NORTH-HOLLAND

Tuning of a Fuzzy Classifier Derived From Data

Shigeo Abe
Hitachi Research Laboratory, Hitachi, Ltd., Hitachi, Japan

Ming-Shong Lan
Asahi Diamond Industrial Co., Ltd., Tokyo, Japan

Ruck Thawonmas
Hitachi Research Laboratory, Hitachi, Ltd., Hitachi, Japan

ABSTRACT

In our previous work we developed a method for extracting fuzzy rules directly from numerical data for pattern classification. The performance of the fuzzy classifier developed using this methodology was comparable to the average performance of neural networks. In this paper, we further develop two methods, a least squares method and an iterative method, for tuning the sensitivity parameters of fuzzy membership functions by which the generalization ability of the classifier is improved. We evaluate our methods using the Fisher iris data and data for numeral recognition of vehicle license plates. The results show that when the tuned sensitivity parameters are applied, the recognition rates are improved to the extent that performance is comparable to or better than the maximum performance obtained by neural networks, but with shorter computational time.

KEYWORDS: ***fuzzy classifiers, rule extraction, tuning, membership function, neural networks, license plate recognition***

I. INTRODUCTION

Multilayered neural networks can learn complex relationships based on numerical input-output data, but analysis of the trained networks is difficult. On the other hand, knowledge acquisition for fuzzy systems is difficult, but once done, analysis or modification of the system is relatively

Address correspondence to Dr. Shigeo Abe, Hitachi Research Laboratories, Hitachi Ltd., 7-1-1 Omika-cho, Hitachi-shi, Ibaraki-ken, 319-12, Japan.

Received January 1994; accepted July 1995.

International Journal of Approximate Reasoning 1996; 14:1–24

655 Avenue of the Americas, New York, NY 10010
0888-613X/96/$15.00
SSDI 0888-613X(95)00076-5

easy. Fuzzy systems with a learning capability have been proposed to fill the gap between these two technologies [1–4]. In [3] we discussed a classifier which used fuzzy rules extracted by the following procedures:

1. Define an activation hyperbox for each class i, by finding the minimum and maximum values of the input data of each class under consideration.
2. If overlapping between the activation hyperboxes of two different classes i and j exists, it is resolved by first defining the overlapping region as an inhibition hyperbox.
3. Then, for the class, i or j, which has data in the inhibition hyperbox, an activation hyperbox is defined in the inhibition hyperbox. If two activation hyperboxes are defined, a similar procedure to procedure 2 is repeated. Procedures 2 and 3 are carried out recursively until the overlapping is resolved.

In [3] we showed that the generalization ability of the fuzzy classifier described was comparable to the average generalization ability of neural networks in terms of successful classification rate.

In Section II of this paper, we describe the fuzzy rule extraction method and the inference mechanism for pattern classification. In Section III, we describe two methods, a least squares method and an iterative method, for tuning the sensitivity parameters of fuzzy membership functions by which the generalization ability of the classifiers is improved. In Section IV, we present the performance evaluation of the proposed approaches with applications to the Fisher iris data and numeral recognition of vehicle license plates, and we also compare the fuzzy classifier with neural networks.

II. FUZZY RULE EXTRACTION AND INFERENCE FOR PATTERN CLASSIFICATION

A. Rule Extraction Method

In the following, we discuss the extraction process for generating fuzzy rules for classifying data with an m-dimensional input vector $\mathbf{x}$ into one of n classes [3]. First assume we have a training data set of input data X_i for class i, where $i = 1, \ldots, n$. Using X_i, an activation hyperbox of level 1, denoted as $A_{ii}(1)$, is defined, which is the maximum region of class i data:

$$A_{ii}(1) = \{\mathbf{x} | v_{iik}(1) \leq x_k \leq V_{iik}(1), k = 1, \ldots, m\}, \tag{1}$$

where x_k = kth element of input vector $\mathbf{x}$;
$v_{iik}(1)$ = minimum value of x_k of $\mathbf{x} \in X_i$;
$V_{iik}(1)$ = maximum value of x_k of $\mathbf{x} \in X_i$.

If the activation hyperboxes $A_{ii}(1)$ and $A_{jj}(1)$ $(j \neq i,\ j = 1, \ldots, n)$ do not overlap, we obtain a fuzzy rule of level 1 for class i as follows:

$$\text{if } \mathbf{x} \text{ is } A_{ii}(1) \text{ then } \mathbf{x} \text{ is class } i. \tag{2}$$

If the activation hyperboxes $A_{ii}(1)$ and $A_{jj}(1)$ overlap, we resolve the overlap recursively by defining the overlapping region as the inhibition hyperbox of level 1 denoted as $I_{ij}(1)$:

$$I_{ij}(1) = \left\{\mathbf{x} \middle| w_{ijk}(1) \leq x_k \leq W_{ijk}(1),\ k = 1, \ldots, m\right\} \tag{3}$$

where $v_{iik}(1) \leq w_{ijk}(1) \leq W_{ijk}(1) \leq V_{iik}(1)$. The minimum and maximum values of inhibition hyperbox $I_{ij}(1)$ are as follows (cf. Figure 1):

1. For $v_{jjk}(1) \leq v_{iik}(1) \leq V_{jjk}(1) < V_{iik}(1)$,

$$w_{ijk}(1) = v_{iik}(1), \qquad W_{ijk}(1) = V_{jjk}(1). \tag{4}$$

2. For $v_{iik}(1) < v_{jjk}(1) \leq V_{iik}(1) \leq V_{jjk}(1)$,

$$w_{ijk}(1) = v_{jjk}(1), \qquad W_{ijk}(1) = V_{iik}(1). \tag{5}$$

3. For $v_{jjk}(1) \leq v_{iik}(1) \leq V_{iik}(1) \leq V_{jjk}(1)$,

$$w_{ijk}(1) = v_{iik}(1), \qquad W_{ijk}(1) = V_{iik}(1). \tag{6}$$

4. For $v_{iik}(1) < v_{jjk}(1) \leq V_{jjk}(1) < V_{iik}(1)$,

$$w_{ijk}(1) = v_{jjk}(1), \qquad W_{ijk}(1) = V_{jjk}(1). \tag{7}$$

However, the inhibition hyperbox defined in this way has a drawback, namely, data that exist on the surface of the inhibition hyperbox may not be classified as either of the two classes as discussed in [3]. To overcome this problem, we expand the originally defined inhibition hyperbox $I_{ij}(1)$, associated with $A_{ii}(1)$ and $A_{jj}(1)$, in the way shown in Figure 2.We denote the expanded inhibition hyperbox as $J_{ij}(1) = \{\mathbf{x} \mid u_{ijk}(1) \leq x_k < U_{ijk}(1),\ k = 1, \ldots, m\}$. The expanded inhibition hyperboxes for $A_{ij}(1)$ and $A_{ji}(1)$ are $J_{ij}(1)$ and $J_{ji}(1)$, respectively, which are different. The expanded inhibition hyperbox $J_{ij}(1)$ is defined as follows (cf. Figure 1):

1. For $v_{jjk}(1) \leq v_{iik}(1) \leq V_{jjk}(1) < V_{iik}(1)$,

$$\begin{aligned} u_{ijk}(1) &= v_{iik}(1), \\ U_{ijk}(1) &= V_{jjk}(1) + \alpha\left[V_{iik}(1) - V_{jjk}(1)\right], \end{aligned} \tag{8}$$

where α $(1 > \alpha > 0)$ is an expansion parameter.

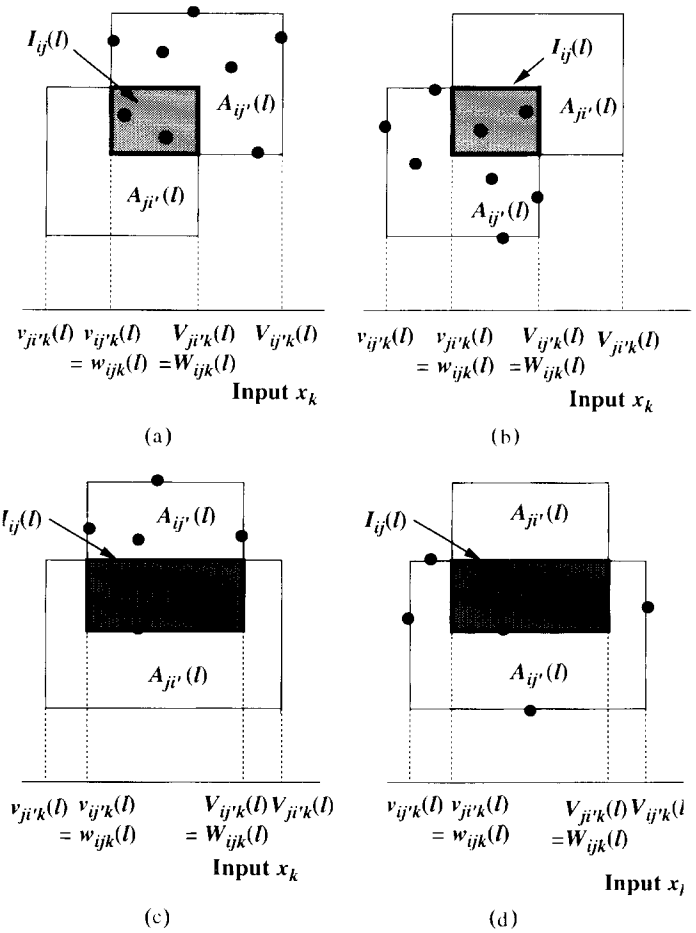


Figure 1. Definition of activation and inhibition hyperboxes ($j' = i$ and $i' = j$ for $l = 1$; $j' = j$ and $i' = i$ for $l \geq 2$). Taken from [3, Figure 3]; © 1995, IEEE.

2. For $v_{iik}(1) < v_{jjk}(1) \leq V_{iik}(1) \leq V_{jjk}(1)$,

$$\begin{aligned} u_{ijk}(1) &= v_{jjk}(1) - \alpha\left[v_{jjk}(1) - v_{iik}(1)\right], \\ U_{ijk}(1) &= V_{iik}(1). \end{aligned} \tag{9}$$

3. For $v_{jjk}(1) \leq v_{iik}(1) \leq V_{iik}(1) \leq V_{jjk}(1)$, we do not expand the inhibition hyperbox for class i, since we need not calculate the degree of membership for the x_k axis. In fact,

$$\begin{aligned} u_{ijk}(1) &= v_{iik}(1), \\ U_{ijk}(1) &= V_{iik}(1). \end{aligned} \tag{10}$$

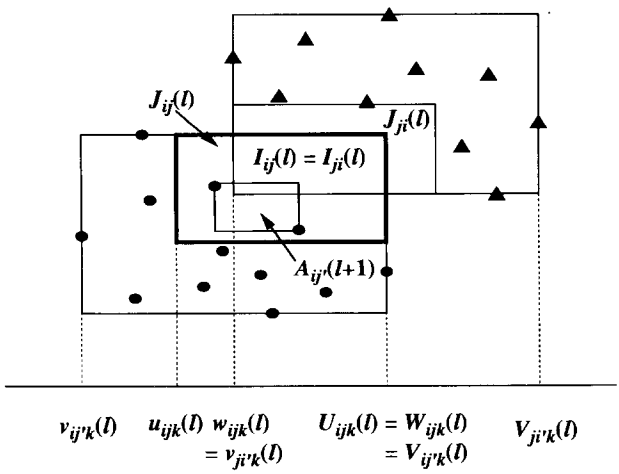


Figure 2. Expansion of the inhibition hyperbox ($j' = i$ and $i' = j$ for $l = 1$; $j' = j$ and $i' = i$ for $l \geq 2$). Taken from [3, Figure 14]; © 1995, IEEE.

4. For $v_{iik}(1) < v_{jjk}(1) \leq V_{jjk}(1) < V_{iik}(1)$,

$$\begin{aligned} u_{ijk}(1) &= v_{jjk}(1) - \alpha\big[v_{jjk}(1) - v_{iik}(1)\big], \\ U_{ijk}(1) &= V_{jjk}(1) + \alpha\big[V_{iik}(1) - V_{jjk}(1)\big]. \end{aligned} \tag{11}$$

Then we define a fuzzy rule of level 1 with inhibition as follows:

$$\text{if } \mathbf{x} \text{ is } A_{ii}(1) \text{ and } \mathbf{x} \text{ is not } J_{ij}(1) \text{ then } \mathbf{x} \text{ is class } i. \tag{12}$$

If $A_{ii}(1)$ is included in $A_{jj}(1)$, i.e., (6) holds for all k, $k = 1, \ldots, m$, then $A_{ii}(1)$ coincides with $I_{ij}(1)$. In this case (12) is a void rule (i.e., it is not created), since no $\mathbf{x}$ can satisfy (12).

If some data belonging to X_i exist in $J_{ij}(1)$, we define the activation hyperbox of level 2, denoted as $A_{ij}(2)$, within the expanded inhibition hyperbox $J_{ij}(1)$ by calculating the minimum and maximum values of x_k based on the data in $J_{ij}(1)$:

$$A_{ij}(2) = \big\{\mathbf{x} \big| v_{ijk}(2) \leq x_k \leq V_{ijk}(2),\ k = 1, \ldots, m\big\} \tag{13}$$

where $\mathbf{x} \in X_i$ and $\mathbf{x} \in J_{ij}(1)$, and where

$v_{ijk}(2)$ = minimum value of x_k where $\mathbf{x} \in X_i$ and $\mathbf{x}$ is in $J_{ij}(1)$,
$V_{ijk}(2)$ = maximum value of x_k where $\mathbf{x} \in X_i$ and $\mathbf{x}$ is in $J_{ij}(1)$,

$$u_{ijk}(1) \leq v_{ijk}(2) \leq x_k \leq V_{ijk}(2) \leq U_{ijk}(1). \tag{14}$$

If there is only one activation hyperbox of level 2 or there are two activation hyperboxes but they do not overlap, we define a fuzzy rule of level 2 for class i as follows:

$$\text{if } \mathbf{x} \text{ is } A_{ij}(2) \text{ then } \mathbf{x} \text{ is class } i. \tag{15}$$

If $A_{ij}(2)$ and $A_{ji}(2)$ overlap, the overlapping region of level 2 is denoted as $I_{ij}(2)$:

$$I_{ij}(2) = \left\{\mathbf{x} \middle| w_{ijk}(2) \le x_k \le W_{ijk}(2), i = 1, \ldots, m\right\}, \tag{16}$$

where $v_{ijk}(2) \le w_{ijk}(2) \le W_{ijk}(2) \le V_{ijk}(2)$.

Similarly to what has been described for level 1, we define the expanded inhibition hyperbox $J_{ij}(2)$:

$$J_{ij}(2) = \left\{\mathbf{x} \middle| u_{ijk}(2) \le x_k \le U_{ijk}(2), k = 1, \ldots, m\right\} \tag{17}$$

where $u_{ijk}(2) \le w_{ijk}(2) \le W_{ijk}(2) \le U_{ijk}(2)$.

Then we define a fuzzy rule of level 2 with inhibition:

$$\text{if } \mathbf{x} \text{ is } A_{ij}(2) \text{ and } \mathbf{x} \text{ is not } J_{ij}(2) \text{ then } \mathbf{x} \text{ is class } i. \tag{18}$$

Fuzzy rules of levels higher than 2 can be defined in a similar manner if an overlap can be defined. In a general form, the fuzzy rule $r_{ij'}(l)$ of level l (≥ 1) without inhibition can be expressed as follows:

$$\text{if } \mathbf{x} \text{ is } A_{ij'}(l) \text{ then } \mathbf{x} \text{ is class } i, \tag{19}$$

where $j' = i$ for $l = 1$ and $j' = j$ for $l \ge 2$. Likewise, the fuzzy rule $r_{ij'}(l)$ of level l with inhibition can be expressed as follows:

$$\text{if } \mathbf{x} \text{ is } A_{ij'}(l) \text{ and } \mathbf{x} \text{ is not } J_{ij}(l) \text{ then } \mathbf{x} \text{ is class } i. \tag{20}$$

The recursion process for defining fuzzy rules terminates when $A_{ij'}(l)$ and $A_{ji'}(l)$ do not overlap or when $A_{ij'}(l) = A_{ji'}(l) = I_{ij}(l-1)$ holds, where $j' = i$ and $i' = j$ for $l = 1$ and $j' = j$ and $i' = i$ for $l \ge 2$. In the latter case, since the overlap cannot be resolved by the recursive process, instead of defining $A_{ij'}(l)$ and $A_{ji'}(l)$, for each datum of class i and/or j in $I_{ij}(l-1)$ we define an activation hyperbox which includes only that datum. We do not further define inhibition and activation hyperboxes of levels higher than l, because as long as no identical data exist in both classes i and j, no overlap exists between the activation hyperboxes of level l.

B. Fuzzy Inference Mechanism

MEMBERSHIP FUNCTION FOR ACTIVATION HYPERBOXES For pattern classification, it is reasonable to assume that the degree of membership of $\mathbf{x}$ for a fuzzy rule given by (19) is 1 if $\mathbf{x}$ is in the activation hyperbox $A_{ij}(l)$, and that the degree of membership decreases as $\mathbf{x}$ moves away from the activation hyperbox. Namely, if all the input variables are normalized to the same scale, e.g., between 0 and 1, then the contour surface on which every location has the same degree of membership is parallel to, and lies at an equal distance from, the surface of the activation hyperbox, as illustrated in Figure 3. To realize a membership function with this characteristic we use the following function, which is similar to that proposed in [1]:

$$m_{A_{ij}(l)}(\mathbf{x}) = \min_{k=1,\ldots,m} m_{A_{ij}(l)}(\mathbf{x}, k), \tag{21}$$

$$m_{A_{ij}(l)}(\mathbf{x}, k) = \left\{1 - \max\left(0, \min\left(1, \gamma_i\left[\nu_{ijk}(l) - x_k\right]\right)\right)\right\} \times \left\{1 - \max\left(0, \min\left(1, \gamma_i\left[x_k - V_{ijk}(l)\right]\right)\right)\right\} \tag{22}$$

where γ_i is the sensitivity parameter for class i. Figure 4 is the one-dimensional membership function given by (22). In [3] we used the same sensitivity parameter for all the rules, but in fact, different values can be used for different classes. Therefore, in this paper, we allow different

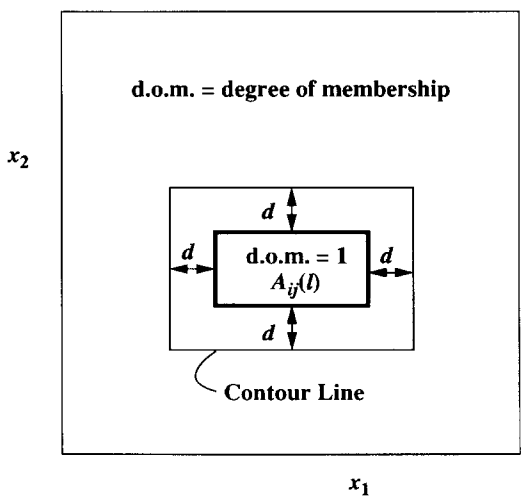


Figure 3. The contour line of the membership function for the activation hyperbox (two-dimensional case). Taken from [3, Figure 4]; © 1995, IEEE.

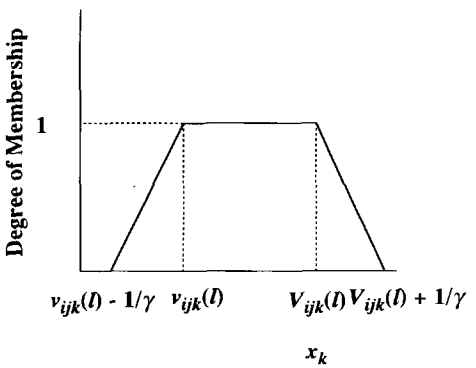


Figure 4. One-dimensional membership function of the activation hyperbox $A_{ij}(l)$. Taken from [3, Figure 5]; © 1995, IEEE.

values for the sensitivity parameters of different classes, and tuning methods are developed to tune the sensitivity parameters.

Thus, the degree of membership of $\mathbf{x}$ for a fuzzy rule $r_{ij}(l)$ given by (19) is

$$d_{r_{ij}(l)}(\mathbf{x}) = m_{A_{ij}(l)}(\mathbf{x}). \tag{23}$$

MEMBERSHIP FUNCTION FOR INHIBITION HYPERBOXES The degree of membership of $\mathbf{x}$ for a fuzzy rule given by (20) is 1 when $\mathbf{x}$ is in the activation hyperbox but not within the expanded inhibition hyperbox, i.e., $\mathbf{x}$ is in $\overline{A_{ij'}(l) - J_{ij}(l)}$, where $\bar{S}$ denotes the closure of the set S, and where $j' = i$ for $l = 1$ and $j' = j$ for $l \geq 2$. If $\mathbf{x}$ moves away from this region, the degree of membership decreases. Namely, in this case it is also favorable for the contour surface to be parallel to, and lie at an equal distance from, the surface of $\overline{A_{ij'}(l) - J_{ij}(l)}$, as shown in Figure 5. [If $A_{ij'}(l) = I_{ij}(l)$, i.e., if the rule is void, we do not calculate the degree of membership for this rule.] To realize this membership function we first define a region $H_{ij}(l)$ associated with $A_{ij'}(l)$ and $I_{ij}(l)$ as follows (cf. Figure 1):

$$\begin{aligned} H_{ij}(l) = \{\mathbf{x} \mid & x_k \leq U_{ijk}(l) \text{ for } v_{ji'k}(l) \leq v_{ij'k}(l) \leq V_{ji'k}(l) < V_{ij'k}(l), \\ & x_k \geq u_{ijk}(l) \text{ for } v_{ij'k}(l) < v_{ji'k}(l) \leq V_{ij'k}(l) \leq V_{ji'k}(l), \\ & -\infty < x_k < \infty \text{ for } v_{ji'k}(l) \leq v_{ij'k}(l) \leq V_{ij'k}(l) \leq V_{ji'k}(l), \\ & u_{ijk}(l) \leq x_k \leq U_{ijk}(l) \text{ for } v_{ij'k}(l) \\ & < v_{ji'k}(l) \leq V_{ji'k}(l) < V_{ij'k}(l), \\ & k = 1, \ldots, m\}, \end{aligned} \tag{24}$$

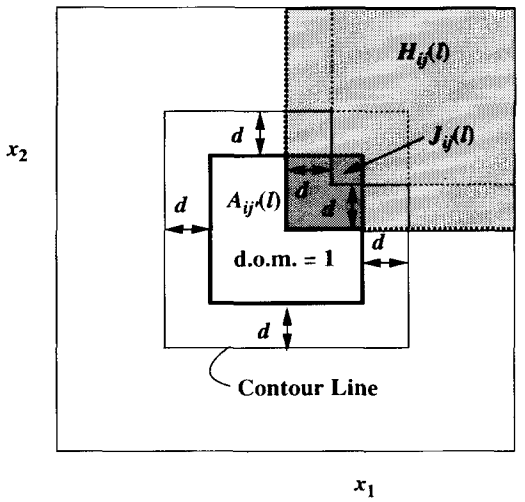


Figure 5. The contour line of the membership function for the activation and inhibition hyperboxes (Two-dimensional case.) Taken from [3, Figure 6]; © 1995, IEEE.

where $j' = i$ and $i' = j$ for $l = 1$, $j' = j$ and $i' = i$ for $l \geq 2$, and $H_{ij}(l)$ and $H_{ji}(l)$ are in general different. According to the definition,

$$H_{ij}(l) \supset J_{ij}(l). \tag{25}$$

The region $H_{ij}(l)$ constitutes an input region where the expanded inhibition hyperbox affects the degree of membership of the rule given by (20). If $\mathbf{x} \notin H_{ij}(l)$, the degree of membership for a fuzzy rule $r_{ij}(l)$ given by (20) is the same as (23). Thus, for $\mathbf{x} \in J_{ij}(l)$ the degree of membership $m_{J_{ij}(l)}(\mathbf{x})$ is given by

$$m_{J_{ij}(l)}(\mathbf{x}) = \max_{k=1,\ldots,m} m_{J_{ij}(l)}(\mathbf{x}, k), \tag{26}$$

where $m_{J_{ij}(l)}(\mathbf{x}, k)$ is the degree of membership of x_k and is calculated as follows:

1. For $v_{ji'k}(l) \leq v_{ij'k}(l) \leq V_{ji'k}(l) < V_{ij'k}(l)$ [cf. Figure 1(a)],

$$m_{J_{ij}(l)}(\mathbf{x}, k) = 1 - \max\bigl(0, \min\bigl(1, \gamma_i\bigl[U_{ijk}(l) - x_k\bigr]\bigr)\bigr). \tag{27}$$

2. For $v_{ij'k}(l) < v_{ji'k}(l) \leq V_{ij'k}(l) \leq V_{ji'k}(l)$ [cf. Figure 1(b)],

$$m_{J_{ij}(l)}(\mathbf{x}, k) = 1 - \max\bigl(0, \min\bigl(1, \gamma_i\bigl[x_k - u_{ijk}(l)\bigr]\bigr)\bigr). \tag{28}$$

3. For $v_{ji'k}(l) \le v_{ij'k}(l) \le V_{ij'k}(l) \le V_{ji'k}(l)$, since $x_k = v_{ij'k}(l)$ and $x_k = V_{ij'k}(l)$ do not constitute the surface of $A_{ij'}(l) - J_{ij}(l)$, it is not necessary to define a membership function in the x_k axis. Thus we set

$$m_{J_{ij}(l)}(\mathbf{x}, k) = 0. \tag{29}$$

Equation (29) holds for all k, where $k = 1, \ldots, m$, only when $A_{ji'}(l) \supset A_{ij'}(l) = I_{ij}(l)$, in other words, when the rule is a void rule. Thus, the x_k axis is ignored when calculating the degree of membership using (29) and (26).

4. For $v_{ij'k}(l) < v_{ji'k}(l) \le V_{ji'k}(l) < V_{ij'k}(l)$ [cf. Figure 1(d)]

$$m_{J_{ij}(l)}(\mathbf{x}, k) = \begin{cases} 1 - \max\bigl(0, \min\bigl(1, \gamma_i\bigl[U_{ijk}(l) - x_k\bigr]\bigr)\bigr) \\ \qquad \text{for} \quad \dfrac{u_{ijk}(l) + U_{ijk}(l)}{2} \le x_k \le U_{ijk}(l), \\ 1 - \max\bigl(0, \min\bigl(1, \gamma_i\bigl[x_k - u_{ijk}(l)\bigr]\bigr)\bigr) \\ \qquad \text{for} \quad u_{ijk}(l) \le x_k \le \dfrac{u_{ijk}(l) + U_{ijk}(l)}{2}. \end{cases} \tag{30}$$

Then the degree of membership for $\mathbf{x} \in H_{ij}(l)$ and $\mathbf{x} \notin J_{ij}(l)$ is obtained by calculating both $m_{A_{ij}(l)}(\mathbf{x})$ and $m_{J_{ij}(l)}(\mathbf{x})$, and taking the minimum, i.e., $\min(m_{A_{ij}(l)}(\mathbf{x}), m_{J_{ij}(l)}(\mathbf{x}))$. Thus $d_{r_{ij}(l)}(\mathbf{x})$ for (20) is given by

$$d_{r_{ij}(l)}(\mathbf{x}) = \begin{cases} m_{A_{ij}(l)}(\mathbf{x}) & \text{for} \quad \mathbf{x} \notin H_{ij}(l), \\ m_{J_{ij}(l)}(\mathbf{x}) & \text{for } \mathbf{x} \in J_{ij}(l), \\ \min\bigl(m_{A_{ij}(l)}(\mathbf{x}), m_{J_{ij}(l)}(\mathbf{x})\bigr) & \text{for} \quad \mathbf{x} \in H_{ij}(l) \text{ and } \mathbf{x} \notin J_{ij}(l) \end{cases} \tag{31}$$

Since $m_{A_{ij}(l)}(\mathbf{x}) = 1$ for $\mathbf{x} \in J_{ij}(l)$, Equation (31) can be rewritten as follows:

$$d_{r_{ij}(l)}(\mathbf{x}) = \begin{cases} m_{A_{ij}(l)}(\mathbf{x}) & \text{for} \quad \mathbf{x} \notin H_{ij}(l), \\ \min\bigl(m_{A_{ij}(l)}(\mathbf{x}), m_{J_{ij}(l)}(\mathbf{x})\bigr) & \text{for } \mathbf{x} \in H_{ij}(l). \end{cases} \tag{32}$$

RULE INFERENCE The final degree of membership of $\mathbf{x}$ for a set of fuzzy rules $\{r_{ij}(l) \mid l = 1, \ldots\}$, denoted as $d_{r_{ij}}(\mathbf{x})$, is given by

$$d_{r_{ij}}(\mathbf{x}) = \max_{l=1,\ldots} d_{r_{ij}(l)}(\mathbf{x}). \tag{33}$$

We take the maximum because the activation hyperbox $A_{ij}(l+1)$, if it exists, is included in the expanded inhibition hyperbox $J_{ij}(l)$, and thus each fuzzy rule in $\{r_{ij}(l) \mid l = 1, \ldots\}$ is exclusive of the others.

Now the degree of membership of $\mathbf{x}$ for class i, denoted as $d_i(\mathbf{x})$, is given by

$$d_i(\mathbf{x}) = \min_{\substack{j \neq i,\ j=1,\ldots,n,\\ A_{ii}(1) \cap A_{jj}(1) \neq \varnothing}} d_{r_{ij}}(\mathbf{x}). \tag{34}$$

When the activation hyperbox of class i overlaps with those of classes j and k, we resolve the conflict, independently, first between classes i and j, then between classes i and k. This process is reflected by taking the minimum in (34). For example, if $d_{r_{ij}}(\mathbf{x}) = 1$ and $d_{r_{ik}}(\mathbf{x}) = 0$, this means that $\mathbf{x}$ is in the region inhibited by the inhibition hyperbox between classes i and k and thus $\mathbf{x}$ should not be classified as class i.

The input $\mathbf{x}$ is finally classified as class i if $d_i(x)$ is the maximum among $d_j(\mathbf{x})$, where $j = 1, \ldots, n$.

Now we consider how the values of sensitivity parameters affect classification. First consider the case when the sensitivity parameters γ_i are large, in other words, the generalization region for each class is small. In this case, a region exists in which data cannot be classified because the degrees of membership for all the classes are zero. By using small sensitivity parameters γ_i, thus increasing the generalization region of each class, all the data in the input space can be classified. If we make the sensitivity parameters γ_i small enough so that the degree of membership for any given point in the input space is greater than 0, then the class boundary does not change even if we make the sensitivity parameter γ_i smaller. This means that as the sensitivity parameters γ_i are decreased from large values to small ones, the recognition rate of test data increases and reaches a plateau. If each input variable is normalized as $[0, 1]$, it is sufficient to set the sensitivity parameter smaller than 1 to obtain the maximum recognition rate. Or if we want to know whether the input data are used for training or not, we may set the sensitivity parameters large. If a datum is not classified because the degree of membership is zero, we know that data which are near to this datum are not used for training.

III. TUNING OF SENSITIVITY PARAMETERS

A. Basic Idea

The fuzzy classifier described in Section II has a 100% recognition rate for the training data set so long as no identical data are presented in different classes. As to the numeral recognition system described in [3], the

generalization ability of the classifier was comparable to, but not better than, the average generalization ability of neural networks when the characteristics of the training data differed significantly from those of the test data set. Furthermore, the reason why the generalization ability of the fuzzy classifier was inferior to the maximum ability of neural networks was that the sensitivity parameters of membership functions used before were not optimized and the same value was used for different classes. The generalization ability of the fuzzy classifier can be improved by tuning the sensitivity parameters; in the following, we discuss why this is possible and present two tuning methods.

Assume that a test datum of class i is misclassified as class j; if its degree of membership with respect to class j is 1, this means that this class i test datum resides in an activation hyperbox of class j. In this case, if we want to make sure that this test datum is to be correctly classified as class i, the existing fuzzy rules need to be modified. However, if the degree of membership with respect to class j is less than 1, the test datum can be correctly classified as class i by decreasing the sensitivity parameter γ_i (i.e., increasing the generalization region of class i) or increasing the sensitivity parameter γ_j (i.e., decreasing the generalization region of class j). Changes should be made so that the data which were correctly classified before the change remain correctly classified.

Therefore, if we can develop a method for tuning the sensitivity parameters, the fuzzy classifier will be more favorable than neural networks because of its low development effort and operational maintenance. That is, suppose we have a set of input-output data available for developing a classifier. If we are going to develop a neural network classifier, usually we first divide the data set into two: the training data set and the test data set. Using the training data set, we train a neural network classifier, and then, using the test data set, we test its performance. If the classification performance is not satisfactory, its improvement is not straightforward, because no good method for further tuning the trained classifier exists. Thus, we need to retrain the network with different initial connection weights (because the convergence of a neural network depends on its initial connection weights), or change the network structure, or repartition the original data set until satisfactory performance is achieved, or train the network using all the available input-output data. The development process described above, then, is very time-consuming, especially when the data set is large, because the process may involve training a neural network with different initial conditions or trying many neural network structures.

On the other hand, a much shorter time is needed to develop a fuzzy classifier, since the process to extract fuzzy rules is very fast; moreover, the

fuzzy classifier can be easily fine-tuned by tuning the sensitivity parameters. To develop a fuzzy classifier, we also first divide the available data set into a training data set and a test data set. Then, the training data set is divided into an extraction data set and a tuning data set. Fuzzy rules are extracted using the extraction data set. If the classification performance of the fuzzy classifier for the test data set is not satisfactory, we can apply one of the tuning methods, which will be discussed later, to tune the sensitivity parameters using the tuning data set. If the performance is still not satisfactory, we can add those tuning data that are not correctly classified to the extraction data set and regenerate fuzzy rules, and then use the remaining tuning data for tuning the sensitivity parameters. In addition to tuning during the development, after the fuzzy classifier is deployed for actual applications, its performance may be further improved by tuning the sensitivity parameters on the fly with new misclassification data.

In the following we discuss two tuning approaches: a batch processing approach based on a least squares method, and an iterative approach suitable for on-line application.

B. Tuning Using a Least Squares Method

Assume that a fuzzy classifier is created using a set of training data. For a given input $\mathbf{x}$, if the degree of membership with respect to class i is less than 1—i.e., $1 > d_i(\mathbf{x})$—then either the input $\mathbf{x}$ is outside the activation hyperboxes of class i, $A_{ij'}(l)$, or it is inside the expanded inhibition hyperboxes of class i, $J_{ij}(l)$. In this case, the degree of membership, $d_{r_{ij}(l)}(\mathbf{x})$, of a fuzzy rule $r_{ij}(l)$ is determined by the maximum or minimum distance between $\mathbf{x}$ and the hyperplane that includes the surface of $A_{ij'}(l)$ or $A_{ij'}(l) - J_{ij}(l)$; the hyperplane is orthogonal to one of the input variable axes. This hyperplane by which $d_{r_{ij}(l)}(\mathbf{x})$ is determined remains the same even if γ_i is changed, because the same γ_i is used for all input variables by class i rules. Let the hyperplane which determines $d_{r_{ij}(l)}(\mathbf{x})$ be orthogonal to the kth input axis and intersect it at b_k. Then the degree of membership $d_{r_{ij}(l)}(\mathbf{x})$ is given by $\min(0, 1 - \gamma_i|x_k - b_k|)$. Also, since we use the same γ_i value for all the input variables in the rules of class i, the hyperplane that determines $d_i(\mathbf{x})$ remains the same even if we vary the value of the sensitivity parameter γ_i.

Based on the above discussion, a very simple least squares method for determining sensitivity parameters is developed. For a given input $\mathbf{x}$ satisfying $d_i(\mathbf{x}) < 1$ for all $i = 1, \ldots, n$, we define a distance vector $\mathbf{t} = (t_1, \ldots, t_n)$, where $t_k = |x_l - b_l|$ and the hyperplane that determines $d_k(\mathbf{x})$ is orthogonal to the lth axis and intersects it at b_l. Let the distance vector of the jth datum of class i be $\mathbf{t}_{ij} = (t_{ij,1}, \ldots, t_{ij,n})$, where $1 \leq j \leq N_i$; N_i is

the number of data of class i for tuning. If

$$t_{ij,i} < t_{ij,k} \qquad \text{for all} \quad k \neq i \tag{35}$$

holds, the input datum $\mathbf{x}_{ij}$ can be correctly classified if the same value is used for all γ_i. Now for a given datum $\mathbf{x}_{ij}$, if $d_k(\mathbf{x}_{ij}) = 1$ for $k \neq i$, the fuzzy classifier is not able to correctly classify the datum, while if $d_i(\mathbf{x}_{ij}) = 1$, the classifier is able to correctly classify the datum, irrespective of the values of the sensitivity parameters. Thus the data which satisfy either of the two conditions are not included in $\mathbf{x}_{ij}$, where $i = 1, \ldots, n$ and $j = 1, \ldots, N_i$, and hence their associated membership vectors are not included in $\mathbf{t}_{ij}$, where $i = 1, \ldots, n$ and $j = 1, \ldots, N_i$.

Since the target value of $d_k(\mathbf{x}_{ij})$ is 1 if $k = i$ and 0 otherwise, the optimum sensitivity parameter for each class can be determined by minimizing the following error function.

$$E = \frac{1}{2}\left(\sum_{i=1}^{n} \sum_{j=1}^{N_i} (\gamma_i t_{ij,i})^2 + \sum_{k=1}^{N} \sum_{\substack{i=1 \\ i \neq k}}^{n} \sum_{j=1}^{N_i} (1 - \gamma_k t_{ij,k})^2 \right). \tag{36}$$

Equation (36) is minimized when the following equation holds:

$$\frac{\partial E}{\partial \gamma_i} = \left(\sum_{k=1}^{n} \sum_{j=1}^{N_k} (t_{kj,i})^2 \right) \gamma_i - \sum_{\substack{k=1 \\ k \neq i}}^{n} \sum_{j=1}^{N_k} t_{kj,i} = 0. \tag{37}$$

Therefore the optimum value for γ_i can be obtained as

$$\gamma_i = \frac{\displaystyle\sum_{\substack{k=1 \\ k \neq i}}^{n} \sum_{j=1}^{N_k} t_{kj,i}}{\displaystyle\sum_{k=1}^{n} \sum_{j=1}^{N_k} (t_{kj,i})^2}. \tag{38}$$

C. Iterative Tuning

In this section we discuss a one-path tuning method in which the tuning sequence is determined by using a graph for data $\mathbf{x}_{ij}$, where $i = 1, \ldots, n$ and $j = 1, \ldots, N_i$. Here, iterative tuning of the membership function is based on the idea that if some class i data are misclassified as class j but no class j data are misclassified as class i, the misclassification may be resolved by expanding the existence region of class i, i.e., decreasing the slope of the membership function for class i. And we assume that for any given $\mathbf{x}_{ij}$ outside of all the activation hyperboxes, its degree of membership with respect to any class is less than 1.

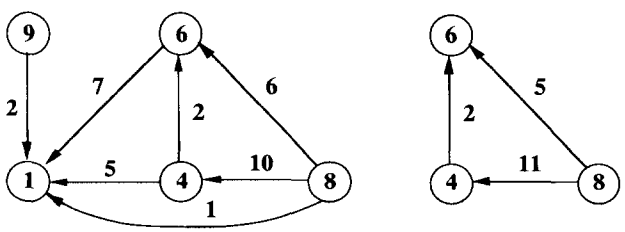
Before describing the iterative tuning method, we first present a graphical representation used to describe the state of misclassification for a given fuzzy classifier and a given set of data. A graph is composed of nodes and directional links, each with an associated number. A node represents a class, a directional link between two nodes indicates that some data of the emanating node are misclassified as the class represented by the ending node, and the number associated with a link represents the number of misclassified data. Figure 6(a) illustrates a graph for a recognition system of vehicle license plates and a test data set. Nodes 0, 2, 3, 5, and 7 are not present; that means that no data are misclassified as class 0, 2, 3, 5, or 7, and no data of classes 0, 2, 3, 5, and 7 are misclassified. A node in a graph may be one of the following three types: (1) a sink node at which links end only, (2) a source node from which links emanate only, and (3) a dual node where links emanate and end. In Figure 6(a), nodes 8 and 9 are source nodes, node 1 is a sink node, and the rest of the nodes are dual nodes.

In this section, we first discuss the range of γ_i that can be changed without causing misclassification to occur for those data that are correctly classified before tuning γ_i. Suppose that a given datum $\mathbf{x}_{ij}$ is correctly classified for a given sensitivity parameter value γ_i. We can increase γ_i to $\gamma_i^U(\mathbf{x}_{ij})$ according to (39), as illustrated in Figure 7, which will not result in misclassifying $\mathbf{x}_{ij}$:

$$\gamma_i^U(\mathbf{x}_{ij}) = \min_{\substack{k=1,\ldots,n \\ k \neq i}} \frac{t_{ij,k}}{t_{ij,i}} \gamma_i. \tag{39}$$

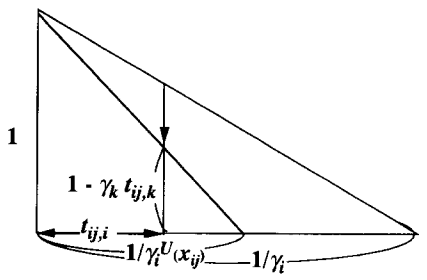
We can also decrease γ_k to $\gamma_k^L(\mathbf{x}_{ij})$ according to (40), which will not result in misclassifying $\mathbf{x}_{ij}$:

$$\gamma_k^L(\mathbf{x}_{ij}) = \frac{t_{ij,i}}{t_{ij,k}} \gamma_k. \tag{40}$$

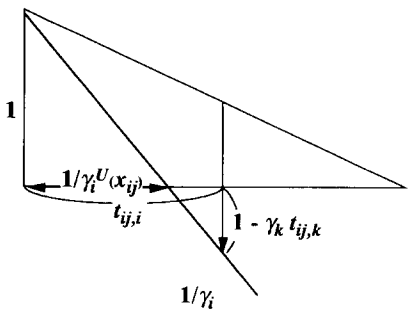


(a) **Before tuning** (b) **After iterative tuning**

Figure 6. A directional graph for misclassification of data for numeral recognition on vehicle license plates (200 training data, 1430 test data, 12 inputs, $\alpha = 0.001$).



(a) $1 - \gamma_k t_{ij,k} > 0$



(b) $1 - \gamma_k t_{ij,k} \leq 0$

Figure 7. Calculation of the upper limit of the sensitivity parameter ($t_{ij,k}/t_{ij,i}$ is the minimum).

Thus if we vary γ_i within

$$\gamma_i^L(\mathbf{x}_{ij}) < \gamma_i < \gamma_i^U(\mathbf{x}_{ij}), \tag{41}$$

the given input $\mathbf{x}_{ij}$ remains correctly classified as class i.

Now let

$$\gamma_i^U = \min_{\mathbf{x}_{ij} \in C_i} \gamma_i^U(\mathbf{x}_{ij}) = \gamma_i \times \min_{\mathbf{x}_{ij} \in C_i} \left(\min_{\substack{k=1,\ldots,n \\ k \neq i}} \frac{t_{ij,k}}{t_{ij,i}} \right) = \gamma_i \times \rho_i^U \tag{42}$$

and

$$\gamma_i^L = \max_{\substack{\mathbf{x}_{kj} \in C_k \\ k=1,\ldots,n, \\ k \neq i}} \gamma_i^L(\mathbf{x}_{kj}) = \gamma_i \times \max_{\mathbf{x}_{kj} \in C_k} \left(\max_{\substack{k=1,\ldots,n, \\ k \neq i}} \frac{t_{kj,k}}{t_{kj,i}} \right) = \gamma_i \times \rho_i^L, \tag{43}$$

where C_i is a set of data belonging to class i which are correctly classified, γ_i^U is the upper limit of the sensitivity parameter for class i, γ_i^L is the lower limit of the sensitivity parameter for class i, ρ_i^U is the upper limit ratio for class i, and ρ_i^L is the lower limit ratio for class i. Thus when varying γ_i within

$$\gamma_i^L < \gamma_i < \gamma_i^U, \tag{44}$$

all the initially correctly classified data remain correctly classified.

Let node i be a sink node, and a given $\mathbf{x}_{kj}$ be classified as class i. Then misclassification can be resolved if γ_i is changed to a value larger than $\gamma_{i,kj}$ which satisfies

$$\gamma_{i,kj} = \frac{t_{ij,k}}{t_{ij,i}} \gamma_i. \tag{45}$$

If $\gamma_{i,kj}$ given by (45) is within the range given by (44), we can resolve misclassification without causing any data which are correctly classified before the sensitivity parameter is changed to become misclassified. We calculate (45) for all the data $\mathbf{x}_{kj}$ which are misclassified as class i. Let the maximum value among $\gamma_{i,kj}$ which satisfy (44) be γ_i', and then change γ_i to that value, which is within the range (44) and larger than γ_i'. By this operation we can reduce the number of misclassifications. Then we can update the upper and lower limits of the sensitivity parameters by updating the upper and lower limit ratios according to their current values and the newly correctly classified data. Namely,

$$\text{new}\,\rho_k^U = \min\left(\rho_k^U, \min_{\mathbf{x}_{kj} \in C_k'} \left(\min_{\substack{i=1,\ldots,n \\ i \neq k}} \frac{t_{kj,i}}{t_{kj,k}} \right) \right), \tag{46}$$

$$\text{new}\,\rho_i^L = \max\left(\rho_i^L, \max_{\mathbf{x}_{kj} \in C_k'} \left(\max_{\substack{k=1,\ldots,n, \\ k \neq i}} \frac{t_{kj,k}}{t_{kj,i}} \right) \right), \tag{47}$$

where C_i' is a set of data newly classified as class i.

Based on a similar discussion to the above, for a source node i, we can decrease the number of misclassifications by decreasing the sensitivity parameter γ_i. After tuning the sensitivity parameters for source and sink nodes, we then tune the sensitivity parameters for dual nodes. When tuning the sensitivity parameters of a dual node, whether its value should be decreased or increased depends on the number of previously misclassified data which are now correctly classified without making previously correctly classified data misclassified.

The range of γ_i given by (44) is a range that ensures that the originally correctly classified data remain correctly classified when γ_i is varied. But if the number of increments of correct classifications exceeds that of increments of misclassifications due to the change of γ_i, a decrease in the total number of misclassifications results. This may happen in particular for dual nodes. If it does, we set the values which violate (44) to γ_i.

Since many local minima may exist in the minimization process, the final outcome may depend on the tuning order. For simplicity, we tune source and sink nodes first and then tune dual nodes. Thus for the case shown in Figure 6(a), the order of tuning is as follows:

$$\{1, 8, 9\} < \{4, 6\}, \tag{48}$$

where nodes in the same set are to be processed together in an arbitrary order, and $A < B$ denotes that nodes in set A need to be processed prior to those in set B.

This iterative tuning approach is well suited for on-the-fly tuning. Namely, when a datum is provided to the classifier, if it is correctly classified, we then update the upper and lower ratios. If it is not correctly classified, we check whether correct classification is possible by changing the sensitivity parameters without causing misclassification of data which were previously classified correctly. If the latter case occurs, we change the sensitivity parameter and update the corresponding upper and lower limit ratios. The major advantage of this method is that past input-output data are not required for tuning: tuning is done by using only the current sensitivity parameter, the upper and lower limit ratios, and the current input-output data.

D. Comparison of the Two Tuning Methods

The least squares method can determine the sensitivity parameters using both correctly classified data and misclassified data, but the iterative method is applicable only when there are misclassified data. Thus the least squares method is suited for use in the development stage of classifiers, while the iterative tuning method is suited for on-line use.

The least squares method can be used in the following way. Suppose we have extraction data for fuzzy rule extraction, and tuning data for tuning the sensitivity parameters. The extraction and tuning data may respectively be the training data and the test data, or we may split the training data into extraction data and tuning data. First we extract fuzzy rules from the extraction data. Then, using tuning data, we tune the sensitivity parameters. The least squares method works better if there are only correctly classified data, because if there are misclassified data, they may act to

suppress the effect of correctly classified data. Thus if there are misclassified tuning data, we add them to the extraction data. We repeat the extraction and tuning process until there are no misclassified data among the tuning data.

IV. PERFORMANCE EVALUATION

To show the improved performance of the fuzzy classifier by properly tuning sensitivity parameter, we used the same data used in [3]. Also, to allow performance comparison between fuzzy classifiers and neural networks, we divided the training data into two: extraction data and tuning data. Using the extraction data, we extracted fuzzy rules and tuned the sensitivity parameters using the tuning data. If a 100% recognition rate for tuning data was not obtained by the least squares method, we added the misclassified data to the extraction data, and again we extracted the fuzzy rules from the extraction data. Usually by this addition, the recognition rate for the tuning data became 100%; but if not, after one more addition of misclassified data to the extraction data, the recognition rate became 100%.

To evaluate the performance of the iterative tuning method, we used the training data as the extraction data and the test data as the tuning data.

A. Iris Data

The Fisher iris data [5] consisted of 150 data with four input features and three classes. In our study, the training data set was composed of the first 25 data of each class, while the test data set was composed of the remaining 25 data of each class.

To examine the performance of the least squares method we divided the training data set into the 38 extraction data and 37 tuning data. In Table 1, case 1 shows the results when all the training data were used for training, and case 2 shows the results when tuning by the least squares method was performed. Using the original extraction and tuning data, four data in the tuning data were misclassified; thus we added them to the extraction data. With this addition all the tuning data were correctly classified. Comparing cases 1 and 2, we saw the number of rules of case 2 was equal or smaller than that of case 1 for the same expansion parameter α. Also, the minimum number of misclassifications, i.e., 1 was achieved in case 2.

To examine the effect of the iterative tuning method we used the training data as the extraction data and the test data as the tuning data. When $\alpha = 0.001$, data of class 0 were all correctly classified, but two of the class 1 data were classified as class 2 and two of the class 2 data were

Table 1. Performance of the Fuzzy Classifier for the Iris Data[a]

	Case 1		Case 2	
α	No. of Rules	No. of Wrongs	No. of Rule	.No. of Wrong
0.001	5	6	5	6
0.1	7	5	7	6
0.2	7	5	7	5
0.3	9	5	7	5
0.4	9	4	7	4
0.5	9	4	8	4
0.6	11	4	9	1
0.7	11	3	9	3
0.8	13	3	10	3
0.9	17	2	11	1
0.99	17	2	11	2

[a] In case 1 all the training data were used for fuzzy rule extraction. In case 2 the training data set was divided into two, one for fuzzy rule extraction and one for sensitivity parameter tuning.

classified as class 1. This meant both nodes 1 and 2 were dual nodes. Meanwhile, since the degrees of membership with respect to the misclassified classes were 1, neither of the tuning methods described in Section III could be used to improve the recognition rate. When $\alpha = 0.9$, nodes 1 and 2 were dual nodes with one misclassification and the degrees of membership were less than 1. Using the least squares method, we could not improve the recognition rate. This was attributed to the following: Since there were only dual nodes, the misclassified data tried to determine the sensitivity parameters so that they were correctly classified. But since the corrections of sensitivity parameters were in opposite directions, they contradicted each other and there was no improvement. Using the iterative tuning method, we could resolve misclassification for class 2 by increasing the sensitivity parameter γ_1 or by decreasing the sensitivity parameter γ_2. With this change, node 2 became the source node with one misclassification, and node 1 became the sink node.

B. License Plate Recognition System

The data used in this study were originally collected to develop a license plate recognition system [6, 7]. Numerals from 0 to 9 were considered. Each of these data consisted of 12 input features extracted from the images of running cars as taken by a TV camera. There were 1630 data, which were divided into training and test data sets with different numbers

of input features; the details are summarized below:

- Data set 1: 200 training data and 1430 test data with 12 input features.
- Data set 2: 810 training data and 820 test data with 12 input features.
- Data set 3: 200 training data and 1430 test data with 4 input features.
- Data set 4: 810 training data and 820 test data with 6 input features.

Data sets 3 and 4 were used to study the effect of tuning when classification was very difficult. The expansion parameter α was set to 0.001 for all the cases except for data set 3. For data set 3, the best performance of the classifier was obtained when α was set to 0.6 without tuning [3]. We compared the performance of the fuzzy classifier with the maximum performance of a three-layered neural network composed of six hidden units; the number of hidden units was determined using the statistical method discussed in [8]. To obtain the maximum performance of the neural network, we trained the network 100 times with initial connection weights randomly assigned between -0.1 and 0.1. For each of the four data sets, we considered the following three cases.

- Case 1: Use all the training data as the extraction data, and do no tuning.
- Case 2: Divide the training data into the same number of extraction data and tuning data.
- Case 3: Exchange the extraction data and the tuning data in case 2.

Table 2 shows the results using the least squares method. After tuning, the performance of fuzzy classifiers was comparable to the maximum performance of neural networks. The largest performance improvement was obtained for data set 1, while for data set 2 with $\alpha = 0.6$ there was no

Table 2. Performance Comparison of Neural Networks and Fuzzy Classifiers for Numeral Recognition[a]

				Fuzzy classifier				
	Neural network			Case 1	Case 2		Case 3	
Data set	Max	Min	Avg	Non	Non	Tuned	Non	Tuned
1	98.25	95.17	96.54	97.06	97.34	98.32	97.06	97.83
2	99.76	98.90	99.41	99.63	99.63	99.63	99.51	99.51
3	77.76	71.33	74.50	72.10	71.61	72.03	72.66	72.73
				75.52[b]	77.06[b]	75.80[b]	75.87[b]	75.45[b]
4	98.66	96.83	97.78	98.05	98.41	98.54	98.04	98.41

[a] In percent; $\alpha = 0.001$ unless otherwise noted. In case 1 all the training data were used for fuzzy rule extraction. In cases 2 and 3, the training data set was divided into two, one for rule extraction and one for sensitivity parameter tuning. The least squares method was used for tuning. In cases 2 and 3 the data for fuzzy rule extraction and the data for fuzzy rule tuning were exchanged.

[b] $\alpha = 0.6$.

performance improvement. Since the least squares method determines the sensitivity parameters by using all the tuning data, the absence of improvement meant that the characteristics of the training data set and test data set were very different.

Table 3 shows the number of rules generated for the cases shown in Table 2. Since the extraction data for cases 2 and 3 are subsets of the extraction data for case 1, case 1 gives the maximum numbers of rules. For data set 1 the number of rules is 10 for all the cases, i.e., one rule per class.

The rule extraction and tuning time for one iteration was less than one second using a 16 MIPS computer for all the cases. Since the final fuzzy system was obtained by one or two iterations of rule extraction and tuning, the training time was very fast. The average training time of the neural network, using the 31 MIPS computer, was 11.7 seconds for data set 1, 2.63 minutes for data set 2, 4.49 minutes for data set 3, and 13.90 minutes for data set 4.

To see the effect of the iterative tuning method, we used the training data as the extraction data and the test data as the tuning data. Table 4 shows the results. We saw that the iterative tuning method was more effective than the least squares method because we could improve the performance step by step. The difference in performance was attributed mainly to dual nodes. Meanwhile, the fuzzy classifier tuned by using the iterative method exceeded the maximum performance of the neural network. For data set 3, when $\alpha = 0.001$ was used, the performance of the fuzzy classifier was inferior to the maximum performance of the neural network; while when $\alpha = 0.6$ was used, the fuzzy classifier outperformed the neural network. Since the misclassification graphs for $\alpha = 0.001$ and $\alpha = 0.6$ were similar, we could obtain a similar recognition rate (i.e., 78.04%) for $\alpha = 0.6$ with the sensitivity parameters tuned for $\alpha = 0.001$.

Table 3. Numbers of Rules Generated for Numeral Recognition.[a]

	Case 1	Case 2		Case 3	
Data set	Non	Non	Tuned	Non	Tuned
1	10	10	10	10	10
2	11	10	11	11	11
3	17	15	16	13	15
	30[b]	22[b]	25[b]	18[b]	28[b]
4	17	14	17	15	16

[a] Cases correspond to those in Table 2. $\alpha = 0.001$ unless otherwise noted.
[b] $\alpha = 0.6$.

Table 4. Improved Performance of the Fuzzy Classifier for Numeral Recognition after Tuning[a]

	Neural network			Fuzzy classifier		
Data set	Max	Min	Avg	Non	LO	Iterative
1	98.25	95.17	96.54	97.06	98.46	98.74
2	99.76	98.90	99.41	99.63	100	100
3	77.76	71.33	74.50	72.10	72.73	75.80
				75.52[b]	75.73[b]	78.18[b]
4	98.66	96.83	97.78	98.05	98.29	98.66

[a] In percent; $\alpha = 0.001$ unless otherwise noted. All the training data were used for fuzzy rule extraction, and test data were used for tuning.
[b] $\alpha = 0.6$.

For data set 1, after the fuzzy classifier was tuned by using the iterative method, the misclassification graph, shown in Figure 6(a), was reduced to the graph shown in Figure 6(b). The number of misclassifications of class 8 data to class 4 increased from 10 to 11; the new one was one of the class 8 data whose degrees of membership with respect to classes 8 and 4, before tuning, were the same, so that it could not be classified as either class at all. Thus none of the originally correctly classified data were misclassified when the tuned sensitivity parameters were used.

For data set 2, a 100% recognition rate was achieved by both methods. This was because there were no dual nodes. If two classes appear as dual nodes in a misclassification graph, the misclassified data belonging to both classes cannot be correctly classified by varying the sensitivity parameters.

VI. CONCLUSIONS

In this paper, we have developed two methods, a least squares method and an iterative method, for tuning the sensitivity parameters of a fuzzy classifier whose fuzzy rules were extracted directly from numerical data. The effectiveness of the tuning approaches was demonstrated using the Fisher iris data, which are commonly used for evaluating the performance of a classifier, as well as by using data on numeral recognition on vehicle license plates. The test results showed that both tuning methods improved the recognition rate of the fuzzy classifier, which was comparable to or better than the maximum performance obtained by neural networks but with less computational time. The results also showed that the fuzzy classifier tuned using the iterative method had slightly better performance

than that tuned using the least squares method. Therefore, we concluded that the generalization ability of the fuzzy classifier could be improved by tuning the sensitivity parameters of membership functions, and iterative tuning has the advantage of being able to be done on the fly.

References

1. Simpson, P. K., Fuzzy min-max neural networks—part 1: Classification, *IEEE Trans. Neural Networks* 3(5), 776–786, Sept. 1992.

2. Wang, L.-X., and Mendel, J. M., Generating fuzzy rules by learning from examples, *IEEE Trans. Systems Man Cybernet.* 22(6), 1414–1427, Nov./Dec. 1992.

3. Abe, S., and Lan, M.-S., A method for fuzzy rules extraction directly from numerical data and its application to pattern classification, *IEEE Trans. Fuzzy Systems* 3(1):18–28, Feb. 1995.

4. Abe, S., and Lan, M.-S., Fuzzy rules extraction directly from numerical data for function approximation, *IEEE Trans. Systems Man Cybernet.* 25(1), 119–129, Jan. 1995.

5. Fisher, R., The use of multiple measurements in taxonomic problems, *Ann. Eugenics* 7(II), 179–188, 1936.

6. Takatoo, M., et al., Gray scale image processing technology applied to vehicle license number recognition system, *Proceedings of the International Workshop on Industrial Applications of Machine Vision and Machine Intelligence*, 76–79, Feb. 1987.

7. Takenaga, H., et al., Input layer optimization of neural networks by sensitivity analysis and its application to recognition of numerals (in Japanese), *Trans. IEE Japan* 111-D(1), 36–44, 1991; English transl., *Elec. Engrg. Japan* 111(4), 130–138, 1991.

8. Kayama, M., et al., Constructing optimal neural networks by linear regression analysis, *Proceedings of Neuro-Nimes '90*, 363–376, 1990.