Abstract—In this paper, fuzzy inference models for pattern classifications have been developed and fuzzy inference networks based on these models are proposed. Most of the existing fuzzy rule-based systems have difficulties in deriving inference rules and membership functions directly from training data. Rules and membership functions are obtained from experts. Some approaches use backpropagation (BP) type learning algorithms to learn the parameters of membership functions from training data. However, BP algorithms take a long time to converge and they require an advanced setting of the number of inference rules. The work to determine the number of inference rules demands lots of experiences from the designer. In this paper, self-organizing learning algorithms are proposed for the fuzzy inference networks. In the proposed learning algorithms, the number of inference rules and the membership functions in the inference rules will be automatically determined during the training procedure. The learning speed is fast. The proposed fuzzy inference network (FIN) classifiers possess both the structure and the learning ability of neural networks, and the fuzzy classification ability of fuzzy algorithms. Simulation results on fuzzy classification of two-dimensional data are presented and compared with those of the fuzzy ARTMAP. The proposed fuzzy inference networks perform better than the fuzzy ARTMAP and need less training samples.

I. INTRODUCTION

CLASSIFICATION of objects is an important area in a variety of fields, including pattern recognition, artificial intelligence, and vision analysis. In a pattern classification problem, if the a priori probabilities and the state conditional densities of all classes are known, Bayes decision theory produce optimal results in the sense that it minimizes the expected error rate. However, in many pattern recognition problems, such information is not available. In this case, many other algorithms such as nearest prototype algorithm, K-nearest neighbor (K-NN) algorithm, and neural network classification algorithms are used.

Conventional nonfuzzy or crisp classification techniques assume that a pattern $X$ belongs to only one class. Fuzzy classification algorithms assign the pattern \( X \) with a distributed membership value to each class. The partitions between fuzzy classes are “soft.” Since 1965, many efforts have been dedicated to fuzzy classification and many algorithms have been presented and applied in pattern recognition and decision systems. Among all the work that has been done, fuzzy $K$-nearest neighbor algorithm by Keller et al. [1] and fuzzy $c$-means algorithm by Bezdek [2] are the most important ones for pattern recognition problems. Pedrycz gave a survey [3] on fuzzy classification methods and their applications.

Fuzzy classifier design can be performed by supervised learning using a set of training data with fuzzy or nonfuzzy labels. When given a pattern, the fuzzy classifier computes the membership value of the pattern in each class and makes decisions based on these membership values. The fuzzy labels of a fuzzy classifier can be “defuzzified” and then the fuzzy classifier becomes a hard classifier but uses the idea of fuzziness in the model. Fuzzy classifier design almost always means arriving at a hard classifier because most pattern recognition systems require hard labels for objects being classified. We can find a better solution to a crisp problem by looking in a larger space at first, which allow the algorithm more freedom to avoid errors. Fuzzy classification algorithms have advantages when a) the decision maker needs the information of classification uncertainty [4], [5]; b) the features of patterns involve uncertainty [5], [6]; and c) it is difficult to find a hard boundary in a classification problem [6], [7].

On the other hand, neural networks (NN’s) have been used as pattern classifiers in many applications in recent years [8]–[14]. Neural network classifiers are model-free estimators [11]–[12]. Neural network classifiers do not make assumptions of how outputs depend on inputs. Instead, they adjust themselves to a given training set by a learning algorithm and decide the boundaries of classes [13], [14].

It is of practical interest to combine fuzzy classification techniques and neural networks while preserving advantages of both and avoiding their problems. Some researchers have combined these two techniques. We have proposed a fuzzy neural network (FNN) for character pattern recognition [15], [16]. This FNN can recognize shifted and distorted training patterns. Keller and Hunt [18] introduced fuzzy set theory into perceptron algorithm. Their fuzzy perceptron converges quickly when the classes are overlapping. This algorithm assigns the fuzzy membership functions before training the perceptron classifiers. Archer and Wang [19] developed a monotonic function neural network model and used it to represent fuzzy membership functions in two-class pattern recognition problems. This model used several small NN’s to construct the boundary of classes. Its applications are limited because it can only deal with two-class monotonic classifications. Kuo et al. proposed a fuzzy neural network
model to realize the weighted Euclidean distance fuzzy classification [20]. Carpenter et al. combined fuzzy set theory with ART algorithm and proposed a fuzzy ART structure for multidimensional maps [21]. Simpson presented fuzzy min-max neural networks for classification and clustering [22], [23]. The last two approaches discussed the applications of fuzzy neural networks on pattern classification problems but did not discuss the learning of fuzzy inference rules and membership functions of fuzzy classes.

Membership function is the key point of fuzzy rule-based systems. However, the conventional method of designing membership functions for fuzzy systems relies on the manual work and experience of experts. This inevitably becomes a bottleneck in fuzzy rule-based system design. In recent years, some researchers have employed the learning ability of neural networks (NN) to learn the membership functions from training data for fuzzy systems. Jang developed an adaptive-network-based fuzzy inference system (ANFIS) which is used as a fuzzy controller [24]. This ANFIS employs a BP type learning procedure to adjust the parameters of the bell-shaped fuzzy membership functions of the input variables.

Wang and Mendel proposed a BP fuzzy system and used it as a nonlinear dynamic system identifier [25]. They showed that fuzzy systems can be viewed as a three-layer feedforward network and developed a BP algorithm to train them to match desired input-output pairs by adjusting the parameters of Gaussian membership functions. Lin and Lee proposed a connectionist model for fuzzy logic control and decision system [26]. This fuzzy control/decision network can be constructed from training samples by machine learning techniques and the neural network structure can be trained to develop fuzzy logic rules and find optimal bell-shaped input/output membership functions. The above papers have discussed the methods of designing membership functions by using BP type learning to adjust the parameters of membership functions with given shapes. The main problem of BP type learning is that it needs many training epochs to converge and this takes a long time to train a network with large training data.

In this paper, two fuzzy inference networks (min-max fuzzy inference network and min-sum fuzzy inference network) are used as classifiers. After being trained by labeled data, the proposed fuzzy inference networks can find the fuzzy and hard partitions between the classes. Efficient self-organizing learning algorithms are also developed. The fuzzy inference network classifiers build decision boundaries by creating subsets of the pattern space. They are model-free estimators and do not make assumptions of how outputs depend on inputs. Instead, they adjust themselves to a given training set by learning algorithms and decide the boundaries of classes. When given an unknown pattern, the fuzzy inference network classifiers use the learned knowledge to estimate the membership value of this pattern in each class and classify the input pattern according to the membership values. The proposed fuzzy inference networks can be expressed by a set of fuzzy inference rules.

In Section II, the fuzzy inference models for pattern classification are developed. In Section III, fuzzy neurons (FN’s) are defined and two fuzzy inference networks (FIN’s) are constructed based on the proposed models by using different types of FN’s in different layers. Maximum membership rule is employed to defuzzify the fuzzy labels. In Section IV, discussions of the proposed FIN classifiers are presented. Simulation results using two-dimensional training data are given in Section V. Section VI gives conclusions on the proposed FIN classifiers. The fuzzy inference classification algorithms are given in Appendix A and the self-organizing learning algorithms for the fuzzy inference networks are presented in Appendix B.

II. FUZZY INFERENCE MODELS FOR PATTERN CLASSIFICATION

Fuzzy rule-based systems have been successfully used in expert systems and control systems. A fuzzy-rule-based system can be expressed by a set of fuzzy inference rules. In each rule, there is a premise and a consequence. The premise is described by a fuzzy proposition and the consequence can be a fuzzy conclusion, or a linear model or some other models. Fuzzy inference methods are algorithms that deduce results from the inference rules and present inputs. Fuzzy inference methods are based on fuzzy logic and in consideration of the special features for practical systems. A fuzzy inference model consists of three basic parts: fuzzification, inference process and defuzzification. Fuzzification is a mapping from the observed input to the fuzzy sets defined in the corresponding universe. Inference Process is a decision making logic which determines fuzzy outputs corresponding to fuzzified inputs, with respect to the fuzzy inference rules. The designers must specify which implication, conjunction and aggregation operators are used. Defuzzification produces nonfuzzy outputs. There are many different choices within each of these three steps, leading to many different models. A fuzzy rule-based system can be used to implement a mapping from $U \subseteq R^N$ to $V \subseteq R^P$.

A pattern recognition system is a mapping from $U \subseteq R^N$ to $[0, 1]^P$ and we can use a fuzzy rule-based system to implement it. Suppose that a pattern $X$ is represented in terms of $N$ features or properties $\{x_1, x_2, \ldots, x_N\}$, i.e., $X \in R^N$. During the classification procedure, a given pattern $X$ is to be assigned into one of the $P$ possible subsets or classes, $c_1, c_2, \ldots, c_P$, based on its feature values. Thus the classification process is a mapping from $U \subseteq R^N$ to $\{0, 1\}^P$. If the subsets or classes $c_1, c_2, \ldots, c_P$ are fuzzy classes, then the system becomes a fuzzy classification system. The classification process now is a mapping from $U \subseteq R^N$ to the unit hypercube $[0, 1]^P$. A fuzzy rule-based system can be used to implement a pattern classification system. For suitability to a pattern recognition system, simplified fuzzy reasoning rules are employed, in which the consequent parts are expressed in real numbers. The inference rules used in this paper are in the form of

\begin{align*}
\text{rule } j: \quad & \text{IF } x_1 \in A_{1j}, \ldots, x_N \in A_{Nj}, \\
& \quad \text{THEN } y_k = y_{k1}, \ldots, y_P = y_{pj}.
\end{align*}

where $A_{ij}$ ($i = 1, 2, \ldots, N$) are fuzzy subsets and $y_{pj}$ ($j = 1, 2, \ldots, M$) are real numbers in $[0, 1]$. For simplicity, only triangle functions are used as the membership functions of $A_{ij}$ in this paper. Several inference methods are developed to
obtain outputs from the inference rules and the present inputs $x_1 = x_1^p$ and $x_2 = x_2^p, \ldots, x_N = x_N^p$.

1) Inference Method I, Min-Max Inference: The compatibility for the antecedent conditions of each rule is

$$c_j = \prod_{i=1}^{N} (A_{ij}(x_i^p)) \quad j = 1, 2, \ldots, M \quad (1)$$

The complete inference results are

$$y_p^f = \max_{j=1}^{M} (c_jy_{jp}) \quad p = 1, 2, \ldots, P. \quad (2)$$

2) Inference Method II, Min-Sum Inference: The compatibility for the antecedent conditions of each rule is

$$c_j = \min_{i=1}^{N} (A_{ij}(x_i^p)) \quad j = 1, 2, \ldots, M. \quad (3)$$

The complete inference results are

$$y_p^f = \begin{cases} \sum_{j=1}^{M} c_jy_{jp}, & \text{if } \sum_{j=1}^{M} c_j \neq 0, \quad p = 1, 2, \ldots, P, \\ 0, & \text{if } \sum_{j=1}^{M} c_j = 0. \end{cases} \quad (4)$$

For nonfuzzy classification system, the Maximum criterion is used for defuzzification, that is, the input pattern should be assigned to the class with the maximum membership value or to the last found class with the maximum membership value.

When triangular functions are used as the membership functions of $A_{ij}$ and “minimum” is used as the fuzzy “AND” operation, the IF part of each rule is actually a fuzzy hyperbox in the pattern space $U \subset R^N$. The centre of the hyperbox is the corresponding prototype of the rule: $\{x_1^j, x_2^j, \ldots, x_N^j\}$ for the $j$th rule. The boundaries of the hyperboxes are determined by the membership functions of fuzzy sets $A_{ij}$ because the membership function defined on each hyperbox is

$$c_j(x) = \prod_{i=1}^{N} (A_{ij}(x_i^p)) \quad j = 1, 2, \ldots, M. \quad (5)$$

Each fuzzy class is an aggregation of the hyperboxes. Assuming $B_j$ represents the $j$th hyperbox ($j = 1, 2, \ldots, M$), $c_j(x)$ is the membership function of $B_j$; then each fuzzy class is an aggregation of $B_j$

$$F_p = \bigcup_{j=1}^{M} (B_j). \quad (6)$$

The union operation is weighted maximum for min-max inference method and is weighted sum for min-sum inference method. $y_p^f$ represents the membership value of the input pattern belonging to the $p$th class $F_p$. $y_p^f$ is determined by the most important prototype for min-max inference method and by all the important prototypes for min-sum inference method. The “most important prototype” for an input pattern regarding the $p$th class is defined as $c_j(x')y_{jp} = \max_{j=1}^{M} (c_j(x')y_{jp})$. An “important prototype” for an input pattern regarding the $p$th class is defined as $c_j(x') \neq 0$ and $y_{jp} \neq 0$.

Based on the proposed fuzzy inference methods, fuzzy min-max and min-sum inference classification algorithms were developed (See Appendix A). The problem now is how to determine the fuzzy inference rules. In the following section, fuzzy inference networks will be constructed to implement the fuzzy inference classifiers and learning algorithms for the fuzzy inference networks will be developed to learn fuzzy inference rules from training patterns.

III. FUZZY INFERENCE NETWORKS FOR PATTERN CLASSIFICATION

A. Fuzzy Neurons

Neurons are the basic computing units of neural networks. In most of the previous researches, it is assumed that all the neurons in a network are the same. A typical neuron sums the weighted inputs and transfers to a nonlinear threshold function. In this paper, we have generalized the definition of neurons as

$$z = h(w_1, x_1, w_2, x_2, \ldots, w_N, x_N)$$

$$s = f(z, t) \quad (7)$$

where $z$ is the net input, $s$ is the state, $w_i (i = 1, 2, \ldots, N)$ are input weights, $h()$ is the aggregation function, $f()$ is the activation function, and $t$ is the activation threshold. The input weights and the activation threshold, that describe the interactions among neurons, could become weight functions and threshold functions and can be adjusted during the learning procedure. A neuron is called a fuzzy neuron (FN) if it has the ability to cope with fuzzy information, i.e., if its inputs are fuzzy variables and the aggregation function $h()$ is a fuzzy operation, or its output is a fuzzy variable, or its weight functions are membership functions. Four types of fuzzy neurons are used in this paper:

1) Transit fuzzy neuron (TRAN-FN): A fuzzy neuron that transfer an input into membership values is called a transit fuzzy neuron or TRAN-FN.

2) Maximum fuzzy neuron (MAX-FN): A fuzzy neuron with maximum as the aggregation function is called a maximum fuzzy neuron or MAX-FN.

3) Minimum fuzzy neuron (MIN-FN): A fuzzy neuron with minimum as the aggregation function is called a minimum fuzzy neuron or MIN-FN.

4) Sum fuzzy neuron (SUM-FN): A fuzzy neuron with sum as the aggregation function is called a sum fuzzy neuron or SUM-FN.

B. The Structure of the Fuzzy Inference Networks

Two types of fuzzy inference networks, min-max fuzzy inference network (MMFIN) and min-sum fuzzy inference network (MSFIN), are developed based on the proposed inference models. The fuzzy inference networks are implemented in feedforward network structures. Different types of fuzzy neurons are used in different layers.

1) Min-Max Fuzzy Inference Network (MMFIN): MMFIN is a three layer feedforward network as shown in Fig. 1. The first layer accepts input pattern into the network. The second layer computes the compatibility for the antecedent condition of each fuzzy inference rule. MIN-FN’s are used in the second layer to represent inference rules, one MIN-FN for each rule.
The number of MIN-FN’s, \(M\), is to be determined by the learning algorithm. The algorithm of the \(j\)th MIN-FN is
\[
s_j^2 = \text{MIN}_{i=1}^{N} (w_{ij}^1(x_i))
\]  
(8)
\[
w_{ij}^1(x_i) = \begin{cases} 
1 + \alpha(x_i - \theta_{ij}), & \text{if } 1 + \alpha(x_i - \theta_{ij}) > \theta_{ij}^2 \\
1 - \alpha(x_i - \theta_{ij}), & \text{if } 1 - \alpha(x_i - \theta_{ij}) > \theta_{ij}^2 \\
0, & \text{otherwise}
\end{cases} 
\]
(9)
in \(i = 1, 2, \ldots, N; \ j = 1, 2, \ldots, M\)

where \(x_i\) is the input of the \(i\)th TRAN-FN, \(w_{ij}^1(x_i)\) is the \(i\)th weighted input of the \(j\)th MIN-FN. \(w_{ij}^1(x_i)\) represents the membership functions of the fuzzy subset \(A_{ij}\) in the fuzzy inference rule set. For simplicity, triangular functions are used for \(w_{ij}^1(x_i)\) (see Fig. 2). The weight functions transform the features of the pattern into membership values. The weights from the \(i\)th input node compute the membership values of the \(j\)th input with respect to the fuzzy sets defined in the \(i\)th projection of \(U\). The information in the IF parts of the inference rules is contained in these membership functions. Positive parameters \(\alpha_i, \theta_{ij}^1\) and \(\theta_{ij}^2\), and parameter \(\Theta_{ij}\) are to be determined by the learning algorithm.

The third layer gives the outputs according to the fuzzy inference method. MAX-FN’s are used in the third layer of the MMFIN. The information in the THEN parts of the inference rules is contained in the weights from the MIN-FN’s in the second layer to the MAX-FN’s in the third layer. The algorithm of the \(p\)th MAX-FN is
\[
s_p^3 = \text{MAX}_{j=1}^{M} (w_{jp}^2, s_j^2)
\]  
(10)
where \(w_{jp}^2\) is the weight from the \(j\)th MIN-FN in the second layer to the \(p\)th MAX-FN in the third layer, which is to be determined by the learning algorithm.

2) Min-Sum Fuzzy Inference Network (MSFIN): MSFIN is also a three layer feedforward network as shown in Fig. 3. As in the MMFIN, TRAN-FN’s are used in the first layer and MIN-FN’s are used in the second layer for the MSFIN. However, the weight functions from the first layer to the second layer of the MSFIN are different from those of the MMFIN (see Fig. 4). The algorithms for the \(j\)th MIN-FN in the second layer of the MSFIN classifier is
\[
s_j^2 = \text{MIN}_{i=1}^{N} (w_{ij}^1(x_i))\quad j = 1, 2, \ldots, M
\]  
(11)
\[
w_{ij}^1(x_i) = \begin{cases} 
1 + \alpha(x_i - \theta_{ij}), & \text{if } 1 + \alpha(x_i - \theta_{ij}) \geq 0 \\
1 - \alpha(x_i - \theta_{ij}), & \text{if } 1 - \alpha(x_i - \theta_{ij}) \geq 0 \\
0, & \text{otherwise}
\end{cases} 
\]
(12)
in \(i = 1, 2, \ldots, N; \ j = 1, 2, \ldots, M\)

where \(\alpha_i, \theta_{ij}^1\) and \(\Theta_{ij}\) are to be determined by the learning algorithm.

SUM-FN’s are used in the third layer of the MSFIN. Now the summations of the SUM-FN’s combine the information of all the inference rules and the compatibilities of the input pattern. Then weighted averages are given as the inference outputs. The algorithm is
\[
s_p^3 = \begin{cases} 
\frac{\sum_{j=1}^{M} (w_{jp}^2 s_j^2)}{\sum_{j=1}^{M} s_j^2}, & \text{if } \sum_{j=1}^{M} s_j^2 \neq 0 \\
0, & \text{if } \sum_{j=1}^{M} s_j^2 = 0
\end{cases} 
\]  
(13)
where \(w_{jp}^2\) is the connection weight between the \(j\)th MIN-FN in the second layer and the \(p\)th SUM-FN in the third layer which is to be determined by the learning algorithm.

C. The Learning of Fuzzy Inference Networks

The most widely used learning method for fuzzy neural networks is backpropagation method. However, this method
suffers from the inherent prerequisite problems, such as an advanced setting of the number of inference rules. Expert knowledge is required to determine the number of inference rules. In this paper, self-organizing learning algorithms are proposed for the fuzzy inference networks. In these learning algorithms, the number of inference rules, the membership functions in the antecedent parts, and the outputs in the conclusion parts of inference rules will be automatically determined during the training procedure.

In the learning algorithms, prototypes are selected from the training patterns and then these prototypes are fuzzified and used to establish inference rules. If a training pattern \( \{x_1 = a_1, x_2 = a_2, y = b\} \) is chosen as a prototype, then an inference rule of \( \text{“IF } x_1 \text{ is near } a_1 \text{ and } x_2 \text{ is near } a_2, \text{ THEN } y \text{ is near } b' \text{”} \) will be established. The membership functions for \( “x_1 \text{ is near } a_1”, “x_2 \text{ is near } a_2” \) and \( “y \text{ is near } b’” \) will be determined by the learning algorithm according to all the training samples. The prototypes are chosen in such a way that the resultant network can recall all the training patterns within a specified error tolerance \( E_c \). The MIN-FN’s in the hidden layer represent inference rules. The weights of the neurons are used to store the information of the membership functions and are adjusted according to all the selected prototypes. The learning algorithms for min-max and min-sum fuzzy inference networks are presented in Appendix B.

By using these self-organizing learning algorithms, fuzzy inference networks are constructed during the training procedure. The learning can always converge because the worst case is to select all the training patterns as prototypes. The learning will be finished in a few epochs. This is different from the backpropagation learning algorithm, which adjusts parameters to minimize the sum of the square of estimated errors.

IV. DISCUSSIONS

The MMFIN and the MSFIN are used as fuzzy classifiers. After being trained, they can estimate the membership values of the input patterns to each class and find the fuzzy partitions. The first layers of the FIN classifiers accept data into the networks. The weight functions between the first layer and the second layer then transfer the input data into membership values. These membership values describe the distances or the similarities of each input to the corresponding feature value of the prototypes. Because the weight functions are triangle membership functions (triangle membership functions are used for simplicity), the FIN classifiers create a triangle fuzzy number for each feature value of a prototype. Consequently, the FIN classifiers establish fuzzy patterns in their structures for the prototypes. The threshold and slope parameters, \( t^1, t^2 \) and \( \alpha \) for the MMFIN classifier, or \( \alpha^1 \) and \( \alpha^2 \) for the MSFIN classifier, control the fuzzy extent of the corresponding prototype. The way of selecting threshold and slope parameters will affect the computation of membership values and the number of MIN-FN’s in the second layer.

The MIN-FN’s in the second layers represent prototypes. The minimum operation of the MIN-FN makes a fuzzy “AND” for all the fuzzy outputs from the first layers. When triangular membership functions are used as the weight functions for the TRAN-FN’s in the first layer, each MIN-FN represents a fuzzy hyperbox. The fuzzy hyperboxes are fuzzy sets defined in the pattern space \( U \subset \mathbb{R}^N \). The output of each MIN-FN gives the membership value of the input pattern to the corresponding hyperbox that describe how much the input pattern is similar to the corresponding prototype.

MAX-FN’s are used in the third layer of the MMFIN. The weights between the second layer and the third layer contain the membership information of all the prototypes belonging to each fuzzy class. The MAX-FN’s choose the maximum weighted input as the membership value of the input pattern to each class. The min-max fuzzy inference network implements a “most important prototype” classification algorithm.

In the MSFIN classifier, SUM-FN’s are used in the third layer. The weighted average output is used as the membership value of the input pattern to each class. The min-sum fuzzy inference network implements a “weighted average of the important prototypes” classification algorithm.

The numbers of MIN-FN’s in the second layers are determined by the learning algorithms, \( E_c \), and the training patterns. The numbers of TRAN-FN’s in the first layers and the numbers of MAX-FN’s or SUM-FN’s in the third layers are determined, respectively, by the numbers of inputs and outputs of training samples.

The proposed self-organizing learning algorithms are improved leader clustering algorithms. The first stage of the learning is leader clustering. The learning algorithms select the first training pattern as the first prototype. The next training pattern is compared with the first prototype. It follows the first prototype if the maximum output error is less than or equal to the error tolerance \( E_t \). Otherwise, it is selected as a new prototype. This process is repeated for all the following training patterns. In the second stage of the learning, each training pattern is fed to the network to check the output error. If the maximum output error is greater than \( E_t \), a new prototype is added. The second stage is repeated until the maximum output error is less than or equal to \( E_t \) for all the training patterns. The information of prototypes is stored in the structures of the networks. The number of prototypes thus grows with time and depends on \( E_t \), the similarities between the training patterns, and the method of calculating outputs. The proposed self-organizing learning algorithms perform better than the leader clustering due to the addition of the second stage.

The proposed fuzzy inference networks are trained to match all the training patterns within the error tolerance. The criterion
for the proposed learning algorithms is

$$\sigma = \max_{k=1}^{K} \left( \max_{j=1}^{P} |y_{kp} - d_{kpl}| \right) \leq E_t$$  \tag{14} $$

where $\sigma$ is the maximum estimated error, $d_{kpl}$ is the desired output, and $y_{kp}$ is the $k$th network output for $k$th training pattern. The training will be finished in a few epochs. This is different from backpropagation-type learning algorithms,

---

**TABLE II**

TRAINING DATA SET 2: 36 TRAINING SAMPLES WITH FUZZY LABELS OF FUZZY EXCLUSIVE OR PROBLEM

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>0.0</th>
<th>0.2</th>
<th>0.4</th>
<th>0.6</th>
<th>0.8</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y_1, y_2$</td>
<td>0.100</td>
<td>0.750</td>
<td>0.250</td>
<td>0.250</td>
<td>0.750</td>
<td>1.000</td>
</tr>
</tbody>
</table>

---

**TABLE III**

LEARNING RESULTS OF THE MMFIN CLASSIFIER

<table>
<thead>
<tr>
<th>Training Set</th>
<th>$E_t$</th>
<th>N</th>
<th>M</th>
<th>P</th>
<th>t(sec)</th>
<th>$\sigma$</th>
<th>$R$</th>
<th>error rate</th>
<th>Fig. No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Data Set 1</td>
<td>0.1</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>0.05</td>
<td>0.00</td>
<td>2</td>
<td>8.60%</td>
<td>Fig.6</td>
</tr>
<tr>
<td>Training Data Set 2</td>
<td>0.1</td>
<td>2</td>
<td>14</td>
<td>2</td>
<td>0.07</td>
<td>0.10</td>
<td>2</td>
<td>12.1%</td>
<td>Fig.7</td>
</tr>
<tr>
<td>Training Data Set 3</td>
<td>0.1</td>
<td>2</td>
<td>40</td>
<td>2</td>
<td>1.43</td>
<td>0.10</td>
<td>3</td>
<td>3.37%</td>
<td>Fig.10a</td>
</tr>
<tr>
<td>Training Data Set 3</td>
<td>0.2</td>
<td>2</td>
<td>20</td>
<td>2</td>
<td>1.79</td>
<td>0.20</td>
<td>4</td>
<td>4.53%</td>
<td>Fig.11a</td>
</tr>
</tbody>
</table>

---

**TABLE IV**

LEARNING RESULTS OF THE MSFIN CLASSIFIER

<table>
<thead>
<tr>
<th>Training Set</th>
<th>$E_t$</th>
<th>N</th>
<th>M</th>
<th>P</th>
<th>t(sec)</th>
<th>$\sigma$</th>
<th>$R$</th>
<th>error rate</th>
<th>Fig. No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Data Set 1</td>
<td>0.1</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>0.05</td>
<td>0.00</td>
<td>2</td>
<td>5.98%</td>
<td>Fig.8</td>
</tr>
<tr>
<td>Training Data Set 2</td>
<td>0.1</td>
<td>2</td>
<td>31</td>
<td>2</td>
<td>0.22</td>
<td>0.00</td>
<td>3</td>
<td>6.72%</td>
<td>Fig.9</td>
</tr>
<tr>
<td>Training Data Set 3</td>
<td>0.1</td>
<td>2</td>
<td>42</td>
<td>2</td>
<td>1.32</td>
<td>0.09</td>
<td>3</td>
<td>2.22%</td>
<td>Fig.12a</td>
</tr>
<tr>
<td>Training Data Set 3</td>
<td>0.2</td>
<td>2</td>
<td>33</td>
<td>2</td>
<td>1.26</td>
<td>0.18</td>
<td>3</td>
<td>2.33%</td>
<td>Fig.12b</td>
</tr>
<tr>
<td>Training Data Set 4</td>
<td>0.1</td>
<td>2</td>
<td>197</td>
<td>2</td>
<td>19.4</td>
<td>0.00</td>
<td>3</td>
<td>2.22%</td>
<td>Fig.13a</td>
</tr>
<tr>
<td>Training Data Set 4</td>
<td>0.2</td>
<td>2</td>
<td>174</td>
<td>2</td>
<td>24.6</td>
<td>0.20</td>
<td>4</td>
<td>2.22%</td>
<td>Fig.13b</td>
</tr>
</tbody>
</table>

---

**TABLE V**

FUZZY INFERENCE RULES EXTRACTED BY MMFIN WHEN TRAINED BY TRAINING DATA SET 1

| Rule 1: | IF: $x_1$ is small and $x_2$ is small | THEN: $y_1$ is 0 and $y_2$ is 1 |
| Rule 2: | IF: $x_1$ is small and $x_2$ is large | THEN: $y_1$ is 1 and $y_2$ is 0 |
| Rule 3: | IF: $x_1$ is large and $x_2$ is small | THEN: $y_1$ is 1 and $y_2$ is 0 |
| Rule 4: | IF: $x_1$ is large and $x_2$ is large | THEN: $y_1$ is 0 and $y_2$ is 1 |

---

**TABLE VI**

FUZZY INFERENCE RULES EXTRACTED BY MSFIN WHEN TRAINED BY TRAINING DATA SET 1

| Rule 1: | IF: $x_1$ is small and $x_2$ is small | THEN: $y_1$ is 0 and $y_2$ is 1 |
| Rule 2: | IF: $x_1$ is small and $x_2$ is medium | THEN: $y_1$ is 0 and $y_2$ is 0 |
| Rule 3: | IF: $x_1$ is small and $x_2$ is large | THEN: $y_1$ is 1 and $y_2$ is 0 |
| Rule 4: | IF: $x_1$ is medium and $x_2$ is small | THEN: $y_1$ is 0 and $y_2$ is 0 |
| Rule 5: | IF: $x_1$ is medium and $x_2$ is large | THEN: $y_1$ is 1 and $y_2$ is 0 |
| Rule 6: | IF: $x_1$ is large and $x_2$ is small | THEN: $y_1$ is 1 and $y_2$ is 0 |
| Rule 7: | IF: $x_1$ is large and $x_2$ is medium | THEN: $y_1$ is 1 and $y_2$ is 0 |
| Rule 8: | IF: $x_1$ is large and $x_2$ is large | THEN: $y_1$ is 0 and $y_2$ is 1 |
TABLE VII

CLASSIFICATION RESULTS OF FIN CLASSIFIERS COMPARING WITH FUZZY ARTMAP FOR CIRCLE-IN-THE-SQUARE PROBLEM

<table>
<thead>
<tr>
<th></th>
<th>MMFIN</th>
<th>MSFIN</th>
<th>Fuzzy ARTMAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Training samples</td>
<td>100</td>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td>Training epochs</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Recalling Rate on Testing Set</td>
<td>91.4%</td>
<td>96.6%</td>
<td>94.0%</td>
</tr>
</tbody>
</table>
selected as a new prototype and a new MIN-FN is established to represent it. Then the slope parameters of the output functions of the TRAN-FN’s are adjusted. If the error is acceptable (say $|s_p^3 - d_{jk}| \leq E_t$), this training pattern is treated as a pattern that is similar to one of the prototypes. The number of MIN-FN’s in the second layer can be determined only after the learning procedure is finished. It depends on $E_t$ and the similarities among training patterns.

The proposed classification algorithms and learning algorithms allow overlapping in the hyperboxes and the fuzzy classes. Fuzzy boundaries of the fuzzy classes can be found by training the networks using fuzzy or nonfuzzy labeled data. Nonfuzzy boundaries can also be found by utilizing the maximum membership rule.

V. SIMULATIONS

The proposed FIN classifiers were trained on a PC-486 (33 MHz) using C language. In the simulations, the proposed FIN classifiers were trained by four sets of two-dimensional training samples (Training Data Sets 1, 2, 3 and 4). Training Data Set 1 and Training Data Set 2 (given in Tables I and II) are training samples with fuzzy labels from the “fuzzy Exclusive OR” (or a quasi-XOR problem) [28]. There are two fuzzy classes in the fuzzy XOR problem, $Y^1$ and $Y^2$. Training Data Set 2 has more input-output samples than Training Data Set 1. Fuzzy labels are available in practical applications. For example, in pattern recognition problems, the “typicalness” of training samples can be considered and used as the fuzzy labels of the training samples. We can also assign class membership values to training pattern according to some rules. Keller et al. proposed three methods to assign class memberships to the training samples [1]. Training Data Sets 3 and 4 are samples from the circle-in-the-square problem. The labels in these two sets are nonfuzzy. There are two classes in the square: inside the circle and outside the circle. The area inside the
circle equals to the area outside the circle. This is a typical classification problem. There are 100 samples in Training Data Set 3 and 1000 samples in Training Data Set 4. These samples are randomly distributed in the square.

In the simulation experiments, different values of $E_k$ were used to investigate its effects on the FIN classifiers. The simulation results of the MMFIN classifier are listed in Table III. The simulation results of the MSFIN classifier are listed in Table IV. In Tables III and IV $N_k, M_k$, and $P$ represent, respectively, the numbers of FN’s in the first, the second, and the third layers. $t$ is the learning time (in seconds). $\sigma$ is the maximum estimated error. $R$ is the number of epochs of the training samples being fed to the networks. Fuzzy inference rules are established during the learning procedure. The membership functions of the inference rules are automatically determined during the learning. Table V gives the fuzzy inference rules extracted by the MMFIN when trained by Training Data Set 1. Table VI gives the fuzzy inference rules extracted by the MSFIN when trained by Training Data Set 1. The membership functions of the inference rules are given in Fig. 5. For illustration, the classification results of the FIN classifiers are compared with those of the fuzzy ARTMAP [21] for the circle-in-the-square problem. Table VII gives the comparison results of the MMFIN and the MSFIN with the fuzzy ARTMAP.

Figs. 6 and 7 show the fuzzy partitions for the fuzzy XOR problem by the MMFIN classifier. Figs. 8 and 9 show the fuzzy partitions for the fuzzy XOR problem by the MSFIN classifier. Figs. 10 and 11 are the classification results of the MMFIN classifier for the circle-in-the-square problem with different learning parameters and trained by different number of training samples. Figs. 12 and 13 are the classification results of the MSFIN classifier for the circle-in-the-square problem.

From the simulation results, we note that the smaller $E_k$ is, the more MIN-FN’s are needed in the second layer and the
Fig. 10. Classification results of the MMFIN for the circle-in-the-square problem (100 training samples). (a) $E_t = 0.1, M = 40$ and (b) $E_t = 0.2, M = 20$.

Fig. 11. Classification results of the MMFIN for the circle-in-the-square problem (1000 training samples). (a) $E_t = 0.1, M = 91$ and (b) $E_t = 0.2, M = 62$.

Fig. 12. Classification results of the MSFIN for the circle-in-the-square problem (100 training samples). (a) $E_t = 0.1, M = 42$ and (b) $E_t = 0.2, M = 33$.

Fig. 13. Classification results of the MSFIN for the circle-in-the-square problem (1000 training samples). (a) $E_t = 0.1, M = 197$ and (b) $E_t = 0.2, M = 174$. 
classification results are better. However, the number of the MIN-FN’s will not exceed the number of training patterns. The value of $E_t$ affects the number of MIN-FN’s in the second layer, and the classification results. If more training samples are learned, the classification results are better. The MMFIN needs fewer number of MIN-FN’s in the second layer than the MSFIN. However, the MSFIN can get smoother membership functions for fuzzy classification problems and lower error rate for nonfuzzy classification problems than the MMFIN when trained by the same set of training data. The learning of the proposed FIN classifiers is fast and can be finished in a few epochs. From the recognition results of the FIN classifiers and those of the fuzzy ARTMAP, we note that the classification results of the FIN classifiers are better than those of fuzzy ARTMAP.

VI. CONCLUSION

In this paper, fuzzy inference models have been developed for classification problems. Two fuzzy inference networks have been constructed and used as classifiers. Efficient self-organizing learning algorithms for the FIN classifiers have also been developed. The proposed FIN classifiers can learn fuzzy inference rules and membership functions directly from training data and then determine fuzzy and hard partitions. The following advantages are noticeable for the proposed FIN classifiers.

1) The FIN classifiers are “intelligent” by using a set of fuzzy inference rules to describe classification problems, i.e., they have the ability to classify patterns using rules that are similar to those used by humans. These rules can be easily understood by humans and are extracted automatically from the training data through the self-organizing learning algorithms.

2) The proposed FIN classifiers have nonlinear separability. The application of the FIN to the circle-in-the-square problem is a good example. Because fuzzy subsets are used in the clustering procedure, fuzzy partitions are obtained automatically from training data. Consequently, the FIN classifiers also have the ability to deal with classes with overlapping characteristics.

3) The proposed fuzzy inference networks are adaptive to environments because the second layer is constructed during the learning procedures. The learning is fast by using the prototype selecting procedure to reduce the output error instead of trying to find a set of optimal parameters based on all the training samples.

4) The proposed fuzzy inference networks and self-organizing algorithms could be applied to any problem that can be expressed by a set of fuzzy inference rules as a mapping from $U \subseteq \mathbb{R}^N$ to $V \subseteq \mathbb{R}^P$, such as fuzzy control problems, fuzzy decision systems and pattern recognition systems. The advantage of using the FIN is that it has parallel structure and the fuzzy inference rules and membership functions can be extracted from examples by the self-organizing learning algorithms.

APPENDIX A

FUZZY INFERENCE CLASSIFICATION ALGORITHMS

Let \{IF $x_1$ is $A_{1j}$, ..., and $x_N$ is $A_{Nj}$, THEN $y_1$ is $y_{j1}$, ..., and $y_P$ is $y_{jp}$, $|j = 1, 2, ..., M|\}$ be the set of fuzzy inference rules for the classification system. The classification algorithms are shown in the following two algorithms.

Min-Max Fuzzy Inference Classification Algorithm

BEGIN

input pattern $X = \{x_1, x_2, ..., x_N\}$, of unknown classification
initialize $j = 1$

DO UNTIL $(j > M)$

the membership value of $X$ to the $j$th hyperbox is computed as

$e_j = \min_{i=1}^{N} \mu_{ij}(x_i)$

where $\mu_{ij}()$ is the membership function of fuzzy set $A_{ij}$
increment $j$

END DO UNTIL

initialize $p = 1$

DO UNTIL $(p > P)$

calculate the membership values of $X$ to the fuzzy classes $F_p$ by finding the most important prototype:

$y_p = \max_{j=1}^{M} (e_j \cdot y_{jp})$

where $y_{jp}$ is the membership value of the prototype $X_j$ to class $F_p$
increment $p$

END DO UNTIL

END
Min-Sum Fuzzy Inference Classification Algorithm

BEGIN

input pattern \( X = \{x_1, x_2, \ldots, x_N\} \), of unknown classification.
initialize \( j = 1 \)
DO UNTIL \((j > M)\)
the membership value of \( X \) to the \( j \)th hyperbox is computed as
\[
c_j = \max_{i=1}^{N} m_i(x_i)
\]
where \( m_i(\cdot) \) is the membership function of fuzzy set \( A_i \)
increment \( j \)
END DO UNTIL

initialize \( p = 1 \)
DO UNTIL \((p > P)\)
calculate the membership values of \( X \) to the fuzzy classes \( F_p \) by
finding all the important prototypes:
\[
y_{jp} = \begin{cases} \frac{\sum_{j=1}^{M} (c_j \cdot y_{jp})}{\sum_{j=1}^{M} c_j} & \text{if } \sum_{j=1}^{M} c_j \neq 0 \\ 0 & \text{if } \sum_{j=1}^{M} c_j = 0 \end{cases}
\]
where \( y_{jp} \) is the membership value of the prototype \( X_j \) to class \( F_p \)
increment \( p \)
END DO UNTIL

END

APPENDIX B
LEARNING ALGORITHMS FOR FUZZY INFERENCE NETWORKS

Assuming there are \( K \) training patterns and \( X_k = \{x_{1k}, \ldots, x_{N_k}\} \subset \mathbb{R}^N, D_k = \{d_{k1}, \ldots, d_{kP}\} \subset \mathbb{R}^P \) is the \( k \)th training pattern, where \( x_{ik} \) is the \( i \)th input and \( d_{kp} \) is the desired \( p \)th output of the \( k \)th training sample. \( D \) is the largest distance between any two of the training patterns. See the following two algorithms.

Self-organizing learning algorithm for the MMFIN classifier:

BEGIN
set \( E_t \), set \( \alpha = 1/D \), establish layer 1 and layer 3
establish the 1st MIN-FN in layer 2 for the 1st training sample
adjust the weight functions \( w_{1k}^{2} \) and weights \( w_{2p}^{2} \)
initialize \( m = 1, k = 2 \)
DO UNTIL \((k \geq K)\)
input the \( k \)th training pattern, compute the output errors:
\[
e_p = s_p^2 - d_{kp}, \quad (p = 1 \text{ to } P)
\]
IF (there is one \( e_p > E_t \) \((p = 1 \text{ to } P)\)), THEN
adjust \( t_{ij} \), and \( t_{ij}^2 (i = 1 \text{ to } N, j = 1 \text{ to } m) \) until \( e_p \leq E_t \) \((p = 1 \text{ to } P)\).
IF (there is one \( e_p < -E_t \) \((p = 1 \text{ to } P)\)), THEN
add a new MIN-FN in layer 2 and increment \( m \),
adjust the weight functions \( w_{1m}^{2} \)(\(i = 1 \text{ to } N\)) and weights \( w_{mp}^{2} \)(\(p = 1 \text{ to } P\)),
increment \( k \)
END DO UNTIL
let
\[
\sigma = \max_k \left( \max_p (|e_p|) \right)
\]
DO UNTIL \((\sigma < E_t)\)
initialize \( k = 1 \)
DO UNTIL \((k \geq K)\)
input the \( k \)th training samples and compute the output error
\[
e_p = s_p^2 - d_{kp}, \quad (p = 1 \text{ to } P)
\]
IF (there is one $e_p > E_t (p = 1 \text{ to } P) \sigma > E_t$), THEN
adjust $r_{ij}^1$, and $r_{ij}^2 (i = 1 \text{ to } N, j = 1 \text{ to } m)$ until $e_p \leq E_t (p = 1 \text{ to } P)$.

IF (there is one $e_p < -E_t (p = 1 \text{ to } P)$) THEN
add a new MIN-FN in layer 2 and increment $m$,
adjust the weight functions $w_{in}^k (i = 1 \text{ to } N)$ and weights $w_{mp}^2 (p = 1 \text{ to } P)$.

increment $k$
END DO UNTIL
END DO UNTIL
END

Self-organizing learning algorithm for the MSFIN classifier:

BEGIN
set $E_t$, establish layer 1 and layer 3,
establish the 1st MIN-FN in layer 2 for the 1st training sample,
adjust the weight functions $w_{ij}^1$ and weights $w_{mp}^2$.
initialize $m = 1, k = 2$.

DO UNTIL ($k \geq K$)
input the $k$th training pattern, compute the output errors:
$e_p = s_{kp} - d_{kp}, \quad (p = 1 \text{ to } P)$

IF (there is one $|e_p| > E_t (p = 1 \text{ to } P)$), THEN
add a new MIN-FN in layer 2 and increment $m$,
adjust weight function $w_{ij}^1 (i = 1 \text{ to } N, j = 1 \text{ to } m)$ and $w_{mp}^2 (j = 1 \text{ to } m, p = 1 \text{ to } P)$.
increment $k$
END DO UNTIL

DO UNTIL ($\sigma < E_t$)
let
$\sigma = \max_k \left( \max_p |e_p| \right)$
initialize $k = 1$.

DO UNTIL ($k \geq K$)
ininput the $k$th training samples and compute the output error

IF (there is one $|e_p| > E_t (p = 1 \text{ to } P)$), THEN
add a new MIN-FN in layer 2 and increment $m$,
adjust weight function $w_{ij}^1 (i = 1 \text{ to } N, j = 1 \text{ to } m)$ and $w_{mp}^2 (j = 1 \text{ to } m, p = 1 \text{ to } P)$.
increment $k$
END DO UNTIL
END DO UNTIL
END

REFERENCES


Hon Keung Kwan (M’81–SM’86) received the B.Sc. degree from the University of London, London, U.K., and the M.Phil. degree from the Chinese University of Hong Kong, in 1976 and 1977, respectively. He received the D.I.C. and Ph.D. degrees in 1981 from Imperial College, University of London. From 1977 to 1978, he was with Interquartz Ltd., Hong Kong, as an Assistant Design Engineer, and later Tek-Devices Ltd., Hong Kong, as a Design Engineer, while serving as a Visiting Lecturer in the Department of Electronic Engineering, Hong Kong Polytechnic. In 1981, he joined the Department of Electronic Engineering, Hong Kong Polytechnic, as a Lecturer. Since December 1981, he had been with the Department of Electrical and Electronic Engineering, University of Hong Kong, as a Lecturer. In 1988, he joined the Department of Electrical Engineering, University of Windsor, Windsor, Ont., Canada, as an Associate Professor; he has been a full Professor since 1989. His research interests include digital filters, fuzzy neural systems, and multimedia signal processing.

Dr. Kwan is a member of the Digital Signal Processing Technical Committee and a member of the Neural Systems and Applications Technical Committee of the IEEE Circuits and Systems Society, a Fellow of the Institution of Electrical Engineers, a Chartered Electrical Engineer in United Kingdom, a registered Electronic Engineer in Hong Kong, and a licensed Professional Engineer in Ontario, Canada.

Lynn Yaling Cai (S’94–M’95) received the B.E. degree in electronics engineering from Tsinghua University, China, in 1985, the M.E. degree in telecommunications from the Second Institute of Aerospace Ministry, China, in 1988, and the Ph.D. degree in electrical engineering from the University of Windsor, Windsor, Ont., Canada, in 1995.

From May 1995 to February 1996, she was a Research Engineer at the Hearing Health Care Research Unit, University of Western Ontario, London, Ont., Canada. Since February 1996, she has been a Research Scientist with the Ginston Research Center, Varian Associate, Palo Alto, CA. Her research interests include signal processing, pattern recognition, fuzzy systems, neural networks, and telecommunications.