Rotation Effect of Training Data Subsets in Parallel Distributed Fuzzy Genetics-based Machine Learning

Y. Nojima*
Department of Computer Science and Intelligent Systems, Osaka Prefecture University, Osaka 599-8531, Japan
Fax:+81-72-254-9198
E-mail: nojima@cs.osakafu-u.ac.jp
*Corresponding author

S. Mihara
Department of Computer Science and Intelligent Systems, Osaka Prefecture University, Osaka 599-8531, Japan
Fax:+81-72-254-9198
E-mail: mihara@ci.cs.osakafu-u.ac.jp

H. Ishibuchi
Department of Computer Science and Intelligent Systems, Osaka Prefecture University, Osaka 599-8531, Japan
Fax:+81-72-254-9198
E-mail: hisaoi@cs.osakafu-u.ac.jp

Abstract: Genetics-based machine learning (GBML) is one of the promising evolutionary algorithms for classifier design. It can construct rule-based classifiers from numerical data sets. These classifiers have comparable classification ability to those by other machine learning techniques and are more understandable than the others. Although GBML has high search ability, it needs long computation time especially for large data sets because a large number of patterns should be classified for string evaluation. We have already applied a parallel distributed implementation to our fuzzy GBML. In our method, we divide not only a population but also a training data set into subgroups. A sub-population and a training data subset are assigned to one CPU core. To avoid the overfitting to certain training data subset, the re-assignment of the training data subsets is periodically performed. The main advantage is that the speed-up rate of our method is larger than the number of used CPU cores without deterioration in the test data accuracy. In this paper, we examine the effect of the re-assignment of the training data subsets together with the scalability of our parallel distributed approach using different numbers of sub-populations (i.e., CPU cores).

Keywords: fuzzy genetics-based machine learning, parallel distributed implementation, pattern classification, large data sets.

Reference to this paper should be made as follows: Nojima, Y., Mihara, S., and Ishibuchi, H. (2010) ‘Rotation effect of training data subsets in parallel distributed fuzzy genetics-based machine learning’ Proceeding of the 14th Asia Pacific Symposium on Intelligent and Evolutionary Systems.

Biographical notes: Dr. Yusuke Nojima received the BS and MS Degrees in mechanical engineering from Osaka Institute of Technology, Osaka, Japan, in 1999 and 2001, respectively, and the Ph.D. degree from Kobe University, Hyogo, Japan, in 2004. Since 2004, he has been with Osaka Prefecture University where he is currently an Assistant Professor in Department of Computer Science and Intelligent Systems. His research interests include multiobjective genetic fuzzy systems, parallel distributed data mining, and ensemble classifier design. Dr. Nojima is a Program Co-Chair of GEFS 2010 and GEFS 2011.

Mr. Shingo Mihara received the BS degree from Osaka Prefecture University, Japan, in 2010. He is currently a master course student in Department of Computer Science and Intelligent Systems,
1 Introduction

Genetic algorithms have been frequently used as a machine learning tool for designing rule-based classification systems. These methods are referred to as genetics-based machine learning (GBML) algorithms. GBML can generate a set of if-then rules (i.e., classifier) as understandable knowledge from a data set (Bull, et al., 2008; Freitas, 2002). Although GBML algorithms are comparable in accuracy to other machine learning techniques, their computation cost of GA is a significant drawback especially for large data sets.

The parallel implementation of a genetic algorithm is a promising approach to reduce the computation time (Alba and Tomassini, 2002; Cantu-Paz, 1997). When we use island models, each sub-population can be assigned to a different processing node (e.g., CPU core). On the other hand, from the viewpoint of data mining, data reduction such as feature selection and instance selection (Liu and Motoda, 1998a, 1998b; Cano et al., 2005, 2006) are frequently-used approaches. As a kind of data reduction approach, in (Bacardit et al., 2004), a training data set is divided into some non-overlapping groups. Then GBML uses a different subgroup for each generation.

Fuzzy rule-based systems have also been optimized in the framework of GBML which is referred to as genetic fuzzy systems (GFS) (Cordon, et al., 2001; Herrera, 2008). There exists the same scalability problem as general GBML algorithms. We have proposed parallel distributed implementation of genetic fuzzy rule selection for pattern classification problems in (Nojima, et al., 2009). In our parallel distributed implementation, we divide not only a population but also a training data set into subgroups. A sub-population and a training data subset are assigned to one CPU core. In order to avoid the overfitting of each sub-population to a specific training data subset, training data subset re-assignment is periodically performed (e.g., every 100 generation). In (Nojima, et al., 2009, 2010a), the experimental results showed that we can reduce the computational time to 1/9 with no deterioration in the test data accuracy when we used three CPU cores in parallel.

In our previous paper (Nojima, et al., 2010b), we applied the parallel distributed implementation of the genetic fuzzy rule selection (Nojima, et al., 2009) to the fuzzy GBML (Ishibuchi, et al. 2005). Our fuzzy GBML can optimize the combination of antecedent conditions and the number of rules. Although there is a high possibility that fuzzy GBML can obtain more accurate classifiers than genetic fuzzy rule selection thanks to the larger search space of fuzzy GBML, the computation cost of fuzzy GBML is much heavier than genetic fuzzy rule selection. In (Nojima, et al., 2010b), we confirmed the applicability of our parallel distributed implementation. In this paper, we deeply analyze our parallel distributed fuzzy GBML algorithm. Especially, we examine the effect of the rotation of the training data subsets together with the scalability on the number of sub-populations (i.e., the number of used CPU cores).

This paper is organized as follows. First we explain a fuzzy rule-based classifier and our parallel distributed fuzzy GBML algorithm in Section 2. Next we examine the effect of the rotation of training data subsets and the number of sub-populations through computational experiments in Section 3. Then in Section 4, we examine the behavior of our approach during evolution in detail. Finally we conclude this paper in Section 5.

2 Parallel distributed implementation of fuzzy genetics-based machine learning

In the previous paper (Nojima et al., 2010), we proposed parallel distributed implementation of fuzzy GBML for the design of fuzzy classifiers. In this section, we will give brief explanations on fuzzy classifiers, our fuzzy GBML, and its parallel distributed implementation.

2.1 Fuzzy rule-based classifier

Let us assume that we have \( m \) training (i.e., labeled) patterns \( x_p = (x_{p1}, ..., x_{pm}) \), \( p = 1, 2, ..., m \) from \( M \) classes in an \( n \)-dimensional pattern space where \( x_{pi} \) is the attribute value of the \( p \)th pattern for the \( i \)th attribute (\( i = 1, 2, ..., n \)). For the simplicity of explanation, we assume that all the attribute values have already been normalized into real numbers in the unit interval \([0, 1]\). For our classification problem, we use fuzzy if-then rules of the following type:

\[
\text{Rule } R_{q}: \text{ If } x_{i1} \text{ is } A_{q1} \text{ and } ... \text{ and } x_{ip} \text{ is } A_{qm} \text{ then Class } C_{q} \text{ with } CF_{q},
\]

where \( R_q \) is the label of the \( q \)th fuzzy rule, \( \textbf{x} = (x_1, ..., x_n) \) is an \( n \)-dimensional pattern vector, \( A_{qi} \) is an antecedent fuzzy set (\( i = 1, 2, ..., n \)), \( C_q \) is a class label, and \( CF_q \) is a rule...
weight. We denote the antecedent fuzzy sets of \( R_q \) as a fuzzy vector \( A_q = (A_{q1}, A_{q2}, ..., A_{qn}) \).

We use 14 fuzzy sets in four fuzzy partitions with different granularities in Fig. 1. In addition to those 14 fuzzy sets, we also use the domain interval \([0, 1]\) as a special antecedent fuzzy set for representing a \textquoteleft don't care\textquoteright condition.

![Fig. 1. Homogeneous fuzzy partitions used in this paper.](image)

The consequent class \( C_q \) and the rule weight \( CF_q \) of each fuzzy rule \( R_q \) are specified from training patterns compatible with its antecedent part \( A_q = (A_{q1}, A_{q2}, ..., A_{qn}) \) in a heuristic manner (Ishibuchi, et al., 2004).

### 2.2 Hybrid genetics-based machine learning

Our fuzzy GBML is a hybrid version of Pittsburgh approach and Michigan approach (Ishibuchi, et al., 2005). Its main framework is based on Pittsburgh approach in which a rule set is codified as a string. Michigan approach is used as a local search in which a rule is codified as a string.

Each fuzzy rule \( R_q \) is represented by its antecedent fuzzy sets \( A_{qi} \) \((i = 1, 2, ..., n)\) as an integer substring of length \( n \), where \( n \) is the dimensionality of the pattern space (i.e., \( n \) is the number of attributes of each pattern). We use 15 symbols (e.g., 0, 1, ..., 9, a, b, ..., e) to represent \textquoteleft don't care\textquoteright and the 14 antecedent fuzzy sets as shown in Fig. 1.

A rule set (i.e., classifier) \( S \) is handled as an individual and coded as a concatenated integer string where each substring of length \( n \) represents a single fuzzy if-then rule. The number of fuzzy rules in each rule set is not fixed in our fuzzy GBML. This means that we use strings of variable length as individuals.

The procedure of our fuzzy GBML is as follows.

#### Step 1: Generate an initial population of \( N_{\text{pop}} \) rule sets by a heuristic rule initialization method where \( N_{\text{pop}} \) is the population size.

#### Step 2: Evaluate the initial population.

#### Step 3: Select a pair of parent rule sets from the current population by binary tournament selection.

#### Step 4: Generate an offspring rule set by inheriting a randomly specified subset of fuzzy rules from each parent with a prespecified crossover probability \( P_c \).

#### Step 5: Randomly replace each antecedent fuzzy set in the offspring rule set with another fuzzy set as a mutation operation with a prespecified mutation probability \( P_m \).

#### Step 6: Apply a single iteration of the following Michigan-style GBML algorithm to the offspring rule set with the probability of 0.5.

6.1. Classify each training pattern by the rule set \( S \). The fitness value of each rule is calculated as the number of correctly classified training patterns by that rule.

6.2. Generate \( N_{\text{replace}} \) fuzzy rules by the genetic operations and the heuristic rule initialization.

6.3. Replace the worst \( N_{\text{replace}} \) fuzzy rules in \( S \) with the newly generated \( N_{\text{replace}} \) fuzzy rules.

#### Step 7: Return to Step 3 until \( N_{\text{pop}} \) offspring rule sets are generated.

#### Step 8: Remove unnecessary fuzzy rules with non-positive rule weights from each offspring rule set.

#### Step 9: Evaluate the offspring population.

#### Step 10: Combine the current population and the offspring population into a merged one. Then choose the best \( N_{\text{pop}} \) rule sets from the merged population to construct the next population.

#### Step 11: If a prespecified stopping condition is not satisfied, return to Step 3. Otherwise terminate the execution of the algorithm and choose the best rule set in the current population as the final classifier.

The heuristic rule initialization method in Step 1 and Step 6.1 is a method for generating a fuzzy if-then rule from one pattern of the training data (misclassified patterns in Step 6.1). Each antecedent condition is randomly specified according to the compatibility grade of the selected pattern. Once all the antecedent fuzzy sets are specified, each antecedent condition is replaced with a \textquoteleft don't care\textquoteright condition according to the compatibility grade and the rule weight of each fuzzy rule in the rule set \( S \). The misclassified rate (%) on the training data by \( S \) is calculated as 

\[
\text{Misclassified Rate} = \frac{\text{Number of misclassified patterns by } S}{\text{Total number of patterns}} \times 100%
\]

The single winner rule is identified by using the compatibility grade and the rule weight of each fuzzy rule in the rule set \( S \). The input pattern \( x_p \) is classified as the consequent class \( C_q \) of the winner rule \( R_q \). When multiple fuzzy rules with different consequent classes have the same maximum value, the classification of \( x_p \) is rejected. If there is no compatible fuzzy rule with \( x_p \), its classification is also rejected.

We use the following weighted-sum fitness function for obtaining accurate and simple fuzzy classifiers.

\[
\text{fitness}(S) = w_1 f_1(S) + w_2 f_2(S) + w_3 f_3(S)
\]

where \( w_1, w_2 \) and \( w_3 \) are non-negative weights. Of course, this fitness function is to be minimized in our fuzzy GBML.
It should be noted that a full scan of all the training patterns is needed whenever a new fuzzy if-then rule is generated in order to specify the consequent class and the rule weight. This process leads to the increase in the computation cost of our fuzzy GBML algorithm.

Due to the page limitation, we do not explain the details of the other genetic operations. Please see (Ishibuchi, et al., 2005; Ishibuchi and Nojima, 2007; Nojima, et al., 2010) for more details.

2.3 Parallel distributed implementation

We use a workstation with multiple CPU cores for the parallel distributed implementation of our fuzzy GBML. Our implementation can be written as follows.

STEP 1: Generate $N_{\text{pop}}$ integer strings as an initial population by the heuristic rule initialization method where $N_{\text{pop}}$ is the population size.

STEP 2: Randomly divide the current population $P$ with $N_{\text{pop}}$ integer strings into $N_{\text{CPU}}$ sub-populations of the same size $\{P_1, P_2, \ldots, P_{N_{\text{CPU}}}\}$. Randomly divide the training data set $D$ with $m$ patterns into training data subsets of the same size $\{D_1, D_2, \ldots, D_{N_{\text{CPU}}}\}$ as Fig 2.

STEP 3: Assign a sub-population and a training data subset to each of the $N_{\text{CPU}}$ CPU cores.

STEP 4: Perform the fuzzy GBML algorithm in each CPU core.

STEP 5: If the prespecified termination condition is satisfied, go to STEP 8.

STEP 6: If the prespecified rotation interval is satisfied, rotate the $N_{\text{CPU}}$ training data subsets as shown in Fig. 2 over the $N_{\text{CPU}}$ CPU cores.

STEP 7: Return to STEP 4.

STEP 8: Choose the best rule set $S_{\text{best}}$ from the whole population in terms of the training data accuracy (i.e., $f_1(S)$) for the whole training data set.

In STEP 2, we divided a training data set into subsets remaining the class balance of the whole training data set.

In STEP 4, the Step 3 to Step 10 of the fuzzy GBML explained in Section 2.2 is performed in each CPU core.

It should be noted that, in STEP 6 of our parallel distributed approach, the consequent class and the rule weight of each rule in all the rule sets in the current sub-population are updated by using the newly assigned training data subset. Whenever these parameters are specified or updated, the compatibility grade of each training pattern in the training data subset is stored to reduce their repeated calculation.

3 Computational experiments

In this paper, we examine the effect of the rotation of the training data subsets together with the scalability of our parallel distributed approach using different numbers of sub-populations (i.e., CPU cores).

3.1 Experimental setting and data sets

In our computational experiments, we used three data sets in Table 1 available from the KEEL-dataset repository (http://keel.es; Alcala-Fedz, et al., 2010). The generalization ability was evaluated by iterating the ten-fold cross validation procedure (10CV) three times with different random seeds (i.e., 30 runs in total).

Table 1. Data sets used in our computational experiments.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Number of attributes</th>
<th>Number of patterns</th>
<th>Number of classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phoneme</td>
<td>5</td>
<td>5404</td>
<td>2</td>
</tr>
<tr>
<td>Satimage</td>
<td>36</td>
<td>6435</td>
<td>6</td>
</tr>
<tr>
<td>Pendig</td>
<td>16</td>
<td>10992</td>
<td>10</td>
</tr>
</tbody>
</table>

We used a workstation with two Xeon 2.93 GHz quad core processors (i.e., eight CPU cores in total). We codified our parallel distributed fuzzy GBML algorithm by Java 1.6 on Windows 7 operating system. In this paper, we examined four different specifications on the number of CPU cores: one, three, five, and seven CPU cores. We specified the population size as 210. That is, each sub-population size for our parallel distributed implementation was 70, 42, and 30, respectively. The remaining setting was as follows:

Table 2. Average results over 3x10CV by the non-parallel fuzzy GBML for three data sets

<table>
<thead>
<tr>
<th>Rotation interval</th>
<th>Training data accuracy [%]</th>
<th>Test data accuracy [%]</th>
<th>Number of rules</th>
<th>Average rule length</th>
<th>Computation time [min]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phoneme</td>
<td>84.59</td>
<td>82.98</td>
<td>16.50</td>
<td>3.05</td>
<td>60.3</td>
</tr>
<tr>
<td>Satimage</td>
<td>85.72</td>
<td>83.94</td>
<td>18.10</td>
<td>4.03</td>
<td>128.4</td>
</tr>
<tr>
<td>Pendig</td>
<td>96.42</td>
<td>95.22</td>
<td>34.50</td>
<td>4.76</td>
<td>392.9</td>
</tr>
</tbody>
</table>
The number of fuzzy rules in each initial rule set: 30,
Upper limit on the number of fuzzy rules: 60,
Lower limit on the number of fuzzy rules: 1,
Probability of don’t care \((P_{\text{don't care}}}) = (n-1)/n,
\((n\) is the number of attributes),
Probability of the Michigan-style algorithm: 0.5,
Crossover probability in the main part \((P_C): 0.9,
Crossover probability in the Michigan-style part: 0.9,
Mutation probability in the main part \((P_M): 1/ (n \mid S\mid),
Mutation probability in the Michigan-style part: 1/ n,
Termination condition: 10,000 generations,
The weight vector in Eq. (2): \(w = (100, 1, 1)\).

We used a large value for the first element of the
weight vector in Eq. (2) to find accurate classifiers. Since
obtained classifiers strongly depend on the weight vector
specification, we need further experiments with different
specifications as a future study.

We examined the following specifications of the
rotation interval for training data subsets rotation.

Rotation interval (generation): 10, 50, 100, 200, 500, None,
where “None” means that training data subsets were not
rotated. When the rotation interval was specified as 50,
training data subsets were rotated every 50 generations.

### 3.2 Results by non-parallel fuzzy GBML

First, we show the average results by the non-parallel
fuzzy GBML. The experiments were done by individually
using three CPU cores in the same workstation. Table 2
shows the results on the average classification rates on the
whole training data and the test data, the average number of
rules and the average rule length in the obtained classifier.
The average rule length means the average number of
antecedent conditions except for don’t care per rule. The
average computation time for each data set is also listed in
Table 2. We can see that the computation time strongly
depends on the number of patterns.

### 3.3 Results by parallel distributed fuzzy GBML

Figure 3 shows the average results on the whole
training data accuracy by our parallel distributed approach
with different specifications for the Phoneme data set. The
black boxes represent better results by our parallel
distributed approach than that of the non-parallel one shown
in Table 2. The better test data accuracy was obtained by
our parallel distributed approach than that by the non-
parallel one in more cases (i.e., nine of 18 specifications).

Fig. 3. Training data accuracy for the Phoneme data set.

Fig. 4. Test data accuracy for the Phoneme data set.

Fig. 5. The number of rules for the Phoneme data set.

Figures 5 and 6 show the average results on the
number of fuzzy rules and the average rule length in the
obtained classifier. The frequent rotations of the training
data subsets decreased the number of rules except for the
cases using seven sub-populations. The average rule length
was almost the same in most cases. The effect of the number
of sub-populations seems to be small with respect to the
number of rules and the average rule length.
Figure 7 shows the average computation time per run. The maximum value of the z-axis (i.e., “Computation time (min)”) is the same as the computation time by the non-parallel distributed approach. We can see that our parallel distributed approach can drastically reduce the computation time depending on the number of sub-populations. The speeding-up rates were larger than the number of sub-populations (i.e., the number of the used CPU cores) in all the specifications.

Figures 8 - 12 show the average results by our parallel distributed approach with different specifications for the Satimage data set. We can observe the similar results to those for the Phoneme data set. The most different point from the previous observation is that the better test data accuracy by our parallel distributed approach than that by the non-parallel one was obtained even in the cases where the number of sub-populations was seven. Another interesting observation is that there exists a trade-off between the number of rules and the average rule length according to the rotation interval (see Figs. 10 and 11). As in Fig. 7, we can see that our parallel distributed approach can drastically reduce the computation time depending on the number of sub-populations in Fig. 12. The speeding-up rates were larger than the number of sub-populations (i.e., the number of used CPU cores) in all the specifications.
Fig. 12. Computation time for the Satimage data set. The range of the z-axis is [0.0, 128.4].

Figures 13 - 17 show the average results by our parallel distributed approach with different specifications for the Pendig data set. The better results on the training data accuracy and the test data accuracy were not obtained by our parallel distributed approach for the Pendig data set. Figures 13 and 14 clearly show that both the training data accuracy and test data accuracy were deteriorated as the number of sub-populations increased. As in Figs. 8 and 9, there may exist an appropriate rotation interval for each specification on the number of sub-populations. As in Figs. 7 and 12, our parallel distributed approach can drastically reduce the computation time shown in Fig. 17. The speeding-up rates were larger than the number of sub-populations in all the specifications.

4 Analysis on evolutionary process

In this section, we examined how each (sub-) population was changed during evolution. We focussed on the following three specifications:

(1) Parallel distributed approach with rotation of seven training data subsets (Rotation interval: 50 generation),
(2) Parallel distributed approach with no rotation of seven training data subsets,
(3) Non-parallel approach.

It should be noted that the first approach (with rotation of seven training data subsets at every 50 generations) had
the shortest computation time among all specifications of our parallel distributed implementation in this paper.

Figure 18 shows how the training data accuracy was improved during the evolution for 10,000 generations by each approach for the Satimage data set. In the case of the parallel distributed approach with/without rotation, we recalculated the fitness of each classifier using the whole training data set to choose the best classifier at each generation. This recalculcation, which is only for plotting Fig. 18, was not used in the evolution. The average error rate on the whole training data in Fig. 18 was calculated over 3x10CV using the best classifier at each generation of each run. The average error rate on the test data in Fig. 19 was also calculated using the same best classifier. In these figures, we show the results at every 50 generations for the first approach (i.e., just after the training data subsets were rotated). For the second and third approaches, we show the results at every 200 generations in Fig. 18 and Fig. 19.

In Fig. 18, we can observe early convergence phenomena of the second and third approaches. We can also see that the simple parallel distributed implementation with no rotation (i.e., the second approach) led to severe performance deterioration. However, such a negative effect of parallel distributed implementation seems to be remedied by the rotation of the training data subsets.

In Fig. 19, the error rates by the second and third approaches were converged at early generations according to the training data accuracy as Fig. 18. While the error rates on the whole training data by the first and third approaches were almost the same at the 10,000th generation in Fig. 18, the error rate on the test data by the first approach was clearly better than that by the third approach at the final generation in Fig. 19.

Figure 20 shows the error rates on the training data subsets by the first approach (Rotation interval: 50 generation). Each thin gray line in Fig. 20 represents the error rate of a classifier with the best fitness value in each sub-population on the training data subset. Two black lines are the same in Fig. 18. The periodicical deterioration in accuracy can be observed at every 50 generation in the right window of Fig. 20. This may be because the distribution of each training data subset was different each other even by remaining the class balance among the training data subsets.

Figure 21 shows the error rates on the assigned training data subsets to each sub-population by the second approach (no rotation). All the error rates on the assigned training data subsets were better than those by the first approach in Fig. 20, but the training data accuracy was worse. This means each sub-population was overfitting to the assigned training data subset too much by the second approach. From these observations, the positive effect of the periodicical rotation of training data subsets was confirmed.
Due to the page limitation, we do not show the results on the Phoneme data, but almost the same observations were confirmed.

Figures 22 and 23 show the error rates on the whole training data and the test data by each approach for the Pendig data set. From these figures, we can see that the accuracy by the second approach (no rotations) can be improved by rotating the training data subsets (i.e., the first approach). But the accuracy by the first approach was still far from that by the third approach (non-parallel one). This means that all the approaches need more generations.

We examined 10 times longer generations for the Pendig data set. Due to the long computation time, we used a single 10 CV procedure (i.e., 10 runs). Figures 24 and 25 are the error rates on the whole training data and the test data by the first and third approaches for 100,000 generations. We can see that the accuracy of the classifier obtained by the third approach (non-parallel one) was converged around 20,000th generation, but that by the first approach was improving along 100,000 generations.

Table 3 and Table 4 show the training data accuracy, the test data accuracy, and the computation time of the first and third approaches with 10,000 and 100,000 generations, respectively. From these tables, we can see that our parallel distributed approach with 100,000 generations (259.1 [min]) obtained more accurate classifier than the non-parallel one with 10,000 generations (392.9 [min]). This means that we can effectively utilize the computation time reduced by our parallel distributed approach.

Finally, we compare the test data accuracy between the proposed parallel distributed fuzzy GBML and the parallel distributed genetic fuzzy rule selection (Nojima, et al., 2010a) as reference. The results shown in Table 5 are the best test data accuracy of each method with three sub-populations. The best specifications on the rotation interval for each method were used. From Table 5, we can see that the parallel distributed fuzzy GBML is more promising approach than the parallel distributed genetic fuzzy rule selection in terms of the search ability.
References

Aid for Young Scientists (B): KAKENHI (22700239).

Acknowledgement

This study is partially supported by the Japanese Grant-in-Aid for Young Scientists (B): KAKENHI (22700239).

5 Conclusion

In this paper, we examined the effect of the periodical rotation of training data subsets in our parallel distributed fuzzy GBML algorithm for large data sets. We also examined the scalability of our parallel distributed implementation with different specifications on the number of sub-populations (i.e., the number of used CPU cores).

For the Phoneme and Satimage data sets, our parallel distributed approach with rotation of training data subsets obtained more accurate classifiers than the non-parallel one in very short computation time. The experimental results clearly showed that the rotation of training data subsets can avoid the overfitting to certain training data subset. There may exist an appropriate rotation interval for each data set. A dynamic specification of the rotation interval will be an interesting future research topic.

Regarding the effect of the number of sub-populations, the experimental results showed that the speeding-up rates of our parallel distributed approach were more than the number of sub-populations (i.e., used CPU cores) in all the cases. For the Phoneme and Pendig data sets, the error rates on both training and test data were deteriorated by using a larger number of sub-populations. On the other hand, for the Satimage data set, we did not observe that effect. This means that the effect of the number of sub-populations must be dependent on the data itself. The data characterization will be another interesting future topic.

There are a number of remaining tasks. For example, we are planning to apply our parallel distributed method to the multiobjective formulations like (Ishibuchi and Nojima, 2007) by using evolutionary multiobjective optimization.

Acknowledgement

This study is partially supported by the Japanese Grant-in-Aid for Young Scientists (B): KAKENHI (22700239).

Table 5. Comparison of the test data accuracy by the parallel distributed implementation of GBML and genetic rule selection.

<table>
<thead>
<tr>
<th>Method</th>
<th>Phoneme</th>
<th>Satimage</th>
<th>Pendig</th>
</tr>
</thead>
<tbody>
<tr>
<td>GBML</td>
<td>84.03</td>
<td>84.68</td>
<td>94.74</td>
</tr>
<tr>
<td>Rule Selection</td>
<td>80.31</td>
<td>82.03</td>
<td>88.71</td>
</tr>
</tbody>
</table>


