Application of Parallel Distributed Genetics-based Machine Learning to Imbalanced Data Sets

Yusuke Nojima, Shingo Mihara, and Hisao Ishibuchi
Department of Computer Science and Intelligent Systems
Osaka Prefecture University
Sakai, Osaka, Japan
{nojima@, mihara@ci., hisaoi@}cs.osakafu-u.ac.jp

Abstract—Real world data sets are often imbalanced with respect to the class distribution. Classifier design from those data sets is relatively new challenge. The main problem is the lack of positive class patterns in the data sets. To deal with this problem, there are two main approaches. One is to additionally sample minority class patterns (i.e., over-sampling). The other is to sample a part of majority class patterns (i.e., under-sampling). In our previous research, we have proposed a parallel distributed genetics-based machine learning for large data sets. In our method, not only a population but also a training data set is divided into subgroups, respectively. A pair of a sub-population and a training data subset is assigned to an individual CPU core in order to reduce the computation time. In this paper, our parallel distributed approach is applied to imbalanced data sets. The training data subsets are constructed by a composition of subsets divided majority class patterns with the entire set of non-divided minority class patterns. Through computational experiments, we show the effectiveness of our parallel distributed approach with the proposed data subdivision schemes for imbalanced data sets.

Keywords – Imbalanced data, classifier design, fuzzy genetics-based machine learning, parallel distributed approach

I. INTRODUCTION

From late 90’s, classifier design from class imbalanced data sets has attracted growing attention in practical applications such as fraud detection, medical diagnosis, network intrusion detection [1]-[3]. In those data sets, there are a larger number of patterns of some classes than the others. In such a case, majority class and minority class are referred to as negative class and positive class, respectively. Positive class is often more important than negative one. Using standard classifiers, all the patterns tend to be classified as negative class. That is, a small number of patterns with positive (or important) class tend to be ignored. To deal with this problem, there are two main approaches. One approach is to additionally sample positive class patterns (i.e., over-sampling). The simplest way is to sample those patterns repeatedly. This is similar to assigning a weight to positive class patterns. The more powerful way is a synthetic minority oversampling technique (SMOTE) which creates artificial patterns around existing minority patterns [4]. The other main approach is to sample a part of negative class patterns (i.e., under-sampling). This approach is a kind of data cleaning techniques like Tomek links and ENN [5]. As the more sophisticated way, there are some hybrid approaches of over-sampling and under-sampling approaches [6].

In our previous research [7]-[9], we have proposed a parallel distributed approach of genetic fuzzy systems (GFS) for large data sets (i.e., data sets with a large number of patterns). Our parallel distributed approach is like an island model of genetic algorithms in which a population is divided into sub-populations and each sub-population is assigned to an island [10], [11]. In addition to the population, a training data set is also divided into subsets and each subset is assigned to an island in [7]-[9]. That is, a pair of a sub-population and a training data subset is assigned to an island. In each island, our GFS is performed with an individual CPU core. A periodical rotation of training data subsets is performed to avoid the over-fitting of each sub-population to the corresponding training data subset. For large data sets, our parallel distributed fuzzy GBML can drastically reduce the computation time and also improve its search ability.

In this paper, we utilize the structure of our parallel distributed model of GFS for imbalanced data sets. Whereas the whole training data set is randomly divided into data subsets for large data, the only negative class patterns (i.e., majority class patterns) are divided into subsets for imbalanced data sets. That is, each training data subset is composed of the divided negative class patterns and the non-divided positive class patterns. As an individual island level, this model can be regarded as an under-sampling approach, because a subset of negative class patterns is used in each island. On the other hand, as the entire level, this model can be regarded as an over-sampling approach, because positive class patterns are replicated and used in each island. Through computational experiments with several imbalanced data sets available from the KEEL data repository (http://www.keel.es), we examine the effectiveness of our GFS with the proposed data subdivision schemes.

This paper is organized as follows. We first briefly explain our hybrid fuzzy genetics-based machine learning (GBML) [10] which is one of the powerful GFS methods. We also briefly explain its parallel distributed model in Section II. Next we explain two data subdivision schemes for imbalanced data sets in Section III. Then we show several experimental results with imbalanced data sets in Section IV. Finally we conclude this paper in Section V.
II. PARALLEL DISTRIBUTED GBML

A. Fuzzy Rule-based Classifier

Consider fuzzy rule-based classifiers for an M-class n-dimensional pattern classification problem in the n-dimensional pattern space \([0, 1]^n\) where \(m\) training patterns \(x_p = (x_{p1}, ..., x_{pn})\), \(p = 1, 2, ..., m\) from \(M\) classes are given. Each attribute value \(x_{pi}\) has already been normalized into real numbers in the unit interval \([0, 1]\) for \(i = 1, 2, ..., n\) and \(p = 1, 2, ..., m\). We use fuzzy rules of the following type [13]:

\[
\text{Rule } R_q: \text{If } x_1 \text{ is } A_{q1} \text{ and } ... \text{ and } x_n \text{ is } A_{qn} \text{ then Class } C_q \text{ with } CF_q, \tag{1}
\]

where \(R_q\) is the label of the qth fuzzy rule, \(A_{qi}\) is an antecedent fuzzy set \((i = 1, 2, ..., n)\), \(C_q\) is a class label, and \(CF_q\) is a real number in the unit interval \([0, 1]\) which represents a rule weight.

As antecedent fuzzy sets, we use 14 fuzzy sets in Fig. 1 and “don't care”. The “don't care” condition is represented by the unit interval \([0, 1]\), which is always fully compatible with any attribute value \(x_{pi}\). The use of “don't care” conditions enables us to simplify a fuzzy rule [14]. The number of antecedent conditions (except for “don't care” conditions) is often referred to as the rule length. We use 15 symbols to represent “don't care” and the 14 antecedent fuzzy sets. These symbols can be viewed as integers. Thus the antecedent part of each fuzzy rule is represented by an integer string of length \(n\).

![Figure 1. Fourteen antecedent fuzzy sets used in this paper.](image)

B. Hybrid Fuzzy Genetics-based Machine Learning

In our hybrid fuzzy GBML [10], Pittsburgh approach is used as the main framework for optimizing a fuzzy rule-based classifier with \(N\) fuzzy rules. That is, a fuzzy rule-based classifier is coded as a concatenated integer string of length \(nN\). As a local search operation, a single iteration of Michigan approach is performed for optimizing an individual fuzzy rule which is coded as an integer string of length \(n\).

The procedure of our hybrid 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.

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.

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

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.

In Step 1, initial fuzzy rules are directly generated from training patterns. “Don’t care” conditions are used with the probability \((n-5)/n\) in initial fuzzy rules. An initial fuzzy rule-based classifier with 30 fuzzy rules is generated from randomly selected 30 training patterns (see [10] for more detail).

In Step 2 and Step 9, the fitness value of a fuzzy rule-based classifier \(S\) is calculated by the following fitness function.

\[
\text{fitness}(S) = w_1 f_1(S) + w_2 f_2(S) + w_3 f_3(S), \tag{2}
\]

where \(w_1, w_2, w_3\) are non-negative constant weights, \(f_1(S)\) is the error rate on training patterns in percentage, \(f_2(S)\) is the number of fuzzy rules in \(S\), and \(f_3(S)\) is the total rule length over fuzzy rules in \(S\). By minimizing this fitness function, we obtain fuzzy rule-based classifiers with high accuracy and low complexity. The weight values are specified as \(w_1=100, w_2=1\) and \(w_3=1\) in our computational experiments. The effect of this specification is left as future experimental study.

In Step 4, an arbitrary number of rules are chosen from each parent. Thus, an offspring has a different number of rules. The upper and lower limits on the number of fuzzy rules in each classifier are specified as 60 and 1 in our computational experiments, respectively. In Step 6, a single iteration of Michigan approach is performed to newly generate \(N_{\text{replace}}\) rules by genetic operations among rules and heuristic rule generation with misclassified patterns. \(N_{\text{replace}}\) is a minimum integer value more than 0.2 \(N\). See [10] for more detail.
C. Parallel Distributed GBML

In our parallel distributed model, we combine two major approaches for large data sets. One is parallel implementation of genetic algorithms like an island model [11], [12]. The other is data reduction like feature selection and instance selection [15]-[18]. Not only a population but also a training data set is divided into subgroups, respectively. In this paper, we use a workstation with eight processors: Intel Xeon X5570 (4 core 2.93GHz) × 2. Seven out of the eight CPU cores are used for parallel computations while the other is used for the operating system. As the conventional parallel distributed model [8], [9], we divide a population into seven sub-populations of the same size. We also randomly divide a training data set into seven subsets while remaining the class distributions. A pair of a sub-population and a training data subset is assigned to each of the seven processors for parallel computation as in Fig. 2.

![Figure 2. Our parallel distributed model for large data sets.](image1)

The procedure of our parallel distributed model can be written as follows.

Step 1: Randomly divide a population and a training data set into subgroups of the same size, respectively.

Step 2: Assign a pair of a sub-population and a training data subset to each of CPU cores.

Step 3: Initialize each sub-population by the heuristic rule initialization method from the corresponding training data subset.

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 training data subsets as shown in Fig. 2 over the CPU cores.

Step 7: Return to Step 4.

Step 8: Combine sub-populations and training data subsets into a single population and the entire training data set, respectively.

Step 9: Classify the entire training data set and remove unnecessary rules from each classifier.

Step 10: Choose the best classifier with respect to the fitness function in (2).

In island models, migration operators are often used to move a copy of a good individual in a sub-population to another one. For the sake of simplicity, we do not use migration operators in our computational experiments.

III. Application of Parallel Distributed GBML to Imbalanced Data Sets

A. Two Subdivision Schemes for Imbalanced Data Sets

In this paper, we apply our parallel distributed GBML to imbalanced data sets. As a simple extension, we propose the following two data subdivision schemes. The key idea is to divide only negative class patterns into subsets. Then each training data subset is composed of the divided negative class patterns and the non-divided positive class patterns.

Simple subdivision of negative class patterns

Negative class patterns (i.e., majority class patterns) are simply divided into seven training data subsets of the same size when the number of CPU cores used in the experiment is seven. Positive class patterns are replicated and used in each island (i.e., each CPU core) as shown in Fig. 3. At each island level, negative class patterns are under-sampled (e.g., from 28 to 4 patterns in Fig. 3). At the entire level, positive class patterns are over-sampled (e.g., from 2 to 14 patterns in Fig. 3).

![Figure 3. Simple subdivision of negative class patterns.](image2)

IR-based subdivision of negative class patterns

Negative class patterns are divided into several training data subsets according to the class imbalanced rate (IR) [19]. IR is calculated as:

\[
IR = m_{\text{negative}} / m_{\text{positive}},
\]

where \(m_{\text{negative}}\) is the number of negative class patterns. The \(m_{\text{positive}}\) is the number of positive class patterns. Let \(IR'\) be the nearest integer value of \(IR\). In this subdivision scheme, the negative class patterns are divided into \(IR'\) subsets. In Fig. 4, \(IR\) is 14 (i.e., 28/2). Thus, there are 14 subsets of negative class patterns. Since we use seven CPU cores, half of 14 subsets are not assigned to any sub-populations. However, through the periodical rotation of the training data subsets, all the subsets are used for optimization process. Positive class patterns are replicated and used in each CPU core as shown in Fig. 4.
**B. Evaluation Criteria**

To evaluate the performance of the obtained classifiers for imbalanced data sets, we follow a specialized literature [19] and use the area under the ROC curve (AUC) calculated from a confusion matrix in Table I.

\[
AUC = \frac{1 + TP_{rate} - FP_{rate}}{2},
\]

where \( TP_{rate} = TP / (TP + FN) \), the percentage of positive class patterns correctly classified as belonging to the positive class. \( FP_{rate} = FP / (FP + TN) \), the percentage of negative class patterns misclassified as belonging to the positive class.

**TABLE I. CONFUSION MATRIX FOR A TWO-CLASS PROBLEM.**

<table>
<thead>
<tr>
<th>Positive prediction</th>
<th>Negative prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive class</td>
<td>True positive (TP)</td>
</tr>
<tr>
<td></td>
<td>False negative (FN)</td>
</tr>
<tr>
<td>Negative class</td>
<td>False positive (FP)</td>
</tr>
<tr>
<td></td>
<td>True negative (TN)</td>
</tr>
</tbody>
</table>

It should be noted that we do not use AUC as the fitness function of our hybrid fuzzy GBML. We used the weighted-sum fitness function in (2) for imbalanced data sets. Of course, there is a possibility that we can obtain better results when using AUC instead of \( f_1(S) \) in (2). We will investigate this possibility in our future study.

IV. COMPUTATIONAL EXPERIMENTS

A. Settings of Computational Experiments

In this section, we show the effectiveness of the proposed parallel distributed GBML for imbalanced data sets. We used eight imbalanced data sets available from KEEL dataset repository (http://www.keel.es) in Table II.

**TABLE II. IMBALANCED DATA SETS USED IN THIS PAPER.**

<table>
<thead>
<tr>
<th>Data set</th>
<th># of patterns</th>
<th># of attributes</th>
<th># of classes</th>
<th>IR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segment0</td>
<td>2308</td>
<td>19</td>
<td>2</td>
<td>6.02</td>
</tr>
<tr>
<td>Yeast3</td>
<td>1481</td>
<td>8</td>
<td>2</td>
<td>8.10</td>
</tr>
<tr>
<td>Page-blocks0</td>
<td>5472</td>
<td>10</td>
<td>2</td>
<td>8.79</td>
</tr>
<tr>
<td>Shuttle0vs4</td>
<td>1829</td>
<td>9</td>
<td>2</td>
<td>13.87</td>
</tr>
<tr>
<td>Yeast4</td>
<td>1484</td>
<td>8</td>
<td>2</td>
<td>28.10</td>
</tr>
<tr>
<td>Yeast1289vs7</td>
<td>947</td>
<td>8</td>
<td>2</td>
<td>30.57</td>
</tr>
<tr>
<td>Yeast5</td>
<td>1484</td>
<td>8</td>
<td>2</td>
<td>32.73</td>
</tr>
<tr>
<td>Yeast6</td>
<td>1484</td>
<td>8</td>
<td>2</td>
<td>41.40</td>
</tr>
</tbody>
</table>

We used the following parameter specifications:

- Terminal condition: 50,000 generations,
- Population size: 210 (sub-population size: 30),
- Crossover probability in both parts: 0.9,
- Mutation probability in Pittsburgh part: \( 1/nN \),
- Mutation probability in Michigan part: \( 1/n \),
- Maximum number of rules in a classifier: 60,
- Periodical rotation interval: 20, 50, 100.

We performed five-fold cross validation (5CV) with three different random seeds. That is, the AUC is averaged over 15 runs (3 x 5CV).

B. Results by Non-Parallel GBML

First we show the experimental results by the non-parallel hybrid fuzzy GBML. We examined average AUC values of the non-parallel hybrid fuzzy GBML and that with SMOTE. SMOTE is one of the most well-known over-sampling methods [4]. It generates synthetic patterns along the line segments joining any/all of the \( k \) minority class nearest neighbors. In Table III, “Normal” represents our GBML without SMOTE. “SMOTE” represents our GBML with SMOTE. Comparing between “Normal” and “SMOTE”, better AUC is highlighted by bold face. From Table III, we can see that AUC tends to be worse as IR increases. We can also see that better AUC values were obtained using “SMOTE” except for Shuttle0vs7 data set.

**TABLE III. COMPARISON OF AVERAGE AUC VALUES BY NON-PARALLEL GBML WITH AND WITHOUT SMOTE.**

<table>
<thead>
<tr>
<th>Data set</th>
<th>Normal</th>
<th>SMOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segment0</td>
<td>98.74%</td>
<td>99.16%</td>
</tr>
<tr>
<td>Yeast3</td>
<td>85.06%</td>
<td>91.51%</td>
</tr>
<tr>
<td>Page-blocks0</td>
<td>86.50%</td>
<td>90.98%</td>
</tr>
<tr>
<td>Shuttle0vs4</td>
<td>99.91%</td>
<td>99.72%</td>
</tr>
<tr>
<td>Yeast4</td>
<td>60.41%</td>
<td>79.11%</td>
</tr>
<tr>
<td>Yeast1289vs7</td>
<td>62.82%</td>
<td>67.65%</td>
</tr>
<tr>
<td>Yeast5</td>
<td>83.18%</td>
<td>95.38%</td>
</tr>
<tr>
<td>Yeast6</td>
<td>50.00%</td>
<td>85.70%</td>
</tr>
</tbody>
</table>

C. Results by Parallel Distributed Models

Next we show the experimental results by the parallel distributed models with two data subdivision schemes. We performed three specifications of the rotation interval (i.e., 20, 50, 100 generations). Comparing between simple subdivision and IR-based subdivision, better AUC values are highlighted by underline in Table IV. Comparing with Table III, better AUC values are highlighted by bold face in Table IV.

In most of cases in Table IV, IR-based subdivision was better than simple subdivision. For data sets with lower IR (e.g., Segment0, Yeast3, Page-blocks0), the results were similar in both schemes, because almost the same data subdivision was performed. On the other hand, for data sets with high IR like Yeast6, we did not obtain good results by simple subdivision. This may be because the number of negative class patterns was still big. At an island level, IR is 5.9 when using simple subdivision for Yeast6 data set.
From the comparison between non-parallel models in Table III and parallel distributed models in Table IV, we can see that better results were obtained from the parallel distributed model with IR-based subdivision. Table IV shows that an appropriate specification of the rotation interval of training data subsets was 50. Thus, we use those results for further comparisons.

**TABLE IV. AVERAGE AUC VALUES BY PARALLEL DISTRIBUTED GBML WITH TWO DATA SUBDIVISION SCHEMES.**

<table>
<thead>
<tr>
<th>Data set</th>
<th>Rotation Interval</th>
<th>Simple subdivision</th>
<th>IR-based subdivision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segment0</td>
<td>100</td>
<td>99.32%</td>
<td>99.22%</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>99.11%</td>
<td>99.46%</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>99.31%</td>
<td>99.49%</td>
</tr>
<tr>
<td>Yeast3</td>
<td>100</td>
<td>90.50%</td>
<td>91.92%</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>90.30%</td>
<td>92.11%</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>90.46%</td>
<td>90.06%</td>
</tr>
<tr>
<td>Page-blocks0</td>
<td>100</td>
<td>94.15%</td>
<td>93.71%</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>94.15%</td>
<td>93.35%</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>93.38%</td>
<td>93.05%</td>
</tr>
<tr>
<td>Shuttle0vs4</td>
<td>100</td>
<td>100.00%</td>
<td>99.88%</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td><strong>100.00%</strong></td>
<td>99.91%</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td><strong>100.00%</strong></td>
<td>99.76%</td>
</tr>
<tr>
<td>Yeast4</td>
<td>100</td>
<td>74.04%</td>
<td>79.80%</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>71.31%</td>
<td>83.27%</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>73.12%</td>
<td>82.25%</td>
</tr>
<tr>
<td>Yeast1289vs7</td>
<td>100</td>
<td>63.87%</td>
<td>65.97%</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>62.07%</td>
<td>66.99%</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>64.65%</td>
<td>63.28%</td>
</tr>
<tr>
<td>Yeast5</td>
<td>100</td>
<td>92.23%</td>
<td>95.00%</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>89.64%</td>
<td><strong>96.09%</strong></td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>91.55%</td>
<td><strong>96.08%</strong></td>
</tr>
<tr>
<td>Yeast6</td>
<td>100</td>
<td>78.38%</td>
<td>85.13%</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>76.74%</td>
<td><strong>85.89%</strong></td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>78.93%</td>
<td>83.55%</td>
</tr>
</tbody>
</table>

**D. Comparison with Alternative Methods**

We compare the results by the proposed parallel distributed models with alternative methods. In Table V, we referred some results from the recent paper [19]. In Chi-3-LTR, Chi-5-LTR, HF-GBML-LTR, the authors applied local tuning of the rules. Besides, it should be noted that all the alternative methods were performed together with SMOTE.

In Table V, better AUC values are highlighted by bold face for each data set. For three out of eight data sets, our parallel distributed model with IR-based subdivision obtained the best AUC values. It is, however, difficult to say which method is the best in Table V. Thus, we applied Friedman test to Table V using MULTIPLETTEST package (http://sci2s.ugr.es/sicidm) [20]. Table VI shows the average rankings of the algorithms. Our parallel distributed model with IR-based subdivision obtained the lowest rank. The $p$-value computed by Friedman test was 0.024.

**TABLE VI. AVERAGE RANKINGS OF THE ALGORITHMS.**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple subdivision</td>
<td>5.19</td>
</tr>
<tr>
<td>IR-based subdivision</td>
<td><strong>2.75</strong></td>
</tr>
<tr>
<td>Chi-3-LTR</td>
<td>4.13</td>
</tr>
<tr>
<td>Chi-5-LTR</td>
<td>5.63</td>
</tr>
<tr>
<td>HF-GBML-LTR</td>
<td>2.94</td>
</tr>
<tr>
<td>Ripper</td>
<td>4.56</td>
</tr>
<tr>
<td>C4.5</td>
<td>2.81</td>
</tr>
</tbody>
</table>

Table VII shows the results by Holm / Hochberg tests for $\alpha = 0.05$. The control algorithm is our parallel distributed model with IR-based subdivision. We can say from this table that our parallel distributed model with IR-subdivision were better than that with simple subdivision and Chi-5-LTR, but there were no statistical difference between IR-based subdivision and some alternative methods (i.e., Ripper, Chi-3-LTR, HF-GBML-LTR, and C4.5).

**V. CONCLUSIONS**

In this paper, we applied our parallel distributed hybrid fuzzy genetics-based machine learning to the classifier design from imbalanced data sets. In the proposed data subdivision schemes, we divided only negative class patterns into subsets. On the other hand, we also replicated positive class patterns. Then we assigned the divided negative class patterns and the non-divided positive class patterns to each island as a training data subset. Our parallel distributed GBML with the proposed data subdivision scheme can be regarded as a hybrid model of over-sampling and under-sampling approaches.

Through the computational experiments, we showed the superiority of the proposed data subdivision scheme (especially, IR-based subdivision) comparing with the non-parallel GBML with and without SMOTE. We also showed the performance of the proposed method is very comparable to the state-of-the-art methods for imbalanced data sets although we did not use SMOTE.

The good point is that we do not generate any synthetic patterns for a small number of positive class patterns and do not remove any negative class patterns. We believe that this is a better way from the viewpoint of data mining, because classifiers can be generated only based on the original data set.

As future research topics, we should apply our method to more data sets. Especially, the application to imbalanced data sets with a large number of patterns would be more interesting. There is a possibility that we can obtain more accurate classifiers with very short computation time thanks to the parallel distributed architecture. We also should examine the effects of different fitness functions on the performance of classifiers. We expect that better results will be obtained using the AUC value in the fitness function of GBML.
REFERENCES


