Linguistic Rule Extraction by Genetics-Based Machine Learning

Hisao Ishibuchi
Dept. of Industrial Engineering
Osaka Prefecture University
1-1 Gakuen-cho, Sakai, Osaka 599-8531, JAPAN
E-mail: hisaoi@ie.osakafu-u.ac.jp
Phone: +81-722-54-9350

Tomoharu Nakashima
Dept. of Industrial Engineering
Osaka Prefecture University
1-1 Gakuen-cho, Sakai, Osaka, 599-8531, JAPAN
E-mail: nakashi@ie.osakafu-u.ac.jp
Phone: +81-722-54-9351

Abstract

This paper shows how linguistic classification knowledge can be extracted from numerical data for pattern classification problems with many continuous attributes by genetic algorithms. Classification knowledge is extracted in the form of linguistic if-then rules. In this paper, emphasis is placed on the simplicity of the extracted knowledge. The simplicity is measured by two criteria: the number of extracted linguistic rules and the length of each rule (i.e., the number of antecedent conditions involved in each rule). The classification ability of extracted linguistic rules, which is measured by the classification rate on given training patterns, is also considered. Thus our task is formulated as a linguistic rule extraction problem with three objectives: to maximize the classification rate, to minimize the number of extracted linguistic rules, and to minimize the length of each rule. For tackling this problem, we propose a multi-objective genetics-based machine learning (GBML) algorithm, which is a hybrid algorithm of Michigan approach and Pittsburgh approach. Our hybrid algorithm is basically a Pittsburgh-style algorithm with variable string length. A Michigan-style algorithm is combined as a kind of mutation for partially modifying each string.

1. INTRODUCTION

Recently fuzzy rule-based systems have been often applied to pattern classification problems (Ishibuchi et al., 1992, Rhee & Krishnapuram, 1993, and Cordon et al., 1999). Main characteristic features of fuzzy rule-based classification systems are their nonlinearity and comprehensibility. Since fuzzy rule-based systems are universal approximators of nonlinear functions (Kosko, 1992, and Wang, 1992), they can handle highly nonlinear classification problems. Many studies on fuzzy rule-based classification systems tried to improve their classification ability. Some studies used neuro-fuzzy techniques (Uebele et al., 1995, Mitra et al., 1997, Nauck & Kruse, 1999), and other studies employed genetic algorithms for designing fuzzy rule-based classification systems (Ishibuchi et al., 1994, Yuan & Zhuang, 1996, and Cordon et al., 1998). In those studies, emphasis was placed on the classification ability of fuzzy rule-based systems rather than their comprehensibility. That is, fuzzy rule-based systems were tuned for maximizing classification rates on given training patterns using various learning techniques. Another research direction in fuzzy rule-based classification is to extract linguistic classification knowledge from numerical data in the form of linguistic if-then rules (Ishibuchi et al., 1997). In this research direction, emphasis is placed on the comprehensibility of extracted linguistic knowledge rather than the classification performance.

Our task in this paper is to extract a small number of comprehensible linguistic rules from numerical data for high-dimensional pattern classification problems with many continuous attributes. We use linguistic rules of the following form for an *n*-dimensional pattern classification problem with *c* classes and *n* continuous attributes:

$$\text{Rule } R_j: \text{If } x_1 \text{ is } A_{j1} \text{ and ... and } x_n \text{ is } A_{jn} \text{ then Class } C_j \text{ with } CF_j, \quad j = 1,2,...,N, \quad (1)$$

where R_j is the label of the *j*-th linguistic rule, $\mathbf{x} = (x_1,...,x_n)$ is an *n*-dimensional pattern vector, A_{ji} is a linguistic value such as *small* and *large* for the *i*-th attribute, C_j is a consequent class (i.e., one of the *c* classes), CF_j is a certainty grade in the unit interval [0,1], and N is the number of linguistic rules.

Let us assume that we have K_i linguistic values for describing the *i*-th attribute of our *n*-dimensional pattern classification problem ($i = 1,2,...,n$). One of those K_i linguistic values is used as the antecedent linguistic value A_{ji} for the *i*-th attribute in each linguistic rule. In this case, the total number of possible combinations of the linguistic values is $K_1 \times K_2 \times ... \times K_n$. It is impractical to examine all the possible combinations for extracting linguistic rules when our pattern classification problem involves many attributes (i.e., when n is large). For example, the total number of the possible combinations of the linguistic values is more than one billion when $n = 13$ and $K_i = 5$ for $i = 1,2,...,13$. This example shows that the exponential increase in the number of linguistic rules prevents the use of any exhaustive examination method in the case of high-dimensional pattern classification problems. Such exponential increase can be also explained by the exponential decrease in the covered area by each linguistic rule. That is, a linguistic rule with many antecedent conditions can cover only a tiny fraction of the *n*-dimensional pattern space. Thus a large number of linguistic rules with many antecedent conditions are necessary for covering the whole pattern space. Due to the exponential increase in the number of linguistic rules, the comprehensibility of linguistic knowledge is drastically impaired as the dimensionality of the pattern space increases. The increase in the number of antecedent conditions also impairs the comprehensibility of each linguistic rule. In general, it is much easier for users to intuitively understand short linguistic rules with only a few antecedent conditions than long rules with many conditions.

In this paper, we try to extract comprehensible classification knowledge from numerical data in the form of a small number of simple linguistic rules. Only a few antecedent conditions are specified by linguistic values in simple linguistic rules. The other conditions are viewed as "*don't care*" conditions. This can be implemented by considering "*don't care*" as an additional linguistic value. In this case, each antecedent linguistic value A_{ji} of our linguistic rules in (1) is selected from the given K_i linguistic values and "*don't care*". Because "*don't care*" conditions can be omitted, linguistic rules with many "*don't care*" conditions are written as simple rules. For example, in a computer simulation described in Section 5, we extracted the following linguistic rules for a 13-dimensional pattern classification problem:

R_1: If x_7 is *medium* and x_{11} is *medium*
then Class 1 with $CF_1 = 0.56$, (2)

R_2: If x_{10} is *small* then Class 2 with $CF_2 = 0.94$, (3)

R_3: If x_7 is *small* and x_{13} is *medium small*
then Class 3 with $CF_3 = 0.84$. (4)

Because we use "*don't care*" as an additional linguistic value for each of the n attributes, the total number of the possible combinations of the linguistic values is calculated as $(K_1 + 1) \times ... \times (K_n + 1)$. Our task in this paper is to find a small number of simple linguistic rules in the huge search space consisting of those combinations of the linguistic values. The difficulty of our task lies in the size of the search space. In the following sections, we show how genetic algorithms can tackle this difficult task.

2. PROBLEM FORMULATION

Let us assume that we have m training patterns $\mathbf{x}_p = (x_{p1}, ..., x_{pn})$, $p = 1,2,...,m$ from c classes in our *n*-dimensional pattern classification problem. For simplicity of explanation, each attribute value is assumed to be a real number in the unit interval [0,1]. This means that the pattern space of our pattern classification problem is the *n*-dimensional unit hyper-cube $[0,1]^n$. In computer simulations of this paper, every attribute value was normalized into a real number in the unit interval [0,1].

We also assume K_i linguistic values and their membership functions have already been given by users for describing the *i*-th attribute ($i = 1,2,...,n$). We do not modify the given membership function of each linguistic value because the modification may cause a gap between the modified membership function and the users' understanding of each linguistic value. Any shapes of membership functions can be handled by our approach in this paper. The point is that the membership function of each linguistic value should be consistent with the users' domain knowledge and intuition. In computer simulations of this paper, we used a typical set of linguistic values with triangular membership functions for each attribute because we have no domain knowledge about data sets used in our computer simulations. An example of such a typical set of linguistic values is shown in Fig. 1.

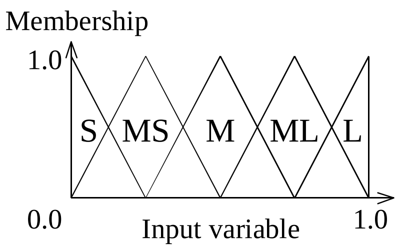

Figure 1: Typical examples of linguistic values (S: *small*, MS: *medium small*, M: *medium*, ML: *medium large*, and L: *large*).

When we use the five linguistic values in Fig. 1 and "DC: *don't care*" for each attribute of our *n*-dimensional pattern classification problem, all the 6^n combinations of the six antecedent linguistic values can be written as

$$\text{If } x_1 \text{ is } \left\{\begin{matrix} \text{S} \\ \text{MS} \\ \text{M} \\ \text{ML} \\ \text{L} \\ \text{DC} \end{matrix}\right\} \text{ and ... and } x_n \text{ is } \left\{\begin{matrix} \text{S} \\ \text{MS} \\ \text{M} \\ \text{ML} \\ \text{L} \\ \text{DC} \end{matrix}\right\} \text{ then } C_j \text{ with } CF_j . \tag{5}$$

In this case, our task is to find a small number of linguistic rules from these 6^n rules. For example, the total number of the linguistic rules in (5) is over 13 billion in a wine data set with 13 attributes used in our computer simulations. It should be noted that the determination of the consequent class C_j and the certainty grade CF_j is not included in our linguistic rule extraction problem. This is because C_j and CF_j of each linguistic rule can be easily determined by a heuristic method in Ishibuchi et al.(1992) from training patterns (see Section 3).

As we have already mentioned, the comprehensibility of extracted linguistic knowledge is impaired by the increase in the number of linguistic rules. Thus we try to minimize the number of extracted rules. From the viewpoint of the comprehensibility of each linguistic rule, many antecedent conditions deteriorate the comprehensibility. Thus we also try to minimize the number of antecedent conditions involved in each rule. This means that we try to use "*don't care*" conditions in each rule as many as possible. At the same time, we want to extract linguistic knowledge that can correctly classify the given training patterns as many as possible. Thus the number of correctly classified training patterns is maximized. Let us denote a set of extracted linguistic rules by *S*. Then our linguistic rule extraction is formulated as the following three-objective optimization problem:

$$\text{Maximize } f_1(S)\text{, minimize } f_2(S)\text{, and minimize } f_3(S)\text{,} \tag{6}$$

where $f_1(S)$ is the number of correctly classified training patterns by *S*, $f_2(S)$ is the number of linguistic rules in *S*, and $f_3(S)$ is the total number of antecedent conditions in *S*. For example, when the rule set *S* consists of the three linguistic rules in (2)-(4), $f_2(S)$ and $f_3(S)$ are calculated as $f_2(S)=3$ and $f_3(S)=5$, respectively. The first objective $f_1(S)$ is calculated by classifying all the given training patterns by the rule set *S*. In Section 3, we briefly describe how each training pattern is classified by linguistic rules in the rule set *S*. As we will show later using simulation results, there exists a tradeoff among the above three objectives. Thus the task of genetic algorithms in this paper is to find non-dominated solutions (i.e., non-dominated rule sets) of our linguistic rule extraction problem. Several GA-based approaches have already been proposed for finding non-dominated solutions of multi-objective optimization problems (Fonseca & Fleming, 1995, and Zitzler & Thiele, 1999).

3. LINGUISTIC CLASSIFICATION

Before describing genetic algorithms for solving our linguistic rule extraction problem, we briefly mention some related issues: determination of the consequent part of each linguistic rule, classification of new patterns by a set of linguistic rules, and specification of the membership function of each linguistic value.

Basically the consequent part of each linguistic rule is determined from training patterns in the subspace specified by its antecedent part (Ishibuchi et al., 1992). First we calculate the compatibility grade $\mu_j(\mathbf{x}_p)$ of each training pattern $\mathbf{x}_p=(x_{p1}, ..., x_{pn})$ with the linguistic rule R_j by the product operation as

$$\mu_{R_j}(\mathbf{x}_p)=\mu_{j1}(x_{p1})\times \ ... \ \times\mu_{jn}(x_{pn}), \tag{7}$$

where $\mu_{ji}(\cdot)$ is the membership function of the antecedent linguistic value A_{ji}. For example, the membership function of "S: *small*" in Fig. 1 is written as

$$\mu_{small}(x)=\max\{0, 1-4x\} \text{ for } 0\le x\le 1. \tag{8}$$

Then we calculate the total compatibility grade of the given training patterns from each class with the linguistic rule R_j as

$$\beta_{\text{Class } h}(R_j) = \sum_{\mathbf{x}_p \in \text{Class } h} \mu_{R_j}(\mathbf{x}_p), \; h = 1,2,...,c. \tag{9}$$

The consequent class C_j is determined as the class with the maximum total compatibility grade:

$$\beta_{\text{Class } C_j}(R_j) = \text{Max}\{\beta_{\text{Class } 1}(R_j), ..., \beta_{\text{Class } c}(R_j)\}. \tag{10}$$

The certainty grade CF_j is specified as

$$CF_j = \{\beta_{\text{Class } C_j}(R_j) - \bar{\beta}\} / \sum_{h=1}^{c} \beta_{\text{Class } h}(R_j), \tag{11}$$

where

$$\bar{\beta} = \sum_{\substack{h=1 \\ h \neq C_j}}^{c} \beta_{\text{Class } h}(R_j) / (c-1). \tag{12}$$

If no training pattern is compatible with R_j, we can not determine the consequent part of R_j. That is, we can not extract the linguistic rule R_j.

Let *S* be a set of extracted linguistic rules. We classify all the training patterns by *S* for calculating the first objective $f_1(S)$. Each training pattern $\mathbf{x}_p$ is classified by a single winner rule R_{j*}, which satisfies the following condition (Ishibuchi et al., 1992):

$$\mu_{R_{j*}}(\mathbf{x}_p) \cdot CF_{j*} = \max\{\mu_{R_j}(\mathbf{x}_p) \cdot CF_j \mid R_j \in S\}. \tag{13}$$

In our linguistic rule extraction problem, it is possible to define each linguistic value by a non-fuzzy interval (see Fig. 2). Because linguistic values should be consistent with the users' domain knowledge and intuition in a particular pattern classification problem, the choice between crisp partitions and fuzzy partitions is totally dependent on the users' preference. Here we compare these two types of partitions from the viewpoint of the performance of linguistic rule-based systems.

As we have already explained, the consequent part of each linguistic rule R_j in (1) is determined by training patterns compatible with its antecedent part. This means that we can not generate any linguistic rule when there is no training pattern in the subspace $A_{j1} \times ... \times A_{jn}$. As we can see from Fig. 2, the main difference between two types of partitions is the existence of overlaps between neighboring linguistic values in the case of fuzzy partitions. For clearly demonstrating the effect of such overlaps on the rule generation, let us consider how many linguistic rules can be generated from a single training pattern. In the case of crisp partitions, only a single rule can be generated from a single training pattern as shown in Fig. 2. This is because there is no overlap between subspaces corresponding to neighboring linguistic rules. In Fig. 2, a linguistic rule in the shaded area can be generated. The generated linguistic rule classifies new patterns in the same shaded area. Thus the training pattern has an influence on the classification of new patterns within that subspace (i.e., the shaded area) including the training pattern. On the other hand, if we use the fuzzy partition in Fig. 1, four fuzzy if-then rules can be generated from the single training pattern in Fig. 2. The generated four linguistic rules cover a much larger subspace than the shaded area. In general, 2^n linguistic rules can be generated from a single training pattern in our *n*-dimensional pattern classification problem in the case of fuzzy partitions.

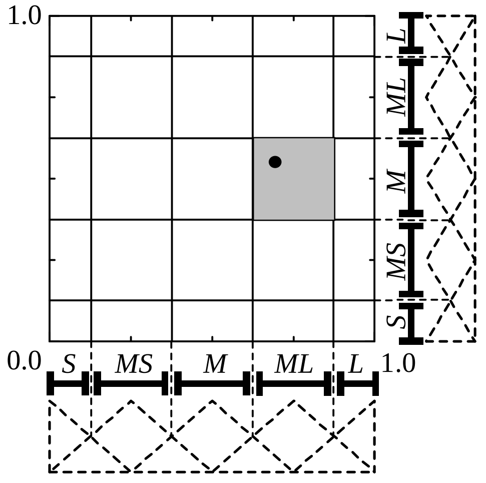

Figure 2: Non-fuzzy partition.

For comparing fuzzy partitions with crisp partitions, we examine the classification performance of generated linguistic rules through computer simulations on iris data. The iris data set involves 150 patterns with four continuous attributes from three classes. For the iris data set, we used *K* linguistic values for each attribute where *K* was specified as $K = 3,4,5,6$ (e.g., $K = 5$ in Fig. 1 and Fig. 2). In computer simulations of this section, all the K^4 combinations of the *K* linguistic values are examined for generating linguistic rules. This means that the four-dimensional pattern space was divided by the $K \times K \times K \times K$ grid. We used such a simple specification in the rule generation procedure just for comparing fuzzy partitions with crisp partitions.

We examined the generalization ability of generated linguistic rules by computer simulations where only five patterns from each class were used as training data. The other 135 patterns were used as test data for calculating the classification rate. This calculation was iterated 50 times using different selections of training patterns in our computer simulations. Simulation results are summarized in Table 1. This table clearly shows that the generalization ability of fuzzy partitions is superior to crisp partitions. This superiority is due to the overlaps of neighboring rules in the case of fuzzy partitions.

Table 1: Classification rates on test data for the iris data set when only 15 patterns were used as training data.

# of linguistic values	3	4	5	6
Fuzzy partition	92.0	87.1	92.1	91.7
Crisp partition	72.9	50.5	52.8	35.6

4. HYBRID GBML ALGORITHM

4.1. BASIC IDEA

In our previous studies on genetics-based machine learning (GBML), we showed that Michigan-style GBML algorithms can effectively find good linguistic rules (Ishibuchi et al., 1996, 1999). We also showed that the search ability of Pittsburgh-style GBML algorithms to find good linguistic rules in a large search space is inferior to Michigan-style algorithms (Ishibuchi et al., 1996). Michigan-style algorithms, however, can not directly optimize the three objectives in our linguistic rule extraction problem because the evolution of linguistic rules is driven only by the classification performance of each linguistic rule. On the contrary, Pittsburgh-style algorithms can directly optimize the three objectives because each objective can be handled as a part of a fitness function. Thus we try to hybridize these two approaches into a single hybrid algorithm. Characteristic features of our hybrid algorithm is as follows:

(1) Our hybrid algorithm is basically a Pittsburgh-style GBML algorithm where an individual (i.e., a string) corresponds to a set of linguistic rules.

(2) A Michigan-style GBML algorithm is used as a kind of mutation in every generation of our hybrid algorithm for partially modifying each rule set generated by Pittsburgh-style genetic operations.

(3) For adjusting the number of linguistic rules, the string length is variable (not fixed). The length of each string is adjusted by a crossover operation.

(4) Our hybrid algorithm is a three-objective genetic algorithm that is devised for finding non-dominated rule sets of our linguistic rule extraction problem. We use two tricks in our hybrid algorithm as in our previous studies on multi-objective scheduling (Murata & Ishibuchi, 1995 and Ishibuchi & Murata, 1998). One is to store a tentative pool of non-dominated solutions separately from the current population during the execution of our hybrid algorithm. Some non-dominated solutions are randomly selected from the tentative pool and added to the current population as elite solutions. The other is to randomly update weight values for the three objectives whenever a pair of parents is selected from the current population.

The outline of our hybrid algorithm can be written as follows (some steps are explained in detail in the following subsections):

Step 1) Initialization: Generate an initial population consisting of N_{set} rule sets (i.e., N_{set} individuals) where N_{set} is the population size.

Step 2) Evaluation: Calculate the three objectives for each rule set in the current population, and then update the tentative pool of non-dominated rule sets.

Step 3) Selection: Repeat the following procedures to select ($N_{\text{set}} - N_{\text{elite}}$) pairs of rule sets.

a) Randomly specify three weight parameters w_i as

$$w_i = random_i / (random_1 + random_2 + random_3), \quad i = 1,2,3. \tag{14}$$

where $random_i$ is a non-negative random real number (or non-negative random integer).

b) Calculate the value of the following scalar fitness function for each rule set S in the current population using the randomly specified three weight parameters in (14).

$$fitness(S) = w_1 \cdot f_1(S) - w_2 \cdot f_2(S) - w_3 \cdot f_3(S). \tag{15}$$

Then select a pair of rule sets based on the fitness value of each rule set S. We specify the selection probability of S in the current population Ψ using the roulette wheel selection with the linear scaling:

$$P(S) = \frac{fitness(S) - f_{\min}(\Psi)}{\sum_{S \in \Psi} \{ fitness(S) - f_{\min}(\Psi) \}}, \tag{16}$$

where $f_{\min}(\Psi)$ is the minimum fitness value in the current population Ψ.

Step 4) Crossover and mutation: Generate a new rule set from each pair of selected rule sets by crossover and mutation operations. These two operations are used with prespecified probabilities. By the genetic operations, ($N_{\text{set}} - N_{\text{elite}}$) rule sets are generated.

Step 5) Michigan-style GBML algorithm: With a prespecified probability, apply a Michigan-style GBML algorithm to each of the generated ($N_{\text{set}} - N_{\text{elite}}$) rule sets. In the Michigan-style GBML algorithm, the fitness value of each linguistic rule is defined by the number of correctly classified training patterns by that rule (Ishibuchi et al., 1999).

Step 6) Elitist strategy: Randomly select N_{elite} non-dominated rule sets from their tentative pool, and then add the selected elite rule sets to the generated ($N_{\text{set}} - N_{\text{elite}}$) rule sets for constructing a new population with N_{set} rule sets.

Step 7) Termination test: If a prespecified stopping condition is satisfied, end the algorithm. Otherwise, return to Step 2.

4.2. CODING

Because the consequent part of each linguistic rule can be easily determined by the heuristic procedure from training patterns as described in Section 3, only its antecedent part is coded. Each linguistic rule R_j in (1) is represented by a string of the length *n* in the form of $R_j = A_{j1}A_{j2} \cdots A_{jn}$. Thus a set of N linguistic rules is represented by a concatenated string of the length $n \times N$. It should be noted that the number of linguistic rules is minimized by our hybrid algorithm. Thus the string length is variable.

4.3. GENETIC OPERATIONS

The number of linguistic rules in each string is adjusted by a crossover operation in Step 4. We use a modified version of the uniform crossover where two parents have different masks as shown in Fig. 3. In this figure, R_j denotes a linguistic rule (i.e., a sting of the length *n*). The point in our crossover operation is that the probability of each linguistic rule to be inherited to the child is not always 0.5. The probability is randomly and uniformly specified in the unit interval [0,1] for each parent whenever the crossover operation is performed.

A mutation operation in Step 4 randomly replaces an antecedent linguistic value with a different linguistic value. For example, a linguistic rule $R_j = A_{j1}A_{j2} \cdots A_{jn}$ in a rule set (i.e., in a string) may be modified by the mutation operation as $A_{j1}B_{j2} \cdots A_{jn}$.

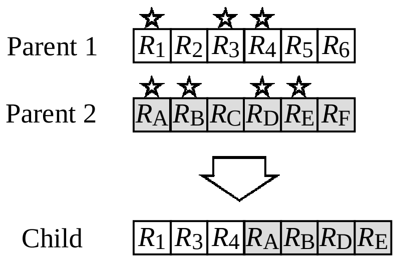
Parent 1
R_1 R_2 R_3 R_4 R_5 R_6
Parent 2
R_A R_B R_C R_D R_E R_F
Child
R_1 R_3 R_4 R_A R_B R_D R_E

Figure 3: Crossover operation.

4.4. MICHIGAN-STYLE GBML ALGORITHM

In Step 5, we use a Michigan-style GBML algorithm for partially modifying each rule set generated by the genetic operations in Step 4. Such modification is mainly for generating good rules and removing unnecessary rules. As a Michigan-style GBML algorithm, we use our fuzzy classifier system (Ishibuchi et al.,1999). In this algorithm, each linguistic rule is evaluated by the number of correctly classified training patterns by that rule. New rules are generated from parent rules by the standard uniform crossover. The mutation operation described in the previous subsection is also used after the crossover operation. Poor rules are removed from the current rule set, and newly generated rules are added. The number of added rules is the same as that of removed rules. For preventing the increase in the CPU time, we use a single iteration of our fuzzy classifier system in Step 5.

5. COMPUTER SIMULATIONS

We applied our hybrid algorithm to a wine data set (available from UC Irvine Database). The wine data set is a three-class pattern classification problem with 178 training patterns involving 13 continuous attributes. We use the five linguistic values in Fig. 1 and "*don't care*" as antecedent linguistic values.

In the main part (i.e., Pittsburgh-style operations), we used the following parameter specifications:

Population size: 50 (i.e., 50 rule sets),
The number of rules in each initial rule set: 100,
Crossover probability: 0.8,
Mutation probability: 0.01,

The number of elite solutions: 5,
Stopping condition: 2000 generations.

In Step 5 of our hybrid algorithm, the Michigan-style algorithm (i.e., fuzzy classifier system) was applied to each rule set with the probability 0.5. We used the following parameter specifications in Step 5:

Crossover probability: 0.8,
Mutation probability: 0.05,
The number of replaced rules: 2,
Stopping condition: one generation.

The application of our hybrid algorithm to the wine data set was iterated ten times from different initial populations. Nineteen non-dominated rule sets in Table 2 were obtained from those ten independent trials. From this table, we can see that a small number of simple linguistic rules can correctly classify many training patterns. For example, the three linguistic rules in (2)-(4) in Section 1 were obtained as a non-dominated rule set, which can correctly classify 165 training patterns (i.e., 92.7% of the given 178 training patterns). In Table 2, we can observe a clear tradeoff between the classification performance and the number of linguistic rules.

For examining the effect of the hybridization, we applied our Pittsburgh-style algorithm to the wine data. That is, the Michigan-style algorithm in Step 5 was removed from our hybrid algorithm in this computer simulation. The other conditions were the same as the above computer simulation. While Step 5 in our hybrid algorithm was skipped, the average CPU time did not decrease (13.2 minutes by the hybrid algorithm and 13.9 minutes by the Pittsburgh-style algorithm on a PC with a Pentium II 400MHz). This is because the number of linguistic rules in each rule set (i.e., in each string) influences the CPU time. The hybrid algorithm decreased the size of rule sets more quickly in its execution than the Pittsburgh-style algorithm.

From ten independent trials with the Pittsburgh-style algorithm, we obtained very similar results to those in Table 2 obtained by the hybrid algorithm. While the final results obtained by these two algorithms were very similar, the hybrid algorithm could find good rule sets more efficiently. In Table 3, we summarize intermediate results during the execution of each algorithm. Table 3 shows the average value of the highest classification rate at each generation over ten independent trials. From this table, we can see that the search ability of the Pittsburgh-style algorithm was improved by the hybridization with the Michigan-style algorithm.

Table 2: Simulation results by the hybrid algorithm.

Number of rules	Classification rates	Average rule length
1	39.9%	0
2	65.7%	0.5
2	66.9%	1.0
2	68.0%	1.5
3	86.0%	0.7
3	90.4%	1.0
3	91.6%	1.3
3	92.7%	1.7
4	94.9%	1.0
4	95.5%	1.3
4	96.6%	1.5
5	96.6%	1.0
5	97.2%	1.2
5	98.3%	1.4
6	97.8%	1.0
6	98.9%	1.3
6	99.4%	1.5
7	100%	2.0
8	100%	1.6

Table 3: Average value of the highest classification rate at each generation over ten independent trials.

Generation	50	100	200	500	1000	2000
Hybrid	48.5	86.9	95.4	98.1	98.9	99.4
Pittsburgh	25.6	58.7	87.4	96.6	98.5	99.1

We also examined the fuzzy classifier system (i.e., Step 5 of our hybrid algorithm) by applying it to the wine data set using the following parameter specifications:

The number of linguistic rules: 10,
Crossover probability: 0.8,
Mutation probability: 0.05,
The number of replaced rules: 2,
Stopping condition: 2000 generations.

This computer simulation was iterated ten times from different initial populations. Since only a single rule set exists in each generation of the Michigan-style GBML algorithm, the CPU time drastically decreased from 13.2 minutes (by the hybrid algorithm) to 0.33 minutes. The following average results were obtained:

Average classification rate: 97.2%,
Average rule length: 1.43.

Since the fuzzy classifier system could not minimize the number of linguistic rules, this computer simulation was performed with the very small population size (i.e., 10 linguistic rules). Thus we could not obtain high classification rates. If we use more linguistic rules (e.g., 60 rules) in each population of the fuzzy classifier system, we can obtain very high classification rates (see Ishibuchi et al., 1999). In this case, the comprehensibility of linguistic knowledge is impaired by a large number of extracted rules.

6. CONCLUSIONS

In this paper, we formulated the linguistic rule extraction from numerical data for high-dimensional pattern classification problems as a three-objective optimization problem. Our goal was to present classification knowledge to users in a human-understandable manner. That is, emphasis was placed on the comprehensibility of extracted linguistic knowledge. For finding non-dominated rule sets of the three-objective rule extraction problem, we proposed a hybrid algorithm where a Michigan-style GBML algorithm was used as a kind of mutation in a Pittsburgh-style GBML algorithm. Its effectiveness was demonstrated by computer simulations on the wine data. That is, it was shown that our hybrid algorithm could find a small number of comprehensible linguistic rules with high classification ability.

References

O. Cordon, M. J. del Jesus, and F. Herrera (1998), “Genetic learning of fuzzy rule-based classification systems cooperating with fuzzy reasoning methods,” *International Journal of Intelligent Systems* 13, 1025-1053.

O. Cordon, M. J. del Jesus, and F. Herrera (1999), “A proposal on reasoning methods in fuzzy rule-based classification systems,” *International Journal of Approximate Reasoning* 20, 21-45.

C. M. Fonseca and P. J. Fleming (1995), “An overview of evolutionary algorithms in multiobjective optimization,” *Evolutionary Computation* 3, 1-16.

B. Kosko (1992), “Fuzzy systems as universal approximators,” *Proc. of 1st IEEE International Conference on Fuzzy Systems*, 1153-1162.

H. Ishibuchi, K. Nozaki, and H. Tanaka (1992), “Distributed representation of fuzzy rules and its application to pattern classification,” *Fuzzy Sets and Systems* 52, 21-32.

H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka (1994), “Construction of fuzzy classification systems with rectangular fuzzy rules using genetic algorithms,” *Fuzzy Sets and Systems* 65, 237-253.

H. Ishibuchi, T. Nakashima, and T. Murata (1996), “Genetic-algorithm-based approaches to the design of fuzzy systems for multi-dimensional pattern classification problems,” *Proc. of 3rd IEEE International Conference on Evolutionary Computation*, 229-234.

H. Ishibuchi, T. Murata, and I. B. Turksen (1997), “Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems,” *Fuzzy Sets and Systems* 89, 135-149.

H.Ishibuchi and T.Murata (1998), “A multi-objective genetic local search algorithm and its application to flowshop scheduling,” *IEEE Trans. on Systems, Man, and Cybernetics - Part C: Applications and Reviews* 28 392-403.

H. Ishibuchi, T. Nakashima, and T. Murata (1999), “Performance evaluation of fuzzy classifier systems for multi-dimensional pattern classification problems,” *IEEE Trans. on SMC - Part B: Cybernetics* 29, 601-618.

S. Mitra, R. K. De, and S. K. Pal (1997), “Knowledge-based fuzzy MLP for classification and rule generation,” *IEEE Trans. on Neural Networks* 8, 1338-1350.

T. Murata and H. Ishibuchi (1995), “MOGA: Multi-objective genetic algorithms,” *Proc. of 2nd IEEE International Conference on Evolutionary Computation*, 289-294.

D. Nauck and R. Kruse (1999), “A neuro-fuzzy method to learn fuzzy classification rules from data,” *Fuzzy Sets and Systems* 89, 277-288.

F. C. -H. Rhee and R. Krishnapuram (1993), “Fuzzy rule generation methods for high-level computer vision,” *Fuzzy Sets and Systems* 60, 245-258.

V. Uebele, S. Abe, and M. -S. Lan (1995), “A neural-network-based fuzzy classifier,” *IEEE Trans. on Systems, Man, and Cybernetics* 25, 353-361.

L. Wang (1992), “Fuzzy systems are universal approximators,” *Proc. of 1st IEEE International Conference on Fuzzy Systems*, 1163-1170.

Y. Yuan and H. Zhuang (1996), “A genetic algorithm for generating fuzzy classification rules,” *Fuzzy Sets and Systems* 84, 1-19.

E. Zitzler and L. Thiele (1999), “Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto Approach,” *IEEE Trans. on Evolutionary Computation* 3, 257-271.