Fuzzy Classification Using Hierarchical Genetic Algorithm with Multiple Rule Gene Tables

Wen-Gong Chen and Shie-Jue Lee
Department of Electrical Engineering
National Sun Yat-Sen University
Kaohsiung 804, Taiwan
E-mail: wgc@water.ee.nsysu.edu.tw

Abstract

Building a rule-based classification system from a training data set is an important research topic in the area of data mining, knowledge discovery, and expert systems. In this paper, a new method, using hierarchical genetic algorithms with multiple rule gene tables, is presented. By using multiple rule gene tables, a higher recognition rate can be achieved.

Keywords: Hierarchical genetic algorithm, rule gene tables, fuzzy rules, data mining.

1 Introduction

Knowledge engineering plays a very important role in expert systems. It involves knowledge discovery, knowledge representation, and human-machine interaction. Knowledge discovery can be supplied by experts but it is usually difficult. The better way for knowledge discovery is by extracting knowledge from data directly. However, there are much uncertainty and vagueness exists in the real world data. Fuzzy set theory proposed by L. A. Zadeh [1] can deal with the uncertainty and vagueness of real world data. Furthermore, the fuzzy systems are capable of handling complex, nonlinear, and mathematically intangible dynamic system using simple solutions. However, it is difficult to determine the shape of membership functions and fuzzy rules for more complex systems. Therefore, to obtain optimal membership functions and fuzzy rules is a difficult task.

Many approaches have been developed to extract knowledge from data directly. Some of them are based on neural networks and fuzzy set theory [2, 3, 4]. In [4], Wu and Chen proposed a fuzzy learning algorithm based on the α-cuts of fuzzy sets to divide numerical data into different partitions and to automatically derive membership functions for each partition. However, there are two problems exist in their method. One is that the cost of time and space for the matrix operation becomes high when the number of training data is large, and the other one is that the recognition rate for special data structure,such as spiral data, is too low.

Genetic Algorithm(GA) have been proved as a powerful method for solving the optimization problems for complex solution space [5, 6]. GA are stochastic search techniques based on the mechanism of natural selection and genetic operations. It simulates the evolutionary process in nature by creating a population of individuals represented by chromosomes. Therefore, the GA can perform machine learning well [7]. In recent years, many researchers proposed hybrid systems by combining fuzzy theory, neural networks, and genetic algorithms to effectively solve fuzzy classification problems [8, 9].

In this paper, we propose a new method for designing an efficient fuzzy classifier using the hierarchical genetic algorithm(HGA) with multiple rule gene tables. A precise hierarchical genetic structure is formed for the purpose of optimizing the fuzzy membership functions and rules. A similar idea has been proposed in [10] for fuzzy rule optimization and applied to fuzzy controller by [11]. In order to improve the classification performance, we use a rule gene table for each class in the hierarchical genetic algorithms.

The rest of this paper is organized as follows: The overview of HGA is introduced in Section 2. Our fuzzy classifier is described in Section 3. The algorithm of our method is described in Section 4. The simulation and experimental results are presented in Section 5. Finally, concluding remarks are given in Section 6.

2 Overview of HGA

Compared with conventional GA, the structure of HGA is hierarchical [10, 11, 12]. A chromosome consists of two types of genes—the control genes and the parametric genes. The control genes are used for pattern features selection while the parametric genes are usually used to represent the parameters of membership functions of fuzzy

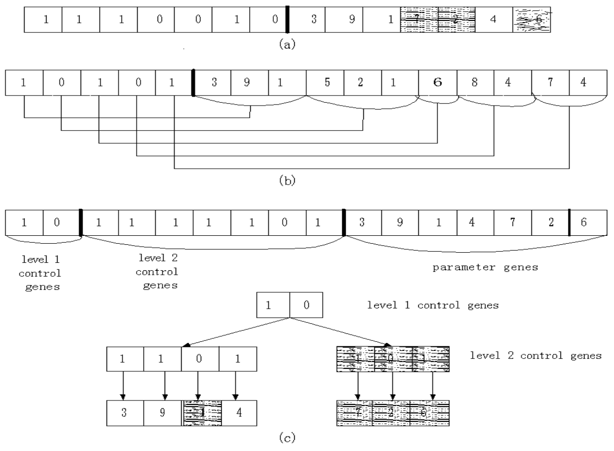


Figure 1: A hierarchical chromsome structure of HGA.

rules. Generally, the control genes are coded as binary digits, while the parametric genes are coded as binary digits, real numbers, or symbols. If the value of a control gene is 1, then the associated parametric gene(s) are enabled, otherwise the associated parametric gene(s) are disabled. By means of hierarchical structure, not only the basic genetic computations are maintained, but also the flexibility, the robustness, and the complexity of parametric modeling are improved. A hierarchical chromosome structure is depicted in Figure 1. Figure 1-(a) and 1-(b) show a two-level gene structure in which the left part is control genes, while the right part is parametric genes. Each control gene is mapped to its corresponding parametric gene(s). In figure 1-(a), each control gene corresponds to one parametric gene, while in figure 1-(b), each control gene corresponds to one set of parametric gene(s)—singular gene or plural genes for each set. In figure 1-(a), we can see that the genes 3, 9, 1, and 4 are enabled, and genes 7, 2, and 6 are disabled. In figure 1-(b) the gene sets (3, 9, 1), (6) and (7,4) are enabled, and gene sets (5, 2, 1) and (8,4) are disabled. Figure 1-(c) is a three-level gene structure, which contains two levels of control genes. The upper level genes can turn on or off the lower level genes. Thus, the genes 3, 9, and 4 are enabled, and genes 1, 7, 2, and 6 are all disabled.

3 Our Fuzzy Classifier

In this paper, a new method for designing an efficient fuzzy classifier using the hierarchical genetic algorithm with multiple rule gene tables is proposed. Firstly, we use a feasible fuzzy partition to construct a population of chromosomes on the feature space and create a rule gene table randomly for each class. Then, the population is evolved by the HGA operations with multiple rule gene tables. Finally, the optimal chromosome and rule gene tables are generated and used to classify test data. The detail is described as follows.

3.1 Chromosome Representation

As mentioned earlier, a chromosome consists of two types of genes,i.e.,control genes and parametric genes. The control genes are coded as binary numbers while the parametric genes are coded as real numbers. The number of control genes is decided by the total number of the fuzzy membership functions of all feature dimensions and the number of output classes. Further, the number of parametric genes is decided by the parameters of membership functions for all fuzzy membership functions. For example, there are four feature dimensions in the iris data set, i.e.,sepal length, sepal width, petal length, and petal width. If we define seven fuzzy triangular membership functions for each feature such as specially short, very short, short, medium, long, very long, and specially long. Note that each membership function is defined with 3 parameters. Therefore, there are 21 parametric genes needed for each feature. Totally, a chromosome for iris data consists of 93 parametric genes. Each value of control genes is randomly generated from 0,1 while each value of parametric genes is flexibly determined as shown in subsection 3.2.

3.2 Flexible Fuzzy Subsets

Suppose x is in the range of $[V_l,V_r]$, a flexible triangular fuzzy membership function shown in figure 2 is defined as the following equation.

$$\mu(x) = \begin{cases} 0 & \text{if } x \le a \text{ or } x \ge c \\ \frac{b-x}{b-a} & \text{if } x < b \\ \frac{c-x}{c-b} & \text{if } x > b \end{cases}$$

where a and c satisfy $|a-c| \cong (V_l - V_r)/N$ and $V_l \le a \le b \le c \le V_r$, b is randomly decided between a and c, N is the number of fuzzy membership functions of each feature space. Note that the variables a, b, and c determine the shape of membership function.

3.3 Multiple Rule Gene Tables

In order to increase the performance of classification, we create several rule gene tables. The number of dimensions of a rule gene table is decided by the number of input features. Every entry of each table is initialized with a random integer representing the class number. For the example of three-classes spiral data, if we define seven fuzzy membership functions for each input

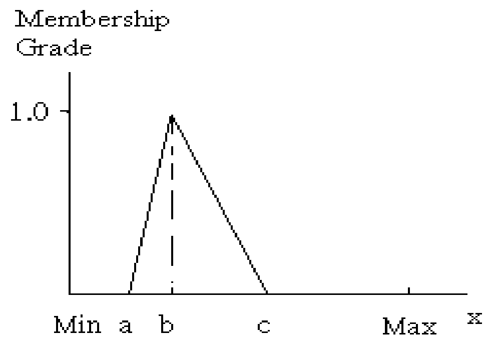


Figure 2: A flexible membership function.

feature, then each table is a 7x7 array as shown in Table 1.

Table 1: A Rule Gene Table for Spiral Data

3	2	1	2	2	1	2
1	2	3	1	3	2	1
2	1	2	1	2	3	2
1	3	3	1	1	3	3
2	1	1	1	3	1	1
1	2	2	1	3	2	2
3	1	3	1	2	1	3

3.4 Fitness Function

Two objectives for designing an efficient fuzzy classificaton are as follows:

1. Maximize the recognition rate.
2. Minimize the number of fuzzy rules.

Considering these two objectives simultaneously, we formulate an optimization fitness function as follows:

$$f = c - wt_1 \times r + wt_2 \times n \qquad (1)$$

where c is a constant with value 3000, wt_r and wt_n are weights and set as 20 and 5 respectively. r represents the number of correctly recognized patterns, and n represents the number of created fuzzy rules.

We use an example to explain how to count the r value as follows:

1. Assume the values of the control genes of a chromosome for three-classes spiral data are shown in the Figure 3 and three rule gene tables are shown in the Table 2. In Figure 3, the first seven binary values represent the seven defined fuzzy membership functions for x-axis feature, the next seven binary values represent the seven defined fuzzy subsets for y-axis feature, and the last three binary values represent the seven fuzzy subsets for output classes. Each entry in each rule gene table is randomly coded as 1, 2, or 3.

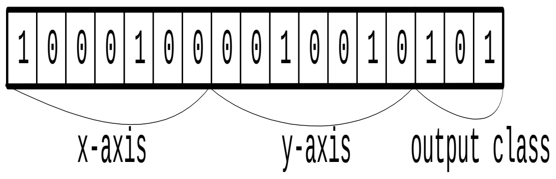


Figure 3: Values of control genes in a chromosome for spiral data.

Table 2: Three Rule Gene Tables for Three-classes Spiral Data

3	2	1	2	2	**2**	2
1	2	3	1	3	2	1
2	1	2	1	2	3	2
1	3	3	1	1	3	3
2	1	1	1	3	1	1
1	2	2	1	3	2	2
3	1	3	1	2	1	3
1	2	3	2	1	**3**	2
2	1	1	3	2	2	3
3	2	1	3	1	2	1
3	1	2	3	1	2	3
1	3	2	2	1	3	1
2	3	3	1	2	2	1
2	2	1	3	3	1	2
3	2	1	2	2	**1**	2
3	1	2	2	1	3	2
1	3	1	3	1	1	1
2	2	1	3	3	2	1
3	2	3	2	1	3	2
3	1	1	3	2	1	3
2	3	1	2	3	2	1

2. From figure 3, we see that the first gene and the fifth gene of x-axis are enabled. Also, the third gene and the sixth gene of y-axis are enabled. There are 4 combinations—(1,3),(1,6),(5,3), and (5,6) between x-axis and y-axis. Initially, let r and n be equal to 0.

3. If an input pattern matches one of these combinations, say (1,6), and it belongs to class 1, we do defuzzification of Center Of Area(COA) to get the output value by means of the (1,6) entry value(2—bold) in the first rule gene table(top in Table 2), and then calculate the difference between the output value(defuzzification result) and the target value(1) of the input pattern.

4. If the difference is less than 0.5, count the r value and check whether to count the n value at the same time. Otherwise, we keep the combination for rule gene tables adjustment and try the same (1,6) entry from the second rule gene table(middle in Table 2). Definitely, the difference got by the same (1,6) entry value(3—bold) in the second rule gene table will not be less than 0.5. Therefore, we keep the combination and try the third rule gene table(bottom in Table 2) again.

5. Repeat step 3 to step 4 for all input patterns, we can get a chromosome's fitness by equation 1

The n value is counted when the r value increases unless we check the same antecedents and consequent as before.

3.5 Rule Gene Tables Adjustment

After fitness calculation of a population, adjust each entry in rule gene tables whose difference between defuzzification result and input target never less than 0.5. The new entries values will be assigned with another class number randomly under a defined probability. The probability is defined as 0.01.

3.6 Fuzzy Rule and Fuzzy Reasoning

The fuzzy rule i for the n-dimensional pattern classification problem is expressed as follows:

R_i : If x_1 is A_{i1} and x_2 is A_{i2} ... and x_n is A_{in} then Class C_i , $i = 1, 2, \ldots, N$,
where $A_{i1}, \ldots, A_{in}$ are antecedent fuzzy subsets, C_i represents a consequent class, and N is the total number of fuzzy rules for the designed fuzzy classifier. The way we classify pattern to some class is almost same as counting the r value described in Section 3.4.

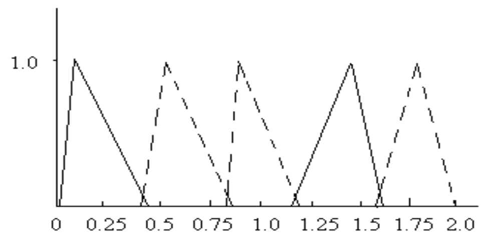
4 Algorithm of Our Method

4.1 Algorithm

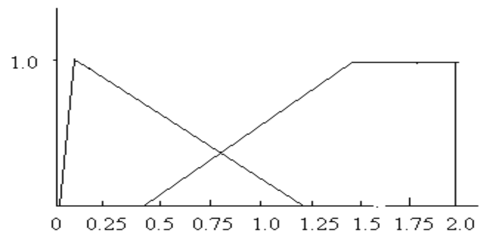
Our algorithm is outlined as follows :

1. Initialization: Randomly initialize k individuals for the population and several rule gene tables. Crossover rate is set as 0.6 and the mutation rate is set as 0.01 at first. During the evolution, they will be randomly changed according to the degree of convergence.

2. Recovery: To ensure that there are no undefined regions for the fuzzy membership functions of parameter genes, an adjustment is operated [11]. There is an example shown in Figure 4(a) if the control genes are 1 0 0 1 0. The result after recovery is shown in Figure 4(b).



(a)



(b)

Figure 4: (a) The membership functions before recovery ; (b) The membership functions after recovery.

3. Fitness calculation: According to the definition of fitness function described in Section 3.4, Calculate each individual's fitness using all training data. After that, sort all chromosomes for selection operation afterwards. During fitness calculation, keep the entries' indices(combinations) of rule gene tables whose difference between defuzzification result and input target always greater than 0.5 for rule gene tables adjustment later.

4. Selection: Choose the best half of ranked chromosomes as part 1 for creating new population of next generation.

5. Crossover: For control genes, select $k/2$ pairs of individuals from the sorted population for crossover. One crossover point is used.

The position is decided by the crossover rate which is set initially. The crossover rate is kept as a constant(0.6) until no variation happens over 20 generations. For the parameter genes, choose the point which corresponds to the crossover point of control genes. Maintaining the parameter genes sequence has to been done after crossover operation.

6. Mutation: Mutation rate is treated as the same way as the crossover rate according the variation of convergence. When no variation happens over 20 generation, mutation rate can be changed a little bit.

7. Rule gene tables adjustment: As described in Section 3.5.

8. Fitness calculation again: After crossover and mutation, we do the same fitness calculation as step 3 to get another better half as part 2 for creating new population of next generation.

9. Next generation: Combine part 1 and part 2 to form a new generation population(still 30 individuals in all). By this way, we can keep the best chromosomes and still have the chance to get better chromosomes generation to generation.

10. Termination checking: Every 10 generations, we do classification testing by reading test data. When the recognition rate is more than some defined threshold, we stop the program in advance. Otherwise, the program terminates after 200 generations.

4.2 An Illustration

In the following, we use a simple data set shown in Figure 5 and Table 3 to illustrate our algorithm as follows:

Table 3: Training data

x-axis	y-axis	class
0.4	0.2	1
0.4	0.4	1
0.6	0.2	1
0.6	0.4	1
1.4	0.2	2
1.4	0.4	2
1.6	0.2	2
1.6	0.4	2
0.9	1.5	3
0.9	1.7	3
1.1	1.5	3
1.1	1.7	3

1. Create a population of size 4 whose control genes are shown in Table 4 and parametric genes are shown in Table 5. Note that the first five binary digits, the second five binary digits, and the last three binary digits in Table 4 represent the 5 fuzzy subsets of x-axis feature, the 5 fuzzy subsets of y-axis feature, and the 3 fuzzy subsets of output classes, repectively, while the top, middle, and bottom tables in Table 5 represent the 5 fuzzy subsets of x-axis feature, the 5 fuzzy subsets of y-axis feature, and the 3 fuzzy subsets of output classes, repectively. We use these rule gene tables as shown in Table 6.

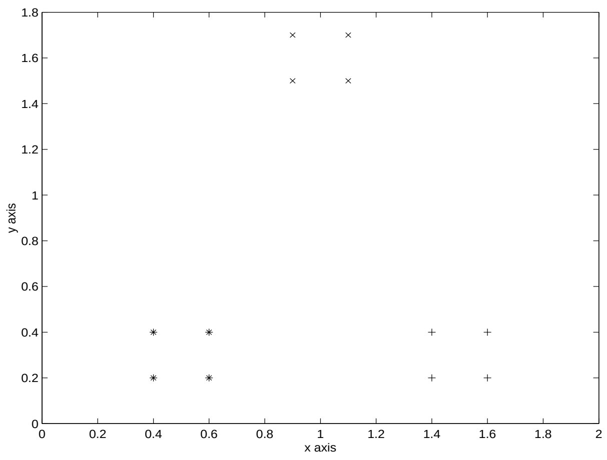


Figure 5: A simple data set—3 classes,12 points.

Table 4: Control genes of a population(size 4)

1	0	0	0	0	0	1	0	1	1	1	1	0
1	0	0	1	0	1	1	1	0	1	1	1	1
0	0	0	0	0	1	0	1	1	0	0	1	1
0	1	1	0	0	0	0	1	1	1	0	1	0

Table 6: Three Rule Gene Tables in illustration

2	1	2	3	2
1	1	1	1	3
2	3	2	3	2
1	1	2	3	2
3	1	1	1	1

1	3	1	3	2
1	1	3	1	2
3	2	1	3	1
1	3	3	2	3
1	2	1	3	2

1	2	2	1	3
2	3	3	3	3
3	2	2	2	1
3	3	3	3	3
2	1	3	2	2

2. Recover the parametric genes according to the control genes. Table 7 shows the result.

3. Calculate the fitness of every chromosome as follows:

   (a) Chromosome 1 : From row 1 of table 4, we see that only the first gene of x-axis is enabled and the second gene , the fourth gene , and the fifth gene of y-axis are enabled . Therefore, there are 3 combinations—(1,2), (1,4), and (1,5)

Table 5: Parametric genes of a population(size 4)

0.00	0.03	0.42	0.40	0.63	0.82	0.80	0.89	1.22	1.20	1.53	1.62	1.60	1.81	2.02
0.00	0.23	0.42	0.40	0.56	0.82	0.80	0.82	1.22	1.20	1.21	1.62	1.60	1.67	2.02
0.00	0.38	0.42	0.40	0.68	0.82	0.80	0.81	1.22	1.20	1.41	1.62	1.60	1.90	2.02
0.00	0.06	0.42	0.40	0.57	0.82	0.80	1.04	1.22	1.20	1.44	1.62	1.60	1.90	2.02

0.00	0.31	0.42	0.40	0.44	0.82	0.80	0.99	1.22	1.20	1.51	1.62	1.60	1.94	2.02
0.00	0.10	0.42	0.40	0.71	0.82	0.80	0.91	1.22	1.20	1.56	1.62	1.60	1.69	2.02
0.00	0.16	0.42	0.40	0.61	0.82	0.80	0.84	1.22	1.20	1.33	1.62	1.60	1.69	2.02
0.00	0.38	0.42	0.40	0.51	0.82	0.80	1.18	1.22	1.20	1.34	1.62	1.60	1.63	2.02

0	1	2	1	2	3	2	3	4
0	1	2	1	2	3	2	3	4
0	1	2	1	2	3	2	3	4
0	1	2	1	2	3	2	3	4

Table 7: Recovered parametric genes of a population(size 4)

0.00	0.03	2.00	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
0.00	0.23	1.22	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	0.40	1.21	2.00	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.00	0.57	0.82	0.80	1.04	2.00	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0

-1.0	-1.0	-1.0	0.00	0.44	1.22	-1.0	-1.0	-1.0	0.80	1.51	1.62	1.60	1.94	2.02
0.00	0.10	0.42	0.40	0.71	0.82	0.80	0.91	1.62	-1.0	-1.0	-1.0	1.20	1.69	2.02
0.00	0.16	0.82	-1.0	-1.0	-1.0	0.40	0.84	1.22	1.20	1.33	2.00	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	0.00	1.18	1.22	1.20	1.34	1.62	1.60	1.63	2.02

0	1	2	1	2	3	2	3	4
0	1	2	1	2	3	2	3	4
0	1	2	1	2	3	2	3	4
0	1	2	1	2	3	2	3	4

between x-axis's fuzzy subsets and y-axis's fuzzy subsets. Initially, let r and n be equal to 0. Then, input the training data and do the following calculations:

i. For the first pattern—(0.4,0.2,1), it matches the combination(1,2). From the (1,2)-entry value(1) in the first rule gene table(upper left in Table 6), the difference between the defuzzification result and the target value(1) is 0.24 which is less than 0.5, Then, we increase the r value and increase the n value at the same time if there is a new rule is created.

ii. For the second pattern—(0.4,0.4,1), the third pattern—(0.6,0.2,1), and the fourth pattern—(0.6,0.4,1), they all match the combination(1,2) and have the same result as the first pattern.

iii. For the fifth pattern—(1.4,0.2,2), it also matches the combination—(1,2), but the difference got from the (1,2) entry value(1) in the first rule gene table(upper left in Table 6) is 1.13(>0.5), Therefore, try the same (1,2) entry value(3) in the second rule gene table(upper right in Table 6) again. However, the difference(1) is still greater than 0.5, therefore, continue to try the same (1,2) entry value(2) in the third rule gene table((bottom in Table 6), we get a difference of 0 which is less than 0.5 finally, therefore, count the r value again and check whether to count the n value at the same time if there is a new rule is created.

iv. For the sixth pattern—(1.4,0.4,2), the seventh pattern—(1.6,0.2,2), and the eightieth pattern—(1.6,0.4,2), they match the combination—(1,2) and have the same result as the fifth pattern.

v. For the ninth pattern—(0.9,1.5,3), it matches the combination—(1,4). we get a difference of 0 from the (1,4)-entry in the first rule gene table.

vi. For the tenth pattern—(0.9,1.7,3), it matches the combination—(1,5), but we get a difference of 0 from the (1,5)-entry in the third rule gene table.

vii. For the eleventh pattern—(1.1,1.5,3), it matches the combination—(1,4), but we get a difference of 0 from the (1,4)-entry in the first rule gene table.

viii. For the twelfth pattern—(1.1,1.7,3), it matches the combination—(1,5), but we get a difference of 0 from the (1,5)-entry in the third rule gene table.

ix. The fitness of chromosome 1 is 75, where r is 12,n is 3.

(b) Chromosome 2, chromosome 3, and chromosome 4 are processed similarly as above. The fitness values obtained are 215, 300, and 115, respectively.

4. Select the best half of chromosomes(i.e.,chromosome 1 and chromosome 4) as part 1 for creating new population of next generation.

5. Select two pairs—chromosome 1 and chromosome 3, chromosome 2 and chromosome 4. Perform the operations of crossover and mutation on the two pairs.

6. Adjust rule gene tables shown in Table 8.

Table 8: Three Rule Gene Tables after adjustment

2	1	2	3	2
1	1	1	1	3
2	3	2	3	1
1	1	2	3	2
3	1	1	1	1

1	3	1	3	2
1	1	3	1	2
3	2	1	3	2
1	3	3	2	3
1	2	1	3	2

1	2	2	2	3
2	3	3	3	3
3	2	2	2	1
3	3	3	3	3
2	1	3	2	2

7. Calculate fitness for the population after crossover and mutation by the same way as step 3. The fitness values of chromosome 1, chromosome 2, chromosome 3 , and chromosome 4 are 115, 300, 70, and 125, respectively.

8. Select the best half of chromosomes(i.e.,chromosome 1 and chromosome 3) as part 2 for creating new population of next generation.

9. Create a new population for next generation by combining part1 and part2, shown in table 9.

Table 9: Control genes of a new population(size 4)

1	0	0	0	0	0	1	0	1	1	1	1	0
0	1	1	0	0	0	0	1	1	1	0	1	0
1	0	0	0	0	0	0	0	1	1	0	1	0
1	0	1	0	0	0	0	0	1	1	0	1	0

10. Check termination: read test data shown in table 10 and use the best chromosome of the last population to calculate the recognition rate to decide whether the algorithm should be terminated or not. Here, we use the best chromosome of the population in generation 1 and get the result as follows:

    (a) For the first test pattern—(0.4,0.4,1), the difference between defuzzificaton result and target value is 0.26

    (b) For the second test pattern—(0.6,0.4,1), the difference between defuzzificaton result and target value is 0.24

    (c) For the third test pattern—(1.4,0.4,2), the difference between defuzzificaton result and target value is 0

    (d) For the fourth test pattern—(1.6,0.4,2), the difference between defuzzificaton result and target value is 0

    (e) For the fifth test pattern—(0.9,1.7,3), the difference between defuzzificaton result and target value is 0

    (f) For the sixth test pattern—(1.1,1.7,3), the difference between defuzzificaton result and target value is 0

    Therefore, the recognition rate is 100%

Table 10: test data

x-axis	y-axis	class
0.4	0.4	1
0.6	0.4	1
1.4	0.4	2
1.6	0.4	2
0.9	1.7	3
1.1	1.7	3

5 Experimental Results

We present two data sets to show the performance of our classification algorithm.

5.1 The Iris Data Set

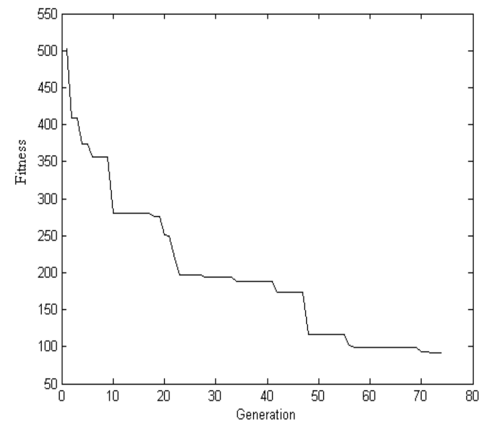
The iris data set consists of 150 patterns with 4 features and 3 classes. In our algorithm, the control genes of each chromosome are randomly coded as 31 binary numbers to correspond to the fuzzy triangular membership functions of input features and output classes. Therefore, the parametric genes are randomly coded as 93 real numbers to represent the membership functions of fuzzy subsets of input features and output classes. We use three rule gene tables whose entries are randomly coded as integer numbers(1-3) to represent output classes. We use all of them to train our algorithm and randomly choose half of them to test. The population size is 30. All other parameters defined in our algorithm are described in Section 3.6.

Figure 6(a) shows the fitness convergence versus generations with one rule gene table for iris data set. Figure 6(b) shows the fitness convergence versus generations with three rule gene tables for Iris data set. From these two figures, we can see that the recognition rate of our algorithm with three rule gene tables is faster than the one with only one rule gene table.

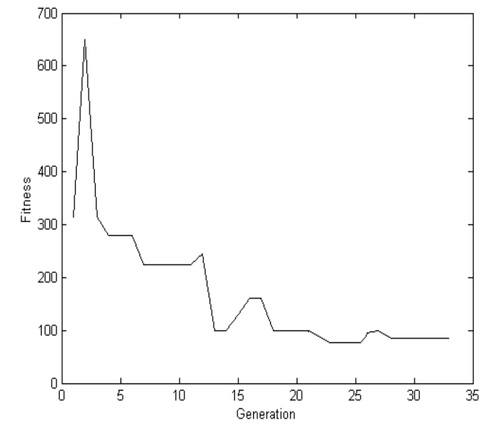
The obtained fuzzy rules using 3 rule gene tables are shown as follows:

R_1: IF sepal length is SL_0 AND sepal width is SW_0 AND petal length is PL_3 AND petal width is PW_1 THEN the flower is Setosa.

R_2: IF sepal length is SL_0 AND sepal width is SW_0 AND petal length is PL_3 AND petal width is PW_3 THEN the flower is Virginica.

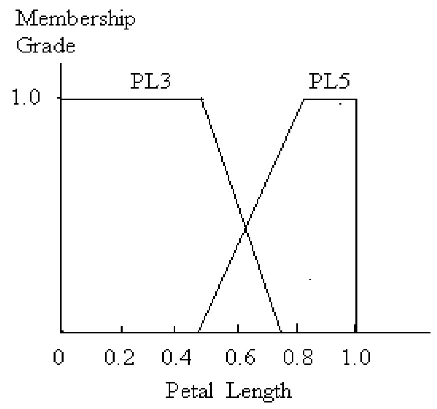


(a)

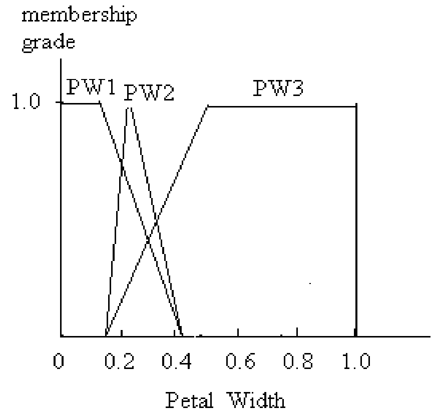


(b)

Figure 6: (a) The fitness convergence vs generations with one rule gene table; (b) The fitness convergence vs generations with 3 rule gene tables.



(a)



(b)

Figure 7: (a) The recovered membership function of petal length fuzzy set.; (b) The recovered membership function of petal width fuzzy set.

R_3: IF sepal length is SL_0 AND sepal width is SW_0 AND petal length is PL_5 AND petal width is PW_2 THEN the flower is Versicolor.

where SL_0 denotes the recovered fuzzy subset of sepal length feature. It is negligible. SW_0 denotes the recovered fuzzy subset of sepal width feature. It is negligible too. PL_3 denotes the third recovered fuzzy subset of petal length feature. PL_5 denotes the fifth recovered fuzzy subset of petal length feature,...etc. The recovery mechanism is derived from [11]. PL_3,PL_5,PW_1,PW_2, and PW_3 are shown in Figure 7(a)and Figure 7(b).

A comparison between Hong's algorithm [8], Wu's algorithm [4], and the proposed algorithm is shown in Table 11. From this table, we can see that the recognition rate of the proposed algorithm is better than those of Hong's algorithm and Wu's algorithm.

5.2 The Spiral Data Sets

The second data set we select is a more complicated spiral data set which is a union of K subsets,s_0,....,s_{K-1}, with the data in subset s_k, $0 \leq k \leq K-1$, belonging to class k. Each subset s_k consists of the following two-attribute points :

$$s_k : \begin{cases} x = \rho \cos(\theta + \frac{2k\pi}{K}) \\ y = \rho \sin(\theta + \frac{2k\pi}{K}) \end{cases}$$

where $\rho = \alpha\theta$, $\alpha = 0.8$, $0.25\pi \leq \theta \leq 4\pi$. We only show a three classes spiral data in Figure 8.

The spiral data contains 2 features. We generate four spiral data sets—2,3,4, and 5 classes.

Table 11: A comparison of the recognition accuracy rate.

	Hong and Lee's algorithm [3]	Wu and Chen's algorithm [4]	The proposed algorithm
Average recognition rate	95.57%	96.21%	99.20%
Average Number of rules	6.21	3	3

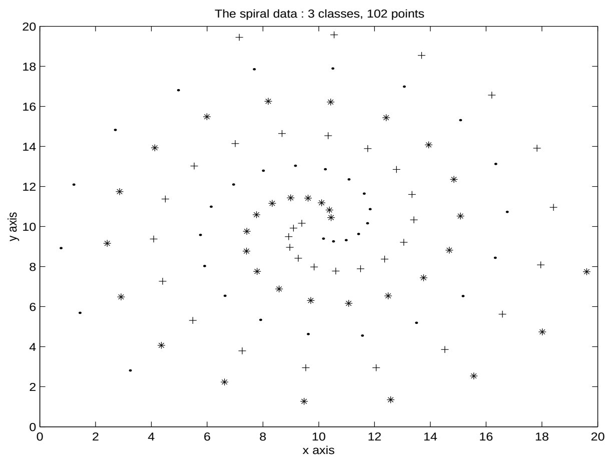
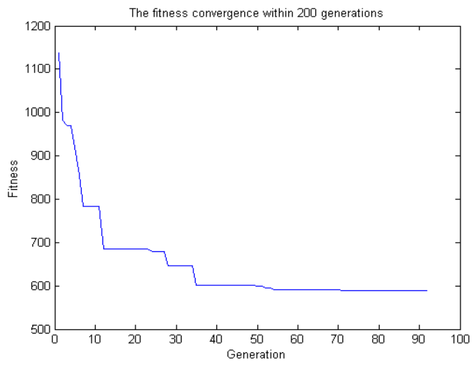


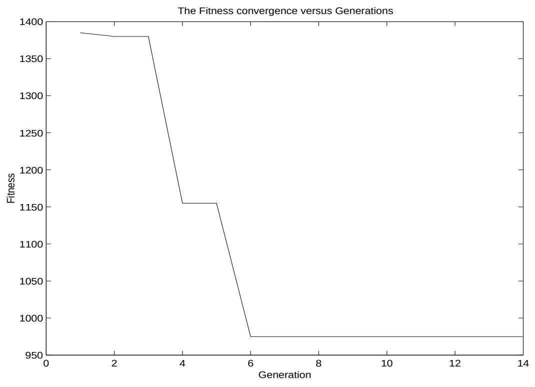
Figure 8: The spiral data—3 classes,102 points

Each class has 34 points. Therefore, for three-classes spiral data, the control genes of each chromosome are randomly coded as 17 binary numbers to correspond to the fuzzy subsets of input features and output classes and the parametric genes are randomly coded as 51 real numbers to represent the membership function of fuzzy subsets of input features and output classes if we define triangular fuzzy sets for each input feature and output class. For the rule gene tables, their entries are randomly coded as integer numbers(1-3) to represent output classes. We use all of them to train our algorithm and randomly choose half of them to test our algorithm.

A comparison on fitness convergence between the case with one rule gene table and the case with three rule gene tables is shown in Figure 9.



(a)



(b)

Figure 9: (a) The Fitness convergence vs generations with one rule gene table ; (b) The Fitness convergence vs generations with 3 rule gene tables.

Table 12: Comparison of two algorithms for spiral data.

			The proposed Algorithm			Wu and Chen's Algorithm		
# of Feat.	# of Class	# of Pats	# of Secs	# of rules	Rec. rate	# of Secs	# of rules	Rec. rate
2	2	68	485	2	1.00	112	1	0.47
2	3	102	616	3	0.96	482	1	0.33
2	4	136	980	4	0.86	720	1	0.28
2	5	170	1320	4	0.82	830	1	0.22

In Table 12, # of Feat. denotes the number of features of input patterns. # of Pats. denotes the number of input patterns. # of secs. denotes the number of seconds for simulation time. Rec. means Recognition. From this table, we can find that the recognition rate derived from our algorithm is higher than the one of Wu's method because of the recovery mechanism [11] and the rule gene tables.

6 Conclusion

In this paper, we have presented a new algorithm based on a hierarchical genetic algorithm with rule gene tables to design an efficient fuzzy classifier. Computer simulation results show that our algorithm performs better than other methods. However, how to decrease the space needed for rule gene tables will be researched in the future.

References

[1] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. 338–353, 1965.

[2] L. X. Wang and J. M. Mendel, "generating fuzzy rules by learning from examples," *IEEE Trans. Syst., Man, Cybern*, vol. 22, no. 6, pp. 1421–1427, 1992.

[3] T. Hong and C. Lee, "Introduction of fuzzy rules and membership functions from training examples," *Fuzzy Sets Syst*, vol. 84, pp. 33–47, 1996.

[4] T.-P. Wu and S.-M. Chen, "A new method for constructing membership functions and fuzzy rules from training examples," *IEEE Trans. Syst., Man, Cybern.-PartB*, vol. 29, pp. 25–40, 1999.

[5] L. RenHou and Y. Zhang, "Fuzzy logic controller based on genetic algorithms," *Fuzzy Sets Syst.*, vol. 83, pp. 1–10, 1996.

[6] B. Carse, C. Terence, F. Munro, and Alistair, "Evolving fuzzy rule based controllers using genetic algorithms," *Fuzzy Sets Syst*, vol. 80, pp. 273–293, 1996.

[7] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison Wesley, 1989.

[8] H. Narazaki, I. Shigaki, and T. Watanabe, "A method for extracting appropriate rules from neural networks," *in proc. FUZZ-IEEE, Yokohama, Japan*, pp. 1865–1870, 1995.

[9] N. K. Kasabov, "Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems," *Fuzzy Sets Syst*, vol. 82, pp. 135–149, 1996.

[10] T. Yoshikawa, T. Furuhashi, and Y. Uchikawa, "Emergence of effective fuzzy rules for controlling mobile robots using dna coding method," in *Proc. ICEC'96*, (Nagoya Japan), pp. 581–586, May. 1996.

[11] K.-s. Tang, K. F. Man, Z.-f. Liu, and S. Kwong, "Minimal fuzzy memberships and rules using hierarchical genetic algorithms," *IEEE Trans. Industrial Electronics*, vol. 45, no. 3, pp. 162–169, Aug.,1998.

[12] K. F. Man, K. S. Tang, and S. Kwong, *Genetic Algorithms.* Addison Wesley, 1999.