CHAPTER 5

GENETIC ALGORITHMS FOR DESIGNING FUZZY CLASSIFICATION SYSTEMS

5.1 INTRODUCTION

Fuzzy systems based on fuzzy if-then rules have been applied to various control problems [64, 102]. Fuzzy if-then rules in those fuzzy systems were usually derived from human experts. Recently several approaches have been proposed for automatically generating fuzzy if-then rules from numerical data without domain experts (see, for example, Jang [52], Sugeno & Yasukawa [103], Takagi & Sugeno [110], and Wang & Mendel [117]). Self-learning methods have been also proposed for adjusting membership functions of fuzzy sets in fuzzy rules. For example, Ichihashi & Watanabe [32] and Nomura *et al.*[85] proposed gradient descent methods for the learning of fuzzy rules. Horikawa *et al.*[29], Jang [51], Lin & Lee [67], and Takagi & Hayashi [109] proposed neural-network-based methods for the generation of fuzzy rules and their tuning.

Genetic algorithms [23,27] have been widely used for generating fuzzy if-then rules and tuning the membership functions of antecedent and consequent fuzzy sets. For example, Feldman [12], Kropp & Baitinger [61] and Thrift [113] employed genetic algorithms for generating fuzzy if-then rules. Membership functions were adjusted by genetic algorithms in Herrera *et al.*[25], Janikow [53,54], Karr [57], Karr & Gentry [58], and Surmann *et al.*[105]. Both the generation of fuzzy if-then rules and the tuning of membership functions were performed by genetic algorithms in Homaifar & McCormick [28], Kinzel *et al.*[59], Park *et al.*[92], and Satyadas & Krishnakumar [97]. The number of fuzzy if-then rules was also determined by genetic algorithms in Carse *et al.*[5], Fukuda *et al.*[18], Ishigami *et al.*[50], Lee & Takagi [65], Liska & Melsheimer [68], and Nomura *et al.*[86]. That is, both the number of fuzzy sets and the membership function of each fuzzy set were determined. Hierarchical structures of fuzzy if-then rules were determined by genetic algorithms in Matsushita *et al.*[71]

and Shimojima *et al.*[99]. In those genetic-algorithm-based methods, a rule set (*i.e.*, a rule table) of fuzzy if-then rules was coded as an individual.

The above-mentioned methods were mainly applied to fuzzy control problems such as cart centering problems [12,25,28,57,59,61,65,113], a pH control problem [58], a spacecraft attitude control problem [97], a truck backing problem [28], and a dc series motor control problem [92]. Some approaches were applied to function approximation problems [50,68,86, 93,99,105,116].

Genetics-based machine learning approaches are usually categorized into two approaches: Michigan approach and Pittsburg approach. In Michigan approach, each rule is handled as an individual called classifier. Thus this approach is referred to as classifier systems in Booker [4]. On the other hand, Pittsburgh approach [104] handles a rule set as an individual in genetic algorithms. All the above-mentioned genetic-algorithm-based methods for generating fuzzy if-then rules and tuning membership functions are categorized as Pittsburgh approach where a set of fuzzy if-then rules (*i.e.*, a fuzzy rule base) was handled as an individual in genetic algorithms. Our approach in this chapter is also a kind of Pittsburgh approach. On the other hand, a single fuzzy if-then rule was coded as an individual in fuzzy classifier systems of Furuhashi *et al.*[19], Nakaoka *et al.*[83], Parodi & Bonelli [93], and Valenzuela-Rendon [116]. Thus rule generation methods in [19,83,93,116] were referred to as fuzzy classifier systems.

While various methods have been proposed for generating fuzzy rules and adjusting membership functions, only a few approaches have dealt with classification problems. For pattern classification problems, Abe *et al.*[1,2] proposed a rule generation method and a rule tuning method where each fuzzy if-then rule was represented by a hyper-box in a multi-dimensional pattern space. Such a hyper-box was also used as a fuzzy if-then rule in fuzzy min-max neural networks [100]. Neural networks were also used as adaptive fuzzy classification systems in [24,72,73,91,94,114]. Ishibuchi *et al.*[45,46] proposed a generation method of fuzzy rules from numerical data for classification problems. Genetic-algorithm-based methods for selecting fuzzy rules were proposed in Ishibuchi *et al.*[47,48] where a small number of fuzzy rules were selected from a large number of candidate rules by genetic algorithms.

In this chapter, we introduce a genetic-algorithm-based method to the construction of a fuzzy classification system with linguistic rules. A small number of linguistic rules are selected by a genetic algorithm to construct a compact fuzzy classification system. The main advantage of the approach explained in this chapter over our former work [47,48] is the clarity of the selected

rules. That is, human decision makers can easily understand each of the selected rules because they are linguistic rules. We employ prespecified membership functions for antecedent fuzzy sets. The effectiveness of the method is illustrated by computer simulations on a numerical example and the well-known iris data (see, Fisher [13]). A hybrid method that incorporates a learning procedure [87,88] into the genetic algorithm is also constructed in order to improve the performance of fuzzy classification systems. The grade of certainty of each linguistic rule is adjusted by the learning procedure during the execution of the genetic algorithm. It is demonstrated by computer simulations on the iris data that the hybrid algorithm can find a small number of linguistic rules with high classification power.

We also introduce another genetic-algorithm-based method for adjusting the membership functions of antecedent fuzzy sets in fuzzy classification rules. Both the number of fuzzy rules and the membership function of each antecedent fuzzy set are determined simultaneously. By this method, an appropriate fuzzy partition of a pattern space is automatically generated from numerical data. The consequent class of the fuzzy rule corresponding to each fuzzy subspace is determined according to the given training patterns in that fuzzy subspace [45]. We introduce a new coding method of fuzzy partition of a pattern space. The coding method is a modified and extended version of Nomura's coding method [86] that was proposed for function approximation problems. While Nomura *et al*.[86] applied their method to approximation problems of single-input functions (*i.e.*, single-dimensional problems), we apply the method introduced in this chapter to classification problems with multiple attributes (*i.e.*, multi-dimensional problems). We also combine the error-correction learning procedure [87,88] with the genetic algorithm. High performance of our method is illustrated by computer simulations on the iris classification problem [13].

In this chapter, first we describe a rule generation procedure and a fuzzy reasoning method for pattern classification problems. Next we introduce a genetic-algorithm-based method to the construction of a fuzzy classification system with linguistic rules. Then we combine an error-correction learning procedure with the genetic algorithm. We also show another genetic-algorithm-based method for adjusting the membership functions of antecedent fuzzy sets in fuzzy classification rules. We also construct a hybrid method that incorporates the learning procedure into the genetic algorithm.

5.2 FUZZY CLASSIFICATION SYSTEMS

In this section, we describe a generation method of fuzzy classification rules from numerical data and a fuzzy reasoning method of new patterns.

5.2.1 Generation of fuzzy classification rules

Let us consider a classification problem in an n-dimensional pattern space $[0, 1]^n$. It is assumed that m patterns $\boldsymbol{x}_p = (x_{p1}, x_{p2}, \ldots, x_{pn})$, $p = 1,2,\ldots,m$, are given as training data from c classes (Class 1, Class 2, ..., Class c).

The following fuzzy rules are employed for the classification problem in $[0, 1]^n$.

$$\text{Rule } R_j: \text{If } x_{p1} \text{ is } A_{j1} \text{ and } \ldots \text{ and } x_{pn} \text{ is } A_{jn} \text{ then Class } C_j \text{ with } CF_j, \quad j = 1,2,\ldots,r, \tag{5.1}$$

where R_j is the label of the rule, A_{ji} is an antecedent fuzzy set on the i-th axis (*i.e.*, the i-th attribute), C_j is the consequent class, CF_j is the grade of certainty of this rule, and r is the total number of fuzzy rules. The consequent class C_j and the grade of certainty CF_j of each fuzzy rule can be determined from the given patterns $\boldsymbol{x}_p = (x_{p1}, x_{p2}, \ldots, x_{pn})$, $p = 1,2,\ldots,m$, as in [45]. We should specify the membership functions of the antecedent fuzzy sets before applying the rule generation procedure.

In the rule selection method [47,48], the linguistic interpretation of selected fuzzy if-then rules was not always easy because various fuzzy sets shown in Fig. 5.1 were used as antecedent fuzzy sets. In order to select fuzzy if-then rules that can be always interpreted linguistically, we restrict the antecedent fuzzy sets of candidate fuzzy if-then rules to the six linguistic values (*i.e.*, S: *small*, MS: *medium small*, M: *medium*, ML: *medium large*, L: *large*, and DC: *don't care*) in Fig. 5.2 (see, Ishibuchi *et al.*[39,40]). That is, the antecedent fuzzy set A_{ji} in (5.1) is one of the six linguistic values. Fuzzy if-then rules with linguistic values in their antecedent parts were referred to as "linguistic classification rules."

When we use the six linguistic values in Fig. 5.2 for each axis of the n-dimensional pattern space, $r = 6^n$ linguistic classification rules can be generated from the training patterns

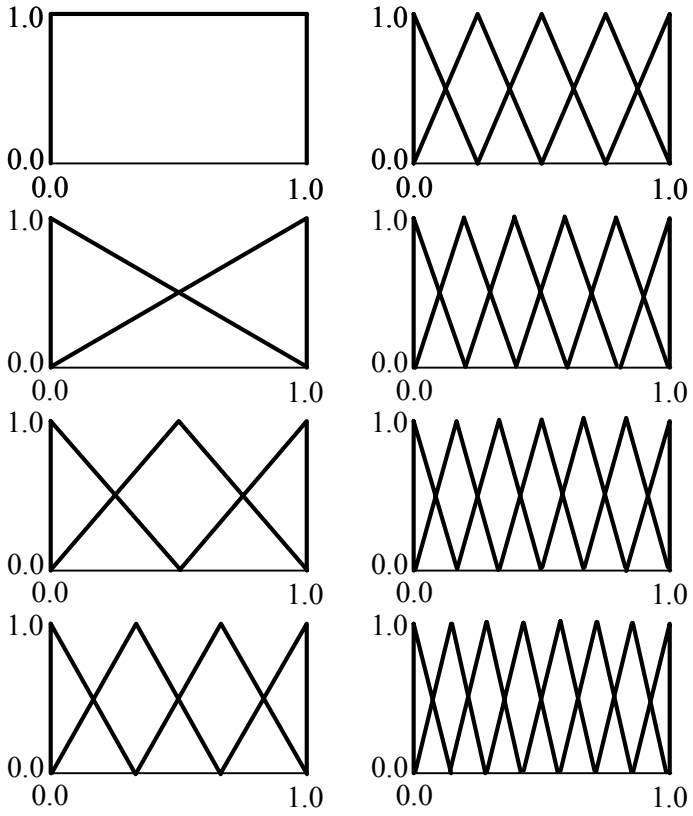

Fig. 5.1 Various antecedent fuzzy sets.

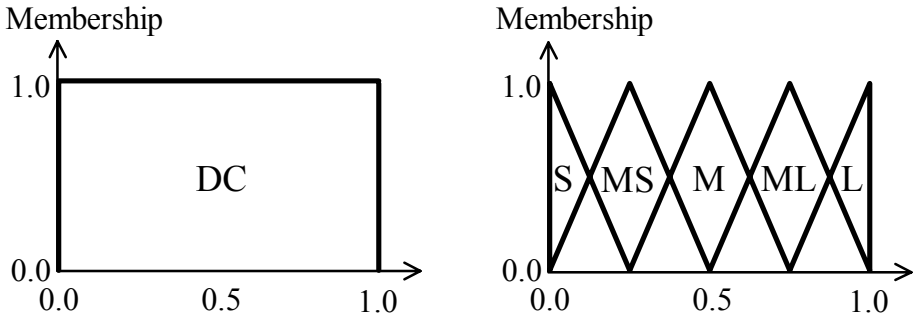

Fig. 5.2 Antecedent fuzzy sets of linguistic classification rules (DC: *don't care*, S: *small*, MS: *medium small*, M: *medium*, ML: *medium large*, and L: *Large*).

$\boldsymbol{x}_p = (x_{p1},\ldots,x_{pn})$, $p = 1,2,\ldots,m$ because each antecedent fuzzy set A_{ji} in (5.1) may assume one of the six linguistic values. For example, $6^2 = 36$ linguistic classification rules can be generated for the two-dimensional pattern space $[0,1]^2$. In this case, 36 fuzzy subspaces are

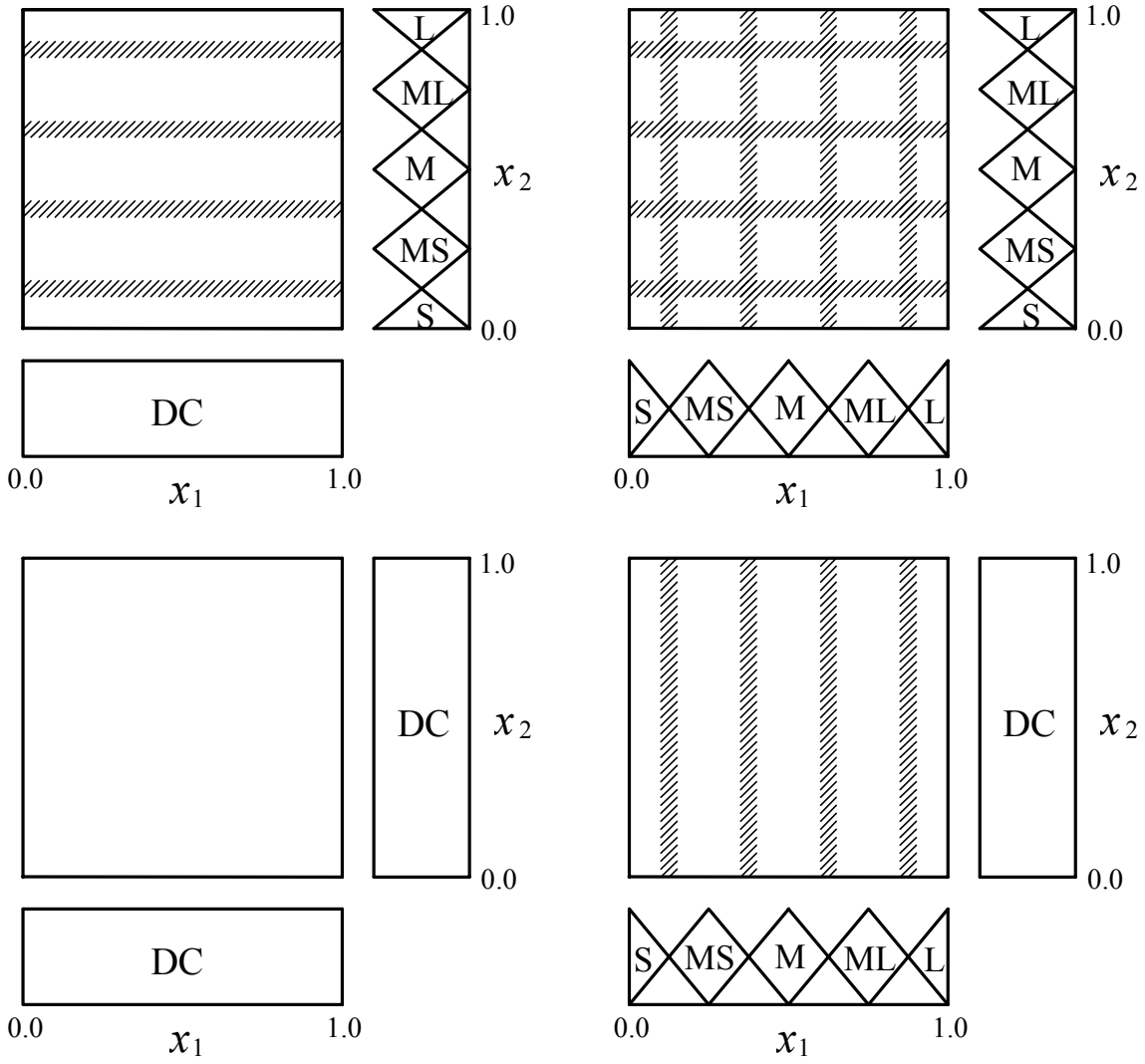

Fig. 5.3 Fuzzy partitions of two-dimensional pattern space by the six linguistic values.

generated in the pattern space $[0,1]^2$ as shown in Fig. 5.3, and a linguistic classification rule is assigned to each fuzzy subspace. From Fig. 5.3, we can see that several linguistic rules are overlapping with each other in the pattern space. This means that some of the 36 linguistic rules in Fig. 5.3 may be redundant for the classification task. Using these membership functions of antecedent fuzzy sets, we define the consequent class and the grade of certainty of the fuzzy rules.

Let us define the grade of compatibility of $\boldsymbol{x}_p$ to the fuzzy rule R_j in (5.1) as

$$\mu_j(\boldsymbol{x}_p) = \mu_{j1}(x_{p1}) \cdot \ \ldots \ \cdot \mu_{jn}(x_{pn}), \tag{5.2}$$

where $\mu_{ji}(\cdot)$ is the membership function of the antecedent fuzzy set A_{ji} which corresponds to one of six linguistic values in Fig. 5.2. When the antecedent fuzzy sets A_{ji}'s are given, the consequent class C_j and the grade of certainty CF_j can be determined as follows:

Step 1: The total grade of compatibility to the fuzzy rule R_j is calculated for each class as

$$\begin{aligned}\beta_{\text{Class } h} &= \sum_{\boldsymbol{x}_p \in \text{ Class } h} \mu_j(\boldsymbol{x}_p) \\ &= \sum_{\boldsymbol{x}_p \in \text{ Class } h} \mu_{j1}(x_{p1}) \cdot \ \ldots \ \cdot \mu_{jn}(x_{pn}), \quad h = 1,2,...,c,\end{aligned} \tag{5.3}$$

where $\beta_{\text{Class } h}$ is the total grade of compatibility of the given training patterns in Class *h* to the fuzzy rule R_j in (5.1).

Step 2: The consequent class C_j of the fuzzy rule R_j is determined as the class with the maximum total grade of compatibility. That is, C_j is determined as Class $\hat{h}$ by

$$\beta_{\text{Class } \hat{h}} = \max\{\beta_{\text{Class } 1}, \beta_{\text{Class } 2}, \ldots, \beta_{\text{Class } c}\}. \tag{5.4}$$

If Class $\hat{h}$ is not determined uniquely (*i.e.*, if two or more classes have the same maximum value in (5.4)), we assign ϕ to C_j where ϕ means an empty class. In this chapter, fuzzy rules with ϕ in the consequent part are referred to as "dummy rules" because those rules have no effect on the classification phase of new patterns.

Step 3: The grades of certainty of all dummy rules are specified as $CF_j = 0$. For non-dummy rules, the grade of certainty CF_j is determined by $\beta_{\text{Class } h}$'s as

$$CF_j = \frac{\beta_{\text{Class } \hat{h}} - \overline{\beta}}{\sum_{h=1}^{c} \beta_{\text{Class } h}}, \tag{5.5}$$

where

$$\overline{\beta} = \left(\sum_{h \neq \hat{h}} \beta_{\text{Class } h}\right) / (c-1). \tag{5.6}$$

Thus the grade of certainty is a real number in the closed interval [0, 1].

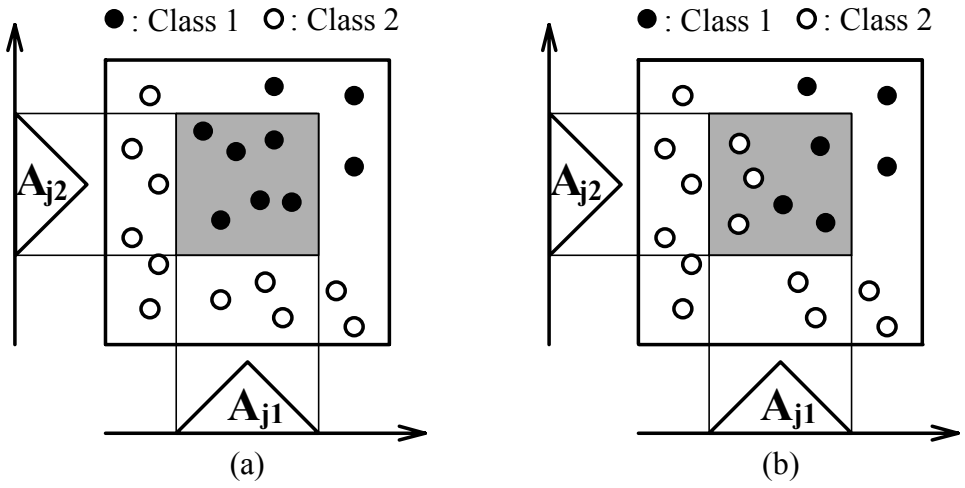

Fig. 5.4 Antecedent fuzzy sets and given patterns.

The grade of certainty CF_j is maximum (*i.e.*, $CF_j = 1$) when $\beta_{\text{Class } \hat{h}} > 0$ and $\beta_{\text{Class } h} = 0$ for $h \neq \hat{h}$. That is, if all the patterns compatible with the fuzzy rule R_j belong to the same class, the grade of certainty CF_j of this rule is equal to 1 (the maximum grade of certainty). For example, the consequent class C_j is determined as Class 1 with $CF_j = 1$ in Fig. 5.4 (a) since all the patterns in the fuzzy subspace $A_{j1} \times A_{j2}$ come from Class 1. On the contrary, when the total grades of compatibility for the *c* classes are similar to one another as shown in Fig. 5.4 (b), the grade of certainty is nearly equal to 0 (the minimum grade of certainty).

5.2.2 Fuzzy reasoning for classifying new patterns

By applying the above rule generation method to all the fuzzy rules in (5.1), we have the *r* fuzzy rules including dummy rules. Let us denote the set of fuzzy rules by *S*. A new pattern $\boldsymbol{x}_p = (x_{p1}, \ldots, x_{pn})$ is classified by the fuzzy rules in *S* as follows [45]:

Step 1: Calculate $\alpha_{\text{Class } h}$ for $h = 1, 2, \ldots, c$ as

$$\alpha_{\text{Class } h} = \max\{\mu_j(\boldsymbol{x}_p) \cdot CF_j \mid C_j = \text{Class } h \text{ and Rule } R_j \in S\}, \qquad (5.7)$$

where $\mu_j(\boldsymbol{x}_p)$ is the grade of compatibility of $\boldsymbol{x}_p$ to the fuzzy rule R_j, which is defined by (5.2). In (5.7), $\alpha_{\text{Class } h}$ is the maximum product of the compatibility and the grade of certainty of the fuzzy rules with Class *h* in the consequent part.

Step 2: Find the maximum value of $\alpha_{\text{Class } h}$'s as

$$\alpha_{\text{Class } \hat{h}} = \max\{\alpha_{\text{Class } 1}, \ldots, \alpha_{\text{Class } c}\}. \tag{5.8}$$

If two or more classes take the same maximum value in (5.8), then the classification of $\boldsymbol{x}_p$ is rejected (*i.e.*, $\boldsymbol{x}_p$ is left as an unclassifiable pattern), else assign $\boldsymbol{x}_p$ to Class $\hat{h}$ determined by (5.8).

In this procedure, a new pattern $\boldsymbol{x}_p = (x_{p1}, \ldots, x_{pn})$ is classified by the fuzzy rule that has the maximum product of the compatibility $\mu_j(\boldsymbol{x}_p)$ and the grade of certainty CF_j.

5.3 GENETIC ALGORITHMS FOR LINGUISTIC RULE SELECTION

In this section, we describe GAs for linguistic rule selection. First we explain a rule selection problem. Next we combine an error-correction learning procedure [87,88] with the genetic algorithm. The performance of the genetic-algorithm-based method is examined by applying it to two-dimensional and four-dimensional classification problems.

5.3.1 Rule selection problem

Let us denote the set of the generated *r* fuzzy classification rules by S_{ALL}:

$$S_{\mathrm{ALL}} = \{\text{Rule } R_j \mid j = 1,2,\ldots,r\}. \tag{5.9}$$

All the fuzzy classification rules in S_{ALL} are used in the rule selection problem as candidate rules.

Our rule selection problem is to select a small number of fuzzy rules from the rule set S_{ALL} to construct a compact classification system *S* with high classification performance. Therefore our problem can be written as follows:

$$\text{Maximize } NCP(S) \text{ and minimize } |S|, \tag{5.10}$$

$$\text{subject to } S \subseteq S_{\mathrm{ALL}}, \tag{5.11}$$

where $NCP(S)$ is the number of correctly classified training patterns by the fuzzy rules in the rule set *S*, and $|S|$ is the number of the fuzzy rules in *S*. Because *S* is a subset of the rule set S_{ALL}, all the generated *r* fuzzy rules are not used for the fuzzy reasoning procedure for classifying new patterns in Subsection 5.2.2.

In order to directly apply a genetic algorithm for single-objective optimization in Section 2.2, we introduce constant weights to combine the two objectives in (5.10) as follows:

$$\text{Maximize } w_{\mathrm{NCP}} \cdot NCP(S) - w_{\mathrm{S}} \cdot |S|, \tag{5.12}$$

$$\text{subject to } S \subseteq S_{\mathrm{ALL}}, \tag{5.13}$$

where w_{NCP} and w_{S} are non-negative constant weights assigned to the two objectives

$NCP(S)$ and $|S|$, respectively. In general, classification power of a classification system is more important than its compactness. Therefore the weights in (5.12) is usually specified as $0 < w_S < w_{NCP}$.

The other way to apply a genetic algorithm for single-objective optimization is to transform one of two objectives to a constraint condition. For example, if we want to maximize the number of correctly classified training patterns (*i.e.*, to maximize $NCP(S)$) using five linguistic classification rules at best, the rule selection problem can be written as follows:

$$\text{Maximize} \quad NCP(S), \tag{5.14}$$

$$\text{subject to} \quad |S| \leq 5, \tag{5.15}$$

$$S \subseteq S_{ALL}. \tag{5.16}$$

Let N_{rule} be the right-hand side constant of the constraint condition (5.15), the rule selection problem with a single objective can be rewritten as follows:

$$\text{Maximize} \quad NCP(S), \tag{5.17}$$

$$\text{subject to} \quad |S| \leq N_{rule}, \tag{5.18}$$

$$S \subseteq S_{ALL}. \tag{5.19}$$

We can also introduce a constraint condition on the number of correctly classified training patterns. Let us assume that the number of correctly classified training patterns should be larger than or equal to $N_{pattern}$. In this case, our rule selection problem can be written as

$$\text{Minimize} \quad |S|, \tag{5.20}$$

$$\text{subject to} \quad NCP(S) \geq N_{pattern}, \tag{5.21}$$

$$S \subseteq S_{ALL}. \tag{5.22}$$

In this chapter, we consider only the rule selection problem in (5.12)-(5.13) using a genetic algorithm for single-objective optimization.

5.3.2 Application of genetic algorithms

A genetic algorithm for single objective optimization is applied to the rule selection problem (5.10)-(5.11) for selecting a small number of linguistic classification rules from a large number of candidate rules in S_{ALL}. A scalar fitness value of S as in (5.12) is defined from the two objectives in (5.10) using constant weights as follows:

$$f(S) = w_{\mathrm{NCP}} \cdot NCP(S) - w_{\mathrm{S}} \cdot |S|, \tag{5.23}$$

where w_{NCP} and w_{S} are non-negative constant weights assigned to the two objectives $NCP(S)$ and $|S|$, respectively.

Each individual (*i.e.*, each rule set S) is represented by a string as $S = s_1 s_2 \ldots s_r$ where r is the number of the linguistic rules in S_{ALL} and $s_j = 1, -1$ or 0 denotes the following:

(i) $s_j = 1$ means that the j-th rule R_j is included in the rule set S,

(ii) $s_j = -1$ means that the j-th rule R_j is not included in the rule set S,

(iii) $s_j = 0$ means that the j-th rule R_j is a dummy rule.

Since dummy rules have no effect on the classification of new patterns, they should be excluded from a rule set S. Therefore the special coding ($s_j = 0$) is assigned to dummy rules in order to prevent S from including them. A string $S = s_1 s_2 \ldots s_r$ is decoded as

$$S = \{\text{Rule } R_j \mid s_j = 1;\ j = 1,2,\ldots,r\}. \tag{5.24}$$

A set of strings (*i.e.*, a set of rule sets) is treated as a population (*i.e.*, as a generation) in the genetic algorithm. While each bit position in a string has one of three values $s_j = 1, -1$ or 0, we can apply genetic operators for binary strings because we can ignore the value "0" in each string.

In order to maximize the fitness function defined by (5.23), we construct the following genetic algorithm where t is the number of generations and $t_{\max}$ is the maximum number of generations that is prespecified to terminate the algorithm.

Step 0 (Initialization): Let $t := 0$. Generate an initial population containing N_{pop} strings where N_{pop} is the number of strings in each population. Each string S is generated by assigning 0 to dummy rules and randomly assigning 1 or -1 to each of the other rules with the probability of 0.5.

Step 1 (Rule elimination and evaluation): Classify all the given training patterns by linguistic classification rules included in each string S. Exclude non-active rules from S. That is, if a linguistic classification rule in S is not used for classifying any pattern, that rule is excluded from S. This rule elimination procedure is applied to all strings in the current population. Thus, every string consists of only active rules after this rule elimination procedure. After this rule elimination procedure, each classification system S is evaluated by (5.23).

Step 2 (Selection): Let Ψ_t be the population in the t-th generation. Select $N_{\text{pop}}/2$ pairs of strings from the current population Ψ_t. The selection probability $P_{\text{s}}(S)$ of a string S in a population Ψ_t is specified as

$$P_{\text{s}}(S) = \frac{f(S) - f_{\min}(\Psi_t)}{\sum_{S' \in \Psi_t} \{f(S') - f_{\min}(\Psi_t)\}}, \tag{5.25}$$

where

$$f_{\min}(\Psi_t) = \min\{f(S') \mid S' \in \Psi_t\}. \tag{5.26}$$

Step 3 (Crossover): For each of the selected pairs, randomly choose bit positions. Each bit position is chosen with the probability of 0.5. Interchange the bit values at the chosen positions in the selected pair.

Step 4 (Mutation): To each bit of the generated strings by the crossover operator, apply the following mutation operator:

$s_j = 1 \rightarrow s_j = -1$ with the probability $P_{\text{m}}(1 \rightarrow -1)$,

$s_j = -1 \rightarrow s_j = 1$ with the probability $P_{\text{m}}(-1 \rightarrow 1)$.

Step 5 (Elitist strategy): Randomly remove one string from the N_{pop} strings newly generated by the above operations, and add the string with the maximum fitness value in the previous population to the current one.

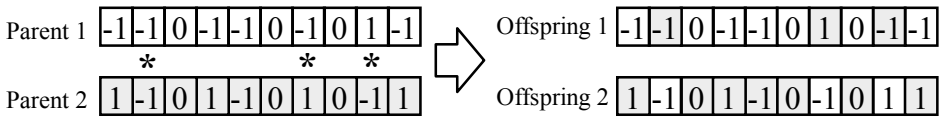

Fig. 5.5 Crossover operator for rule selection problems.

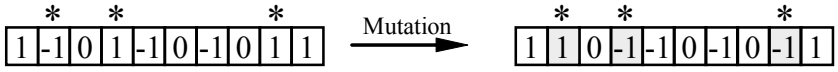

Fig. 5.6 A mutation operator for rule selection problems.

Step 6 (Termination test): Let $t := t + 1$. If $t = t_{\max}$, stop the algorithm. Otherwise, return to Step 1.

The rule elimination procedure in Step 1 is added to the genetic algorithm in our former work [47,48]. The crossover operator in Step 3 was called the uniform crossover in Syswerda [107]. We show this crossover operator in Fig. 5.5. In Step 4, different mutation probabilities $P_{\mathrm{m}}(1 \to -1)$ and $P_{\mathrm{m}}(-1 \to 1)$ are assigned to the mutations from 1 to -1 and from -1 to 1, respectively. A larger probability is usually assigned to $P_{\mathrm{m}}(1 \to -1)$ than to $P_{\mathrm{m}}(-1 \to 1)$ in order to reduce the number of linguistic classification rules in each individual. The mutation operator is illustrated in Fig. 5.6.

The genetic algorithm was applied to a two-class classification problem in a two-dimensional pattern space $[0, 1]^2$ shown in Fig. 5.7 with the following parameter specifications:

Weights in the fitness function: $w_{\mathrm{NCP}} = 10$, $w_{\mathrm{S}} = 1$,

Population size: $N_{\mathrm{pop}} = 50$,

Crossover probability: 1.00,

Mutation probabilities: $P_{\mathrm{m}}(1 \to -1) = 0.1$, $P_{\mathrm{m}}(-1 \to 1) = 0.001$,

Stopping condition: $t_{\max} = 200$ (*i.e.*, 200 generations).

First we generated $6^2 = 36$ linguistic classification rules corresponding to the fuzzy partitions in Fig. 5.3. Then the genetic algorithm was applied to the rule selection problem for selecting a small number of significant rules from the generated 36 rules. By the genetic algorithm, the following three linguistic classification rules were selected for the classification problem in Fig. 5.7:

This figure is inserted by cut-and-paste.

Fig. 5.7 An example pattern classification problem in a two-dimensional pattern space $[0, 1]^2$.

If x_1 is *don't care* and x_2 is *don't care* then Class 1 with $CF = 0.19$,

If x_1 is *don't care* and x_2 is *large* then Class 2 with $CF = 0.84$,

If x_1 is *medium* and x_2 is *don't care* then Class 2 with $CF = 0.75$.

The classification boundary obtained by the selected three linguistic classification rules is shown in Fig. 5.8. From Fig. 5.8, we can see that all the given patterns are correctly classified by the selected rules.

Since the grade of certainty of the first linguistic classification rule is very small (*i.e.*, 0.19), this rule is employed in the classification of a new pattern only when the other rules do not have large grades of compatibility to the new pattern. Therefore we have the following classification rule from the above three rules by ignoring "*don't care*" attributes.

If x_1 is *medium* or x_2 is *large* then Class 2, else Class 1.

From the configuration of the given patterns in Fig. 5.7, we can see that this classification rule agrees with our intuitive recognition of the given patterns.

We also applied the genetic algorithm to the well-known iris data (see, for example, Fisher [13]) for selecting linguistic classification rules. The iris data consist of the following four-dimensional patterns from three classes:

Class 1 (Iris setosa): $\boldsymbol{x}_p = (x_{p1}, x_{p2}, x_{p3}, x_{p4}),\ p = 1,2,\ldots,50$,

Class 2 (Iris versicolor): $\boldsymbol{x}_p = (x_{p1}, x_{p2}, x_{p3}, x_{p4}),\ p = 51,52,\ldots,100$,

Class 3 (Iris virginica): $\boldsymbol{x}_p = (x_{p1}, x_{p2}, x_{p3}, x_{p4}),\ p = 101,102,\ldots,150$,

This figure is inserted by cut-and-paste.

Fig. 5.8 Classification boundary by the selected three rules.

where x_{p1} is the sepal length, x_{p2} is the sepal width, x_{p3} is the petal length, and x_{p4} is the petal width. In computer simulations of this section, all the attribute values were normalized into real numbers in the unit interval $[0,1]$ as

$$x_{pi} := (x_{pi} - \min\{x_{pi}\}) / (\max\{x_{pi}\} - \min\{x_{pi}\}), \quad p = 1,2,\ldots,150; \; i = 1,2,3,4, \quad (5.27)$$

where

$$\min\{x_{pi}\} = \min\{x_{pi} \mid p = 1,2,\ldots,150\}, \quad i = 1,2,3,4, \quad (5.28)$$

$$\max\{x_{pi}\} = \max\{x_{pi} \mid p = 1,2,\ldots,150\}, \quad i = 1,2,3,4. \quad (5.29)$$

Therefore the iris data were transformed into a three-class classification problem in the four-dimensional unit cube $[0,1]^4$.

Since the iris data have four attributes, $6^4 = 1296$ linguistic classification rules were generated as candidate rules. Thus our rule selection problem is to find a compact rule set from the 1296 rules. The total number of possible rule sets is $2^{1296} \cong 1.36 \times 10^{390}$.

By the genetic algorithm with the same parameter specifications as in the above computer simulation, five linguistic rules in Fig. 5.9 were selected. The last column (# of patterns) in Fig. 5.9 shows the number of training patterns that were correctly classified by each rule. Therefore we can see that 147 patterns (98% of the given 150 patterns) are correctly classified by the selected five rules. By ignoring "*don't care*" attributes denoted by rectangles in Fig. 5.9, we have the following linguistic classification rules from the selected rules:

No.	x_1	x_2	x_3	x_4	Class	CF	# of patterns
1					1	1.00	50
2					2	0.95	48
3					3	0.78	28
4					3	1.00	14
5					3	1.00	7

Fig. 5.9 Selected linguistic classification rules for the iris data.

If x_3 is *small* then Class 1 with *CF*=1.00,

If x_3 is *medium* and x_4 is *medium* then Class 2 with *CF*=0.95,

If x_2 is *medium small* and x_4 is *medium large* then Class 3 with *CF*=0.78,

If x_1 is *medium* and x_2 is *medium* and x_4 is large then Class 3 with *CF*=1.00,

If x_1 is *large* and x_2 is *medium* then Class 3 with *CF*=1.00.

5.3.3 Extension to hybrid genetic algorithm

In the rule selection problem, we generated candidate linguistic rules by the rule generation procedure in Subsection 5.2.1 where the grade of certainty of each rule was determined with no tuning procedure. In this subsection, first we briefly describe how the grade of certainty of each rule can be adjusted to improve the performance of a fuzzy-rule-based classification system. Then we introduce a hybrid algorithm that incorporates a learning procedure of the grade of certainty [87,88] into our genetic algorithm. It is shown by computer simulations on the iris data that a small number of linguistic rules with high classification power are selected by the hybrid genetic algorithm.

A. Adjustment of the grade of certainty

From the fuzzy reasoning procedure for classifying a pattern $\boldsymbol{x}_p = (x_{p1}, \ldots, x_{pn})$ in Subsection 5.2.2, we can see that $\boldsymbol{x}_p$ is classified by a linguistic classification rule $R_{\hat{j}}$ that

satisfies the following relation:

$$\mu_{\hat{j}}(\boldsymbol{x}_p) \cdot CF_{\hat{j}} = \max\{\mu_j(\boldsymbol{x}_p) \cdot CF_j \mid \text{Rule } R_j \in S\}. \tag{5.30}$$

If the consequent class $C_{\hat{j}}$ of this rule is the same as the actual class of $\boldsymbol{x}_p$, $\boldsymbol{x}_p$ is correctly classified, otherwise $\boldsymbol{x}_p$ is misclassified.

When $\boldsymbol{x}_p$ is correctly classified by the linguistic classification rule $R_{\hat{j}}$, the grade of certainty $CF_{\hat{j}}$ of this rule is increased as the reward of the correct classification [87,88]:

$$CF_{\hat{j}}^{\text{new}} = CF_{\hat{j}}^{\text{old}} + \eta_1 \cdot (1 - CF_{\hat{j}}^{\text{old}}), \tag{5.31}$$

where η_1 is a positive learning constant for increasing the grade of certainty. On the contrary, when $\boldsymbol{x}_p$ is misclassified by the linguistic classification rule $R_{\hat{j}}$, the grade of certainty $CF_{\hat{j}}$ of this rule is decreased as the punishment of the misclassification [87,88]:

$$CF_{\hat{j}}^{\text{new}} = CF_{\hat{j}}^{\text{old}} - \eta_2 \cdot CF_{\hat{j}}^{\text{old}}, \tag{5.32}$$

where η_2 is a positive learning constant for decreasing the grade of certainty.

In this procedure, the grade of certainty of each linguistic rule is always in the unit interval $[0,1]$ if the positive learning constants η_1 and η_2 are less than unity (*i.e.*, $0 < \eta_1 < 1$ and $0 < \eta_2 < 1$). Since there are usually much more correctly classified patterns than misclassified patterns, a larger value is assigned to η_2 than η_1. Therefore the learning constants should satisfy the inequality $0 < \eta_1 < \eta_2 < 1$. In computer simulations of this section, we specified η_1 and η_2 as $\eta_1 = 0.001$ and $\eta_2 = 0.1$.

B. Hybrid genetic algorithm

The learning procedure of the grade of certainty CF_j is combined with the genetic algorithm. Since the learning procedure is applicable to any rule set *S*, we apply it to all the rule sets (*i.e.*, all the strings) generated by the crossover and mutation operators in the genetic algorithm. That is, the following procedure is inserted between Step 5 and Step 6 of the genetic algorithm described in Subsection 5.3.3:

[Learning procedure of the grade of certainty]

Step 5.5 (Learning): Apply the learning procedure to each rule set S generated by the crossover and mutation operators. The learning procedure for each rule set S is iterated N_{learning} times for all the training patterns where N_{learning} is the number of iteration of this learning procedure.

C. Simulation results

Using the same parameter specifications as in Subsection 5.3.3, we applied the hybrid genetic algorithm to the iris data. We specified N_{learning} as $N_{\text{learning}} = 20$. This means that the learning procedure was iterated 20 times for each of the generated strings in the hybrid genetic algorithm.

After 200 generations, we obtained seven linguistic rules that can correctly classify all the 150 patterns (classification rate: 100%). The selected rules are shown in Fig. 5.10. In order to examine the average performance of the hybrid genetic algorithm, the same computer simulation with a different initial population was iterated ten times. We specified N_{learning} as $N_{\text{learning}} = 0, 5, 10, 20$. When $N_{\text{learning}} = 0$, the hybrid genetic algorithm is the same as the genetic algorithm with no learning procedure. Table 5.1 shows average results over ten trials. From the comparison of these results, linguistic rules selected by the hybrid algorithm have higher classification power than those by the non-hybrid genetic algorithm.

No.	x_1	x_2	x_3	x_4	Class	CF	# of patterns
1					1	1.00	50
2					2	0.30	20
3					2	0.15	4
4					2	0.88	26
5					3	1.00	26
6					3	0.09	17
7					3	1.00	7

Fig. 5.10 Selected linguistic classification rules for the iris data.

Table 5.1 Average results of the hybrid genetic algorithm over ten trials.

N_{learning}	0	5	10	20
$\|S\|$	7.0	9.3	9.4	12.0
$NCP(S)$	146.9	149.0	148.9	149.6
Classification rate	97.9%	99.3%	99.3%	99.7%