5.4 GENETIC ALGORITHMS FOR FUZZY PARTITION OF PATTERN SPACE

In this section, we also introduce another genetic-algorithm-based method for adjusting the membership functions of antecedent fuzzy sets in fuzzy classification rules [36,37]. 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 for fuzzy partitions 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. The differences between these two coding methods are as follows:

(i) The types of fuzzy rules are different. We use fuzzy rules with class labels in the consequent part for pattern classification problems while Nomura *et al.*[86] employed simplified fuzzy rules with real numbers in the consequent part for function approximation problems.

(ii) Our coding method can represent a "*don't care attribute*" by utilizing the whole domain of that attribute as an antecedent fuzzy set. The "*don't care attribute*" is represented by an interval with the full membership value in the whole domain.

(iii) Our coding method uses not only triangular membership functions but also trapezoidal membership functions.

(iv) We use different mutation probabilities for increasing and decreasing the number of membership functions. The probability of adding a new membership function is less than that of eliminating existing one.

(v) We introduce a mutation operator for fine tuning of membership functions. The mutation slightly modifies the shape of a membership function by exchanging adjacent two bit values.

The aim of the modifications in (ii) ~ (iv) is to reduce the number of membership functions for classification problems. On the other hand, the aim of introducing the new mutation in (v) is to improve the classification performance of fuzzy rules. 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 section 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 as in the previous section. High performance of our method is illustrated by computer simulations on the iris classification problem [13].

5.4.1 Coding of fuzzy partition

Nomura *et al*.[86] proposed the determination method of the fuzzy partition of an input space. There is a restriction on their definition of membership functions. Therefore we modify their definition.

Let $\mu_{ji}(x_i)$ be the membership function for the fuzzy set A_{ji} in the fuzzy rule R_j in (5.1) ($j = 1,2,...,r$, where r is the total number of generated fuzzy rules). Fig. 5.11 shows membership functions defined by Nomura's method [86]. In Fig. 5.11, all the membership functions are triangular. The width of each membership function is defined by the length between the centers of neighboring membership functions. The arrangement of membership functions can be expressed in terms of a string $L_i = l_1 l_2 \cdots l_{length_i}$ consisting of "0" and "1" where $length_i$ means the prespecified length of the string L_i ($i = 1,2,...,n$). In the string L_i, $l_k = 1$ means that the *k*-th position in the string L_i is the center position of a membership function ($k = 1,2,...,length_i$). A string of $length_i = 10$ is shown in Fig. 5.11 where Nomura's coding method is used. In Nomura's coding method, there is the restriction that there must be the value "1" at both edge positions of the string L_i as shown in Fig. 5.11. Let r_i^{set} be the number of fuzzy sets on the *i*-th axis. This means that r_i^{set} is the number of 1's in the string L_i. Therefore the restriction in Nomura's coding method leads to $r_i^{set} \geq 2$.

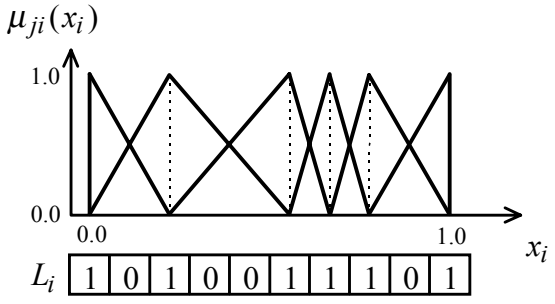

Fig. 5.11 Membership functions defined by Nomura's method.

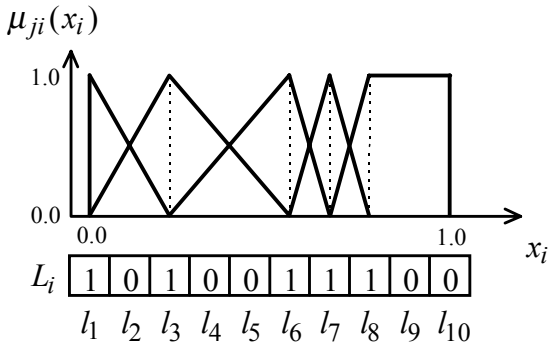

Fig. 5.12 Membership functions defined by the extended coding method.

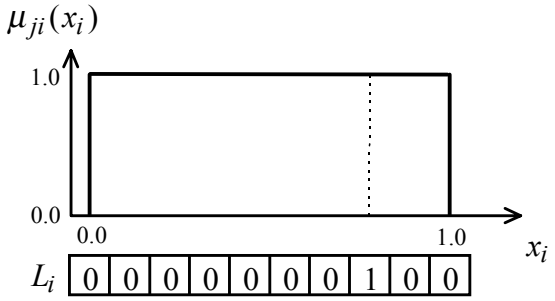

Fig. 5.13 Special membership function defined by L_i where $r_i^{set} = 1$.

In this section, we relax this restriction on the values at both edge positions of the string L_i. That is, there can be "0" at both the edge positions. Fig. 5.12 shows example membership functions $\mu_{ji}(x_i)$ defined by the extended coding method. In Fig. 5.12, the shape of the membership function on the right edge of this axis is trapezoidal since the value of l_{10} is 0.

When the number of 1's in the string L_i is less than two, we specially define just one membership function for the *i*-th axis as follows:

$$\mu_{ji}(x_i) = \begin{cases} 1, \text{ if } 0 \le x_i \le 1, \\ 0, \text{ otherwise.} \end{cases} \tag{5.33}$$

The shape of this fuzzy set is illustrated in Fig. 5.13. When this membership function is

assigned for a certain attribute, we can ignore that attribute because the membership value is always equal to 1 for all its possible attribute values.

We use the fuzzy sets defined by the string L_i as the antecedent fuzzy sets A_{ji}'s for the *i*-th attribute. Since there are r_i^{set} fuzzy sets for each axis of the *n*-dimensional pattern space $[0, 1]^n$, we can generate $r_1^{set} \cdot r_2^{set} \cdot \ldots \cdot r_n^{set}$ fuzzy rules. Therefore the total number *r* of the fuzzy rules in (5.1) is $r = r_1^{set} \cdot r_2^{set} \cdot \ldots \cdot r_n^{set}$. The fuzzy classification system that has the $r_1^{set} \cdot r_2^{set} \cdot \ldots \cdot r_n^{set}$ fuzzy rules is denoted by a combined string as $S = L_1 L_2 \cdots L_n$. In this section, the combined string *S* is treated as an individual in the genetic algorithm. Therefore, an individual *S* denotes a rule set (*i.e.*, a classification system) for pattern classification problems. Hereafter, *S* and L_i are called a string and a substring, respectively. It should be noted that the consequent class C_j and the grade of certainty CF_j of the fuzzy rule R_j in (5.1) corresponding to each fuzzy subspace are determined according to the given training patterns in that fuzzy subspace using the rule generation method in Subsection 5.2.1. Therefore the combined string *S* does not include the consequent part of each fuzzy rule.

5.4.2 Genetic algorithm for adjusting membership functions

Our problem is to find a compact fuzzy classification system *S* with high classification power. Therefore our problem has the same objectives as in (5.10) and (5.11): (i) to maximize the number of correctly classified training patterns by the fuzzy rules in *S*, and (ii) to minimize the number of the fuzzy rules in *S*. By combining these two objectives, the following problem is formulated in the same manner as in (5.12):

$$\text{Maximize} \quad w_{\text{NCP}} \cdot NCP(S) - w_{\text{S}} \cdot |S|, \tag{5.34}$$

where w_{NCP} and w_{S} are non-negative constant weights assigned to the two objectives $NCP(S)$ and $|S|$, respectively. As we have already explained in the last subsection, $|S|$ is calculated from the combined string $S = L_1 L_2 \ldots L_n$ as $|S| = r_1^{set} \cdot r_2^{set} \cdot \ldots \cdot r_n^{set}$ where r_i^{set} is the number of fuzzy sets specified by L_i. A similar problem was formulated for function approximations in Fukuda *et al*.[18].

The value of the objective function in (5.34) is used as the fitness value of each individual.

That is, the fitness function $f(S)$ is defined as

$$f(S) = w_{\mathrm{NCP}} \cdot NCP(S) - w_{\mathrm{S}} \cdot |S| . \tag{5.35}$$

In order to maximize the fitness function defined by (5.35), 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 (*i.e.*, N_{pop} individuals) where N_{pop} is the number of strings in each population. In this operation, each string S is generated by assigning "1" with the probability P_1 and "0" with the probability P_0 to each bit in S where $P_1 + P_0 = 1$. The fitness value of each string S is calculated by (5.35).

Step 1 (Selection): Let Ψ_t be the population in the t-th generation. Select $N_{\mathrm{pop}}/2$ pairs of strings from the current population Ψ_t. The selection probability $P_{\mathrm{s}}(S)$ of a string S in the population Ψ_t is specified as

$$P_{\mathrm{s}}(S) = \frac{f(S) - f_{\min}(\Psi_t)}{\sum_{S' \in \Psi_t} \{f(S') - f_{\min}(\Psi_t)\}}, \tag{5.36}$$

where

$$f_{\min}(\Psi_t) = \min\{f(S') \mid S' \in \Psi_t\} . \tag{5.37}$$

Step 2 (Crossover): For each of the selected pairs, randomly choose substrings L_i's from one string. Each substring L_i is chosen with the probability of 0.5. Interchange the substring L_i between the selected pair. Fig. 5.14 shows an example of this crossover operator.

Step 3 (Mutation): To each bit value of the generated strings by the crossover operator, apply the following two mutation operations:

Step 3.1 : Exchange adjacent bit values in L_i with the probability P_{swap}.

Step 3.2 : Reverse the value of each bit with the probabilities $P_{\mathrm{reverse}}(1 \rightarrow 0)$ and

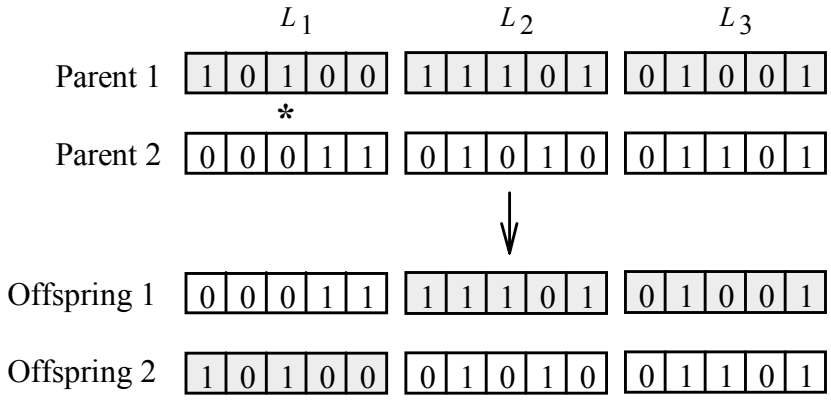

Fig. 5.14 Crossover operator.

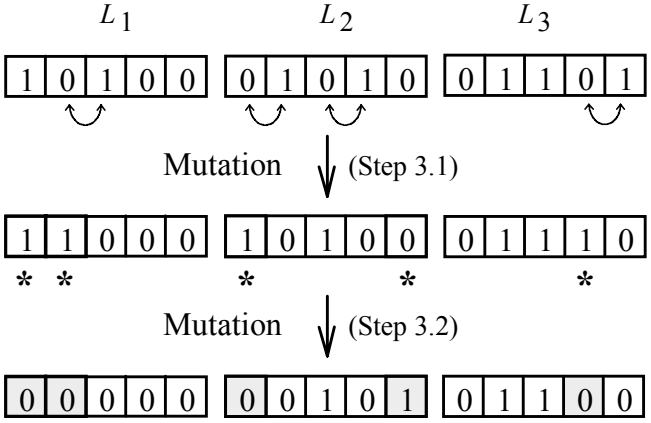

Fig. 5.15 Mutation operators.

$P_{\text{reverse}}(0 \to 1)$:

$l_k = 1 \ \to \ l_k = 0$ with the probability $P_{\text{reverse}}(1 \to 0)$,

$l_k = 0 \ \to \ l_k = 1$ with the probability $P_{\text{reverse}}(0 \to 1)$.

These mutation operators are illustrated in Fig. 5.15.

Step 4 (Elitist strategy): Randomly remove one string from the N_{pop} strings generated by the above genetic operations, and add the string with the maximum fitness value in the previous population to the current one.

Step 5 (Termination test): Let $t := t + 1$. If $t = t_{\max}$, stop the algorithm. Otherwise, return to Step 1.

The characteristics of the proposed genetic algorithm are as follows.

(i) The crossover operator in Step 2 interchanges substrings L_i's between the selected pair of strings. This means that our crossover is a multi-point crossover with crossover points between substrings. Because crossover points are not chosen within substrings, the membership functions of antecedent fuzzy sets are not destroyed by the crossover operator. It is possible to implement an alternative crossover operation by randomly selecting crossover points within substrings as well as between substrings. In this case, the membership functions are destroyed by the crossover operator.

(ii) The mutation operation in Step 3.1 is for fine tuning of membership functions. The shape of a membership function is slightly modified by this mutation.

(iii) The mutation operation in Step 3.2 is to generate a new membership function or to eliminate existing one. By assigning a larger value to the mutation from $l_k = 1$ to $l_k = 0$ (*i.e.*, $P_{\text{reverse}}(1 \to 0) > P_{\text{reverse}}(0 \to 1)$), we can reduce the number of fuzzy rules in *S*.

5.4.3 Computer simulations

In this section, we apply the genetic algorithms with the extended coding method to a two-dimensional pattern classification problem and the well-known iris data [13] in the four-dimensional pattern space.

A. A two-dimensional pattern classification problem

First let us consider the two-class classification problem in Fig. 5.16. For this classification problem, a fine fuzzy partition is required for the left half of the pattern space but a coarse fuzzy partition is appropriate for the right half. The genetic algorithm with the extended coding method described in the last section was applied to the classification problem with the following parameter specifications.

Length of each substring L_i of the string $t = t_{\max}$: $length_i = 21$,

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

Population size: $N_{\text{pop}} = 10$,

Assigning probabilities of "1" and "0" in the initial population: $P_1 = 0.5$, $P_0 = 0.5$,

This figure is inserted by cut-and-paste.

Fig. 5.16 A two-class classification problem in the two-dimensional pattern space $[0, 1]^2$.

Crossover probability: 1.00,

Mutation probabilities: $P_{\text{swap}} = 0.1$, $P_{\text{reverse}}(1 \rightarrow 0) = 0.1$, and $P_{\text{reverse}}(0 \rightarrow 1) = 0.05$,

Stopping condition: $t_{\max} = 500$ (*i.e.*, 500 generations).

For examining the effect of the coding restriction of $l_1 = 1$ and $l_{length_i} = 1$, we also applied the genetic algorithm with Nomura's coding method. Fig. 5.17 (a) shows the fuzzy partition and the classification boundary between two classes obtained by the genetic algorithms with Nomura's coding method. Fig. 5.17 (b) shows those obtained by the extended coding method. The hatched areas, the dotted areas, and the painted areas in Fig. 5.17 indicate that the consequents of the corresponding fuzzy classification rules are Class 1, Class 2 and ϕ, respectively. All the given patterns were correctly classified by each method in Fig. 5.17. The genetic algorithm with Nomura's coding method generated sixteen fuzzy rules, while the algorithm with the extended coding method found eight fuzzy rules. From Fig. 5.17 (b), we observe that the extension to trapezoidal membership functions reduced the number of fuzzy rules effectively. The genetic algorithm with the extended coding method could find the appropriate fuzzy partition for this problem.

B. Iris classification problem

We also applied the genetic algorithm with the extended coding method to the well-known iris data [13]. The same preprocessing procedure in Subsection 5.3.3 was applied to the data. In the coding method, the length of each substring L_i (*i.e.*, [illegible]) should be prespecified. When

These figures are inserted by cut-and-paste.

(a) Nomura's coding ($r_i^{set} \geq 2$) (b) The extended coding ($r_i^{set} \geq 1$)

Fig. 5.17 The fuzzy partitions and classification boundaries.

a large value is used for $length_i$, the genetic algorithm can adjust membership functions elaborately, but its execution requires long computation time. On the contrary, if a small value is used for $length_i$, the computation time can be reduced but the tuning of membership functions is not fine. In order to examine the effect of the specifications of $length_i$ on the performance of the genetic algorithm, we employed the following five values of $length_i$ as follows:

Length of each substring L_i of the string $S = L_1 L_2 L_3 L_4$: $length_i = 6, 11, 16, 21, 31$.

In order to generate almost the same number of initial fuzzy rules in these five trials, we specified the assigning probabilities of "1" and "0" in the initial population as $P_1 = 4 / length_i$ and $P_0 = 1 - P_1$. This means that each substring has four positions with "1" on the average. Therefore, each string S has 4^4 fuzzy rules on the average in the initial population.

In the same manner as in the last subsection, we applied the genetic algorithm to the iris classification problem. We employed the same parameter specifications of the genetic algorithm except for the length of each L_i. All the 150 patterns of the iris classification problem were used for constructing a fuzzy-rule-based classification system by the genetic algorithm. Table 5.2 shows the classification rate, the number of fuzzy rules and CPU time. CPU time in the fourth column were required for the genetic algorithm with each value of

$length_i$. The classification rate was calculated for all the training patterns. From Table 5.2, we can see that large values of $length_i$ deteriorated the ability of the genetic algorithm to generate good fuzzy classification rules. This deterioration may be explained as follows. Because the iris classification problem is a four-dimensional problem, the length of each string S is $4 \times length_i$. Therefore the total number of possible solutions (*i.e.*, possible combinations of "0" and "1" for S) is $2^{4 \times length_i}$. When $length_i$ increases, $2^{4 \times length_i}$ exponentially increases. This exponential increase of the search space may deteriorate the ability of the genetic algorithm to generate good fuzzy classification rules. Of course, if we implement our genetic algorithm using a large number of population (*e.g.*, 1000 strings) and a large number of generations (*e.g.*, 10,000 generations), we may have a elaborated fuzzy partition by a large number of $length_i$.

In order to find a rule set with high classification power, we performed another computer simulation with different weights in the fitness function and a different stopping condition. We defined the weights as $w_{\mathrm{NCP}} = 1000$, $w_{\mathrm{S}} = 1$, and $t_{\max} = 1000$. The length of each substring L_i was specified as $length_i = 21$. By this computer simulation, we got 56 fuzzy rules that correctly classified all the 150 training patterns. Fig. 5.18 shows the membership functions for each axis obtained by the genetic algorithm. We can see from Fig. 5.18 that the attribute x_1 can be negligible.

5.4.4 Hybrid genetic algorithm

The learning procedure of the grade of certainty CF_j [87,88] is combined with our genetic algorithm in the same manner as in Subsection 5.3.3. 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 4 and Step 5 of the genetic algorithm described in Subsection 5.4.2:

Table 5.2 Performance of genetic algorithms with various $length_i$'s.

$length_i$	Classification rate (%)	Number of rules in S	CPU time (sec.)
6	97.3	6	278.2
11	98.0	12	755.9
16	97.3	9	1776.9
21	97.3	18	5524.3
31	96.0	40	17458.0

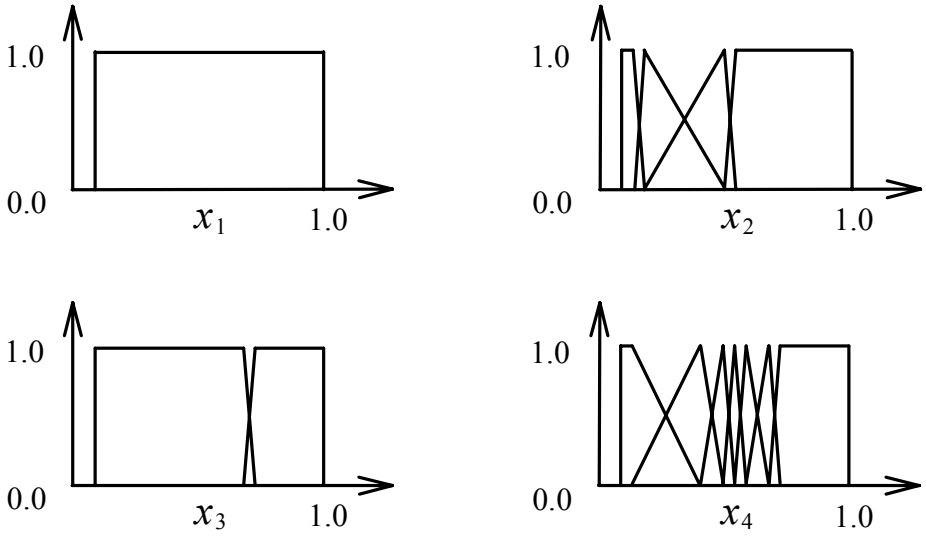

Fig. 5.18 Membership functions for each attribute.

Table 5.3 Performance of the hybrid genetic algorithm with various $length_i$'s.

$length_i$	Classification rate (%)	Number of rules in S	CPU time (sec.)
6	100.0	24	2649.0
11	98.7	12	6588.2

[Learning procedure of the grade of certainty]

Step 4.5 (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.

The hybrid algorithm was applied to the iris data using the same parameter specifications as in the genetic algorithm in Subsection 5.4.3 except for the length of each substring L_i (*i.e.*, $length_i$). We employed the hybrid genetic algorithms with $length_i = 6,11$. Table 5.3 shows the results of computer simulations with $w_{\text{NCP}} = 10$, $w_{\text{S}} = 1$, and $N_{\text{learning}} = 10$. The learning algorithm improved the classification rate of the generated fuzzy classification rules while it required much longer computation time. Because of heavy computation load of the hybrid algorithm, large values of $length_i$ are inappropriate.

5.5 SUMMARY

In this chapter, we introduced two genetic-algorithm-based approaches for constructing fuzzy classification systems. We first applied a genetic-algorithm-based method to the construction of a fuzzy classification system with linguistic rules. The method was illustrated by computer simulations on a numerical example and the well-known iris data. A hybrid approach that incorporates a learning procedure [87,88] into the genetic algorithm was also designed in order to improve the performance of fuzzy classification systems. It was 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 applied 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 were determined simultaneously by this method. 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 introduced a new coding method of fuzzy partitions of a pattern space. The new coding method was a modified and extended version of Nomura's coding method [86] that was proposed for function approximation problems. We also combined the error-correction learning procedure [87,88] with the genetic algorithm. High performance of our method was illustrated by computer simulations on the iris classification problem [13].