic value. Assume that there are M input linguistic variables, $A_1, A_2, \ldots, A_M$, and N output linguistic variables, $F_1, F_2, \ldots, F_N$. The number of linguistic values associated with an input linguistic variable A_h is a_h, and these (input) linguistic values are denoted by $\mathscr{A}_{h,1}, \mathscr{A}_{h,2}, \ldots, \mathscr{A}_{h,a_h}$. The number of linguistic values associated with an output linguistic variable F_l is f_l, and these (output) linguistic values are denoted by $\mathscr{F}_{l,1}, \mathscr{F}_{l,2}, \ldots, \mathscr{F}_{l,f_l}$. Two exemplar fuzzy rules are given below:

$R_{1,1,1}$: If A_1 is $\mathscr{A}_{1,1}$ and A_2 is $\mathscr{A}_{2,1}$ and … and
A_M is $\mathscr{A}_{M,1}$ (AND_1), then F_1 is $\mathscr{F}_{1,1}$,
R_{Q,N,f_N}: If A_1 is $\mathscr{A}_{1,a_1}$ and A_2 is $\mathscr{A}_{2,a_2}$ and … and
A_M is $\mathscr{A}_{M,a_M}$ (AND_Q), then F_N is $\mathscr{F}_{N,f_N}$,

where the rules are numbered $R_{i,j,k}$, indicating the fuzzy rule with antecedent AND_i, output linguistic variable F_j, and the output linguistic value $\mathscr{F}_{j,k}$.

The fuzzy neural network is a five-layer dedicated neural network, as shown in Fig. 1, designed according to the working process of fuzzy controller systems [7, 8]. The adjustable weights for rule learning are on the input links of the nodes in Layer IV. Initially, it is constructed to contain all the possible rules of a fuzzy control system. The number of nodes and connections of the initial network are summarized in Table 1. The semantic meaning and functions of the nodes in the proposed network are as follows.

Layer I (*input layer*): Each node in Layer I represents an input linguistic variable of the network and is used as a buffer to broadcast the input to the next layer, i.e., to the membership-function nodes of its linguistic values. The range (or space) of each input linguistic variable is determined by the application and need not be constrained to within [0, 1].

Layer II (*membership-function layer*): Each node in Layer II represents the membership function of a linguistic value associated with an input linguistic variable. These nodes are called MF nodes. The output of such an MF node is in the range [0, 1] and represents the membership grade of the input with respect to the membership function. Therefore, an MF node is a fuzzifier. The most commonly used membership functions are in the shape of triangle, trapezoid, and bell. The functions of the nodes in this layer are determined and formulated by applications. The weights of the input links in this layer are unity.

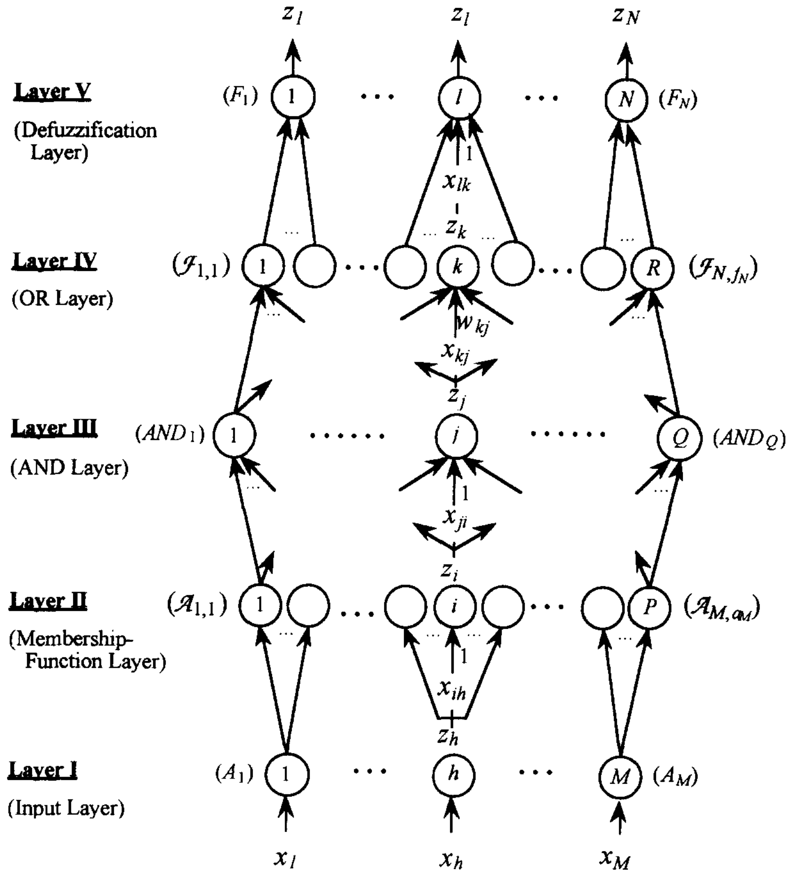


Fig. 1. The network structure of the fuzzy neural network.

Table 1
Summary of the nodes and links in the initial fuzzy neural network

Layer	For a node in the layer			Total		
	Index	# Input links	# Output links	# Nodes	# Input links	# Output links
I	h	1	a_h	M	M	P
II	i	1	Q/a_h	P	P	MQ
III	j	M	R	Q	MQ	QR, QN
IV	k	Q	1	R	QR, QN	R
V	l	f_l	1	N	R	N

Note: M, the number of the input linguistic variables; N, the number of the output linguistic variables; A_h, an input linguistic variable represented by node h in Layer I; a_h, the number of the linguistic values of A_h; F_l, an output linguistic variable represented by node l in Layer V; f_l, the number of the linguistic values of F_l; $P = \sum_{h=1}^{M}(a_h)$; $Q = \prod_{h=1}^{M}(a_h)$; $R = \sum_{l=1}^{N}(f_l)$; QR, the number of the initial fuzzy rules; QN, the number of the rules in a sound fuzzy rule base.

Layer III (*AND layer*): Each node in Layer III represents a possible *IF*-part for fuzzy rules. These nodes are called AND nodes. In fuzzy set theory, there are many different operators for fuzzy intersection [2, 18]. We choose the most commonly used one, i.e., the *min*-operator suggested by Zadeh [16], as the function of an AND node. The main reason is that the *min*-operator is simple and effective and has strong characteristics of competition. Therefore, the operation performed by node j in Layer III is defined as follows:

$$z_j = \min_{i \in P_j}(x_{ji}) = MIN_j, \tag{1}$$

where P_j is the set of indices of the nodes in Layer II that have an output link connected to node j and $x_{ji} = z_i$. The link weights w_{ji} on the input links in this layer are unity. From Eq. (1) for z_j, it is obvious that the output value of node j in Layer III is determined by the output of a node i in Layer II, which provides the minimum among all the output values of the nodes connected to node j. Node i is called a dominant node of node j.

Layer IV (*OR layer*): Each node in Layer IV represents a possible *THEN*-part for fuzzy rules. The nodes in this layer are called OR nodes. The operation performed by an OR node is to combine fuzzy rules with the same consequent. Initially, the links between Layers III and IV are fully connected so that all the possible fuzzy rules are embedded in the structure of the network. The weight w_{kj} of an input link in Layer IV represents the certainty factor of a fuzzy rule, which comprises the AND node j in Layer III as the *IF*-part and the OR node k in Layer IV as the *THEN*-part. Hence, these weights are adjustable while learning the knowledge of fuzzy rules.

In fuzzy set theory, there are many different operators for fuzzy union [2, 18]. We choose the most commonly used one, i.e., the *max*-operator suggested by Zadeh [16], as the function of an AND node for the same reason as that of the *min*-operator. Therefore, the operation performed by a node k in Layer IV is defined as follows:

$$z_k = \max_{j \in P_k}(x_{kj} w_{kj}) = MAX_k, \tag{2}$$

where P_k is the set of indices of the nodes in Layer III that have an output link connected to node k and $x_{kj} = z_j$. During the training phase, the link weights w_{kj} in Layer IV are learnable nonnegative real numbers.

As was the case with Eq. (1) for z_j, from Eq. (2) for z_k, it is obvious that the output value of node k in Layer IV is determined by the maximum among the products of the output values x_{kj} of the nodes in the previous layer and the weights w_{kj} associated with the links connected to node k. Suppose that the maximal product is provided by node j in Layer III. Then, node j is called a dominant node of node k.

Layer V (*defuzzification layer*): Each node in Layer V represents an output linguistic variable and performs defuzzification, taking into consideration the effects of all the membership functions of the linguistic values of the output.

Suppose that the correlation-product inference and the fuzzy centroid defuzzification scheme [4] are used. Then the function of node l in Layer V is defined as follows:

$$z_l = \frac{\sum_{k \in P_l}(x_{lk} a_{lk} c_{lk})}{\sum_{k \in P_l}(x_{lk} a_{lk})}, \tag{3}$$

where P_l is the set of indices of the nodes in Layer IV that have an output link connected to node l, $x_{lk} = z_k$, and a_{lk} and c_{lk} are the area and centroid of the membership function of the output linguistic value represented by node k in Layer IV, respectively. Since it is assumed that the membership functions of the output linguistic values are known, the areas and centroids can be calculated before learning. The link weights of Layer V are unity.

3. Error backpropagation (EBP) learning algorithm for the FNN with competitive node functions

In conventional neural networks, most of the neurons perform summation and sigmoid functions. However, more and more alternatives for these functions have been presented while the applications of neural networks explored recently [6]. In this section, we describe the derivation of the EBP learning algorithm for the FNN with the node functions defined in the previous section.

In the training phase, the concept of error backpropagation is used to minimize the least mean square (LMS) error function:

$$E = \frac{1}{2} \sum_{l=1}^{N} (T_l - z_l)^2, \tag{4}$$

where N is the number of nodes in Layer V and T_l and z_l are the target and actual outputs of the node l in Layer V, respectively. The methods for adjusting the learnable weights w_{kj} in Layer IV of the fuzzy rules are based on the gradient descent search.

Let the delta value, δ, of a node in the network be defined as the influence of the node output with respect to E. The definition of the delta values δ_k and δ_l for nodes in Layers IV and V, the evaluation of the gradients of E ($\nabla E_{w_{kj}}$) with respect to the learnable weights, and the adjustments of the weights (Δw_{kj}) are described as follows:

The definition of the Delta values: For a defuzzification node l in Layer V, the delta value δ_l is defined as follows:

$$\delta_l = \frac{\partial E}{\partial z_l} = -(T_l - z_l). \tag{5}$$

For an OR node k in Layer IV, the Delta value δ_k is defined as follows:

$$\begin{aligned}\delta_k &= \frac{\partial E}{\partial z_k} = \frac{\partial E}{\partial z_l} \frac{\partial z_l}{\partial z_k} \\ &= \delta_l \frac{a_{lk}(c_{lk} - z_l)}{\sum_{k' \in P_l}(x_{lk'} a_{lk'})},\end{aligned} \tag{6}$$

where the output link of node k is connected to exactly one node l in Layer V.

The evaluation of the gradients of E: For the weight w_{kj} of the link connected from node j in Layer III to node k in Layer IV, the definition of the gradient of E with respect to w_{kj} is defined as follows:

$$\begin{aligned}\nabla E_{w_{kj}} &= \frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial z_k} \frac{\partial z_k}{\partial w_{kj}} \\ &= \delta_k \cdot \begin{cases} x_{kj} & \text{if } x_{kj} w_{kj} = MAX_k, \\ 0 & \text{otherwise.} \end{cases}\end{aligned} \tag{7}$$

The two cases in Eq. (7) for $\nabla E_{w_{kj}}$ indicate that when the EBP training algorithm is processed, there will be different adjustments in the backward pass depending on the situation that occurs in the forward pass. It is obvious that only the weight of the link emitted from a dominant node of node k has a nonzero gradient, i.e., only the weight will be adjusted in the backward pass of the learning algorithm, due to the competitive characteristics of the function performed by the OR node.

The adjustments of the learnable weights: The adjustments of the learnable weights w_{kj}, which are based on the gradient descent search, can be described as follows:

$$\begin{aligned}w_{kj}(t+1) &= w_{kj}(t) + \Delta w_{kj} \\ &= w_{kj}(t) - \beta \nabla E_{w_{kj}},\end{aligned} \tag{8}$$

where β is the learning rate.

The knowledge of the fuzzy rules learned by the EBP algorithm is distributed over the weights of the links between Layers III and IV.

4. The rule-pruning method

Since the knowledge of fuzzy rules learned by EBP algorithm is distributed over the adjustable weights, most of the weights after training have nonzero positive values. Then the fuzzy rule base obtained directly right after the training phase will contain all the rules with nonzero weight, i.e., certainty factor. This causes the size of the rule base very large in most cases. Therefore, a pruning process is required to obtain a concise rule base. At the meantime, the structure of the FNN will also be reduced.

The physical meaning of the weights on the input links in Layer IV is explained in the following. After the EBP training, the learned weights w_{kj} on a set of links from an AND node j in Layer III to the OR nodes of an output linguistic variable F_l (see Fig. 2) are interpreted as the certainty factors of a set of fuzzy rules that have the same *IF*-part and the same output linguistic variable but different linguistic values. These fuzzy rules will be referred to as "incompatible" rules. Fig. 2 can be interpreted as the following incompatible fuzzy rules:

$R_{j,l,1}$: If AND_j, then F_l is $\mathscr{F}_{l,1}$ $(w_{k_1,j})$.

$R_{j,l,2}$: If AND_j, then F_l is $\mathscr{F}_{l,2}$ $(w_{k_2,j})$.

$\vdots$

$R_{j,l,r}$: If AND_j, then F_l is $\mathscr{F}_{l,r}$ $(w_{k_r,j})$.

$\vdots$

R_{j,l,f_l}: If AND_j, then F_l is $\mathscr{F}_{l,f_l}$ $(w_{k_{f_l},j})$.

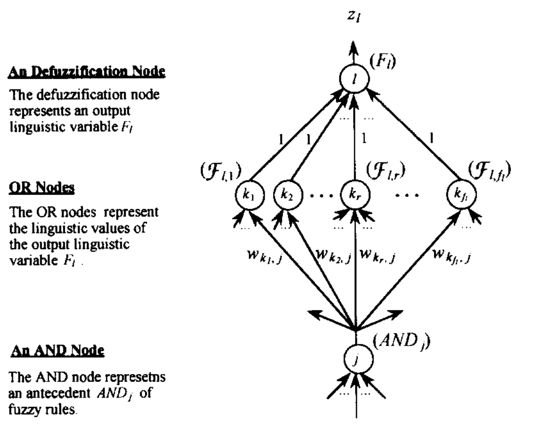

z_l
(F_l)
l
An Defuzzification Node
The defuzzification node represents an output linguistic variable F_l
1
1
1
1
$(\mathcal{F}_{l,1})$
$(\mathcal{F}_{l,r})$
$(\mathcal{F}_{l,f_l})$
k_1
k_2
k_r
k_{f_l}
OR Nodes
The OR nodes represent the linguistic values of the output linguistic variable F_l .
$w_{k_1,j}$
$w_{k_2,j}$
$w_{k_r,j}$
$w_{k_{f_l},j}$
(AND_j)
j
An AND Node
The AND node represetns an antecedent AND_j of fuzzy rules.

Fig. 2. The diagram of the possible fuzzy rules with identical antecedent AND_j for an output linguistic variable F_l.

where AND_j is the antecedent represented by an AND node j in Layer III. The effects of these rules are that when the antecedent AND_j holds, each of the rules is activated to a certain degree represented by the weight value (the certainty factor) associated with that rule. In the previous research that related to the learning of fuzzy rules, only a few discussed about the pruning of the rules and the methods provided were heuristic, such as deleting the rules with weak strengths, i.e., small weights, or choosing the one with the maximum weights among the incompatible rules and deleting the others [8].

In this paper, a method for pruning fuzzy rules is proposed based on the following derivation. For defuzzification node l, the corresponding output with respect to AND node j (and the set of incompatible rules listed above) is denoted by z_l^j and is evaluated as follows:

$$z_l^j = \frac{\sum_{k \in (P_l \cap N_j)} (x_{lk} a_{lk} c_{lk})}{\sum_{k \in (P_l \cap N_j)} (x_{lk} a_{lk})},$$

where

$$x_{lk} = z_k = x_{kj} w_{kj} = z_j w_{kj}.$$

Thus,

$$z_l^j = \frac{\sum_{k \in P_l} (z_j w_{kj} a_{lk} c_{lk})}{\sum_{k \in P_l} (z_j w_{kj} a_{lk})},$$

$$= \frac{\sum_{k \in P_l} (w_{kj} a_{lk} c_{lk})}{\sum_{k \in P_l} (w_{kj} a_{lk})}. \tag{9}$$

From Eq. (9), z_l^j is the centroid of the weighted linguistic values of F_l, where the weights of the linguistic values is equal to the certainty factors of the set of incompatible rules listed above. Hence, z_l^j can also be called the centroid of the set of incompatible rules.

From the illustration given above, it is obvious that the total effects of the set of incompatible rules on the output linguistic variable F_l is equivalent to the centroid of the incompatible rules. Therefore, for rule pruning, an evaluation equation is proposed based on the concept of the centroid of gravity.

The equation for evaluating the centroid of the incompatible fuzzy rules represented by the links from an AND node j in Layer III to the OR nodes of an output linguistic variable F_l is defined as follows:

$$C_{lj} = \frac{\sum_{k \in P_l} (w_{kj} a_{lk} c_{lk})}{\sum_{k \in P_l} (w_{kj} a_{lk})}. \tag{10}$$

For each linguistic value of the output linguistic variable F_l, we define an interval in the space of F_l according to the membership function of the linguistic value and the requirements of the application. The basic criterion for defining the intervals is that they should cover the entire space of F_l. If the centroid C_{lj} computed for the set of incompatible rules listed above is in the interval of a linguistic value $\mathcal{F}_{l,r}$, then the corresponding rule $R_{j,l,r}$ is not redundant and remains in the rule base. In the set of the incompatible rules, all the rules other than the remaining ones are eliminated.

For each AND node in Layer III, the pruning process is performed to delete its redundant output links connected to the OR nodes associated with each output linguistic variable. After the pruning phase, a rule base with much smaller size can be obtained.

In practice, after an fuzzy rule learning, a sound fuzzy rule base is expected to be obtained for many applications. A sound fuzzy rule base is defined as: for each output linguistic variable, there exists exactly one consequent for each possible antecedent in the rule base. In an application, the fuzzy rule base obtained after the pruning phase can be sound if the intervals defined for the linguistic values of each output linguistic variable are nonoverlapped.

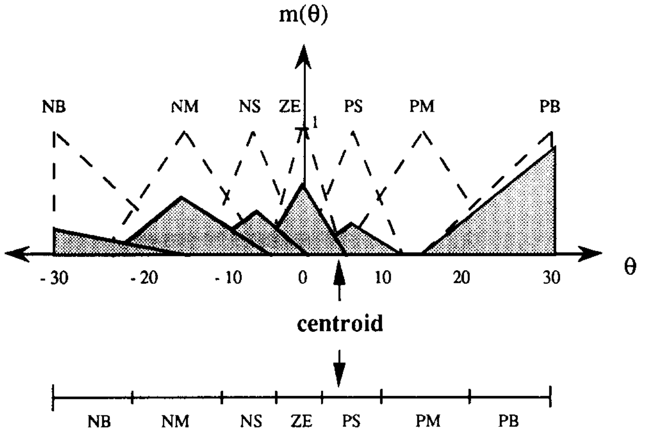


Fig. 3. The effects of the set of incompatible rules with respect to the output linguistic variable θ. The space of θ is divided into seven nonoverlapped intervals. The centroid calculated by Eq. (10) for this set of incompatible rules is in the interval of linguistic value *PS*.

After rule pruning, the fuzzy rule base obtained may be either sound or containing incompatible rules. For a sound fuzzy rule base, the certainty factors of the rules are set to unity. For a fuzzy rule base containing incompatible rules, the certainty factors of the rules can be determined by processing the EBP training once more.

An example is given below for illustrating the rule pruning. From the simulation results of the *truck backer-upper* problem (see Section 5), one of the sets of incompatible fuzzy rules obtained after the EBP training phase is listed as follows:

If x is *LE* and ϕ is *RB*, then θ is *NB*, (0.18337).

If x is *LE* and ϕ is *RB*, then θ is *NM*, (0.42997).

If x is *LE* and ϕ is *RB*, then θ is *NS*, (0.33736).

If x is *LE* and ϕ is *RB*, then θ is *ZE*, (0.52446).

If x is *LE* and ϕ is *RB*, then θ is *PS*, (0.23184).

If x is *LE* and ϕ is *RB*, then θ is *PM*, (0.00379).

If x is *LE* and ϕ is *RB*, then θ is *PB*, (0.84130).

The values of the certainty factors shown in the parentheses are the simulation results of the problem when the space of the output linguistic variable θ was normalized to $[0, 1]$. When the antecedent of the set of incompatible rules holds, the relative effects of the rules on the consequent are shown in Fig. 3. In this figure, the dashed lines represent the membership functions of the seven linguistic values for the output linguistic variable θ as the same as that shown in Fig. 4(b), and the shaded area represents the relative effect of these incompatible rules with respect to θ. The seven intervals defined for the output linguistic values of θ are nonoverlapped and cover the entire space of θ as shown in Figs. 3 and 4(b). The centroid calculated for this set of incompatible rules is in the interval of linguistic value *PS*. Therefore, only the rule

If x is *LE* and ϕ is *RB*, then θ is *PS*

remains after the rule-pruning phase. The remaining rule is identical to the one specified at the left top square of the target fuzzy rule base shown in Fig. 5.

5. System simulation and evaluation

A general purpose simulator of the proposed fuzzy neural network was implemented on a Sun SPARC station. The *truck backer-upper* control problem [3] was used as a benchmark to evaluate the performance of the proposed network. In this fuzzy control system, the truck x-position coordinate

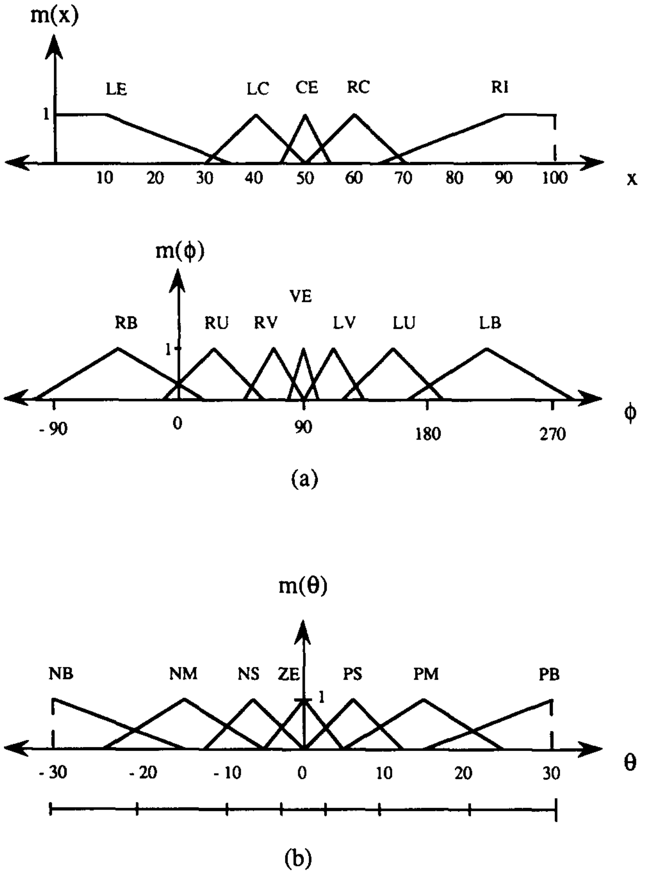


Fig. 4. Fuzzy membership functions of the linguistic values associated with (a) input linguistic variables x and ϕ and (b) output linguistic variable θ for the *truck backer-upper* problem. The space of θ is divided into seven nonoverlapped intervals.

		x				
		LE	LC	CE	RC	RI
	RB	PS	PM	PM	PB	PB
	RU	NS	PS	PM	PB	PB
	RV	NM	NS	PS	PM	PB
ϕ	VE	NM	NM	ZE	PM	PM
	LV	NB	NM	NS	PS	PM
	LU	NB	NB	NM	NS	PS
	LB	NB	NB	NM	NM	NS

Fig. 5. The target FAM-bank matrix for the *truck backer-upper* controller.

x and the truck angle ϕ were chosen as the input linguistic variables and the steering-angle control signal θ was chosen as the output linguistic variable. The input linguistic variable x had five linguistic values: *LE*, *LC*, *CE*, *RC*, and *RI*; the input linguistic variable ϕ had seven linguistic values: *RB*, *RU*, *RV*, *VE*, *LV*, *LU*, and *LB*; and the output linguistic variable θ had seven linguistic values: *NB*, *NM*, *NS*, *ZE*, *PS*, *PM*, and *PB*. The membership functions of the linguistic values of these linguistic variables are given in Fig. 4. There were 245 ($= 5 \times 7 \times 7$) possible fuzzy rules. A target fuzzy rule base, referred to as the FAM-bank matrix, is specified in Fig. 5. In the simulation of the different adaptive controller systems, the truck trajectories

				x		
		LE	LC	CE	RC	RI
	RB	PS	*PB	*PB	PB	PB
	RU	*NM	*ZE	PM	PB	PB
	RV	NM	*NM	*NS	*PS	PB
ϕ	VE	NM	NM	*NM	*ZE	*PB
	LV	*NM	NM	*NM	*NS	*PB
	LU	*NM	*NM	NM	*NM	*PM
	LB	*NM	*NM	NM	NM	*NM

* : a fuzzy rule that is different from the target one

(a)

				x		
		LE	LC	CE	RC	RI
	RB	PS	PM	PM	PB	PB
	RU	*ZE	*PM	PM	PB	PB
	RV	*NS	*ZE	PS	PM	PB
ϕ	VE	NM	NM	ZE	PM	PM
	LV	NB	*NB	NS	PS	*PS
	LU	NB	NB	*NB	NS	PS
	LB	NB	NB	NM	NM	NS

(b)

Fig. 6. (a) The FAM bank generated by the neural control system. (b) The DCL-estimated FAM bank.

produced according to the membership functions and the target fuzzy rule base shown in Figs. 4 and 5, respectively, were regarded as the ideal trajectories and were used as the training samples [3]. Besides, the target FAM-bank matrix was used as a basis for comparing the performance of different adaptive controller systems.

In [3], two adaptive controller systems with different learning methodologies were analyzed. One was a neural system, which consisted of a multilayer feedforward neural network with the conventional backpropagation gradient-descent algorithm. The neural system was trained by applying 35 sample vectors to the network, and more than 100000 iterations (epochs) were required to complete the training process. The second system was the adaptive fuzzy *truck backer-upper*, which consisted of a laterally inhibitive differential competitive learning (DCL) network trained with the DCL algorithm. The adaptive fuzzy system was trained by applying 2230 sample vectors to it. In order to generate the fuzzy rules, i.e., the FAM banks, product-space clustering was applied to the training results of the neural system and the adaptive fuzzy system. The FAM banks generated by these two different adaptive controller systems, are shown in Fig. 6.

To illustrate briefly the performance of these two adaptive controller systems, a set of random input vectors were tested to estimate the retrieving error rates of the FAM banks generated by these two systems in comparison with the target FAM bank in Fig. 5. The error rates of the neural system and the adaptive fuzzy system were 11.42% and 3.74%, respectively.

In the simulations for our fuzzy neural network, the fuzzy membership functions and the FAM-bank matrix shown in Figs. 4 and 5, respectively, were used to generate the sample vectors for training. A total of 350 training samples were generated randomly, about 10 samples in the product space of each antecedent. The network contained 2 neurons in Layer I, 12 ($= 5 + 7$) neurons in Layer II, 35 ($= 5 \times 7$) neurons in Layer III, 7 neurons in Layer IV, and 1 neuron in Layer V. Initially, there were 245 ($= 5 \times 7 \times 7$) links between Layers III and IV; each link represented a possible fuzzy rule.

During the EBP training of fuzzy rules in the first phase of the proposed learning procedure, the training error was reduced very quickly in each epoch iteration and the learning was completed in a small number of epochs. Simulation results for the EBP training with random initial weights on the input links of Layer IV are shown in Fig. 7. The training iteration was terminated after the error rate was reduced to 1%.

After the EBP training, the rule-pruning process was performed in the second phase of the learning procedure. The space of the output linguistic variable θ, $-30 \leqslant \theta \leqslant 30$, was divided into seven nonoverlapped intervals, shown in Fig. 4(b), corresponding to the seven linguistic values of θ. Only 35 of the 245 initial fuzzy rules remained after pruning. For analysis, a FAM bank was generated by the

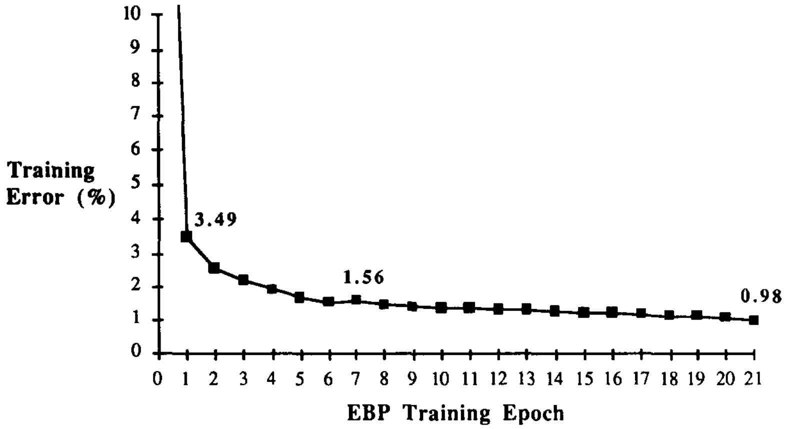


Fig. 7. Training error rates of the EBP learning algorithm for the *truck backer-upper* problem: training error versus EBP training epochs. The number of the random training samples is 350.

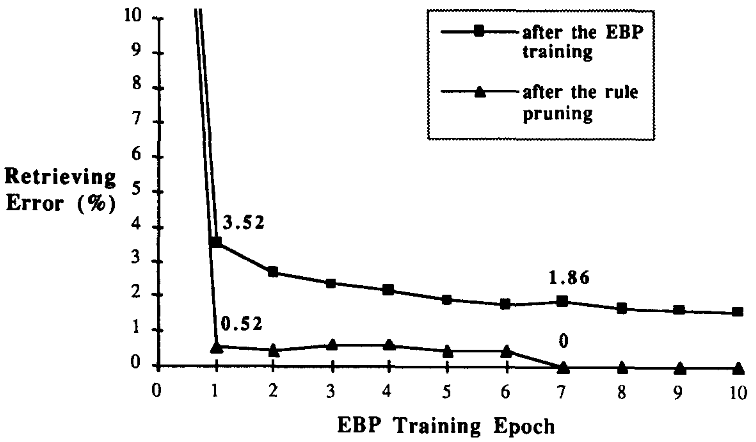


Fig. 8. Retrieving error rates of the fuzzy neural network after the EBP training and after the rule pruning, respectively, for the *truck backer-upper* problem: retrieving error versus EBP training epochs.

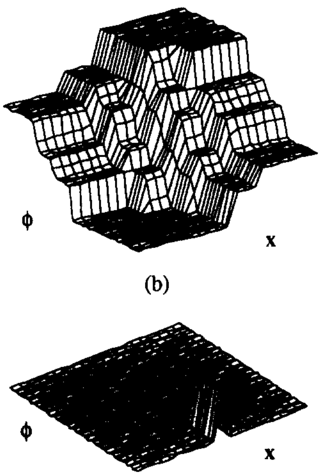
pruning process after each EBP training epoch. The sets of random input vectors that were generated to test the performance of the other two adaptive controller systems mentioned previously were tested to estimate the retrieving error rates of the FAM banks. Fig. 8 shows the retrieving error rates of the fuzzy neural networks after the EBP training and after the rule pruning, respectively. From this figure, we can see that the retrieving error of the fuzzy neural network after the rule pruning is smaller than that just after the EBP training. It means that the truck trajectories produced by an FNN with EBP training and rule pruning are closer to the ideal trajectories than that produced by an FNN with EBP training only.

Fig. 9 shows (a) the resulting FAM bank generated by the pruning phase after only one EBP training epoch, (b) the corresponding control surfaces of the FAM bank, and (c) the absolute difference of the control surface and the target FAM surface.

x

	LE	LC	CE	RC	RI
RB	PS	PM	PM	PB	PB
RU	NS	PS	PM	PB	PB
RV	NM	NS	PS	PM	PB
ϕ VE	NM	NM	ZE	PM	PM
LV	NB	NM	NS	PS	PM
LU	NB	NB	NM	NS	PS
LB	NB	NB	* NB	NM	NS

(a)



(b)

(c)

Fig. 9. (a) FAM bank generated by the rule-pruning process when the EBP training was terminated after the first epoch in Fig. 7. (b) The corresponding control surface of the FAM bank in (a). (c) The absolute difference of the control surface in (b) and the target FAM surface.

We can see that the FAM bank in Fig. 9(a) mismatches the target FAM bank in Fig. 5 at only one rule with a slightly different output linguistic value. Moreover, the retrieving error after the pruning phase for seven EBP training epochs is 0%, i.e., the FAM bank generated then is identical to the target FAM bank. Therefore, the target FAM bank can be reproduced by the fuzzy neural network with no error. In other words, the fuzzy neural network can produce the ideal trajectories for the truck backer-upper controller problem after the training and pruning processes.

For different numbers of training samples, the retrieving errors after pruning with different EBP training epochs are shown in Fig. 10. When 35 random training samples are applied to the network, the pruning phase can generate a FAM bank with an error rate of 10.93% after the first EBP training epoch and with an error rate of 5.81% after the 13th training epoch. In addition, when 2230 random training samples were applied to the network, the pruning phase can reproduce the target FAM bank with no error after only one EBP training epoch.

6. Conclusions

A fuzzy neural network for acquiring rules of a fuzzy-logic rule-based control system has been presented. The learning procedure of the network is divided into two phases. The first one is an *error backpropagation* (EBP) training phase, and the second one is a rule-pruning phase. In the first phase, the EBP learning algorithm enables the network to acquire the knowledge of fuzzy rules precisely and quickly. The main reason for these results is that the gradient descent search approach in the EBP algorithm enables the network to learn more precisely than typical competitive learning algorithms, while the dedicated structure of the network and the competitive characteristics of the functions for OR nodes in the network enable the network to converge much more rapidly than conventional backpropagation learning algorithms. The knowledge of fuzzy rules learned in the EBP training phase is distributed over the learnable weights of the network. Therefore, in the second phase of the learning procedure, a pruning process is performed to convert the distributed knowledge of fuzzy rules learned by the EBP training into a precise and sound (or much smaller size) rule base.

In the near future, we plan to analyze the convergence for the learnable items of the design issues for fuzzy control systems, such as the fuzzy rules, the membership functions of the linguistic values, and the fuzzy operators. Furthermore, we plan to extend the layer-structured fuzzy neural network to event-driven acyclic networks [6].

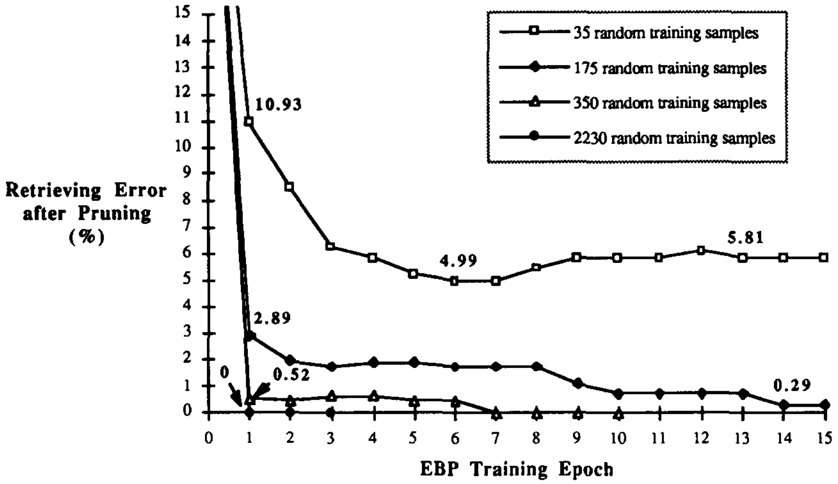


Fig. 10. Retrieving error rates of the proposed fuzzy neural network after the pruning phase for different numbers of training samples in the EBP training phase: retrieving error after pruning versus EBP training epochs.

References

[1] S.I. Horikawa, T. Furuhashi and Y. Uchikawa, On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm, *IEEE Trans. Neural Networks* 3(5) (1992) 801–806.

[2] G.J. Klir and T.A. Folger, *Fuzzy Sets, Uncertainty, and Information* (Prentice-Hall, Englewood Cliffs, NJ, 1988).

[3] S.G. Kong and B. Kosko, Adaptive fuzzy systems for backing up a truck-and-trailer, *IEEE Trans. Neural Networks* 3(2) (1992) 211–223.

[4] B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamic Systems Approach to Machine Intelligence* (Prentice-Hall, Englewood Cliffs, NJ, 1992).

[5] R. Krishnapuram and J. Lee, Fuzzy-set-based hierarchical networks for information fusion in computer vision, *Neural Networks* **5** (1992) 335–350.

[6] R.C. Lacher, S.I. Hruska, and D.C. Kuncicky, Back-propagation learning in expert networks, *IEEE Trans. Neural Networks* 3(1) (1992) 62–72.

[7] C.C. Lee, Fuzzy logic in control systems: fuzzy logic controller – part i and part ii, *IEEE Trans. Systems Man Cybernet.* **20**(2) (1990) 404–435.

[8] C.T. Lin and C.S.G. Lee, Neural-network-based fuzzy logic control and decision system, *IEEE Trans. Comput.* **C-40**(12) (1991) 1320–1336.

[9] D.E. Rumelhart and J.L. McClelland, *Parallel Distributed Processing* (MIT Press, Cambridge, MA, 1986).

[10] J.J. Shann, The structure design and learning methodology for a fuzzy-rule-based neural network, Ph.D. Dissertation, National Chiao-Tung University, ROC (1994).

[11] J.J. Shann and H.C. Fu, A fuzzy neural network for knowledge learning, *Proc. IFSA '93* (1993).

[12] J.J. Shann and H.C. Fu, Backpropagation learning for acquiring fine knowledge of fuzzy neural networks, *Proc. WCNN '93* (1993).

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

[14] L.X. Wang and J.M. Mendel, Back-propagation fuzzy systems as nonlinear dynamic system identifiers, *IEEE Internat. Conf. on Fuzzy Systems* (1992) 1409–1418.

[15] P. Werbos, Beyond regression: new tools for prediction and analysis in the behavioral sciences, Ph.D. Dissertation, Harvard University (1974).

[16] L.A. Zadeh, Fuzzy sets, *Inform. and Control* **8** (1965) 338–353.

[17] L.A. Zadeh, The concept of a linguistic variable and its applications to approximate reasoning I, II, III, *Inform. Sci.* **8** (1975) 199–249, 301–357; **9** (1975) 43–80.

[18] H.J. Zimmermann, *Fuzzy Set Theory – and Its Applications* (Kluwer Academic Publishers, Dordrecht, Boston, MA, 2nd ed., 1991).