Performance Evaluation of Fuzzy Classifier Systems for Multidimensional Pattern Classification Problems

Hisao Ishibuchi, *Member, IEEE*, Tomoharu Nakashima, *Student Member, IEEE*, and Tadahiko Murata, *Member, IEEE*

***Abstract*— We examine the performance of a fuzzy genetics-based machine learning method for multidimensional pattern classification problems with continuous attributes. In our method, each fuzzy if–then rule is handled as an individual, and a fitness value is assigned to each rule. Thus, our method can be viewed as a classifier system. In this paper, we first describe fuzzy if–then rules and fuzzy reasoning for pattern classification problems. Then we explain a genetics-based machine learning method that automatically generates fuzzy if–then rules for pattern classification problems from numerical data. Because our method uses linguistic values with fixed membership functions as antecedent fuzzy sets, a linguistic interpretation of each fuzzy if–then rule is easily obtained. The fixed membership functions also lead to a simple implementation of our method as a computer program. The simplicity of implementation and the linguistic interpretation of the generated fuzzy if–then rules are the main characteristic features of our method. The performance of our method is evaluated by computer simulations on some well-known test problems. While our method involves no tuning mechanism of membership functions, it works very well in comparison with other classification methods such as nonfuzzy machine learning techniques and neural networks.**

I. Introduction

FUZZY rule-based systems have been successfully applied to various control problems [1], [2]. Fuzzy rules in these systems are usually derived from human experts as linguistic if–then rules. Recently several approaches have been proposed for automatically generating fuzzy if–then rules from numerical data without domain experts (see, for example, [3]–[6]). Genetic algorithms [7], [8] have been widely used for generating fuzzy if–then rules and tuning membership functions (see a survey by Carse *et al.* [9]). For example, genetic algorithms were used for generating fuzzy if–then rules in [10], [11], for tuning membership functions in [12]–[16], and for both the rule generation and the membership function tuning in [9], [17]–[21]. Hierarchical structures of fuzzy rule-based systems were also determined by genetic algorithms in [22], [23].

Genetics-based machine learning methods for rule generation fall into two categories: the Michigan approach and the Pittsburgh approach. In the Michigan approach, each rule is handled as an individual, called a classifier. Thus, this approach is referred to as a classifier system [24]. On the other hand, the Pittsburgh approach [25] handles an entire rule set as an individual. All the above-mentioned methods [9]–[23] for generating fuzzy if–then rules and tuning membership functions are categorized as Pittsburgh approaches, in which a set of fuzzy if–then rules was treated as an individual. The Michigan approach was also used for generating fuzzy if–then rules in [26]–[29], where each fuzzy if–then rule was treated as an individual (i.e., as a classifier). Thus, the rule generation methods in [26]–[29] were referred to as fuzzy classifier systems.

While various methods have been proposed for generating fuzzy if–then rules and tuning membership functions, only a few methods are applicable to pattern classification problems. This is because the above-mentioned methods [9]–[23], [26]–[29] lie mainly in the domain of control problems and function approximation problems. For pattern classification problems, Abe *et al.* [30], [31] proposed a rule generation method and a rule tuning method in which each fuzzy if–then rule was represented by a hyperbox in multidimensional pattern spaces. Such a hyperbox was also used as a fuzzy if–then rule in fuzzy min-max neural networks [32]. Neural networks were used as adaptive fuzzy classification systems in [33]–[38]. Neural-network-based fuzzy systems usually have high learning ability, but sometimes lose the comprehensibility of fuzzy if–then rules (i.e., lose the linguistic interpretation of each rule).

If the comprehensibility of fuzzy if–then rules by human users is a criterion in designing a fuzzy rule-based system, a fuzzy partition by a simple fuzzy grid with pre-specified membership functions is preferable. An example of such a fuzzy partition is shown in Fig. 1, where each axis of a two-dimensional (2-D) pattern space is homogeneously partitioned by five linguistic values (S: *small*, MS: *medium small*, M: *medium*, ML: *medium large*, and L: *large*). Ishibuchi *et al.* [39] proposed a heuristic method for generating fuzzy if–then rules for pattern classification problems using such a fuzzy partition. Rule selection methods based on genetic algorithms were proposed in [40], [41], wherein pre-specified membership functions were also used as in [39]. Of course, we do not have to partition each axis homogeneously when a priori knowledge on linguistic values is available from domain experts for specifying membership functions.

It has been often claimed that grid-type fuzzy partitions such as Fig. 1 cannot handle high-dimensional problems with

Manuscript received April 5, 1998; revised January 30, 1999. This paper was recommended by Associate Editor A. Kandel.

H. Ishibuchi and T. Nakashima are with the Department of Industrial Engineering, Osaka Prefecture University, Osaka 599-8531, Japan (e-mail: hisaoi@ie.osakafu-u.ac.jp).

T. Murata is with the Department of Industrial and Information Systems Engineering, Ashikaga Institute of Technology, Tochigi 326-8558, Japan.

Publisher Item Identifier S 1083-4419(99)05272-3.

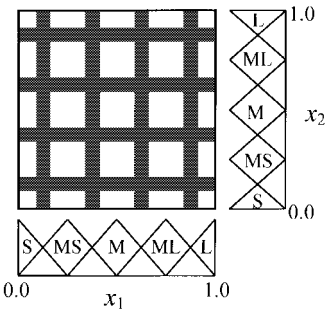

Fig. 1. Example of fuzzy partition by a simple fuzzy grid with five linguistic values for each axis of the 2-D pattern space $[0, 1] \times [0, 1]$.

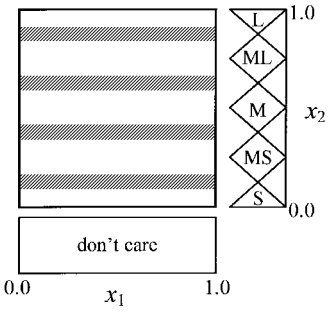

Fig. 2. Example of fuzzy partition of the 2-D pattern space $[0, 1] \times [0, 1]$ with "*don't care*" as an antecedent fuzzy set.

many input variables due to the curse of dimensionality (see, for example, [9]). That is, when we use the grid-type fuzzy partition, the number of fuzzy if–then rules exponentially increases as the number of input variables increases. In this paper, however, we use the grid-type fuzzy partition for pattern classification problems with many continuous attributes (e.g., 13 attributes) because such a fuzzy partition maintains an inherent advantage of fuzzy rule-based systems: the comprehensibility of fuzzy if–then rules. We tackle the curse of dimensionality by utilizing "*don't care*" as an antecedent fuzzy set and generating only a small number of promising fuzzy if–then rules by genetic operations. The antecedent fuzzy set "*don't care*" is represented by an interval-type membership function whose membership value is always unity in the domain of each attribute value [41]. For example, if the domain of the ith attribute (i.e., x_i) is the unit interval $[0, 1]$, the membership function of the antecedent fuzzy set "*don't care*" can be written as

$$\mu_{\text{don't care}}(x_i) = \begin{cases} 1, & \text{if } 0 \le x_i \le 1, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

In Fig. 2, we show an example fuzzy partition that incorporates with the antecedent fuzzy set "*don't care*." A 2-D pattern space is divided into five fuzzy subspaces by a fuzzy grid with "*don't care*" for the first axis (i.e., x_1) and the five linguistic values for the second axis (i.e., x_2). From the comparison of Fig. 1 with Fig. 2, we can see that the introduction of "*don't care*" as an antecedent fuzzy set reduces the number of fuzzy if–then rules. In fact, it also reduces the number of attributes (i.e., the number of antecedent conditions) in fuzzy if–then rules.

The rule selection methods in [40] and [41] consisted of two stages: rule generation using fuzzy grids, and rule selection using genetic algorithms. Because their methods generated all the candidate fuzzy if–then rules in the first stage, we cannot directly apply their methods to multidimensional pattern classification problems with many attributes. In this paper, we propose a fuzzy classifier system to handle such classification problems. The main aim of this paper is to clearly demonstrate that our fuzzy classifier system, based on prespecified linguistic values with fixed membership functions, works well for well-known test problems involving many attributes (e.g., wine data with thirteen attributes and credit approval data with fourteen attributes). Through computer simulations on such test problems, we show that our fuzzy classifier system is comparable to other pattern classification methods such as a genetics-based machine learning method with nonfuzzy if–then rules [42], back-propagation neural networks [43] and the C4.5 algorithm [44]. This means that our fuzzy classifier system can construct high performance fuzzy rule-based systems that can be easily understood by human users through linguistic interpretation.

A fuzzy rule-based system is constructed from a set of labeled samples (i.e., labeled feature vectors). The constructed fuzzy rule-based system assigns a new feature vector to one of given classes. That is, fuzzy rule-based systems are applicable to the same data set as nonfuzzy classification methods (e.g., neural networks, decision trees, and statistical techniques). This means that fuzzy rule-based systems can be viewed as an alternative to other classification methods. One advantage of fuzzy rule-based classification systems is their comprehensibility. We can easily understand them because each fuzzy if–then rule is interpreted through linguistic values such as "*small*" and "*large*." High classification ability is another advantage of fuzzy rule-based systems. These advantages are demonstrated in this paper.

This paper is organized as follows. Section II briefly explains a fuzzy rule generation method with fuzzy grids and a fuzzy reasoning method for pattern classification problems. Section III explains each step of our fuzzy classifier system: generation of an initial population, evaluation of each fuzzy if–then rule, generation of new fuzzy if–then rules by genetic operations, replacement of a part of the current population with the newly generated fuzzy if–then rules, and termination of the algorithm. Section IV compares our fuzzy classifier system with other classification methods by computer simulations on well-known test problems. In Section V, we discuss why our fuzzy classifier system works well while it involves no tuning mechanism of membership functions. In Section VI, we suggest some directions to extend our fuzzy classifier system. Section VII concludes this paper.

II. Rule Generation and Fuzzy Reasoning

A. Pattern Classification Problem

Let us assume that our pattern classification problem is a c-class problem in the n-dimensional pattern space $[0, 1]^n$ with continuous attributes. We also assume that m real vectors $\mathbf{x}_p = (x_{p1}, x_{p2}, \ldots, x_{pn})$, $p = 1, 2, \ldots, m$, are given as training patterns from the c classes $(c \ll m)$. Because the

pattern space is $[0,1]^n$, attribute values of each pattern are $x_{pi} \in [0,1]$ for $p = 1,2,\ldots,m$ and $i = 1,2,\ldots,n$. An example of such a pattern classification problem is shown in Fig. 3 where $c = 2$ (i.e., two-class problem), $n = 2$ (i.e., 2-D pattern space) and $m = 40$ (i.e., 40 training patterns).

B. Rule Generation

In our fuzzy classifier system, we use fuzzy if–then rules of the following type for the c-class pattern classification problem with the n-dimensional pattern space $[0,1]^n$:

$$\text{Rule } R_j\text{: If } x_1 \text{ is } A_{j1} \text{ and } \ldots \text{ and } x_n \text{ is } A_{jn} \text{ then Class } C_j \text{ with } CF = CF_j \tag{2}$$

where

R_j	Label of the jth fuzzy if–then rule;
$A_{j1},\ldots,A_{jn}$	Antecedent fuzzy sets on the unit interval $[0,1]$;
C_j	Consequent class (i.e., one of the given c classes);
CF_j	Grade of certainty of the fuzzy if–then rule R_j.

It should be noted that the grade of certainty CF_j is different from the fitness value of each rule. The fitness value is used in a selection operation of our fuzzy classifier system while CF_j is used in fuzzy reasoning for classifying new patterns.

As antecedent fuzzy sets, we use the five linguistic values in Fig. 1 and "*don't care*" in Fig. 2. Thus, for the n-dimensional pattern classification problem, the total number of fuzzy if–then rules is $(5+1)^n$. It is impossible to use all the $(5+1)^n$ fuzzy if–then rules in a fuzzy rule-based system when the number of attributes (i.e., n) is large. Our fuzzy classifier system searches for a set of a relatively small number of fuzzy if–then rules (e.g., 60 rules).

In our fuzzy classifier system, the consequent class C_j and the grade of certainty CF_j of each fuzzy if–then rule are determined by the following simple heuristic procedure [39]–[41] when its antecedent fuzzy sets $A_{j1},\ldots,A_{jn}$ are specified by genetic operations.

[**Determination of C_j and CF_j**]

Step 1: Calculate the compatibility grade of each training pattern $\mathbf{x}_p = (x_{p1}, x_{p2}, \ldots, x_{pn})$ with the fuzzy if–then rule R_j by the following product operation:

$$\mu_j(\mathbf{x}_p) = \mu_{j1}(x_{p1}) \times \cdots \times \mu_{jn}(x_{pn}) \tag{3}$$

where $\mu_{ji}(x_{pi})$ is the membership function of A_{ji}.

Step 2: For each class, calculate the sum of the compatibility grades of the training patterns with the fuzzy if–then rule R_j

$$\beta_{\text{Class } h}(R_j) = \sum_{\mathbf{x}_p \in \text{Class } h} \mu_j(\mathbf{x}_p), \quad h = 1,2,\ldots,c \tag{4}$$

where $\beta_{\text{Class } h}(R_j)$ is the sum of the compatibility grades of the training patterns in Class h with the fuzzy if–then rule R_j.

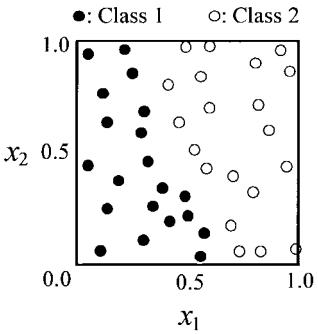

Fig. 3. Two-class classification problem with the 2-D pattern space $[0,1] \times [0,1]$.

Step 3: Find Class $\hat{h}_j$ that has the maximum value of $\beta_{\text{Class } h}(R_j)$

$$\beta_{\text{Class } \hat{h}_j}(R_j) = \text{Max}\{\beta_{\text{Class } 1}(R_j), \ldots, \beta_{\text{Class } c}(R_j)\}. \tag{5}$$

If two or more classes take the maximum value, the consequent class C_j of the fuzzy if–then rule R_j can not be determined uniquely. In this case, let C_j be ϕ. If a single class (i.e., Class $\hat{h}_j$) takes the maximum value, let C_j be Class $\hat{h}_j$. If there is no training pattern compatible with the fuzzy if–then rule R_j (i.e., if $\beta_{\text{Class } h}(R_j) = 0$ for $h = 1,2,\ldots,c$), the consequent class C_j is also specified as ϕ.

Step 4: If the consequent class C_j is ϕ, let the grade of certainty CF_j of the fuzzy if–then rule R_j be $CF_j = 0$. Otherwise the grade of certainty CF_j is determined as follows:

$$CF_j = \{\beta_{\text{Class } \hat{h}}(R_j) - \bar{\beta}\} \Big/ \sum_{h=1}^{c} \beta_{\text{Class } h}(R_j) \tag{6}$$

where

$$\bar{\beta} = \sum_{\substack{h=1 \\ h \neq \hat{h}_j}}^{c} \beta_{\text{Class } h}(R_j)/(c-1). \tag{7}$$

While the determination of the grade of certainty $_{CF}j$ by (6)–(7) seems to be a bit complicated at a glance, we can see that this procedure is intuitively acceptable if we consider two-class classification problems (i.e., $c = 2$). For example, when $\beta_{\text{Class } 1}(R_j) > \beta_{\text{Class } 2}(R_j)$, C_j is Class 1 and CF_j is specified as

$$CF_j = \frac{\beta_{\text{Class } 1}(R_j) - \beta_{\text{Class } 2}(R_j)}{\beta_{\text{Class } 1}(R_j) + \beta_{\text{Class } 2}(R_j)}. \tag{8}$$

In the case of $\beta_{\text{Class } 1}(R_j) = \beta_{\text{Class } 2}(R_j)$, C_j is ϕ and $CF_j = 0$ because we can not specify the consequent class C_j.

Using the pattern classification problem in Fig. 3, we illustrate this rule generation procedure. If we use the fuzzy partition by the 5×5 fuzzy grid in Fig. 1, we can generate 25 fuzzy if–then rules as shown in the rule table in Fig. 4 where C1 and C2 are consequent classes (i.e., Class 1 and Class 2,

x_2 \ x_1	S	MS	M	ML	L
L	C 1 (1.00)	C 1 (0.86)	C 2 (0.99)	C 2 (1.00)	C 2 (1.00)
ML	C 1 (1.00)	C 1 (0.74)	C 2 (0.81)	C 2 (1.00)	C 2 (1.00)
M	C 1 (1.00)	C 1 (0.94)	C 2 (0.38)	C 2 (1.00)	C 2 (1.00)
MS	C 1 (1.00)	C 1 (1.00)	C 1 (0.79)	C 2 (0.80)	C 2 (1.00)
S	C 1 (1.00)	C 1 (1.00)	C 1 (0.85)	C 2 (0.63)	C 2 (1.00)

Fig. 4. Set of generated fuzzy if–then rules with the 5 × 5 fuzzy grid in Fig. 1.

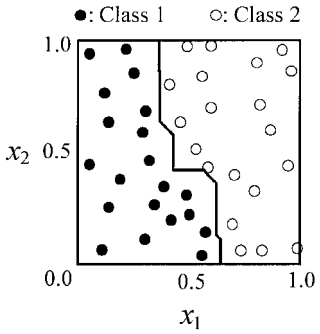

Fig. 5. Classification boundary obtained by the 25 fuzzy if–then rules in Fig. 4.

respectively), and each real number in parentheses denotes the grade of certainty of the corresponding fuzzy if–then rule.

Our fuzzy if–then rules with certainty grades are different from standard ones that are usually used in control problems and function approximation problems. The following traditional type of fuzzy if–then rules is also applicable to pattern classification problems:

$$\text{Rule } R_j\text{: If } x_1 \text{ is } A_{j1} \text{ and } \ldots \text{ and } x_n \text{ is } A_{jn} \text{ then } y_1 \text{ is } B_{j1} \text{ and } \ldots \text{ and } y_c \text{ is } B_{jc} \tag{9}$$

where y_k is the possibility grade of the occurrence of Class k, and B_{jk} is a consequent fuzzy set. Instead of the consequent fuzzy set B_{jk}, we can also use a singleton fuzzy set (i.e., a real number b_{jk}) or a linear function of input values (i.e., $b_{jk}^0 + b_{jk}^1 \cdot x_1 + \cdots + b_{jk}^n \cdot x_n$). Antecedent fuzzy sets and consequent fuzzy sets (or real numbers, linear functions) can be adjusted based on training patterns in the same manner as in neural networks. In this paper, we use fuzzy if–then rules in (1) because they are simpler than those in (9). The simplicity of fuzzy if–then rules is necessary for constructing comprehensible fuzzy rule-based systems. The use of fuzzy if–then rules in (1) also makes the credit assignment in our fuzzy classifier system straightforward. As we will show in this paper, our fuzzy if–then rules with certainty grades in (1) can generate nonlinear complicated decision boundaries even if we use very simple antecedent fuzzy sets.

C. Fuzzy Reasoning

When the antecedent fuzzy sets of each fuzzy if–then rule are given, we can determine the consequent class and the grade of certainty by the heuristic rule generation procedure in the previous section. Here we assume that we have already generated a set of fuzzy if–then rules for a pattern classification problem. Fig. 4 is an example of such a rule set. Let us denote the set of fuzzy if–then rules we generate by S.

An input pattern $\mathbf{x}_p$ to the fuzzy rule-based system with the rule set S is classified by a fuzzy reasoning method. In our fuzzy classifier system, we perform the fuzzy reasoning via the single winner rule. The winner rule $R_{\hat{j}}$ for the input pattern $\mathbf{x}_p$ is determined as

$$\mu_{\hat{j}}(\mathbf{x}_p) \cdot CF_{\hat{j}} = \text{Max}\{\mu_j(\mathbf{x}_p) \cdot CF_j \mid R_j \in S\}. \tag{10}$$

That is, the winner rule has the maximum product of the compatibility $\mu_j(\mathbf{x}_p)$ and the grade of certainty CF_j. If more than one fuzzy if–then rule have the same maximum product but different consequent classes for the input pattern $\mathbf{x}_p$, the classification of that pattern is rejected. The classification is also rejected if no fuzzy if–then rule is compatible with the input pattern $\mathbf{x}_p$ (i.e., $\mu_j(\mathbf{x}_p) = 0$ for $^{\forall}R_j \in S$). This fuzzy reasoning method based on a single winner rule leads to simplicity in the credit assignment algorithm in our fuzzy classifier system, as only one rule is responsible for the classification result of each input pattern. It is possible to modify our fuzzy reasoning method to account for the situation when different classes have the same maximum value in (10), and for when no fuzzy if–then rule is compatible with the input pattern $\mathbf{x}_p$. After this modification, classification rates of fuzzy rule-based systems are improved because rejection rates are decreased. At the same time, there is an increase in error rates. Thus, such a modification is promising when the penalty of rejection is not small.

In Fig. 5, we show classification results by the fuzzy reasoning method based on the single winner rule in (10). The classification boundary in Fig. 5 was generated by the fuzzy rule-based system with the 25 fuzzy if–then rules in Fig. 4.

III. Fuzzy Classifier System

In this section, we propose a fuzzy classifier system for automatically designing a fuzzy rule-based system for multidimensional pattern classification problems with many continuous attributes. Our fuzzy classifier system is very simple enough to be easily implemented as a computer program. Nevertheless it has high classification performance, as shown in the next section. Extensions of our fuzzy classifier system will be discussed in Section VI.

A. Outline of Fuzzy Classifier System

Our fuzzy classifier system does not involve any complicated tuning mechanism for membership functions. It is based on the heuristic rule generation procedure in Section II, basic genetic operations, and a simple credit assignment scheme. Genetic operations such as selection, crossover and mutation are used for generating a combination of antecedent fuzzy sets of each fuzzy if–then rule. A rule's consequent class and certainty grade are determined by the heuristic procedure in Section II. The outline of our fuzzy classifier system is as follows.

Step 1: Generate an initial population of fuzzy if–then rules;

Step 2: Evaluate each fuzzy if–then rule in the current population;

Step 3: Generate new fuzzy if–then rules by genetic operations;

Step 4: Replace a part of the current population with the newly generated rules;

Step 5: Terminate the algorithm if a stopping condition is satisfied, otherwise return to Step 2.

Each step of our fuzzy classifier system will be explained in detail, along with a coding method of fuzzy if–then rules, in the following sections.

B. Coding of Fuzzy If–Then Rule

Because the consequent class and the grade of certainty of a fuzzy if–then rule are easily determined by the heuristic procedure in Section II, only antecedent fuzzy sets are altered by genetic operations in our fuzzy classifier system. We denote the five linguistic values in Fig. 1 and "*don't care*" in Fig. 2 by the following six symbols (i.e., 1, 2, 3, 4, 5, and #) in our fuzzy classifier system:

$$\begin{aligned}
&\mathbf{S}\text{: } \textit{small} && \rightarrow 1,\\
&\mathbf{MS}\text{: } \textit{medium small} && \rightarrow 2,\\
&\mathbf{M}\text{: } \textit{medium} && \rightarrow 3,\\
&\mathbf{ML}\text{: } \textit{medium large} && \rightarrow 4,\\
&\mathbf{L}\text{: } \textit{large} && \rightarrow 5,\\
&\mathbf{DC}\text{: } \textit{don't care} && \rightarrow \#.
\end{aligned}$$

Each fuzzy if–then rule can be denoted by a string of these six symbols. For example, a string "1#3#" denotes the following fuzzy if–then rule for a four-dimensional pattern classification problem:

If x_1 is *small* and x_2 is *don't care* and x_3 is *medium* and x_4 is *don't care* then Class C_j with $CF = CF_j$.

Because the conditions with "*don't care*" can be omitted, this rule is rewritten as follows:

If x_1 is *small* and x_3 is *medium* then Class C_j with $CF = CF_j$.

C. Initial Population

Let us denote the number of fuzzy if–then rules in each population in our fuzzy classifier system by N_{pop} (i.e., N_{pop} is the population size). To construct an initial population, N_{pop} fuzzy if–then rules are generated by randomly selecting their antecedent fuzzy sets from the six symbols corresponding to the five linguistic values and "*don't care*". Each symbol is randomly selected with the probability of 1/6. The consequent class C_j and the grade of certainty CF_j of each fuzzy if–then rule are determined by the heuristic procedure in Section II.

D. Evaluation of Each Rule

Let us denote the set of N_{pop} fuzzy if–then rules in the current population by S. In order to evaluate each fuzzy if–then rule in S, we classify all the given training patterns by the fuzzy rule-based system with the rule set S using the fuzzy reasoning method in Section II-C. As we have already explained in (10), each pattern is classified by a single winner rule. Thus the credit assignment is very simple. In our fuzzy classifier system, we assign a unit reward to the winner rule when a training pattern is correctly classified by that rule. After all the training patterns are examined, the fitness value of each fuzzy if–then rule is defined as follows by the total reward assigned to that rule

$$\textit{fitness}(R_j) = \textit{NCP}(R_j) \tag{11}$$

where $\textit{fitness}(R_j)$ is the fitness value of the fuzzy if–then rule R_j, and $\textit{NCP}(R_j)$ is the number of training patterns that are correctly classified by R_j. The fitness value of each fuzzy if–then rule is updated by (11) at each generation in our fuzzy classifier system. While this credit assignment scheme seems to be too simple to handle real-world classification problems, it works very well for well-known real-world test problems as will be shown in Section IV. An alternative credit assignment scheme with a misclassification penalty will be discussed in Section VI.

When more than one fuzzy if–then rule with the same consequent class in the rule set S have the same maximum value in (10), we assume that the fuzzy if–then rule with the smallest index j is the winner. For example, suppose that R_3 and R_9 in the rule set S are fuzzy if–then rules with the same antecedent and the same consequent (i.e., R_3 and R_9 are the same fuzzy if–then rule). In this case, R_3 and R_9 always have the same maximum value in (10), and R_3 is the winner. This means that the reward is assigned only to R_3. In this manner, only a single rule with the smallest index j among duplicated fuzzy if–then rules can survive the generation update in our fuzzy classifier system. Thus our fuzzy classifier system implicitly prevents a population from being dominated by homogeneous fuzzy if–then rules.

E. Genetic Operations for Generating New Rules

In order to generate new fuzzy if–then rules, first a pair of fuzzy if–then rules is selected from the current population. Each fuzzy if–then rule in the current population (i.e., in the rule set S) is selected by the following selection probability based on the roulette wheel selection with the linear scaling:

$$P(R_j) = \frac{\textit{fitness}(R_j) - \textit{fitness}_{\min}(S)}{\sum_{R_i \in S}\{\textit{fitness}(R_i) - \textit{fitness}_{\min}(S)\}} \tag{12}$$

where $\textit{fitness}_{\min}(S)$ is the minimum fitness value of the fuzzy if–then rules in the current population S.

From the selected pair of fuzzy if–then rules, two rules are generated by the uniform crossover for the antecedent fuzzy sets. The uniform crossover is illustrated in Fig. 6 for a five-dimensional pattern classification problem. It should be noted that only the antecedent fuzzy sets of the selected pair of fuzzy if–then rules are mated.

Each antecedent fuzzy set of the fuzzy if–then rules generated by the crossover operation is randomly replaced with a different antecedent fuzzy set using a pre-specified mutation

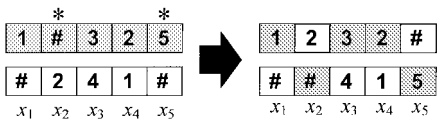

Fig. 6. Uniform crossover for antecedent fuzzy sets ($*$ denotes a crossover position).

Fig. 7. Mutation for antecedent fuzzy sets ($*$ denotes a mutation position).

probability. This mutation operation is illustrated in Fig. 7. As in the crossover operation, the mutation operation is applied to only the antecedent fuzzy sets. The consequent class and the certainty grade of each of the newly generated fuzzy if–then rules are determined after the mutation by the heuristic procedure in Section II.

These genetic operations (i.e., selection, crossover, and mutation) are iterated until a pre-specified number of fuzzy if–then rules are newly generated.

F. Rule Replacement

A prespecified number of fuzzy if–then rules (say, $N_{\rm rep}$) in the current population are replaced with the newly generated rules by the genetic operations. In our fuzzy classifier system, the worst $N_{\rm rep}$ rules with the smallest fitness values are removed from the current population and the newly generated fuzzy if–then rules are added.

In order to generate $N_{\rm rep}$ fuzzy if–then rules, the genetic operations in the previous section are iterated $N_{\rm rep}/2$ times. That is, $N_{\rm rep}/2$ pairs of fuzzy if–then rules are selected from the current population in the selection operation, and two fuzzy if–then rules are generated from each pair by the crossover and mutation operations.

G. Termination Test

We can use any stopping conditions for terminating the execution of our fuzzy classifier system. In computer simulations of this paper, we used the total number of generations as a stopping condition. The final solution obtained by our fuzzy classifier system is the rule set with the maximum classification rate for training patterns over all generations. That is, the final solution is not the final population but the best population. Since the evolution is not based on the performance of an entire rule set but each rule in the Michigan approach including our fuzzy classifier system, the final population is not always the best rule set. In the Michigan approach, a single rule is treated as an individual, and each population corresponds to a rule set. On the contrary, the final population always includes the best rule set in the Pittsburgh approach with an elite strategy, in which an entire rule set is treated as an individual.

H. Algorithm

Our fuzzy classifier system can be written as the following algorithm.

Step 1: Generate an initial population of $N_{\rm pop}$ fuzzy if–then rules by randomly specifying antecedent fuzzy sets of each rule. The consequent class and the grade of certainty are determined by the heuristic procedure in Section II.

Step 2: Classify all the given training patterns by the fuzzy if–then rules in the current population, then calculate the fitness value of each rule by (11).

Step 3: Generate $N_{\rm rep}$ fuzzy if–then rules from the current population by the selection, crossover and mutation operations. The consequent class and the grade of certainty of each fuzzy if–then rule are determined by the heuristic procedure in Section II.

Step 4: Replace the worst $N_{\rm rep}$ fuzzy if–then rules with the smallest fitness values in the current population with the newly generated rules.

Step 5: Terminate the iteration of the algorithm if the pre-specified stopping condition is satisfied, otherwise return to Step 2.

I. Numerical Example

We applied our fuzzy classifier system to the pattern classification problem in Fig. 3. Because this pattern classification problem has two attributes (i.e., x_1 and x_2), we used fuzzy if–then rules of the following type:

$$R_j\text{: If } x_1 \text{ is } A_{j1} \text{ and } x_2 \text{ is } A_{j2} \text{ then Class } C_j \text{ with } CF_j. \tag{13}$$

As the antecedent fuzzy sets A_{j1} and A_{j2}, we used the five linguistic values in Fig. 1 and the "*don't care*" in Fig. 2. Thus the total number of fuzzy if–then rules is $6^2 = 36$. Our problem is to find a compact rule set with high classification performance from these 36 fuzzy if–then rules. It should be noted that the total number of rule sets is very large (i.e., $2^{36} \cong 6.9 \times 10^{10}$ including an empty rule set) even for this small-scale numerical example with only two attributes.

In computer simulations, we applied our fuzzy classifier system to the pattern classification problem in Fig. 3 with various specifications of the population size $N_{\rm pop}$. Parameter specifications used in the computer simulations are as follows:

Population size: $N_{\rm pop} = 2, 3, \ldots, 10$
Crossover probability: 1.0
Mutation probability: 0.1
Number of replaced rules in each population: $N_{\rm rep} = 1$
Stopping condition: 1000 generations.

From these parameter specifications, we can see that only one fuzzy if–then rule was replaced at each generation (i.e., $N_{\rm rep} = 1$) and the algorithm was terminated after 1000 generations.

Simulation results are summarized in Table I. From Table I, we can see that all the training patterns in Fig. 3 can be correctly classified by six fuzzy if–then rules. The following rules were generated by our fuzzy classifier system. Note that antecedent conditions with "*don't care*" are omitted in the

TABLE I
SIMULATION RESULTS FOR A NUMERICAL EXAMPLE

Population size	Classification rate
2	85.0%
3	92.5%
4	97.5%
5	97.5%
6	100%
7	100%
8	100%
9	100%
10	100%

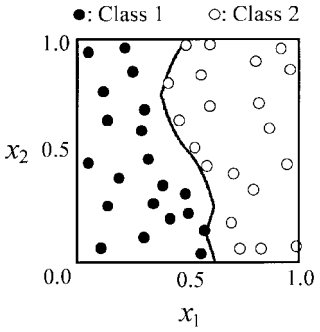

Fig. 8. Classification boundary obtained by the selected fuzzy if–then rules.

following.

If x_1 is *medium small* then Class 1 with $CF = 0.90$
If x_1 is *medium* and x_2 is *small*
then Class 1 with $CF = 0.85$
If x_1 is *medium* and x_2 is *medium small*
then Class 1 with $CF = 0.79$
If x_1 is *large* then Class 2 with $CF = 1.0$
If x_1 is *medium large* then Class 2 with $CF = 0.87$
If x_1 is *medium* and x_2 is *medium large*
then Class 2 with $CF = 0.81$.

The classification boundary generated by these fuzzy if–then rules is shown in Fig. 8. In Fig. 8, all the training patterns are correctly classified by much fewer fuzzy if–then rules (i.e., by six rules) than Fig. 5 (25 rules). The reduction of the number of fuzzy if–then rules was realized by the introduction of "*don't care*" as an antecedent fuzzy set. The effect of "*don't care*" becomes much more significant in multidimensional pattern classification problems with many attributes. This will be discussed in Section VI.

As shown in the above six fuzzy if–then rules, some rules have two antecedent conditions while other rules have a single antecedent condition. The number of antecedent conditions of each fuzzy if–then rule was automatically determined by our fuzzy classifier system from given training patterns. For example, we can see from Fig. 8 that an input pattern belongs to Class 2 if its first attribute (x_1) is "*large*". Thus, the fourth fuzzy if–then rule has only a single condition "x_1 is *large*". On the other hand, we do not know whether an input pattern belongs to Class 1 or Class 2 when we know that its first attribute is "*medium*". Thus the second, third and last fuzzy if–then rules have conditions on the second attribute (x_2) as well as the condition on the first attribute. While the number of antecedent conditions of each fuzzy if–then rule differs, all the fuzzy if–then rules can be handled in the same manner in our fuzzy classifier system. This is because "*don't care*" has a membership function (see (1) in Section I), just as the other linguistic values do. For example, the compatibility of an input pattern $\mathbf{x}_p = (x_{p1}, x_{p2})$ with the first fuzzy if–then rule is calculated from (3) as

$$\mu_1(\mathbf{x}_p) = \mu_{\text{medium small}}(x_{p1}) \times \mu_{\text{don't care}}(x_{p2}). \quad (14)$$

On the other hand, the compatibility with the second fuzzy if–then rule is calculated as

$$\mu_2(\mathbf{x}_p) = \mu_{\text{medium}}(x_{p1}) \times \mu_{\text{small}}(x_{p2}). \quad (15)$$

From (14) and (15), we can see that all the fuzzy if–then rules are handled in the same manner even if the number of antecedent conditions of each rule is different.

IV. PERFORMANCE EVALUATION

A. Performance for Training Data: Wine Classification Data

In order to examine the performance of our fuzzy classifier system for multidimensional pattern classification problems with many continuous attributes, we used wine classification data that consist of 178 samples with 13 continuous attributes from three classes. We used the wine data because

1) this data set is available from the University of California, Irvine, database (available via anonymous ftp from *ftp.ics.uci.edu/pub/machine-learning-databases*);
2) this data set has many continuous attributes;
3) simulation results of our fuzzy classifier system can be compared with reported results by another genetics-based machine learning method in [42].

Corcoran and Sen [42] reported the following results of their genetics-based machine learning system, which was based on the Pittsburgh approach where each individual corresponds to a set of if–then rules (i.e., a rule set)

Best classification rate: 100%
Average classification rate: 99.5%
Worst classification rate: 98.3%.

These results were classification rates obtained by ten independent trials where all the 178 samples were used as training data. Each trial was performed with 60 nonfuzzy if–then rules in each individual (i.e., in each rule set), 1500 individuals in each population, and 300 generations (i.e., $1500 \times 300 = 450\,000$ rule sets of 60 nonfuzzy if–then rules were examined in each trial).

We applied our fuzzy classifier system to the same data set. As a preprocessing of the data set for our fuzzy classifier system, the domain of each attribute was linearly normalized to the unit interval $[0, 1]$. By this preprocessing, the maximum and minimum values of each attribute became 1 and 0, respectively. We also used this preprocessing for other data sets in computer simulations of this paper.

Because the wine data have 13 continuous attributes, we used fuzzy if–then rules of the following type in our fuzzy classifier system:

$$R_j\text{: If } x_1 \text{ is } A_{j1} \text{ and } \ldots \text{ and } x_{13} \text{ is } A_{j13} \text{ then Class } C_j \text{ with } CF_j. \quad (16)$$

As in the computer simulations in Section III, we used the five linguistic values and "*don't care*" as antecedent fuzzy sets of each fuzzy if–then rule. The total number of possible fuzzy if–then rules is $6^{13} \cong 1.3 \times 10^{10}$. The task of our fuzzy classifier system is to find a compact rule set with high classification performance from these 6^{13} fuzzy if–then rules.

In order to compare our fuzzy classifier system with the genetics-based machine learning system in [42] with 60 nonfuzzy if–then rules, we also specified the number of fuzzy if–then rules in each population as 60 (i.e., $N_{\text{pop}} = 60$). The other parameters were specified as follows in our computer simulations:

Crossover probability: 1.0

Mutation probability: 0.1

Number of replaced rules in each population: $N_{\text{rep}} = 12$

Stopping condition: 1000 generations.

From these parameter specifications, we can see that 1000 rule sets of 60 fuzzy if–then rules were examined in our fuzzy classifier system. It should be noted that the total number of examined rule sets in our fuzzy classifier system (i.e., 1000 rule sets) was much smaller than that in the genetics-based machine learning system in Corcoran and Sen (i.e., 450 000 rule sets).

We applied our fuzzy classifier system to the wine data 10 times, and the following results were obtained:

Best classification rate: 99.4%

Average classification rate: 98.5%

Worst classification rate: 97.8%.

From the comparison of these results with the aforementioned results in [42], we can see that slightly inferior classification rates were obtained by our fuzzy classifier system with much fewer rule set examinations (i.e., 1000 examinations) than the genetics-based machine learning system with 450 000 rule set examinations. This clearly demonstrates the ability of our fuzzy classifier system to efficiently search for a compact rule set with high classification performance. The above results by our fuzzy classifier systems can be drastically improved by adjusting the grade of certainty of each fuzzy if–then rule. This will be discussed in Section VI.

In Fig. 9, we show how our fuzzy classifier system evolved fuzzy if–then rules. Each closed circle in Fig. 9 shows the average classification rate over the ten trials for the training data at each generation. From Fig. 9, we can see that the average classification rate was rapidly improved from a very poor performance at the initial generation.

For examining the significance of each of the two genetic operations (i.e., crossover and mutation) in our fuzzy classifier system, we applied it to the wine data by specifying the crossover probability or the mutation probability as zero. The other parameters were specified in the same manner as in the above computer simulations. The average classification rate over ten trials with no crossover was 98.0% after 1000 generations. While this is almost the same as the 98.5% average classification rate with both genetic operations, the crossover operation has an effect on the search ability of our fuzzy classifier system. Open circles in Fig. 9 show the simulation results by the fuzzy classifier system with no crossover operation. The search ability was slightly deteriorated by specifying the crossover probability as zero in Fig. 9. On the other hand, the average classification rate over ten trials with no mutation (i.e., with the zero mutation probability) was 3.6% after 1000 generations. This is terribly bad in comparison with the 98.5% average classification rate by our original fuzzy classifier system with both genetic operations. From these observations, we can see that the search in our fuzzy classifier system is mainly driven by the mutation operation.

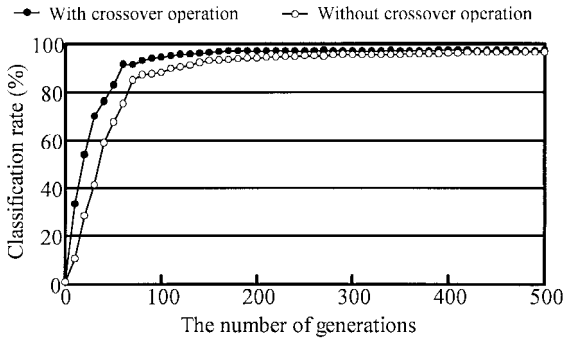

Fig. 9. Average classification rate of the current population at each generation.

We also examined a variant of the crossover operation where a pair of selected fuzzy if–then rules was mated only when they had the same consequent class. When selected fuzzy if–then rules had different consequent classes, a new pair of fuzzy if–then rules was selected. In this manner, the selection procedure was modified so that a pair of selected fuzzy if–then rules always had the same consequent class. We applied this variant of our fuzzy classifier system to the wine data. The average classification rate over ten trials was 98.5% after 1000 generations, which was the same as the result by our original fuzzy classifier system.

B. Performance for Test Data: Credit Approval Data

In the previous section, we demonstrated the performance of our fuzzy classifier system for training data. In this section, we examine its performance for test data by computer simulations on credit approval data. The credit approval data, which are also available from the University of California, Irvine, database, were used in Quinlan [44] for his C4.5 algorithm. This data set consists of 690 samples with 14 attributes from two classes. The performance of the C4.5 algorithm was evaluated by the ten-fold cross-validation in [44]. In the ten-fold cross-validation, all the given samples are divided into ten subsets of the same size (i.e., 69 samples in the case of the

credit approval data). Then, nine subsets are used as training data and the other subset is used as test data. The ten-fold cross-validation consists of ten iterations of the performance evaluation so that each of the ten subsets is used as test data just once.

In [44], the following results were reported as the performance of the C4.5 algorithm for test data (see [44, Table IX-I]).

Best classification rate among various parameter specifications: 85.8%

Average classification rate over various parameter specifications: 84.3%

Worst classification rate among various parameter specifications: 82.5%.

As in [44], we also used the ten-fold cross-validation for evaluating the performance of our fuzzy classifier system for test data. Because the result of the ten-fold cross-validation may depend on the partition of the given data set into ten subsets, we independently performed the ten-fold cross-validation 20 times using different partitions of the given data set (i.e., our fuzzy classifier system was employed 20×10 times in this computer simulation). Our fuzzy classifier system was slightly modified to handle discrete attributes in the credit approval data. For example, the first attribute of the credit approval data has only two attribute values $\{0, 1\}$. Thus we used only the three antecedent fuzzy sets "*small*", "*large*" and "*don't care*" for the first attribute because the other antecedent fuzzy sets such as "medium" have no compatibility with the two attribute values $\{0, 1\}$. From the same reason, we use only the four antecedent fuzzy sets "*small*", "*medium*", "*large*" and "*don't care*" for the fourth attribute with three attribute values $\{1, 2, 3\}$. It should be noted that those attribute values were normalized by the preprocessing procedure of our fuzzy classifier system to $\{0.0, 0.5, 1.0\}$, respectively.

In each trial of the ten-fold cross-validation, the best rule set with the maximum classification rate for training data was selected to evaluate its performance for test data. Our fuzzy classifier system was applied to the credit approval data with the following parameter specifications.

Population size: $N_{\text{pop}} = 100$ (i.e., 100 fuzzy if–then rules)

Crossover probability: 1.0

Mutation probability: 0.1

Number of replaced rules in each population: $N_{\text{rep}} = 20$

Stopping condition: 1000 generations.

From 20 independent iterations of the ten-fold cross-validation, the following average results were obtained for test data.

Classification rate for test data: 86.7%

Error rate for test data: 13.2%

Reject rate for test data: 0.1%.

While the test condition of these results is not exactly the same as that of the above-mentioned results by the C4.5 algorithm, we can see that our fuzzy classifier system (86.7% classification rate) slightly outperformed the C4.5 algorithm (85.8% best classification rate and 84.3% average classification rate) for the credit approval data.

C. Performance for Test Data: Glass Data

Our fuzzy classifier system was also applied to the glass data that consist of 214 samples with nine continuous attributes from six classes. This data set is also available from U.C. Irvine database. This data set is one of 16 data sets used in [45] for examining the performance of his 1R algorithm and Quinlan's C4 algorithm [46]. A random subsampling technique was used in [45] for evaluating the performance of 1R and C4 for test data. In his computer simulations, 2/3 of the given samples were used as training data and the other 1/3 samples were used as test data. In Holte [45], this random split of the given data was iterated 25 times, and the following results for test data were reported (see [45, Table III]).

Classification rate for test data by 1R: 53.8%

Classification rate for test data by C4: 63.2%.

Using the same random sub-sampling technique, we evaluated the performance of our fuzzy classifier system for test data. We used the same parameter specifications as in the computer simulations for the credit approval data. The following classification rates for test data were obtained by our fuzzy classifier system for the glass data.

Classification rate for test data: 64.4%

Error rate for test data: 35.6%

Reject rate for test data: 0.0%.

From the comparison of these results with the above-mentioned results of 1R and C4, we can see that our fuzzy classifier system outperformed the 1R algorithm and was comparable to the C4 algorithm for the glass data.

D. Performance for Test Data: Appendicitis Data

Weiss and Kulikowski [47] examined the performance of various nonfuzzy classification methods by the leaving-one-out procedure for appendicitis data. The same data set was also used in [48] for evaluating the performance of various fuzzy classification methods by the ten-fold cross-validation. The appendicitis data set consists of 106 samples with seven attributes from two classes (for details, see [47]). The following results were reported in [47] as the performance of ten nonfuzzy classification methods such as statistical pattern recognition methods, neural networks and machine learning methods (see [47, Fig. 6.4]).

Best classification rate for test data among ten methods: 89.6%

Average classification rate for test data over ten methods: 84.3%

Worst classification rate for test data among ten methods: 73.6%.

These results were obtained in [47] after carefully tuning parameter values for each method.

On the other hand, Grabisch and Nicolas [48] reported the following results as the performance of six fuzzy classification methods such as a fuzzy k nearest neighbor method and fuzzy pattern matching methods with various criteria (see [48, Table IV]).

Best classification rate for test data among six methods: 87.7%
Average classification rate for test data over six methods: 84.6%
Worst classification rate for test data among six methods: 79.2%.

These results were obtained by the ten-fold cross-validation after carefully tuning parameter values for each method.

Using the leaving-one-out procedure as in [47], we examined the performance of our fuzzy classifier system with the same parameter specifications as in the cases of the credit approval data and the glass data (i.e., we did not customize the parameter values for each test problem in our fuzzy classifier system). The following results were obtained as the performance of our fuzzy classifier system for test data.

Classification rate for test data: 84.9%
Error rate for test data: 13.2%
Reject rate for test data: 1.9%.

From the comparison of these results with the aforementioned results of the other classification methods, we can see that our fuzzy classifier system was comparable to the other classification methods with carefully tuned parameter values. In fact, the 84.9% classification rate obtained by our fuzzy classifier system is larger than or equal to those by six nonfuzzy classification methods such as the nearest neighbor method (classification rate: 82.1%). The classification rate by our fuzzy classifier system is also larger than those by two fuzzy classification methods such as the fuzzy pattern matching with the arithmetic mean operator (classification rate: 80.2%).

E. Performance for Test Data: Cancer Data

Cancer data were used in [47] and [49] for evaluating various classification methods. The cancer data set consists of 286 samples with nine attributes from two classes (for details, see [47]). The following results were reported in [47] as the performance of ten nonfuzzy classification methods (see [47, Fig. 6.6]).

Best classification rate for test data among ten methods: 77.1%
Average classification rate for test data over ten methods: 70.9%
Worst classification rate for test data among ten methods: 65.3%.

These results were obtained from four trials of a random sub-sampling procedure with 70% training patterns and 30% test patterns after carefully tuning parameter values for each method.

Grabisch and Dispot [49] reported the following results as the performance of nine fuzzy classification methods (see [49, Table 5]).

Best classification rate for test data among nine methods: 68.0%
Average classification rate for test data over nine methods: 60.6%
Worst classification rate for test data among nine methods: 55.9%.

These results were obtained from the two-fold cross-validation after carefully tuning parameter values for each method.

Using the ten-fold cross-validation, we examined the performance of our fuzzy classifier system with the same parameter specifications as in the previous computer simulations (i.e., with no customized parameter tuning for this test problem). The following results were obtained as the performance of our fuzzy classifier system for test data from ten independent trials of the ten-fold cross-validation:

Classification rate for test data: 74.0%
Error rate for test data: 25.9%
Reject rate for test data: 0.1%.

We can see from this result that our fuzzy classifier system is comparable to the other classification methods with carefully tuned parameter values examined in [47] and [49]. Since the test condition (ten-fold cross-validation) of this result is not the same as Weiss and Kulikowski (30% test patterns) and Grabisch and Dispot (two-fold cross-validation), we do not rigorously compare our average classification rate (74.0%) with the reported results (e.g., the backpropagation algorithm: 71.5%, and the nearest neighbor method: 65.3%).

Recently, Grabisch [50] reported good results by three fuzzy integral classifiers (i.e., classification methods based on fuzzy integrals) for the cancer data. The following results were reported in Grabisch by the fuzzy integral classifiers trained with the quadratic criterion (QUAD), and the generalized quadratic criterion minimized by the constrained least mean squares algorithm (CLMS) or the heuristic least-mean-squares algorithm (HLMS):

Classification rate by QUAD: 68.5%
Classification rate by CLMS: 72.9%
Classification rate by HLMS: 77.4%.

These results were obtained by the ten-fold cross-validation. We can see from these results that the performance of our fuzzy classifier system is better than the first two versions of the fuzzy integral classifiers. The performance of the fuzzy integral classifier with the HLMS algorithm (i.e., 77.4% classification rate) outperforms our fuzzy classifier system and all the results reported in [47] and [49].

TABLE II
SIMULATION RESULTS WITH VARIOUS POPULATION SIZES

Population size	Classification rate
20	98.2%
40	98.3%
60	98.5%
80	98.7%
100	98.8%
200	99.0%

TABLE III
SIMULATION RESULTS WITH VARIOUS CROSSOVER PROBABILITIES

Crossover probability	Classification rate
0.5	98.5%
0.6	98.6%
0.7	98.7%
0.8	98.6%
0.9	98.7%
1.0	98.5%

TABLE IV
SIMULATION RESULTS WITH VARIOUS MUTATION PROBABILITIES

Mutation probability	Classification rate
0.01	97.8%
0.02	98.4%
0.05	98.4%
0.10	98.5%
0.20	98.8%
0.50	95.2%

TABLE V
SIMULATION RESULTS WITH VARIOUS SPECIFICATIONS OF THE NUMBER OF REPLACED RULES

Replaced rules	Classification rate
3	97.7%
6	98.5%
12	98.5%
18	98.7%
24	98.8%
30	98.7%

TABLE VI
CPU TIME FOR EACH TEST PROBLEM

Data set	Samples	Attributes	CPU Time
Wine	178	13	3.7(min.)
Credit	690	14	25.8(min.)
Glass	214	9	6.3(min.)
Appendicitis	106	7	2.8(min.)
Cancer	286	9	8.3(min.)

TABLE VII
SIMULATION RESULTS WITH DIFFERENT ANTECEDENT FUZZY SETS

Number of fuzzy sets used for each attribute	3	4	5
Classification rate	84.9%	80.2%	84.9%
Error rate	15.1%	16.0%	13.2%
Reject rate	0.0%	3.8%	1.9%

F. Parameter Specifications and Computation Time

In the previous computer simulations for evaluating the performance of our fuzzy classifier system for test data, we used the same parameter specifications for the four different data sets (i.e., credit approval, glass, appendicitis, and cancer). The simulation results in the previous sections demonstrated that our fuzzy classifier system with the common parameter specifications has high classification performance for the four data sets. This implicitly shows that our fuzzy classifier system does not require fine parameter tuning.

In order to show the robustness of our fuzzy classifier system with respect to parameter specifications, we show simulation results with different parameter values. For the wine data, we performed computer simulations by changing one of the four parameter values (i.e., population size, crossover probability, mutation probability, and the number of replaced rules). The other parameters were specified in the same manner as in Section IV-A. Simulation results obtained by ten trials for each parameter specification are summarized in Tables II–V. From these tables, we can see that our fuzzy classifier system works well in a wide-range of parameter values.

Computation time is a very important criterion as well as classification performance when we choose a classification method to be applied to real-world problems. We examined the CPU time of our fuzzy classifier system by using all the given samples in each data set as training data. In Table VI, we show the CPU time for each data set required for a single trial until 1000 generations with 100 fuzzy if–then rules. The CPU time was measured by a workstation (with Ultra SPARC 143 MHz, 32 MB RAM, and Sun OS 5.5). Our fuzzy classifier system was coded as a computer program using C language. From Table VI, we can see that our fuzzy classifier system does not require long computation time even if it is applied to real-world pattern classification problems with many attributes. This clearly shows the applicability of our fuzzy classifier system to real-world pattern classification problems.

We also examined the effect of antecedent fuzzy sets on the performance of our fuzzy classifier system. Instead of the five fuzzy sets in Fig. 1, we used three fuzzy sets and four fuzzy sets. In both cases, each axis of pattern spaces was homogeneously partitioned into triangular fuzzy sets in the same manner as in Fig. 1. Our fuzzy classifier system with the three fuzzy sets and "*don't care*" was applied to the appendicitis data for evaluating its performance for test data by the leaving-one-out procedure in the same manner as in Section IV-D. The four fuzzy sets and "*don't care*" were also examined in the same manner. Simulation results are summarized in Table VII where the previous result with the five fuzzy sets is also shown. From this table, we can see that the performance of our fuzzy classifier system is not very sensitive to the choice of antecedent fuzzy sets.

V. CHARACTERISTIC BEHAVIOR OF FUZZY CLASSIFIER SYSTEM

We have already demonstrated that our fuzzy classifier system has high classification ability while it involves no learning mechanism of membership functions. In the computer simulations of the previous sections, we did not use any expert knowledge for specifying the membership function of each

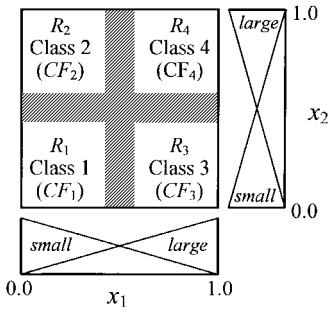

Fig. 10. Fuzzy partition with four fuzzy if–then rules.

antecedent fuzzy set, either. We simply used the homogeneous fuzzy partition for each axis of the pattern space. In this section, we discuss why our fuzzy classifier system with such simple specifications worked well for the real-world pattern classification problems with many attributes.

A. Classification Boundary

If we use a crisp partition for each axis of the pattern space as in many machine learning techniques, classification boundaries between different classes are usually parallel to the axes of the pattern space (for example, see [44, ch. 10]). While our fuzzy classifier system employs the fixed membership functions and the simple fuzzy grid as shown in Fig. 1, it can generate various classification boundaries because it uses the fuzzy reasoning method that takes into account the grade of certainty of each fuzzy if–then rule.

For simplicity of discussion, let us assume that we have only the following four fuzzy if–then rules generated by the fuzzy partition in Fig. 10:

R_1: If x_1 is *small* and x_2 is *small* then Class 1 with $CF = CF_1$,

R_2: If x_1 is *small* and x_2 is *large* then Class 2 with $CF = CF_2$,

R_3: If x_1 is *large* and x_2 is *small* then Class 3 with $CF = CF_3$,

R_4: If x_1 is *large* and x_2 is *large* then Class 4 with $CF = CF_4$.

When these four fuzzy if–then rules have the same grade of certainty (e.g., $CF_j = 1.0$ for $j = 1, 2, 3, 4$), the classification boundary is very simple as shown in Fig. 11(a) where each real number in parentheses denotes the grade of certainty of the corresponding fuzzy if–then rule. Various classification boundaries, however, can be generated as shown in Fig. 11(b)–(d) by assigning different grades of certainty to the fuzzy if–then rules. In Fig. 11(b)–(d), the certainty grade of each fuzzy if–then rule is shown as a real number in parentheses as in Fig. 11(a). There is no region of Class 2 in Fig. 11(d) because we specified the grade of certainty CF_2 of the corresponding fuzzy if–then rule R_2 as $CF_2 = 0$ in this figure. As shown in Fig. 11, our fuzzy classifier system has the ability to generate complicated classification boundaries while each fuzzy if–then rules is very simple. This is one reason of high performance

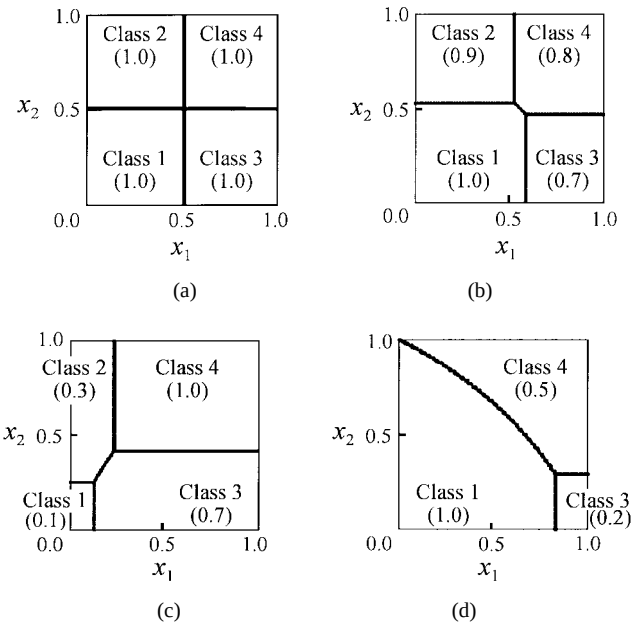

Fig. 11. Classification boundary generated by the four fuzzy if–then rules in Fig. 11. Real numbers in parentheses denote the grades of certainty of the fuzzy if–then rules. In (d), $CF_2 = CF_3 = 0$.

of our fuzzy classifier system. That is, while our fuzzy classifier system involves no tuning mechanism of membership functions, it can generate complicated classification boundaries by means of the heuristic specification of the grade of certainty of each fuzzy if–then rule. From Fig. 11(b)–(d), we can see that the classification boundaries generated by fuzzy if–then rules are not always parallel to the axes of the pattern space. This is one characteristic feature of the fuzzy rule-based classification in comparison with other methods based on nonfuzzy if–then rules. From Fig. 10, we can see that the four fuzzy if–then rules with different consequent classes overlap one another. Such overlap plays an important role in generating various classification boundaries in Fig. 11. The overlap of multiple fuzzy if–then rules with different consequent classes is another characteristic feature of the fuzzy rule-based classification. The conflict is automatically resolved through the membership functions of antecedent fuzzy sets and the grade of certainty of each fuzzy if–then rule.

B. Effect of "Don't Care" Attribute

As we have already mentioned in Section III, the use of "*don't care*" as an antecedent fuzzy set has a significant effect on the performance of our fuzzy classifier system. In order to demonstrate the importance of "*don't care*" as an antecedent fuzzy set, we applied our fuzzy classifier system without "*don't care*" to the wine classification problem. That is, we used only the five linguistic values in Fig. 1 as antecedent fuzzy sets. Simulation results by our fuzzy classifier system over ten trials are as follows.

Average classification rate obtained without *don't care*: 64.5%.

Average classification rate obtained with *don't care*: 98.5% (see Section IV-A).

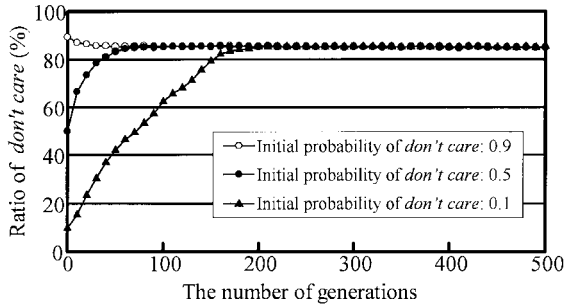

Fig. 12. Average ratio of "*don't care*" at each generation.

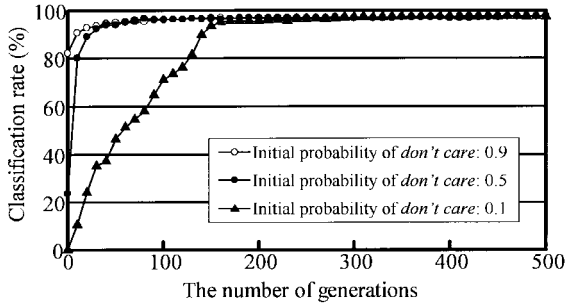

Fig. 13. Average classification rate at each generation.

From the comparison between these two results, we can see that the performance of our fuzzy classifier system was significantly deteriorated when we did not use "*don't care*" as an antecedent fuzzy set.

In order to examine the effect of "*don't care*" further, we monitored the number of "*don't care*" at each generation in computer simulations on the wine data. We applied our fuzzy classifier system to the wine data using three different initial populations of 60 fuzzy if–then rules. Three initial populations were generated by assigning "*don't care*" to each antecedent fuzzy set with the probability of 0.1, 0.5, and 0.9, respectively.

In Fig. 12, we show the ratio of "*don't care*" to 13×60 antecedent fuzzy sets of 60 fuzzy if–then rules at each generation. Fig. 12 is the average result over ten trials. From this figure, we can see that the ratios of "*don't care*" approached the same value in these three conditions while the initial ratios were different. In Fig. 13, we show the corresponding results about classification rates at each generation. From this figure, we can see that the initial classification rates were relatively high when the initial ratios of "*don't care*" were high.

The effect of "*don't care*" can be also explained from the point of view of the covered area by each fuzzy if–then rule. For example, a fuzzy if–then rule with "*small*" for all the 13 attributes of the wine data can cover only $(1/4)^{13}$ of the 13-dimensional (13-D) pattern space because the antecedent fuzzy set "*small*" only covers 1/4 of each axis (see Fig. 1). Thus we can see that such a fuzzy if–then rule can cover only a very small area in the pattern space. The area covered by each fuzzy if–then rule is enlarged by introducing "*don't care*" as an antecedent fuzzy set. For example, a fuzzy if–then rule with "*small*" for two attributes and "*don't care*" for the other 11 attributes of the wine data can cover 1/16 of the 13-D pattern space.

While we demonstrated the positive effect of "*don't care*" on the performance of fuzzy if–then rules by computer simulations, it may have a negative effect at the same time. A fuzzy if–then rule with "*don't care*" attributes can extend the region of its consequent class beyond where there are training data. This may cause the increase of the error rate for test data. While our fuzzy classifier system does not have any safeguard against such over-generalization, the negative effect was not clear in computer simulations of this paper. The inclusion of a misclassification penalty in the fitness function can be viewed as a safeguard against the over-generalization of fuzzy if–then rules. This will be discussed in the next section.

The introduction of "*don't care*" has also a large effect on the comprehensibility of generated fuzzy if–then rules. As shown in Fig. 12, about 85% of antecedent fuzzy sets were "*don't care*" in the generated fuzzy if–then rules for the wine data. This means that each of the generated fuzzy if–then rules has only a few antecedent conditions. Short fuzzy if–then rules with a few antecedent conditions can be understood by human user much easier than long rules with many conditions.

The use of general fuzzy if–then rules with many "*don't care*" conditions was also discussed in Bonarini [51] where fuzzy classifier systems were employed for designing fuzzy logic controllers. Ishibuchi and Murata [52] demonstrated that the introduction of "*don't care*" decreased the number of fuzzy if–then rules in fuzzy rule-based classification systems whose antecedent fuzzy sets were trained by a genetic algorithm in the framework of the Pittsburgh approach. Fuzzy switching functions with "*don't care*" conditions were discussed in [53]–[55] for concise function representation, data reduction and error correction.

VI. Extensions

We have already explained the simplest version of our fuzzy classifier system in the previous sections. In this section, we suggest some directions to extend our fuzzy classifier system.

A. Misclassification Penalty

In our fuzzy classifier system, each fuzzy if–then rule receives a unit reward when it correctly classifies a training pattern. Thus fuzzy if–then rules that correctly classify many training patterns receive a large amount of reward. Now let us consider a fuzzy if–then rule that correctly classifies 20 training patterns and misclassifies ten training patterns. In this case, a high fitness value is assigned to this fuzzy if–then rule (i.e., $\mathit{fitness}(R_j) = 20$) while it also misclassifies many training patterns. In our fuzzy classifier system, such a fuzzy if–then rule survives the generation update because of its high fitness value.

Such a fuzzy if–then rule can be removed during the generation update by introducing the misclassification penalty. Let us denote the penalty for the misclassification of a single training pattern by w_{error}. Using the misclassification penalty, the fitness value of each fuzzy if–then rule in (11) is modified as follows:

$$\mathit{fitness}(R_j) = \mathit{NCP}(R_j) - w_{\text{error}} \cdot \mathit{NMP}(R_j) \qquad (17)$$

TABLE VIII
SIMULATION RESULTS FOR TRAINING DATA WITH VARIOUS MISCLASSIFICATION PENALTIES

Penalty	Classification rate	Error rate	Reject rate
0	81.9%	18.1%	0.0%
1	85.2%	14.2%	0.6%
2	84.3%	12.8%	2.9%
5	82.4%	14.3%	3.3%
10	82.2%	15.0%	2.8%
20	82.0%	14.7%	3.3%

TABLE IX
SIMULATION RESULTS FOR TEST DATA WITH VARIOUSMISCLASSIFICATION PENALTIES

Penalty	Classification rate	Error rate	Reject rate
0	74.0%	25.9%	0.1%
1	76.6%	22.5%	0.9%
2	74.5%	20.0%	5.5%
5	74.9%	18.4%	6.7%
10	75.8%	18.7%	5.5%
20	75.4%	17.7%	6.4%

where $NCP(R_j)$ is the number of correctly classified training patterns by the fuzzy if–then rule R_j, and $NMP(R_j)$ is the number of misclassified patterns.

In order to demonstrate the effect of the misclassification penalty, we applied the modified fuzzy classifier system with the misclassification penalty to the cancer data. We iterated the ten-fold cross-validation procedure ten times in the same manner as in Section IV for each of six penalty values (i.e., $w_{\text{error}} = 0, 1, 2, 5, 10, 20$). Simulation results for training data and test data are summarized in Tables VIII and IX, respectively. It should be noted that the fuzzy classifier system with $w_{\text{error}} = 0$ corresponds to its original version in the previous sections. From these tables, we can observe the following.

(1) By appropriately specifying the misclassification penalty, we can improve the performance of our fuzzy classifier system for the cancer data (see Table IX for test data). In fact, the result with $w_{\text{error}} = 10$ for test data (i.e., 75.8% classification rate, 18.7% error rate, and 5.5% reject rate) can be viewed as outperforming all the reported results of 19 classification methods in [47] and [49], and being comparable to the best result (i.e., 77.4% classification rate, 22.6% error rate, and 0.0% rejection rate) in Grabisch [50].

(2) By assigning large values to the misclassification penalty, high reject rates and low error rates were obtained (see Table VIII for training data and Table IX for test data).

The second observation can be explained by the definition of the fitness value in (17). That is, fuzzy if–then rules that misclassify some training patterns may have negative fitness values in the modified fuzzy classifier system with a large misclassification penalty even if those rules correctly classify many training patterns. Thus such fuzzy if–then rules are replaced with new rules in the next generation. In this way, high rejection rates remain during the generation update.

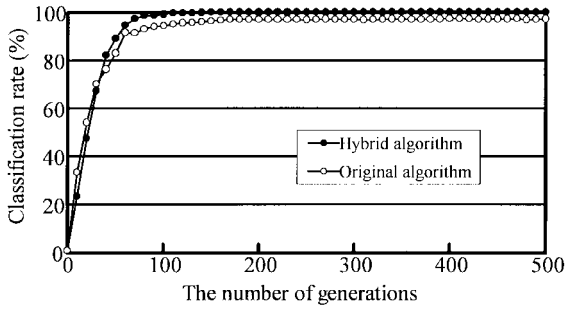

Fig. 14. Simulation results by the hybrid algorithm.

B. Learning of the Grade of Certainty

It has been demonstrated that the performance of fuzzy if–then rules can be improved by adjusting the grade of certainty of each rule in Nozaki *et al.* [56]. The learning procedure in [56] is the following simple reward-and-punishment scheme. When a training pattern is correctly classified by a fuzzy if–then rule (say, $R_{\hat{j}}$), its grade of certainty is increased as the reward of the correct classification as

$$CF_{\hat{j}}^{\text{new}} = CF_{\hat{j}}^{\text{old}} + \eta_1 \cdot \left(1 - CF_{\hat{j}}^{\text{old}}\right) \tag{18}$$

where η_1 is the learning rate for increasing the grade of certainty. On the other hand, if the training pattern is misclassified by the fuzzy if–then rule $R_{\hat{j}}$, its grade of certainty is decreased as the punishment of the misclassification as

$$CF_{\hat{j}}^{\text{new}} = CF_{\hat{j}}^{\text{old}} - \eta_2 \cdot CF_{\hat{j}}^{\text{old}} \tag{19}$$

where η_2 is the learning rate for decreasing the grade of certainty.

This learning procedure can be combined with our fuzzy classifier system to construct a hybrid algorithm. In the hybrid algorithm, the learning procedure is applied to each population of fuzzy if–then rules before the fitness value is assigned to each rule.

In order to demonstrate the effect of the learning procedure, we applied the hybrid algorithm to the wine data in the same manner as in Section IV. The learning procedure was iterated ten times for each pattern (i.e., ten epochs) at each generation. Simulation results are summarized in Fig. 14 together with the previous results by the original fuzzy classifier system in Section IV. This figure shows the average classification rate for training data at each generation over ten trials. From this figure, we can observe that the hybrid algorithm can find fuzzy if–then rules with high classification performance in early generations. In fact, 100% classification rate for training data was obtained in all the ten trials. On the average, all the training patterns were correctly classified after 138 generations. This means that 138 rule set examinations in the hybrid algorithm were required for 100% classification rate on the average. This result clearly outperformed the reported result of the genetics-based machine learning system in [42].

C. Alternative Coding

Let us consider a classification problem in Fig. 15. All the given training patterns may be correctly classified by the

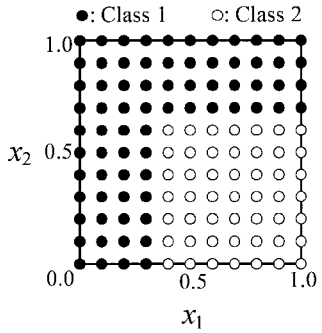

Fig. 15. Example of a pattern classification problem.

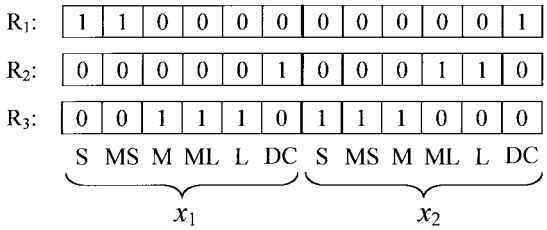

Fig. 16. Alternative coding method.

following three fuzzy if–then rules if we introduce combined antecedent fuzzy sets such as "*small* or *medium small*":

R_1: If x_1 is *small* or *medium small* and x_2 is *don't care* then Class 1 with $CF = CF_1$,

R_2: If x_1 is *don't care* and x_2 is *medium large* or *large* then Class 1 with $CF = CF_2$,

R_3: If x_1 is *medium* or *medium large* or *large* and x_2 is *small* or *medium small* or *medium* then Class 2 with $CF = CF_3$.

These fuzzy if–then rules can not be generated by our fuzzy classifier system because only a single antecedent fuzzy set is assigned to each attribute in the coding of each fuzzy if–then rule.

In our fuzzy classifier system, we can introduce an alternative coding method for representing the above fuzzy if–then rules. The alternative coding method has six boxes for each attribute as shown in Fig. 16 where each of the above three fuzzy if–then rules is denoted by a string with 12 boxes. In the alternative coding, "1" means that the corresponding antecedent fuzzy set appears in the fuzzy if–then rule. A similar coding method has already been used in [26] where "*don't care*" was not explicitly used as an antecedent fuzzy set. A string with all 1's in [26] corresponds to "*don't care*". But they are not equivalent to each other because "*don't care*" has a rectangular membership function as shown in Fig. 2 while the string with all 1's in [26] is the union of triangular membership functions.

D. Biased Probabilities

As we have already demonstrated, the antecedent fuzzy set "*don't care*" plays a very important role in our fuzzy classifier system. In fact, it did not work well without this special antecedent fuzzy set as shown in Section V-B. In order to effectively utilize "*don't care*" in our fuzzy classifier system, we can use the following two tricks:

1) To assign a larger probability to "*don't care*" than the other antecedent fuzzy sets when an initial population is generated. For the effect of this trick, see Fig. 13 in Section V.
2) To bias the mutation probability in order that the mutation to "*don't care*" has a larger mutation probability than the mutation from "*don't care*" to the other antecedent fuzzy sets.

These tricks force fuzzy if–then rules to have many "*don't care*" symbols in their antecedent parts. Because attributes with "*don't care*" can be omitted in fuzzy if–then rules, we can obtain general fuzzy if–then rules with a small number of linguistic conditions in the antecedent parts by these two tricks.

E. Heuristic Procedure for Generating an Initial Population

As Nakaoka *et al.* [29] have already pointed out, the performance of a randomly generated initial population is usually very poor (see also Figs. 9 and 13). This is because many training patterns have no compatible initial fuzzy if–then rules. That is, the classification of many training patterns is rejected by the initial population of randomly generated fuzzy if–then rules. Nakaoka *et al.* [29] proposed an idea to count the number of antecedent fuzzy sets that are compatible with an input vector in their fuzzy classifier system for fuzzy control problems. Bonarini [51] proposed a fuzzy classifier system to efficiently design fuzzy logic controllers, in which a special rule generation mechanism was employed to insure that useless fuzzy if–then rules with no compatible input cases were never generated.

For pattern classification problems, an initial population with high classification performance can be easily generated as follows:

Step 1: For each training pattern, find a fuzzy if–then rule that has the largest compatibility with the training pattern. Because "*don't care*" has the largest compatibility to any attribute value, this special fuzzy set is not considered as an antecedent fuzzy set in this stage. For example, from a three-dimensional training pattern $\mathbf{x}_p = (0.02, 0.51, 0.98)$, the following fuzzy if–then rule is found as the most compatible rule when we use the five linguistic value in Fig. 1:

If x_1 is *small* and x_2 is *medium* and x_3 is *large* then Class C_j with $CF = CF_j$.

In this step, m fuzzy if–then rules are generated from the given m training patterns.

Step 2: With a pre-specified probability, replace antecedent fuzzy sets with "*don't care*".

Step 3: From the generated m fuzzy if–then rules, randomly select N_{pop} rules where N_{pop} is the population size. In this stage, we can also use the rule selection method in [40], [41].

F. Variable Population Size

Because the number of fuzzy if–then rules in each population of our fuzzy classifier system is constant, it can not be directly applied to the rule selection problem with two objectives: to maximize the classification rate and to minimize the number of fuzzy if–then rules. When we try to minimize the number of fuzzy if–then rules as well as to maximize the classification rate, unnecessary fuzzy if–then rules have to be removed from each population in our fuzzy classifier system. This can be done by replacing all the fuzzy if–then rules that have no reward (i.e., *fitness*$(R_j) = 0$) with a small number of newly generated fuzzy if–then rules. In this way, unnecessary fuzzy if–then rules are removed from a current population at each generation. It should be noted that the population size is not constant in this scheme.

Another idea for minimizing the number of fuzzy if–then rules is to assign a flag to each fuzzy if–then rule to indicate whether the corresponding fuzzy if–then rule should be included in the fuzzy rule-based system or not. That is, a fuzzy if–then rule is included in the fuzzy rule-based system if its flag is "1". Otherwise (i.e., its flag is "0"), the corresponding fuzzy if–then rule is not included in the fuzzy rule-based system. In this scheme, the population size is constant. A similar idea was used in Shimojima *et al.* [22] and Ishibuchi *et al.* [40], [41] for the Pittsburgh approach.

VII. Conclusion

In this paper, we demonstrated high performance of a fuzzy classifier system for pattern classification problems with many continuous attributes. Our fuzzy classifier system is very simple because

1) antecedent fuzzy sets of each fuzzy if–then rule are prespecified linguistic values with fixed membership functions;
2) the consequent class and the grade of certainty of each fuzzy if–then rule are determined by a simple heuristic procedure.

No tuning mechanism of membership functions is involved in our fuzzy classifier system. The simplicity of the algorithm is one advantage of our fuzzy classifier system. Because we use pre-specified linguistic values as antecedent fuzzy sets, fuzzy if–then rules are always linguistically interpreted by human users. The comprehensibility of the generated fuzzy if–then rules is another advantage of our fuzzy classifier system. Since the generated fuzzy if–then rules involve many "*don't care*" conditions (e.g., about 85% of antecedent fuzzy sets were "*don't care*" in the computer simulation on wine data; see Fig. 12), they can be rewritten as short fuzzy if–then rules with a small number of antecedent conditions. This makes the generated fuzzy if–then rules more comprehensible. By computer simulations on real-world test problems, we examined the performance of our fuzzy classifier system. Simulation results demonstrated that our fuzzy classifier system outperformed many other classification methods. We also suggested various directions to extend our fuzzy classifier system in order to improve its performance. Some variants of our fuzzy classifier system were examined in [57] and [58].

It has been often claimed that fuzzy rule-based systems with grid-type fuzzy partitions can not scale up high-dimensional problems with many inputs while such systems work well for low-dimensional problems. This paper clearly demonstrated that our fuzzy classifier system based on simple fuzzy grids worked well for real-world pattern classification problems with more than ten continuous attributes. The main contribution of this paper is to have proposed an approach to the design of fuzzy rule-based systems with both high performance and high comprehensibility, and to have clearly demonstrated its effectiveness by computer simulations on real-world pattern classification problems. One difficulty of our fuzzy classifier system is that the number of fuzzy if–then rules is not minimized. In our fuzzy classifier system, the fitness function is assigned to each fuzzy if–then rule and the quality of the rule set is not taken into account in the evolution of fuzzy if–then rules. Thus, it is difficult to directly minimize the number of fuzzy if–then rules by including it in the fitness function even if we extend the population size to an adaptable parameter. For example, it is known that only five fuzzy if–then rules can classify all the 178 samples in the wine data (see [59]). This result was obtained by a multi-objective genetic algorithm based on the Pittsburgh approach where each individual corresponds to a rule set. Because a fitness value is assigned to each rule set in the Pittsburgh approach, we can minimize the number of fuzzy if–then rules by a genetic algorithm. The computation load of the Pittsburgh approach, however, is much larger than our fuzzy classifier system. Including a minimization procedure of the number of fuzzy if–then rules in our fuzzy classifier system without involving significant increase of computation load is left for future work.

References

[1] M. Sugeno, "An introductory survey of fuzzy control," *Inf. Sci.*, vol. 36, nos. 1/2, pp. 59–83, 1985.
[2] C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller part I and part II," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, pp. 404–435, Mar./Apr. 1990.
[3] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, pp. 116–132, Jan./Feb. 1985.
[4] J.-S. R. Jang, "Fuzzy controller design without domain experts," in *Proc. 1st IEEE Int. Conf. Fuzzy Systems*, San Diego, CA, Mar. 8–12, 1992, pp. 289–296.
[5] L. X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, pp. 1414–1427, Nov./Dec. 1992.
[6] M. Sugeno and T. Yasukawa, "A fuzzy-logic-based approach to qualitative modeling," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 7–31, Feb. 1993.
[7] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. of Michigan Press, 1975.
[8] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[9] B. Carse, T. C. Fogarty, and A. Munro, "Evolving fuzzy rule based controllers using genetic algorithms," *Fuzzy Sets Syst.*, vol. 80, pp. 273–293, June 1996.
[10] P. Thrift, "Fuzzy logic synthesis with genetic algorithms," in *Proc. 4th Int. Conf. Genetic Algorithms*, Univ. Calif., San Diego, July 13–16, 1991, pp. 509–513.
[11] D. S. Feldman, "Fuzzy network synthesis with genetic algorithms," in *Proc. 5th Int. Conf. Genetic Algorithm*, Univ. Illinois, Urbana-Champaign, July 17–21, 1993, pp. 312–317.
[12] C. L. Karr, "Design of an adaptive fuzzy logic controller using a genetic algorithm," in *Proc. 4th Int. Conf. Genetic Algorithms*, Univ. Calif., San Diego, July 13–16, 1991, pp. 450–457.

[13] C. L. Karr and E. J. Gentry, "Fuzzy control of pH using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 46–53, Feb. 1993.
[14] F. Herrera, M. Lozano, and J. L. Verdegay, "Tuning fuzzy logic controllers by genetic algorithms," *Int. J. Approx. Reas.*, vol. 12, pp. 299–315, Apr./May 1995.
[15] C. Z. Janikow, "A genetic algorithm for optimizing fuzzy decision trees," in *Proc. 6th Int. Conf. Genetic Algorithms*, Univ. Pittsburgh, Pittsburgh, PA, July 15–19, 1995, pp. 421–428.
[16] C. Z. Janikow, "A genetic algorithm method for optimizing the fuzzy components of a fuzzy decision tree," in *Genetic Algorithms for Pattern Recognition*, S. K. Pal and P. P. Wang, Eds. Boca Raton, FL: CRC, 1996, pp. 253–281.
[17] H. Nomura, I. Hayashi, and N. Wakami, "A self-tuning method of fuzzy reasoning by genetic algorithm," in *Proc. 1992 Int. Fuzzy Systems Intelligent Control Conf.*, Louisville, KY, Mar. 16–18, 1992, pp. 236–245.
[18] M. A. Lee and H. Takagi, "Integrating design stages of fuzzy systems using genetic algorithms," in *Proc. 2nd IEEE Int. Conf. Fuzzy Systems*, San Francisco, CA, Mar. 28–Apr. 1, 1993, pp. 612–617.
[19] D. Park, A. Kandel, and G. Langholz, "Genetic-based new fuzzy reasoning models with application to fuzzy control," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, pp. 39–47, Jan. 1994.
[20] A. Homaifar and E. McCormick, "Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 129–139, May 1995.
[21] H. Ishigami, T. Fukuda, T. Shibata, and F. Arai, "Structure optimization of fuzzy neural network by genetic algorithm," *Fuzzy Sets Syst.*, vol. 71, pp. 257–264, May 1995.
[22] K. Shimojima, T. Fukuda, and Y. Hasegawa, "Self-tuning fuzzy modeling with adaptive membership function, rules, and hierarchical structure based on genetic algorithm," *Fuzzy Sets Syst.*, vol. 71, pp. 295–309, May 1995.
[23] S. Matsushita, A. Kuromiya, M. Yamaoka, T. Furuhashi, and Y. Uchikawa, "Determination of antecedent structure for fuzzy modeling using genetic algorithm," in *Proc. 3rd IEEE Int. Conf. Evolutionary Computation*, Nagoya University, Nagoya, Japan, May 20–22, 1996, pp. 235–238.
[24] L. B. Booker, D. E. Goldberg, and J. H. Holland, "Classifier systems and genetic algorithms," *Artif. Intell.*, vol. 40, pp. 235–282, Sept. 1989.
[25] S. F. Smith, "A learning system based on genetic algorithms," Ph.D. dissertation, Univ. Pittsburgh, Pittsburgh, PA, 1980.
[26] M. Valenzuela-Rendon, "The fuzzy classifier system: A classifier system for continuously varying variables," in *Proc. 4th Int. Conf. Genetic Algorithms*, Univ. California, San Diego, July 13–16, 1991, pp. 346–353.
[27] A. Parodi and P. Bonelli, "A new approach to fuzzy classifier systems," in *Proc. 5th Int. Conf. Genetic Algorithms*, Univ. Illinois, Urbana-Champaign, July 17–21, 1993, pp. 223–230.
[28] T. Furuhashi, K. Nakaoka, and Y. Uchikawa, "Suppression of excessive fuzziness using multiple fuzzy classifier systems," in *Proc. 3rd IEEE Int. Conf. Fuzzy Systems*, Orlando, FL, June 26–29, 1994, pp. 411–414.
[29] K. Nakaoka, T. Furuhashi, and Y. Uchikawa, "A study on apportionment of credits of fuzzy classifier system for knowledge acquisition of large scale systems," in *Proc. 3rd IEEE Int. Conf. Fuzzy Systems*, Orlando, FL, June 26–29, 1994, pp. 1797–1800.
[30] S. Abe and M.-S. Lan, "A method for fuzzy rules extraction directly from numerical data and its application to pattern classification," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 18–28, Feb. 1995.
[31] S. Abe, M.-S. Lan, and R. Thawonmas, "Tuning of a fuzzy classifier derived from data," *Int. J. Approx. Reas.*, vol. 14, pp. 1–24, Jan. 1996.
[32] P. K. Simpson, "Fuzzy min-max neural networks—Part I: Classification," *IEEE Trans. Neural Networks*, vol. 3, pp. 776–786, Sept. 1992.
[33] S. K. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets, and classification," *IEEE Trans. Neural Networks*, vol. 3, pp. 683–697, Sept. 1992.
[34] W. Pedrycz, "Fuzzy neural networks with reference neurons as pattern classifiers," *IEEE Trans. Neural Networks*, vol. 3, pp. 770–775, Sept. 1992.
[35] S. Mitra and S. K. Pal, "Self-organizing neural network as a fuzzy classifier," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, pp. 385–399, Mar. 1994.
[36] S. Mitra, "Fuzzy MLP based expert system for medical diagnosis," *Fuzzy Sets Syst.*, vol. 65, pp. 285–296, Aug. 1994.
[37] V. Uebele, S. Abe, and M.-S. Lan, "A neural-network-based fuzzy classifier," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 353–361, Feb. 1995.
[38] S. K. Halgamuge, W. Poechmueller, and M. Glesner, "An alternative approach for generation of membership functions and fuzzy rules based on radial and cubic basis function networks," *Int. J. Approx. Reas.*, vol. 12, pp. 279–298, Apr./May 1995.
[39] H. Ishibuchi, K. Nozaki, and H. Tanaka, "Distributed representation of fuzzy rules and its application to pattern classification," *Fuzzy Sets Syst.*, vol. 52, pp. 21–32, Nov. 1992.
[40] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Selecting fuzzy if–then rules for classification problems using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 260–270, Aug. 1995.
[41] H. Ishibuchi, T. Murata, and I. B. Turksen, "Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems," *Fuzzy Sets Syst.*, vol. 89, pp. 135–149, July 1997.
[42] A. L. Corcoran and S. Sen, "Using real-valued genetic algorithms to evolve rule sets for classification," in *Proc. 1st IEEE Int. Conf. Evolutionary Computation*, Orlando, FL, June 27–29, 1994, pp. 120–124.
[43] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing*, D. E. Rumelhart and J. L. MaClelland, Eds. Cambridge, MA: MIT Press, 1986, pp. 318–362.
[44] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.
[45] R. C. Holte, "Very simple classification rules perform well on most commonly used dataset," *Mach. Learn.*, vol. 11, pp. 63–91, 1993.
[46] J. R. Quinlan, "Simplifying decision trees," *Int. J. Man–Mach. Stud.*, vol. 27, pp. 221–234, Sept. 1987.
[47] S. M. Weiss and C. A. Kulikowski, *Computer Systems That Learn*. San Mateo, CA: Morgan Kaufmann, 1991.
[48] M. Grabisch and J.-M. Nicolas, "Classification by fuzzy integral: Performance and tests," *Fuzzy Sets Syst.*, vol. 65, pp. 255–271, Aug. 1994.
[49] M. Grabisch and F. Dispot, "A comparison of some methods of fuzzy classification on real data," in *Proc. 2nd Int. Conf. Fuzzy Logic Neural Networks*, Iizuka, Japan, July 17–22, 1992, pp. 659–662.
[50] M. Grabisch, "The representation of importance and interaction of features by fuzzy measures," *Pattern Recognit. Lett.*, vol. 17, pp. 567–575, 1996.
[51] A. Bonarini, "Evolutionary learning of general fuzzy rules with biased evaluation functions: Competition and cooperation," in *Proc. 1st IEEE Conf. Evolutionary Computation*, June 27–29, 1994, pp. 51–56.
[52] H. Ishibuchi and T. Murata, "Learning of fuzzy classification rules by a genetic algorithm," *Electron. Commun. Jpn.*, vol. 80, pp. 37–46, Mar. 1997.
[53] M. Mukaidono, "An improved method for minimizing fuzzy switching functions," in *Proc. 14th Int. Symp. Multiple-Valued Logic*, 1984, pp. 196–201.
[54] E. T. Lee, "Fuzzy symmetric functions with don't-care conditions and applications," *Kybernetica*, vol. 22, no. 4, pp. 54–68, 1998.
[55] N. Motoki and K. Miyasaki, "Minimization of fuzzy switching functions," *Mem. Tohoku Inst. Technol.,*, ser. I, no. 17, pp. 197–202, 1997.
[56] K. Nozaki, H. Ishibuchi, and H. Tanaka, "Adaptive fuzzy rule-based classification systems," *IEEE Trans. Fuzzy Syst.*, vol. 4, pp. 238–250, Aug. 1996.
[57] H. Ishibuchi and T. Nakashima, "Performance evaluation of various variants of fuzzy classifier systems for pattern classification problems," in *Proc. 16th Annu. Meeting North American Fuzzy Inf. Processing Soc.*, Syracuse, NY, Sept. 21–24, 1997, pp. 245–250.
[58] ———, "Improving the performance of fuzzy classifier systems for pattern classification problems with continuous attributes," *IEEE Trans. Ind. Electron.*, to be published.
[59] H. Ishibuchi and T. Murata, "Minimizing the fuzzy rule base and maximizing its performance by a multi-objective genetic algorithm," in *Proc. 6th IEEE Int. Conf. Fuzzy Syst.*, Barcelona, Spain, July 1–5, 1997, pp. 259–264.

Hisao Ishibuchi (M'93) received the B.S. and M.S. degrees in precision mechanics from Kyoto University, Kyoto, Japan, in 1985 and 1987, respectively, and the Ph.D. degree from Osaka Prefecture University, Osaka, Japan, in 1992.

Since 1987, he has been with Department of Industrial Engineering, Osaka Prefecture University, where he is currently a Professor. He was a Visiting Research Associate at the University of Toronto, Toronto, Ont., Canada, from August 1994 to March 1995 and from July 1997 to March 1998. His research interests include fuzzy rule-based systems, fuzzified neural networks, genetic algorithms, fuzzy scheduling, and evolutionary games.

Tomoharu Nakashima (S'99) received the B.S. and M.S. degrees from the College of Engineering, Osaka Prefecture University, Osaka, Japan, in 1995 and 1997, respectively. He is currently pursuing the Ph.D. degree at Osaka Prefecture University.

From June 1998 to September 1998, he was a Postgraduate Scholar with the Knowledge-Based Intelligent Engineering Systems Centre, University of South Australia. His current research interests include fuzzy systems, machine learning, genetic algorithms, reinforcement learning, game theory, multi-agent systems, and image processing.

Tadahiko Murata (S'96–M'98) received the B.S., M.S., and Ph. D. degrees from the Department of Industrial Engineering, Osaka Prefecture University, Osaka, Japan, in 1994, 1996 and 1997, respectively.

He is currently an Assistant Professor, Department of Industrial and Information Systems Engineering, Ashikaga Institute of Technology, Tochigi, Japan. His research interests include knowledge acquisition, multi-objective optimization, fuzzy logic, genetic algorithms, scheduling problems and pattern classification problems.