Genetic Rule Selection with a Multi-Classifier Coding Scheme for Ensemble Classifier Design

Yusuke Nojima * and Hisao Ishibuchi
Department of Computer Science and Intelligent Systems
Graduate School of Engineering, Osaka Prefecture University
1-1 Gakuen-cho, Naka-ku, Sakai, Osaka 599-8531, Japan

* Corresponding Author:
Dr. Yusuke Nojima
Department of Computer Science and Intelligent Systems
Graduate School of Engineering, Osaka Prefecture University
1-1 Gakuen-cho, Naka-ku, Sakai, Osaka 599-8531, Japan
Phone: +81-72-254-9198 Fax: +81-72-254-9915 E-Mail: nojima@cs.osakafu-u.ac.jp

Abstract: In this paper, we examine the effectiveness of genetic rule selection with a multi-classifier coding scheme for ensemble classifier design. Genetic rule selection is a two-stage method. The first stage is rule extraction from numerical data using a data mining technique. Extracted rules are used as candidate rules. The second stage is evolutionary multiobjective rule selection from the candidate rules. We use a multi-classifier coding scheme where an ensemble classifier is represented by an integer string. Three criteria are used as objective functions in evolutionary multiobjective rule selection to optimize ensemble classifiers in terms of accuracy and diversity. We examine the performance of designed ensemble classifiers through computational experiments on six benchmark datasets in the UCI machine learning repository.

Keywords: Evolutionary multiobjective optimization, interval rule-based ensemble classifier, genetic rule selection, diversity measure.

1. Introduction
Combining multiple classifiers into a single one is a promising approach to the design of reliable classifiers. A number of approaches have been proposed for generating ensemble classifiers [1-6, 11, 13, 26-34, 37]. Some studies indicate that the diversity of component classifiers is important to generate ensemble classifiers with high generalization ability. In [4], the diversity is defined by the output variety of component classifiers and/or the structural difference of component classifiers. Ensemble approaches can be divided into two groups. One group includes approaches which do not directly evaluate the diversity of component classifiers.
These approaches use some tricks in training data selection and/or classifier design for increasing the diversity of component classifiers such as Bagging and Boosting [3, 13]. The other group includes approaches which directly evaluate the diversity of component classifiers using heuristic diversity measures [28, 30, 32, 36, 37]. Diversity measures can be easily incorporated into evolutionary algorithms. Evolutionary design of ensemble classifiers with diversity measures is an active research area. Especially, evolutionary multiobjective optimization (EMO) algorithms [7-10] are suitable for finding a good balance between the accuracy and the diversity of component classifiers. Recently, some EMO-based approaches to ensemble classifier design have been proposed (for example, [1, 4-6, 27, 31-34]).

Genetic rule selection was first formulated as a single-objective combinatorial optimization problem for the design of fuzzy rule-based classifiers. It is a two-stage method [18, 20]. First, a prespecified number of fuzzy rules are extracted from numerical data. Extracted rules are used as candidate rules. Next, a genetic algorithm is used to obtain an accurate and compact fuzzy classifier from the candidate rules. Genetic rule selection has been generalized as multiobjective formulations using EMO algorithms [16-19, 21, 22]. It was also applied to the design of interval rule-based classifiers [19]. In our former study [32], we have proposed a multi-classifier coding scheme for the design of ensemble fuzzy rule-based classifiers by multiobjective genetic rule selection. The proposed coding scheme is a new idea in the field of ensemble classifier design. An ensemble classifier is represented by an integer string in our approach. Each string is evaluated by its overall classification rate as an ensemble classifier and the average classification rate of its component classifiers. Each string is also evaluated by the entropy of outputs from its component classifiers.

In this paper, we explain our coding scheme in detail, and examine its effectiveness through extensive computational experiments. We apply our ensemble approach to the design of interval rule-based ensemble classifiers (see our former study [32] for preliminary results by fuzzy rule-based ensemble classifiers). Through computational experiments, we examine four formulations: two single-objective and two multiobjective formulations. Experimental results demonstrate the effectiveness of our approach based on the multi-classifier coding scheme and the entropy-based diversity measure in comparison with other methods.

This paper is organized as follows. In Section 2, we briefly explain the outline of genetic algorithms used in genetic rule selection. In Section 3, we explain multiobjective genetic rule selection for generating individual rule-based classifiers with high accuracy and high interpretability. In Section 4, we explain multiobjective genetic rule selection with a multi-classifier coding scheme and an entropy-based diversity measure. In Section 5, we examine the performance of our approach by comparing it with other ensemble methods. Finally, we conclude this paper in Section 6.
2. Genetic algorithms

Genetic algorithms are one of the most well-known and frequently-used optimization techniques in the field of evolutionary computation inspired by natural evolution [14]. A candidate solution is typically represented by a binary string in genetic algorithms. We use a binary string to represent a subset of the candidate rules in genetic rule selection. For function optimization, a real number vector is often used instead of a binary string to represent a solution. For combinatorial optimization, a solution is often represented by an integer string (e.g., in integer programming problems) and a permutation of items (e.g., in traveling salesman problems and scheduling problems). In our multi-classifier coding scheme, we use an integer string to represent an ensemble classifier. The choice of a coding method is usually problem-dependent.

Each string is evaluated by a fitness function, which measures the quality of the string as a solution of an optimization problem at hand. The objective function of the optimization problem or its modification is used as the fitness function. First a number of strings are generated as an initial population. The next population is generated from the current population by selection, crossover and mutation. Better strings with higher fitness values are more likely to be selected as parents in the selection phase. Crossover is a genetic operation to generate one or two offspring from two parents by recombining them. On the other hand, mutation is another genetic operation to randomly modify a part of a string. The next population is constructed from the current population and the offspring population generated by the selection, crossover and mutation. Such a generation update procedure is iterated until a prespecified termination condition is satisfied.

A very rough outline of genetic algorithms can be written as follows:

**Step 1:** Initialization \((P)\)

**Step 2:** \(Q = \text{Genetic operation} (P)\)

**Step 3:** \(P' = \text{Generation update} (P \cup Q)\)

**Step 4:** Replace \(P\) with \(P'\) and return Step 2 until the termination condition is satisfied.

Here, \(P\), \(Q\), and \(P'\) are the current population, the offspring population, and the next population, respectively. In Step 1, a prespecified number of strings are generated as an initial population. This is often performed randomly. Of course, we can use domain-specific techniques to generate good initial strings. Each string in the current population \(P\) is evaluated by the fitness function. In Step 2, a prespecified number of pairs of parents are chosen from the current population \(P\). Better strings with higher fitness values are more likely to be chosen as parents using a parent selection method such as tournament selection and roulette selection. A crossover operator is applied to each pair of parents to generate offspring. Figure 1 illustrates typical examples of crossover operators for binary strings. A mutation operator is used after crossover to modify the generated offspring. In Fig. 2, we show the bit-flip mutation operator for binary
strings. Generated offspring are evaluated by the fitness function. In Step 3, the next population $P'$ is constructed from the current population $P$ and the offspring population $Q$. Some form of elitism is usually implemented in the generation update phase to speed up the genetic search. That is, very good strings in the current population are often inherited to the next population with no modifications. Step 4 is the termination test. The number of the total generations, which is usually a good index of computation load, is frequently used as the termination condition in Step 4.

![Fig. 1. Crossover operators.](image)

![Fig. 2. Bit-flip mutation.](image)

3. Design of interval rule-based classifiers by EMO

3.1. Interval rule-based classifiers

Let us assume that we have $m$ training (i.e., labeled) patterns $x_p = (x_{p1}, ..., x_{pn}), p = 1, 2, ..., m$ from $M$ classes in an $n$-dimensional continuous pattern space where $x_{pi}$ is the attribute value of the $p$-th training pattern for the $i$-th attribute ($i = 1, 2, ..., n$). We denote the set of these training patterns by $D$ (i.e., $D = \{x_1, ..., x_m\}$). We also denote the set of training patterns from Class $h$ as $D_{\text{Class } h}$ where $h = 1, 2, ..., M$.

For our $n$-dimensional $M$-class pattern classification problem, we use the following classification rule:

$$\text{Rule } R_q: \text{If } x_1 \text{ is } A_{qi} \text{ and } ... \text{ and } x_n \text{ is } A_{qn} \text{ then Class } C_q \text{ with } CF_q, \tag{1}$$

where $R_q$ is the label of the $q$-th rule, $x = (x_1, ..., x_n)$ is an $n$-dimensional pattern vector, $A_{qi}$ is an antecedent interval ($i = 1, 2, ..., n$), $C_q$ is a class label, and $CF_q$ is a rule weight (i.e., certainty grade). Each antecedent condition “$x_i$ is $A_{qi}$” means the inclusion $x_i \in A_{qi}$ (i.e., the inequality $A_{qi}^L \leq x_i \leq A_{qi}^U$ where $A_{qi} = [A_{qi}^L, A_{qi}^U]$). We denote the antecedent part of the classification rule $R_q$ in (1) by the interval vector $A_q$ where $A_q = (A_{q1}, ..., A_{qn})$. Thus $R_q$ is denoted as “$A_q \Rightarrow \text{Class } C_q$”.

To generate antecedent intervals, we simultaneously use multiple partitions with different granularities (i.e., from coarse partitions into a few intervals to fine partitions into many...
intervals). This is because we usually have no a priori information about an appropriate granularity of the discretization for each attribute. The use of multiple partitions is one characteristic feature of our approach to knowledge extraction. Since we simultaneously use multiple partitions with different granularities, we need no heuristic criteria to compare different granularities (i.e., we do not have to determine the number of intervals for each attribute). In our computational experiments, we use five partitions into $K$ intervals where $K = 1, 2, 3, 4, 5$ (see Fig. 3). That is, each antecedent interval $A_{qi}$ is chosen from 15 intervals in the five partitions in Fig. 3. Thus, the total number of combinations of antecedent intervals is $15^n$ for our $n$-dimensional pattern classification problem. It should be noted that $K = 1$ in Fig. 3 corresponds to the whole domain interval, which means “don’t care”.

![Five partitions with different granularities](image)

Fig. 3. Five partitions with different granularities used in our computational experiments.

As shown in Fig. 3, the domain interval is divided into $K$ intervals. To specify $(K - 1)$ cutting points, we use an optimal splitting method based on the class entropy measure [12, 19]:

$$H(A_1, ..., A_K) = -\sum_{j=1}^{K} \frac{|D_j|}{|D|} \sum_{h=1}^{M} \left( \frac{|D_{jh}|}{|D_j|} \cdot \log_2 \frac{|D_{jh}|}{|D_j|} \right),$$

(2)

where $(A_1, ..., A_K)$ is $K$ intervals generated by the discretization of an attribute, $D_j$ is the set of training patterns in the interval $A_{j\alpha}$ and $D_{jh}$ is the set of training patterns from Class $h$ in $D_j$. Using the optimal splitting method [12], we can efficiently find the optimal $(K - 1)$ cutting points that minimize the class entropy measure in (2). In this manner, we can obtain multiple partitions for various values of $K$ for each attribute. It should be noted that classification rules of the form in (1) do not always have $n$ antecedent conditions (i.e., antecedent conditions can be don’t care with $K = 1$ in Fig. 3). Some rules may have only a few conditions while others may have many conditions.

For determining the consequent class $C_q$ and the rule weight $CF_q$, first the confidence of the rule “$A_q \Rightarrow \text{Class } h$” is calculated for each class $h$. Let $D(A_q)$ be the set of compatible training patterns with the antecedent part $A_q$:

$$D(A_q) = \{ \mathbf{x}_p \mid x_{p1} \in A_{q1}, ..., x_{pn} \in A_{qn} \}.$$  

(3)
When \( D(A_q) \) is empty, we do not generate any rule with the antecedent part \( A_q \). The confidence of “\( A_q \Rightarrow \text{Class } h \)” is calculated as follows:

\[
c(A_q \Rightarrow \text{Class } h) = \frac{|D(A_q) \cap D(\text{Class } h)|}{|D(A_q)|}, \quad h = 1, 2, \ldots, M.
\] (4)

The consequent class \( C_q \) is specified as the class with the maximum confidence:

\[
c(A_q \Rightarrow \text{Class } C_q) = \max \{c(A_q \Rightarrow \text{Class } h) \mid h = 1, 2, \ldots, M\}.
\] (5)

The consequent class \( C_q \) can be viewed as the dominant class among the compatible training patterns with the antecedent part \( A_q \). As we have already mentioned, we do not generate any rule with the antecedent part \( A_q \) when there is no compatible training patterns with \( A_q \).

The rule weight \( CF_q \) is specified by the difference between the confidence of the consequent class and the sum of the confidences of the other classes as:

\[
CF_q = c(A_q \Rightarrow \text{Class } C_q) - \sum_{\substack{h=1 \atop h \neq C_q}}^M c(A_q \Rightarrow \text{Class } h).
\] (6)

The rule weight \( CF_q \) is used in the classification phase in the following manner. When a new pattern is to be classified by a rule-based classification system \( S \) (i.e., a rule set \( S \)), first all compatible rules with the new pattern are found from \( S \). Then a single winner rule \( R_w \) with the largest rule weight is identified among the compatible rules:

\[
R_w = \arg \max \{\mu_{A_q}(x_p) \cdot CF_q \mid R_q \in S\}.
\] (7)

The new pattern is classified as the consequent class of the winner rule. When multiple rules with different consequent classes have the same maximum value in (7), the classification of the input pattern \( x_p \) is rejected. The classification of \( x_p \) is also rejected when there are no compatible rules with positive compatibility grades for \( x_p \). That is, we do not use default rules. The use of default rules are examined later in Subsection 5.3.

### 3.2. Genetic rule selection

We can generate a large number of classification rules by specifying the consequent class and the rule weight for each of the \( 15^n \) combinations of the antecedent intervals. It is, however, very difficult for human users to handle such a large number of rules. It is also very difficult to intuitively understand long rules with many antecedent conditions. Thus we only generate short rules with a small number of antecedent conditions in the first stage of genetic rule selection. Since the antecedent interval corresponding to \( K = 1 \) in Fig. 3 is actually equivalent to a don’t care condition, all don’t care conditions with the antecedent interval for \( K = 1 \) can be omitted. In this paper, the number of antecedent conditions excluding don’t care conditions is referred to as the rule length. We only examine short rules of length \( L_{\text{max}} \) or less (e.g., \( L_{\text{max}} = 3 \)).
restriction on the rule length is to find a small number of short (i.e., simple) rules that are easily understood by human users.

We further decrease the number of rules by choosing only good rules with respect to a heuristic rule evaluation criterion. That is, we choose a prespecified number of short rules for each class using a heuristic rule evaluation criterion. In the field of data mining, the confidence and the support are often used as heuristic rule evaluation criteria [23]. In the previous subsection, we have already shown in (4) the definition of the confidence of the classification rule “\( A_q \Rightarrow \text{Class } h \)”. Its support is defined as

\[
s(A_q \Rightarrow \text{Class } h) = \frac{|D(A_q) \cap D(\text{Class } h)|}{|D|}, \quad h = 1, 2, \ldots, M.
\]  

(8)

In our computational experiments, each rule is evaluated based on the difference between its support value and the total support value of the other rules with the same antecedent conditions and different consequent classes. More specifically, the rule \( R_q \) with the antecedent conditions \( A_q = (A_{q1}, \ldots, A_{qn}) \) and the consequent class \( C_q \) is evaluated as

\[
f(R_q) = s(A_q \Rightarrow \text{Class } C_q) - \sum_{h=1 \atop h \neq C_q}^{M} s(A_q \Rightarrow \text{Class } h).
\]  

(9)

This is a modified version of a heuristic rule evaluation criterion used in an iterative fuzzy genetics-based machine learning algorithm called SLAVE [15].

We extract a prespecified number of classification rules with the largest values of this criterion for each class (say \( N/M \) rules for each of \( M \) classes: \( N \) rules in total). The extracted rules are used as candidate rules. As we have already mentioned, only short rules of length \( L_{\text{max}} \) or less are examined in our heuristic rule extraction phase (i.e., the first stage of genetic rule selection) in order to find interpretable rules.

In the second stage of genetic rule selection, a subset \( S \) of the \( N \) candidate rules is denoted by a binary string of length \( N \) as \( S = s_1 s_2 s_3 \cdots s_N \) where \( s_i = 1 \) and \( s_i = 0 \) mean that the \( i \)-th candidate rule is included in and excluded from the rule set \( S \), respectively. Such a binary string is used as an individual (i.e., a classifier) in genetic rule selection (see Fig. 4).
In our previous studies [17, 19, 21, 22, 24], we have used the following three objectives to find an accurate and compact rule set $S$:

- $f_1(S)$: The number of correctly classified training patterns by $S$,
- $f_2(S)$: The number of rules in $S$,
- $f_3(S)$: The total number of antecedent conditions (rule length) in $S$.

The first objective is maximized while the second and third objectives are minimized. That is,

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

In this paper, we use the following two-objective problem for comparison because ensemble classifier design is formulated as two-objective problems in the next section.

$$\text{Maximize } f_1(S) \text{ and minimize } f_2(S).$$

As an EMO algorithm, we use NSGA-II, which is one of the most well-known and frequently-used EMO algorithms (for details, see [9, 10]). The second stage of genetic rule selection is written as follows:

**Step 1:** Randomly generate $N_{\text{pop}}$ binary strings of length $N$ as an initial population where $N_{\text{pop}}$ is a user-definable parameter called the population size.

**Step 2:** Evaluate each string using the multiobjective fitness evaluation scheme in NSGA-II.

**Step 3:** Iterate the following operations $N_{\text{pop}}$ times to generate an offspring population with $N_{\text{pop}}$ strings.

3.1: Select a pair of parent strings from the current population by binary tournament selection.

3.2: Recombine the selected parent strings to generate an offspring by a genetic operation called crossover. We use the uniform crossover operation where each bit value of the offspring is inherited from one parent randomly chosen from the two parents. This operation is applied with a prespecified probability (e.g., 0.9). When the crossover operation is not applied to the selected pair of parents, one of the two parents is randomly chosen and used as an offspring.
3.3: Apply a biased mutation operation to the offspring. This operation changes a 0 to a 1 with a small probability and a 1 to a 0 with a large probability to decrease the number of 1’s (i.e., the number of selected rules) in the offspring.

3.4: Calculate the first objective function. Some rules in $S$ may be chosen as the winner rules for no training patterns. We refer to those rules as unnecessary rules. We remove unnecessary rules without degrading the accuracy. Then calculate the second objective function.

**Step 4:** Combine the current and offspring populations into the merged population, and evaluate each string as in Step 2. Then select the best $N_{\text{pop}}$ strings from the merged population.

**Step 5:** If a prespecified termination condition is not satisfied, return to Step 3 with the best $N_{\text{pop}}$ strings selected in Step 4 as the population in the next generation. Otherwise, terminate the execution of the algorithm. We use the total number of iterations of the algorithm (i.e., the total number of generations) as the termination condition in this paper.

We can obtain a number of non-dominated classifiers (i.e., non-dominated rule sets) with respect to the accuracy and the complexity.

### 4. Genetic rule selection for ensemble classifier design

#### 4.1. Multi-classifier coding scheme

For ensemble classifier design, we use a multi-classifier coding scheme [32] which can be easily incorporated into any evolutionary algorithms. Each ensemble classifier is represented by an integer string of length $N$ as $S = s_1 s_2 \cdots s_N$, $s_j \in \{0, 1, \ldots, J\}$ where $N$ and $J$ are the total number of candidate rules and the maximum number of component classifiers, respectively. The value of $s_j$ means a classifier number. “$s_j = 0$” means that the $i$-th rule is included in no classifier. $J$ should be specified beforehand. If we have 12 candidate rules as in Fig. 5 and $J = 3$, the string “103210210033” means that the ensemble classifier is composed of three component classifiers: Classifier 1 with $R_A$, $R_E$, and $R_H$, Classifier 2 with $R_D$ and $R_G$, and Classifier 3 with $R_C$, $R_K$, and $R_L$.

<table>
<thead>
<tr>
<th>Candidate Rules</th>
<th>Genotype</th>
<th>Phenotype</th>
</tr>
</thead>
<tbody>
<tr>
<td>If-then rule A</td>
<td>1</td>
<td>Classifier 1 If-then rule A</td>
</tr>
<tr>
<td>If-then rule B</td>
<td>0</td>
<td>If-then rule E</td>
</tr>
<tr>
<td>If-then rule C</td>
<td>3</td>
<td>If-then rule H</td>
</tr>
<tr>
<td>If-then rule D</td>
<td>2</td>
<td>Classifier 2 If-then rule D</td>
</tr>
<tr>
<td>If-then rule E</td>
<td>1</td>
<td>If-then rule G</td>
</tr>
<tr>
<td>If-then rule F</td>
<td>0</td>
<td>Classifier 3 If-then rule C</td>
</tr>
<tr>
<td>If-then rule G</td>
<td>2</td>
<td>If-then rule K</td>
</tr>
<tr>
<td>If-then rule H</td>
<td>1</td>
<td>If-then rule L</td>
</tr>
<tr>
<td>If-then rule I</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>If-then rule J</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>If-then rule K</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>If-then rule L</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 5. An ensemble classifier represented by an integer string.
An input pattern is classified by each component classifier using the single winner-based method explained in Subsection 3.1. Random tie break is not employed in each component classifier. Then the final classification is performed through the simple majority voting scheme based on the classification result by each component classifier. When multiple classes have the same maximum number of votes, one class is randomly chosen among those classes with the maximum vote.

Although a binary string represents a single classifier in Section 3, an integer string represents a number of classifiers (i.e., an ensemble classifier) in Section 4. The main characteristic of this study is that “string = ensemble”, although “string = classifier” in most of evolutionary multiobjective ensemble classifier designs.

4.2. EMO with a diversity criterion

In [4], the structure and output diversities of component classifiers were often used to improve the generalization ability on unseen patterns. We also proposed an entropy-based diversity measure which evaluates the diversity of outputs from component classifiers [32]. Our diversity measure is similar to Kohavi-Wolpert variance measure [28] which evaluates the variability of the predicted class by each component classifier. Kuncheva and Whitaker [30] modified this variance measure to evaluate the variance based only on whether the decision by each component classifier is correct or not. Our entropy-based diversity measure $E$ is defined as:

$$E = \frac{1}{m} \sum_{p=1}^{m} \sum_{c=1}^{M} (-P_{pc} \log P_{pc}),$$

(12)

where $m$ and $M$ are the number of patterns and the number of classes, respectively. In (12), $P_{pc}$ is the rate of the number of component classifiers which classify the pattern $p$ as class $c$. Let us consider five component classifiers in Fig. 6 when three of them output Class 1, one of them outputs Class 2, and the other rejects the classification. We ignore Classifier 5 with the reject decision. Thus, the $P_{p1}$ and $P_{p2}$ are calculated as 3/4 and 1/4, respectively.

![Fig. 6. Calculation of the rate of component classifiers which classify pattern $p$ as the same class.](image-url)
We use the following three objective functions for the design of ensemble classifiers.

\[ f_4(S) \]: Classification rate of an ensemble classifier,
\[ f_5(S) \]: Average classification rate of component classifiers,
\[ f_6(S) \]: Entropy of outputs from component classifiers.

Here, \( f_4(S) \) and \( f_5(S) \) are the accuracy criteria for an ensemble classifier and its component classifiers, respectively. On the other hand, \( f_6(S) \) is the entropy-based diversity measure.

In this paper, we examine the following four formulations:
- **SOP1**: Maximize \( f_4(S) \),
- **SOP2**: Maximize \( f_5(S) \),
- **MOP1**: Maximize \( f_4(S) \) and \( f_6(S) \),
- **MOP2**: Maximize \( f_5(S) \) and \( f_6(S) \).

For the first two single-objective formulations, we use a single-objective genetic algorithm (SOGA) with the same generation update scheme as NSGA-II (i.e., (\( \mu + \lambda \))-ES mechanism with \( \mu = \lambda \)). For the last two multiobjective formulations, we use NSGA-II. In both algorithms, we use the uniform crossover and the biased mutation operations explained in Section 3.2. Figure 7 shows the examples of the uniform crossover and the biased mutation operation. The biased mutation operation means that a larger probability is assigned to the mutation from a non-zero integer to a 0 than that from a 0 to a non-zero integer. Figure 8 shows transition probabilities on the biased mutation operation for the multi-classifier coding scheme. In Fig. 8, \( \alpha \) and \( \beta \) mean the probability from a non-zero integer to a 0 and that from a 0 to a non-zero integer, respectively. This is for efficiently decreasing the number of rules used in each component classifier. Furthermore, we employ the removal of unnecessary rules as explained in Section 3.2.
5. Computational experiments

5.1. Effect of multi-classifier coding and entropy-based diversity measure

We used six data sets with many numerical attributes available from the UCI machine learning repository (Table 1). We used the ten-fold cross-validation (10-CV) method to compare the four formulations in Section 4. The whole 10-CV procedure was executed two times (i.e., $2 \times 10$-CV: 20 runs) with different random seeds in this paper. In each run, we extracted 300 candidate rules for each class from training patterns. SOGA and NSGA-II were employed to find non-dominated ensemble classifiers from $300M$ extracted candidate rules where $M$ is the number of classes. The following parameter specifications were used in SOGA and NSGA-II:

- Population size $N_{\text{pop}}$: 200 strings,
- Crossover probability: 0.9,
- Biased mutation probabilities:
  
  $$p_m(0 \rightarrow a) = 1/300M, \quad p_m(a \rightarrow 0) = 0.1,$$

  where $a$ is a non-zero integer,
- Stopping condition: 5000 generations,
- Number of classifiers: 3, 5, 7.

Table 1. Data sets used in our computational experiments.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Attributes</th>
<th>Patterns</th>
<th>Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast W</td>
<td>9</td>
<td>683*</td>
<td>2</td>
</tr>
<tr>
<td>Diabetes</td>
<td>8</td>
<td>768</td>
<td>2</td>
</tr>
<tr>
<td>Glass</td>
<td>9</td>
<td>214</td>
<td>6</td>
</tr>
<tr>
<td>Heart C</td>
<td>13</td>
<td>297*</td>
<td>5</td>
</tr>
<tr>
<td>Sonar</td>
<td>60</td>
<td>208</td>
<td>2</td>
</tr>
<tr>
<td>Wine</td>
<td>13</td>
<td>178</td>
<td>3</td>
</tr>
</tbody>
</table>

* Incomplete patterns with missing values are not included.

Experimental results are summarized in Tables 2-7. These tables show average error rates on test patterns of the ensemble classifier with the best classification rate on training patterns. That is, we chose the best ensemble classifier in the final population of genetic rule selection with respect to the training data accuracy, and evaluated it in terms of the test data accuracy. In these tables, three figures in parentheses show some statistics about component classifiers. They are the average number of rules in each component classifier, the average total rule length in each component classifier, and the average number of component classifiers in the chosen ensemble classifiers. In each table, the best result in each row among the four formulations is highlighted by bold face. The best result in each table is underlined. From Tables 2-7, we can see that good results were often obtained from MOP2. Of course, the best combination of a formulation and the number of component classifiers depended on the problem.
With respect to the complexity of ensemble classifiers, we can see from Tables 2-7 the number of rules in each component classifier decreases using the classification rate of an ensemble classifier (i.e., \( f_i \)) as objective function(s). In some runs, the best ensemble classifier has a smaller number of component classifiers than the maximum number of component classifiers. This may mean that the number of component classifiers can be optimized by incorporating appropriate complexity measures as objective functions.

**Table 2. Average error rate on test patterns of the ensemble classifier with best training accuracy for Breast W data set.**

<table>
<thead>
<tr>
<th># of classifiers</th>
<th>SOP1 ( (f_i) )</th>
<th>SOP2 ( (f_j) )</th>
<th>MOP1 ( (f_a, f_b) )</th>
<th>MOP2 ( (f_c, f_d) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3.22</td>
<td><strong>3.15</strong></td>
<td>4.02</td>
<td><strong>3.15</strong></td>
</tr>
<tr>
<td></td>
<td>(6.8, 18.0, 3.0)</td>
<td>(12.4, 29.8, 3.0)</td>
<td>(8.8, 15.9, 3.0)</td>
<td>(9.0, 19.5, 3.0)</td>
</tr>
<tr>
<td>5</td>
<td>3.59</td>
<td>3.15</td>
<td>3.66</td>
<td><strong>2.71</strong></td>
</tr>
<tr>
<td></td>
<td>(4.3, 11.0, 5.0)</td>
<td>(11.7, 28.1, 5.0)</td>
<td>(5.4, 9.9, 5.0)</td>
<td>(8.7, 19.7, 5.0)</td>
</tr>
<tr>
<td>7</td>
<td>3.15</td>
<td><strong>2.85</strong></td>
<td>4.61</td>
<td>3.15</td>
</tr>
<tr>
<td></td>
<td>(3.3, 8.5, 6.4)</td>
<td>(11.8, 28.5, 7.0)</td>
<td>(3.6, 6.5, 6.6)</td>
<td>(8.0, 17.8, 7.0)</td>
</tr>
</tbody>
</table>

**Table 3. Average error rate on test patterns of the ensemble classifier with best training accuracy for Diabetes data set.**

<table>
<thead>
<tr>
<th># of classifiers</th>
<th>SOP1 ( (f_i) )</th>
<th>SOP2 ( (f_j) )</th>
<th>MOP1 ( (f_a, f_b) )</th>
<th>MOP2 ( (f_c, f_d) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>25.00</td>
<td><strong>23.30</strong></td>
<td>24.92</td>
<td>23.88</td>
</tr>
<tr>
<td></td>
<td>(11.0, 29.3, 3.0)</td>
<td>(18.6, 47.4, 3.0)</td>
<td>(9.2, 22.0, 3.0)</td>
<td>(12.7, 30.1, 3.0)</td>
</tr>
<tr>
<td>5</td>
<td>24.54</td>
<td>24.34</td>
<td>24.87</td>
<td><strong>23.95</strong></td>
</tr>
<tr>
<td></td>
<td>(6.7, 17.6, 5.0)</td>
<td>(15.0, 37.5, 5.0)</td>
<td>(5.8, 13.0, 5.0)</td>
<td>(9.9, 23.0, 5.0)</td>
</tr>
<tr>
<td>7</td>
<td>24.73</td>
<td><strong>23.95</strong></td>
<td>25.39</td>
<td>24.74</td>
</tr>
<tr>
<td></td>
<td>(5.1, 13.4, 6.9)</td>
<td>(14.0, 35.3, 7.0)</td>
<td>(4.1, 9.2, 6.5)</td>
<td>(8.8, 20.3, 7.0)</td>
</tr>
</tbody>
</table>

**Table 4. Average error rate on test patterns of the ensemble classifier with best training accuracy for Glass data set.**

<table>
<thead>
<tr>
<th># of classifiers</th>
<th>SOP1 ( (f_i) )</th>
<th>SOP2 ( (f_j) )</th>
<th>MOP1 ( (f_a, f_b) )</th>
<th>MOP2 ( (f_c, f_d) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>32.40</td>
<td>31.52</td>
<td>32.29</td>
<td><strong>31.16</strong></td>
</tr>
<tr>
<td></td>
<td>(13.3, 38.3, 3.0)</td>
<td>(23.6, 68.0, 3.0)</td>
<td>(13.2, 37.6, 3.0)</td>
<td>(20.0, 57.3, 3.0)</td>
</tr>
<tr>
<td>5</td>
<td>32.25</td>
<td><strong>31.33</strong></td>
<td>31.76</td>
<td>32.05</td>
</tr>
<tr>
<td></td>
<td>(8.5, 24.5, 5.0)</td>
<td>(25.4, 73.5, 5.0)</td>
<td>(9.6, 27.1, 5.0)</td>
<td>(22.8, 65.8, 5.0)</td>
</tr>
<tr>
<td>7</td>
<td>32.51</td>
<td>32.05</td>
<td><strong>31.34</strong></td>
<td>32.74</td>
</tr>
<tr>
<td></td>
<td>(6.2, 18.0, 7.0)</td>
<td>(27.3, 79.0, 7.0)</td>
<td>(6.9, 19.6, 7.0)</td>
<td>(24.8, 71.7, 7.0)</td>
</tr>
</tbody>
</table>
Table 5. Average error rate on test patterns of the ensemble classifier with best training accuracy for Heart C data set.

<table>
<thead>
<tr>
<th># of classifiers</th>
<th>SOP1 ($f_4$)</th>
<th>SOP2 ($f_5$)</th>
<th>MOP1 ($f_4, f_6$)</th>
<th>MOP2 ($f_5, f_6$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>47.78 (26.6, 78.1, 3.0)</td>
<td>49.63 (80.4, 233.2, 3.0)</td>
<td><strong>47.24</strong> (30.6, 89.2, 3.0)</td>
<td>50.48 (64.3, 186.3, 3.0)</td>
</tr>
<tr>
<td>5</td>
<td>48.61 (16.3, 48.2, 5.0)</td>
<td>47.79 (78.3, 226.0, 5.0)</td>
<td><strong>47.61</strong> (16.2, 47.0, 5.0)</td>
<td>47.43 (62.8, 181.4, 5.0)</td>
</tr>
<tr>
<td>7</td>
<td><strong>45.94</strong> (11.6, 34.2, 7.0)</td>
<td>47.46 (72.4, 208.3, 7.0)</td>
<td><strong>47.43</strong> (10.7, 30.8, 7.0)</td>
<td>47.43 (61.8, 177.9, 7.0)</td>
</tr>
</tbody>
</table>

Table 6. Average error rate on test patterns of the ensemble classifier with best training accuracy for Sonar data set.

<table>
<thead>
<tr>
<th># of classifiers</th>
<th>SOP1 ($f_4$)</th>
<th>SOP2 ($f_5$)</th>
<th>MOP1 ($f_4, f_6$)</th>
<th>MOP2 ($f_5, f_6$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>21.46 (7.2, 14.3, 3.0)</td>
<td>21.45 (10.6, 21.1, 3.0)</td>
<td>25.06 (8.4, 16.8, 3.0)</td>
<td><strong>21.44</strong> (8.6, 17.1, 3.0)</td>
</tr>
<tr>
<td>5</td>
<td>20.49 (4.6, 9.2, 5.0)</td>
<td>21.95 (9.6, 19.1, 5.0)</td>
<td>22.67 (5.7, 11.4, 5.0)</td>
<td><strong>19.30</strong> (7.5, 14.9, 5.0)</td>
</tr>
<tr>
<td>7</td>
<td>23.37 (3.2, 6.4, 6.7)</td>
<td><strong>21.24</strong> (8.9, 17.7, 7.0)</td>
<td>25.26 (3.8, 7.6, 7.0)</td>
<td>22.43 (6.6, 13.1, 7.0)</td>
</tr>
</tbody>
</table>

Table 7. Average error rate on test patterns of the ensemble classifier with best training accuracy for Wine data set.

<table>
<thead>
<tr>
<th># of classifiers</th>
<th>SOP1 ($f_4$)</th>
<th>SOP2 ($f_5$)</th>
<th>MOP1 ($f_4, f_6$)</th>
<th>MOP2 ($f_5, f_6$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4.18 (4.0, 10.6, 3.0)</td>
<td><strong>3.94</strong> (7.8, 21.0, 3.0)</td>
<td>8.73 (9.1, 23.1, 3.0)</td>
<td>7.06 (8.3, 20.7, 3.0)</td>
</tr>
<tr>
<td>5</td>
<td><strong>3.10</strong> (3.0, 7.9, 4.9)</td>
<td>3.68 (9.3, 25.6, 5.0)</td>
<td>9.54 (6.8, 17.6, 5.0)</td>
<td>3.35 (8.3, 21.8, 5.0)</td>
</tr>
<tr>
<td>7</td>
<td>4.51 (2.5, 6.6, 6.3)</td>
<td>3.71 (11.6, 32.4, 7.0)</td>
<td><strong>2.29</strong> (5.6, 14.2, 6.5)</td>
<td>8.19 (7.8, 20.5, 7.0)</td>
</tr>
</tbody>
</table>

For further examining the validity of the use of the component classifier accuracy and the diversity measure (i.e., $f_5$ and $f_6$), we show the non-dominated ensemble classifiers obtained by a single run of each algorithm for each data set in Figs. 9-14. The number of component classifiers was specified as five. We can see that a large number of non-dominated ensemble classifiers were obtained from the multiobjective formulations. Some of the obtained non-dominated ensemble classifiers by MOP2 have very good generalization ability on test data. Of course, ensemble classifiers with higher entropy values are not always better than those with
smaller entropy values in terms of their generalization ability on test data. We need further experiments to examine the relation between the accuracy of ensemble classifiers and the diversity of component classifiers.

Fig. 9. Classification accuracy by a single run for Breast W data set.

Fig. 10. Classification accuracy by a single run for Diabetes data set.
Fig. 11. Classification accuracy by a single run for Glass data set.

Fig. 12. Classification accuracy by a single run for Heart C data set.

Fig. 13. Classification accuracy by a single run for Sonar data set.
5.2. Comparison with other methods

We compare the proposed method with other ensemble methods in Table 8. The proposed method in Table 8 means MOP2 with five component classifiers. In the fuzzy version, we used the same algorithm as the proposed method except that homogeneous fuzzy partitions were used instead of inhomogeneous interval partitions (see [32] for fuzzy rule-based ensemble classifier design). Rule selection in this table is the original genetic rule selection method with binary strings. In this method, the accuracy on training data and the number of rules in a string were used as objective functions (i.e., Eq.(11)). The results of the other ensemble methods are cited from [35].

From the comparison of the proposed method with the fuzzy version, we can see that interval rule-based ensemble classifiers outperform fuzzy rule-based ensemble classifiers in four out of the six data sets.

Comparing with the original rule selection, the genetic rule selection can be improved by using the multi-classifier coding scheme and the diversity measure. The proposed method improves the generalization ability for all data sets.

Regarding other ensemble methods, the proposed method was the best ensemble method for Breast W data set. For Diabetes data set, the proposed method was comparable to the other ensemble methods. For Glass data set, the proposed method was comparable to NN-based methods and inferior to C4.5-based methods. For Sonar data set, the proposed method was comparable to C4.5-based methods and inferior to NN-based methods. It should be noted that the test data accuracy by the proposed method in Table 8 was obtained from the best ensemble classifier in terms of training data accuracy, not test data accuracy. From Figs. 9-14, we can see that there were very good ensemble classifiers with the middle diversity. If we could specify an
appropriate diversity beforehand, we will be able to choose the best ensemble classifier with the highest test data accuracy more than the other ensemble methods.

Table 8. Comparison with other methods on test data accuracy by 10-CV.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Proposed method</th>
<th>Fuzzy version</th>
<th>Rule Selection</th>
<th>Bagging NN</th>
<th>Bagging C4.5</th>
<th>Boosting (Arc) NN</th>
<th>Boosting (Arc) C4.5</th>
<th>Boosting (Ada) NN</th>
<th>Boosting (Ada) C4.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast W</td>
<td><strong>2.71</strong></td>
<td>3.73</td>
<td>3.37</td>
<td>3.4</td>
<td>3.7</td>
<td>3.8</td>
<td>4</td>
<td>4</td>
<td>3.5</td>
</tr>
<tr>
<td>Diabetes</td>
<td>23.95</td>
<td>24.21</td>
<td>24.4</td>
<td><strong>22.8</strong></td>
<td>24.4</td>
<td>24.4</td>
<td>26</td>
<td>23.3</td>
<td>25.7</td>
</tr>
<tr>
<td>Glass</td>
<td>32.05</td>
<td>38.10</td>
<td>32.51</td>
<td>33.1</td>
<td>25.8</td>
<td>32</td>
<td>25.5</td>
<td>31.1</td>
<td><strong>23.3</strong></td>
</tr>
<tr>
<td>Heart C</td>
<td>47.61</td>
<td><strong>45.06</strong></td>
<td>49.95</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Sonar</td>
<td>19.3</td>
<td>21.19</td>
<td>22.12</td>
<td>16.8</td>
<td>25.3</td>
<td><strong>12.9</strong></td>
<td>21.5</td>
<td>13</td>
<td>21.7</td>
</tr>
<tr>
<td>Wine</td>
<td>3.35</td>
<td><strong>2.57</strong></td>
<td>5.93</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

5.3. Effect of a default rule

We also examined the effect of default rules. When there is no compatible rule for a pattern \( x \), our default rule classifies \( x \) as the majority class of the training patterns. Table 9 shows the results of the proposed method with no default rule and that with this default rule. The proposed method in Table 8 means MOP2 with five component classifiers. We can see that the generalization ability was slightly deteriorated by using the default rule for most data sets. This may be because the use of the default rule has a negative effect on the genetic search for good ensemble classifiers.

Table 9. Comparison of the proposed method with no default rule and that with a default rule on test data accuracy by 10-CV.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Proposed method with no default rule</th>
<th>Proposed method with the default rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast W</td>
<td>2.71 ( 8.7, 19.7, 5.0)</td>
<td>3.37 ( 8.5, 19.6, 5.0)</td>
</tr>
<tr>
<td>Diabetes</td>
<td>23.95 ( 9.9, 23.0, 5.0)</td>
<td>23.75 (10.2, 23.6, 5.0)</td>
</tr>
<tr>
<td>Glass</td>
<td>32.05 (22.8, 65.8, 5.0)</td>
<td>32.49 (21.0, 60.7, 5.0)</td>
</tr>
<tr>
<td>Heart C</td>
<td>47.61 (62.8, 181.4, 5.0)</td>
<td>48.61 (64.2, 185.4, 5.0)</td>
</tr>
<tr>
<td>Sonar</td>
<td>19.30 (7.5, 14.9, 5.0)</td>
<td>20.02 (7.5, 19.9, 5.0)</td>
</tr>
<tr>
<td>Wine</td>
<td>3.35 (8.3, 21.8, 5.0)</td>
<td>6.81 (7.5, 19.9, 5.0)</td>
</tr>
</tbody>
</table>
6. Conclusions and future remarks

In this paper, we examined the effect of the multi-classifier coding scheme and the entropy-based diversity measure on the design of interval rule-based ensemble classifiers. Experimental results showed that we can obtain better generalization ability by the proposed method with integer strings than the original genetic rule selection method with binary strings. It was also shown that the multiobjective formulation with the average accuracy of component classifiers and the entropy-based diversity measure can find a number of ensemble classifiers with good generalization ability.

As future research issues, we will examine the relationship between the accuracy and the diversity of ensemble classifiers. The use of different diversity measures will give us some hints to improve our approach. We will further examine other objective functions such as the number of rules, rule length, the number of component classifiers, and so on. The use of inhomogeneous fuzzy partitions is also an interesting future research issue.

Acknowledgement

This work was partially supported by Grant-in-Aid for Scientific Research on Young Scientists (B): KAKENHI (18700228) and Foundation for C&C Promotion.

References


