Effects of Data Reduction on the Generalization Ability of Parallel Distributed Genetic Fuzzy Rule Selection

Yusuke Nojima
Graduate School of Engineering
Osaka Prefecture University
Sakai, Osaka, Japan
E-mail: nojima@cs.osakafu-u.ac.jp

Hisao Ishibuchi
Graduate School of Engineering
Osaka Prefecture University
Sakai, Osaka, Japan
E-mail: hisaoi@cs.osakafu-u.ac.jp

Abstract— **Genetic fuzzy rule selection has been successfully used to design accurate and interpretable fuzzy classifiers from numerical data. In our former study, we proposed its parallel distributed implementation which can drastically decrease the computational time by dividing both a population and a training data set into sub-groups. In this paper, we examine the effect of data reduction on the generalization ability of fuzzy rule-based classifiers designed by our parallel distributed approach. Through computational experiments, we show that data reduction can be realized without severe deterioration in the generalization ability of the designed fuzzy classifiers.**

Keywords-Genetic fuzzy rule selection, parallel distributed implementation, pattern classification, data reduction.

I. Introduction

Genetic algorithms have been frequently used for the design of fuzzy rule-based systems under the name of genetic fuzzy systems (GFSs) [5,7]. One of the most important research issues is their scalability improvement to large data sets [8]. When we use GFSs for large data sets, the evaluation of each individual needs long computational time. There exist two well-known approaches to the decrease in computational costs for the handling of large data sets. One approach is data reduction, which includes feature selection and instance selection [11,12]. The other approach is parallel implementation of genetic algorithms, which is usually based on spatial structures of populations such as island and cellular models [1,4].

In our former study [13-15], we proposed a parallel distributed genetic fuzzy rule selection based on the data subdivision and the parallelization of genetic fuzzy rule selection [10]. The idea is to divide not only a population but also the training data set into sub-groups. A training data subset and a sub-population are assigned to each processing node (e.g., CPU core). In order to avoid the over-fitting of each sub-population to a specific training data subset, training data subset re-assignment is periodically performed (e.g., every 10 generations). In [14], we used a workstation with four CPU cores (a CPU core was used as a server, and the others were used as clients). The experimental results showed that we can reduce the computational time to 1/9 with no deterioration of the test data accuracy.

In our former study, we used all the available training data after dividing them into a number of subsets, each of which was assigned to a CPU core. It has been pointed out in the literature [2,3,11] that all patterns are not always necessary in the learning of classifiers. This means that we may eliminate a part of training patterns without losing their important characteristics. In this paper, we examine the effect of data reduction on the generalization ability of fuzzy classifiers. As a preliminary study, we apply three random sampling methods to our parallel distributed genetic fuzzy rule selection.

This paper is organized as follows. First we explain genetic fuzzy rule selection for designing fuzzy classifiers and its parallel distributed implementation in Section II. In Section III, we examine the effect of data reduction. Finally we conclude this paper in Section IV.

II. Parallel Distributed Classifier Design

A. Fuzzy Rules for Pattern Classificaiton

Let us assume that we have m training (i.e., labeled) patterns $\mathbf{x}_p = (x_{p1}, \ldots, x_{pn})$, $p = 1, 2, \ldots, m$ from M classes in the n-dimensional continuous pattern space where x_{pi} is the attribute value of the p-th training pattern for the i-th attribute ($i = 1, 2, ..., n$). For simplicity of explanation, we assume that all the attribute values have already been normalized into real numbers in [0, 1].

For our n-dimensional pattern classification problem, we use fuzzy rules of the following type:

$$\text{Rule } R_q\text{: If } x_1 \text{ is } A_{q1} \text{ and } \ldots \text{ and } x_n \text{ is } A_{qn} \text{ then Class } C_q \text{ with } CF_q, \quad (1)$$

where R_q is the label of the q-th fuzzy rule, $\mathbf{x} = (x_1, \ldots, x_n)$ is an n-dimensional pattern vector, A_{qi} is an antecedent fuzzy set ($i = 1, 2, \ldots, n$), C_q is a class label, and CF_q is a rule weight (i.e., certainty grade).

We simultaneously use multiple fuzzy partitions with different granularities in fuzzy rule generation. In this paper, we use four homogeneous fuzzy partitions in Fig. 1 (i.e., 14 triangular fuzzy sets in Fig. 1). We also use the domain interval [0,1] as an antecedent fuzzy set in order to represent a *don't care* condition. That is, we use the 15 antecedent fuzzy sets in total.

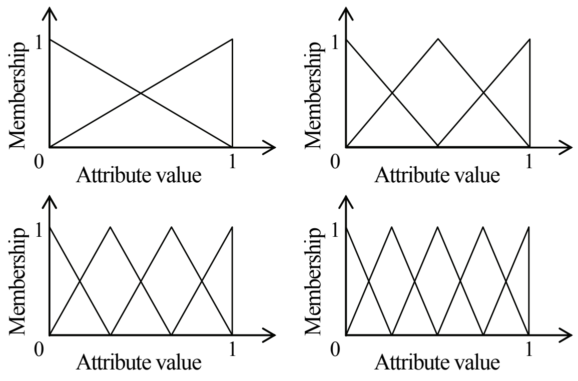


Figure 1. Four fuzzy partitions used in our computation experiments.

The consequent class C_q and the rule weight CF_q of each fuzzy rule R_q can be specified in a heuristic manner by compatible training patterns (by their compatibility grades) with the antecedent part of R_q. For details, see [9,13-15].

B. Fuzzy Rule Generation

Since we use the 15 antecedent fuzzy sets for each attribute of our n-dimensional pattern classification problem, the total number of possible fuzzy rules is 15^n. For high-dimensional data sets, the total number of possible fuzzy rules becomes quite large. Moreover, it is very difficult to intuitively understand long fuzzy rules with many antecedent conditions. Thus, we only generate short fuzzy rules with only a small number of antecedent conditions. In this paper, we examine only short fuzzy rules of length $L_{\max}$ or less (e.g., $L_{\max} = 3$) in order to generate understandable fuzzy rules as candidates in fuzzy rule selection. It should be noted that the length of a fuzzy rule is defined by the number of its antecedent conditions excluding *don't care*.

We generate a prespecified number of promising fuzzy rules as candidate rules in fuzzy rule selection. We use the product value of the confidence and the support of each fuzzy rule as a rule pre-screening criterion. That is, a prespecified number of fuzzy rules with the higher product values are used as candidates.

The confidence of a fuzzy rule "If $\mathbf{A}_q$ then Class h" is calculated for each class ($h = 1, 2, \ldots, M$) as

$$c(\mathbf{A}_q \Rightarrow \text{Class } h) = \frac{\sum_{\mathbf{x}_p \in \text{Class } h} \mu_{\mathbf{A}_q}(\mathbf{x}_p)}{\sum_{p=1}^{m} \mu_{\mathbf{A}_q}(\mathbf{x}_p)}, \tag{1}$$

where $\mu_{\mathbf{A}_q}(\mathbf{x}_p)$ is the compatibility grade of $\mathbf{x}_p$ with the antecedent part $\mathbf{A}_q$, which is calculated by the product operation. On the other hand, the support of the fuzzy rule is calculated as follows:

$$s(\mathbf{A}_q \Rightarrow \text{Class } h) = \frac{\sum_{\mathbf{x}_p \in \text{Class } h} \mu_{\mathbf{A}_q}(\mathbf{x}_p)}{m}. \tag{3}$$

C. Parallel Distributed Implementation

Figure 2 shows the basic framework of our parallel distributed implementation of genetic fuzzy rule selection. To handle a large data set, we use a cluster computer system composed of a server CPU and a number of client CPUs. We can easily set up this type of systems by using some independent desktop computers or a single computer with multi-core CPUs.

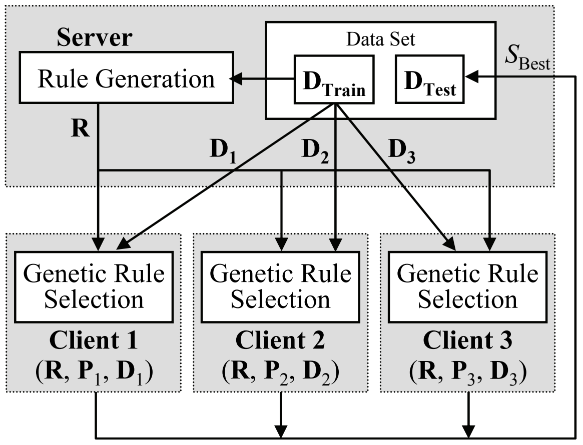


Figure 2. Parallel distributed genetic fuzzy rule selection.

Our parallel distributed genetic fuzzy rule selection is executed in the following manner.

[Parallel Distributed Genetic Fuzzy Rule Selection]

Step 1: Generate a number of promising rules **R** (i.e., N rules) from the whole training patterns.

Step 2: Randomly generate N_{pop} binary strings of length N as an initial population.

Step 3: Randomly divide the current population P and the training patterns D into sub-populations $\{P_1, P_2, \ldots\}$ and sub-groups $\{D_1, D_2, \ldots\}$, respectively.

Step 4: Send the sub-populations and the sub-groups of the training patterns to client CPUs.

Step 5: Iterate genetic rule selection for a prespecified number of generations in each client CPU.

Step 6: Systematically change the sub-groups of the training patterns to optimize each subpopulation for a different sub-group of the training patterns.

Step 7: If a prespecified termination condition is not satisfied, return to Step 5. Otherwise go to Step 8.

Step 8: Choose the best fuzzy classifier S_{best} from the whole population and examine the generalization ability for test patterns.

In Step 2, each binary string of length N is represented 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.

In Step 5, we use tournament selection, uniform crossover, biased mutation to generate an offspring population. The biased mutation changes 0 to 1 with a small probability and 1 to 0 with a large probability to decrease the number of 1's in the offspring.

We use the following three objectives to find an accurate and compact fuzzy classifier S:

$f_1(S)$: The number of correctly classified training patterns in the training data sub-group D_j by S,

$f_2(S)$: The number of selected fuzzy rules in S,

$f_3(S)$: The total number of antecedent conditions in S.

These three objectives are combined into the following weighted sum fitness function:

$$fitness(S) = w_1 \cdot f_1(S) - w_2 \cdot f_2(S) - w_3 \cdot f_3(S), \qquad (4)$$

where w_1, w_2 and w_3 are non-negative weights. This fitness function is maximized in genetic fuzzy rule selection. As a result, the accuracy is maximized while the complexity is minimized.

The first objective is calculated by classifying training patterns $\mathbf{x}_p$. We use a single winner-based (i.e., winner-take-all) classification method. A single winner rule is identified using the product of the compatibility grade and the rule weight of each fuzzy rule.

Since we use a single winner-based classification method, some rules may be used for the classification of no training patterns. Whereas the existence of such an unnecessary rule in the rule set S has no effect on the first objective of the weighted sum fitness function, it deteriorates the second and third objectives. Thus we remove all the unnecessary rules responsible for the classification of no training patterns before calculating the second and third objectives.

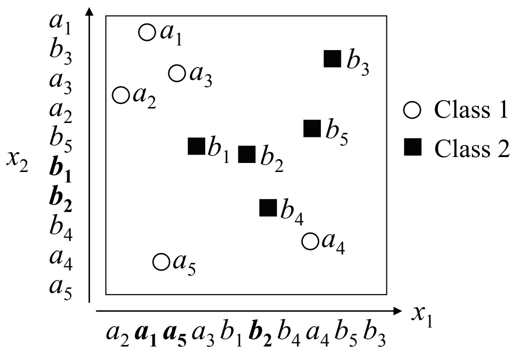
D. Data Reduction

We examine three types of data reduction methods. One is **random sampling**. This method randomly eliminates a prespecified number of the training patterns from each sub-group of the training patterns.

Another type is **random sampling with 1-NN**. In this method, first each training pattern is classified by the other patterns using the nearest neighbor method. Then correctly classified patterns (e.g., a_1, a_2, a_3, b_1, b_2, b_3, b_5 in Fig. 3) are randomly eliminated.

The other is **random sampling with rank**. A rank of a pattern is the number of attributes on which that pattern and its adjacent two neighbors are from the same class. For example, b_5 in Fig. 3 is Rank 0 whereas b_2 is Rank 2 (only b_2 is Rank 2). Rank 1 patterns in Fig. 3 are a_1, a_5 and b_1. A prespecified number of patterns with the highest ranks are eliminated. When we remove three patterns from Fig. 3, first b_2 is removed. Then two of a_1, a_5, b_1 are randomly removed.

The second and third methods are similar to the Condensed Nearest Neighbor Rule [6] which tries to maintain the decision boundary.



a_1 b_3 a_3 a_2 b_5 b_1 b_2 b_4 a_4 a_5
x_2
a_1
a_3
b_3
a_2
b_5
b_1
b_2
b_4
a_4
a_5
○ Class 1
■ Class 2
x_1
a_2 a_1 a_5 a_3 b_1 b_2 b_4 a_4 b_5 b_3

Figure 3. An example of a training data set.

III. COMPUTATIONAL EXPEIRMENTS

Through computational experiments on four benchmark data sets in Table I which are available from the UCI machine learning repository, we examined the effects of data reduction in our parallel distributed genetic fuzzy rule selection. We evaluated the generalization ability of the obtained fuzzy rule-based classifiers by iterating the ten-fold cross validation procedure two times (i.e., 2 x 10 CV). That is, each result is the average over 20 runs. We used a workstation with two Xeon 3.0 GHz dual processors (i.e., four CPU cores, in total). We used one of the four CPU cores as a server CPU. The other three were used as client CPUs.

TABLE I. DATA SETS USED IN OUR COMPUTATIONAL EXPERIMENTS.

Data set	Attributes	Patterns	Classes
Yeast	8	1484	10
Page-Blocks	10	5473	5
Satimage	36	6435	6
Pendig	16	10992	10

At first, we generated 300 fuzzy rules for each class by using the product criterion of confidence and support. The maximum rule length of each fuzzy rule was specified as 2 for the satimage data set, and 3 for the other three data sets.

In genetic fuzzy rule selection, we specified the population size as 300 (i.e., the size of each sub-population was 100). The total number of evaluated strings was specified as 300300. This is equal to the number of strings in initial population with 300 strings and 1,000 generations in the case of genetic fuzzy rule selection. The weight vector in the fitness function in (4) was specified as **w** = (100, 1, 1).

We examined the following five implementations.

Type 0: Only the server performs rule extraction and genetic fuzzy rule selection. No subdivision is used in this case. Thus, this type is the same as the original non-parallel algorithm.

Type 1: Three clients perform genetic fuzzy rule selection. But the sub-groups of the training patterns are never changed during the execution.

Type 2: Three clients perform genetic fuzzy rule selection. The sub-groups are changed every 100 generations.

Type 3: Three clients perform genetic fuzzy rule selection. The sub-groups are changed every 10 generations.

Type 4: Three clients perform genetic fuzzy rule selection. The sub-groups are changed every generation.

Tables II-XVII show the average training data accuracy, the average test data accuracy, the average number of selected fuzzy rules, the average total rule length, and the average CPU time (in second) for each type. We performed statistical tests for examining the statistical significance of the difference between the original non-parallel algorithm (i.e., Type 0) and our parallel distributed algorithm (i.e., Types 1-4) in the test data accuracy. We used a paired student's t-test when the distribution of experimental results can be regarded as a normal distribution. Otherwise, we used a Wilcoxon signed-ranks test [16]. The results were shown in bold face when the results are not significantly different from the results by the non-parallel algorithm (i.e., Type 0). We specified the significance level α as 0.05.

A. *Results on the Yeast Data Set*

From Table II, we can see that there is no statistical difference between Type 0 and all the others. That is, we can drastically reduce the computational time without the deterioration in the test data accuracy. The best setting was Type 3 in terms of the test data accuracy. The computation was about nine times faster than that of Type 0. It should be noted that the number of fuzzy rules in the classifier obtained by Type 3 was also smaller than that by Type 0. This may be a positive effect of data subdivision on avoiding the over-fitting to the training patterns. It should be also noted that the result by Type 1 was not statistically different from that by Type 0. This means that we do not need to use all the training patterns intrinsically, because each client CPU in Type 1 always used the same training data sub-group without the systematical re-assignment like Types 2-4.

Tables III-V show the results by data reduction from 10% to 50%. We can see that we can eliminate 40% of the training patterns by random sampling without the deterioration in the test data accuracy. Comparing with Type 0 with all the training patterns, we can reduce the computational time to more than 1/14. An interesting observation is that the number of fuzzy rules in the classifier obtained by using data reduction was also decreased as the reduction rate increases.

TABLE II. RESULTS ON THE YEAST DATA SET. TYPE 0 IS A NON-PARALLEL APPROACH. TYPE 1-4 ARE PARALLEL APPROACHES.

Type	Train	Test	Rules	Length	Time (s)
0	61.35	57.84	23.1	63.4	2047.5
1	59.59	**57.32**	18.5	50.6	231.1
2	59.96	**57.35**	17.1	47.4	221.4
3	60.63	**57.51**	16.9	48.2	228.9
4	60.20	**57.44**	16.9	48.6	421.0

TABLE III. RESULTS OF TYPE 3 ON THE YEAST DATA SET BY RANDOM SAMPLING.

Rate	Train	Test	Rules	Length	Time (s)
10%	60.43	**57.83**	16.0	45.6	206.5
20%	60.29	**57.92**	15.8	44.5	184.2
30%	60.05	**57.41**	15.2	43.3	162.7
40%	59.70	**57.09**	14.4	41.0	141.9
50%	59.24	57.03	14.0	39.7	120.4

TABLE IV. RESULTS OF TYPE 3 ON THE YEAST DATA SET BY RANDOM SAMPLING WITH 1-NN.

Rate	Train	Test	Rules	Length	Time (s)
10%	60.67	57.04	17.2	49.5	206.0
20%	60.44	56.94	16.8	47.0	184.1
30%	59.89	**57.30**	15.7	44.5	160.4
40%	59.15	56.39	14.8	42.7	150.8
50%	58.38	55.52	14.2	41.0	138.6

TABLE V. RESULTS OF TYPE 3 ON THE YEAST DATA SET BY RANDOM SAMPLING WITH RANK.

Rate	Train	Test	Rules	Length	Time (s)
10%	60.63	**57.96**	17.3	48.8	207.4
20%	60.35	**57.15**	16.5	46.9	185.0
30%	60.08	**57.16**	16.5	46.9	162.6
40%	59.63	56.71	15.7	44.6	142.9
50%	59.40	56.86	15.5	44.5	131.1

B. *Results on the Page-Blocks Data Set*

Table VI shows that the results by Type 2 and Type 4 were not significantly different from that by Type 0. The computation by Type 2 was about nine times faster than that of Type 0. Since Type 2 was faster than Type 4, we used Type 2 to examine the data reduction effects.

TABLE VI. RESULTS ON THE PAGE-BLOCKS DATA SET. TYPE 0 IS A NON-PARALLEL APPROACH. TYPE 1-4 ARE PARALLEL APPROACHES.

Type	Train	Test	Rules	Length	Time (s)
0	90.38	90.24	7.6	15.1	3301.8
1	90.30	90.16	5.7	11.6	362.4
2	90.30	**90.20**	5.9	12.6	364.1
3	90.31	90.16	5.9	13.2	366.6
4	90.23	**90.17**	4.3	9.7	712.4

From Tables VII-IX, we can see that only random sampling with rank could reduce the training data set without the deterioration in the test data accuracy. The computation of Type 2 by random sampling with rank can be more than 13 times faster than that of Type 0. Since the page-blocks data set is highly imbalanced, patterns from minority classes might be removed by random sampling and random sampling with 1-NN.

TABLE VII. RESULTS OF TYPE 2 ON THE PAGE-BLOCKS DATA SET BY RANDOM SAMPLING.

Rate	Train	Test	Rules	Length	Time (s)
10%	90.30	90.18	5.7	12.1	325.3
20%	90.27	90.14	5.1	10.2	288.1
30%	90.26	90.14	5.2	10.4	252.1
40%	90.25	90.18	5.0	10.6	216.2
50%	90.23	90.09	4.6	9.5	179.9

TABLE VIII. RESULTS OF TYPE 2 ON THE PAGE-BLOCKS DATA SET BY RANDOM SAMPLING WITH 1-NN.

Rate	Train	Test	Rules	Length	Time (s)
10%	90.29	90.13	5.6	11.6	323.0
20%	90.30	90.12	5.9	12.2	288.0
30%	90.25	90.07	5.1	10.3	252.9
40%	90.25	90.07	5.1	10.1	216.8
50%	90.22	90.09	4.6	9.0	181.6

TABLE IX. RESULTS OF TYPE 2 ON THE PAGE-BLOCKS DATA SET BY RANDOM SAMPLING WITH RANK.

Rate	Train	Test	Rules	Length	Time (s)
10%	90.31	**90.18**	6.1	13.2	323.0
20%	90.32	**90.17**	6.2	13.4	288.0
30%	90.31	**90.20**	5.9	13.4	252.9
40%	90.31	90.19	5.9	12.4	216.8
50%	90.31	90.18	5.6	12.1	181.6

C. *Results on the Satimage Data Set*

Table X shows that we can reduce the computational time by Types 1-4. We employed Type 3 for data reduction because of the highest test data accuracy.

From Tables XI-XIII, the best reduction rate was obtained from random sampling with rank. Its computational time was more than 22 times shorter than that of Type 0. It may be possible to reduce more patterns by this method.

TABLE X. RESULTS ON THE SATIMAGE DATA SET. TYPE 0 IS A NON-PARALLEL APPROACH. TYPE 1-4 ARE PARALLEL APPROACHES.

Type	Train	Test	Rules	Length	Time (s)
0	83.45	82.11	33.7	67.3	10968.7
1	82.65	**81.82**	25.3	50.1	1161.2
2	82.83	**81.97**	21.1	42.0	1035.6
3	83.22	**82.15**	22.0	43.8	1036.3
4	83.04	**81.74**	22.3	43.5	1449.6

TABLE XI. RESULTS OF TYPE 3 ON THE SATIMAGE DATA SET BY RANDOM SAMPLING.

Rate	Train	Test	Rules	Length	Time (s)
10%	83.16	**82.08**	20.8	41.2	911.9
20%	83.07	**82.14**	20.6	40.8	799.8
30%	82.95	**81.91**	19.5	38.9	692.7
40%	82.89	**81.89**	18.3	36.4	570.7
50%	82.76	81.63	17.5	34.6	471.5

TABLE XII. RESULTS OF TYPE 3 ON THE SATIMAGE DATA SET BY RANDOM SAMPLING WITH 1-NN.

Rate	Train	Test	Rules	Length	Time (s)
10%	83.19	**81.90**	21.2	42.2	905.1
20%	83.11	81.70	21.2	42.0	797.7
30%	83.02	**81.94**	19.6	38.7	685.1
40%	82.95	**82.07**	19.0	37.6	574.9
50%	82.79	81.81	18.0	35.4	468.0

TABLE XIII. RESULTS OF TYPE 3 ON THE SATIMAGE DATA SET BY RANDOM SAMPLING WITH RANK.

Rate	Train	Test	Rules	Length	Time (s)
10%	83.22	**82.07**	21.4	42.2	908.2
20%	83.18	**82.09**	20.6	40.8	792.8
30%	83.14	**82.02**	20.4	40.4	694.5
40%	83.12	**82.00**	20.5	40.5	624.8
50%	82.98	**81.94**	19.8	39.0	490.9

D. Results on the Pendig Data Set

From Table XIV, we can see that the test data accuracy by Type 2 was not statistically different from that by Type 0. We used Type 2 for data reduction.

In Tables XV-XVII, we can remove 40% of the training patterns without the deterioration of the test data accuracy by random sampling with rank. But, when the rate was 30%, there was a statistical difference. This may mean that we need a more careful data reduction method. The computation by random sampling with rank (40%) was more than 17 times faster than that of Type 0.

TABLE XIV. RESULTS ON THE PENDIG DATA SET. TYPE 0 IS A NON-PARALLEL APPROACH. TYPE 1-4 ARE PARALLEL APPROACHES.

Type	Train	Test	Rules	Length	Time (s)
0	89.50	88.73	52.4	156.2	30187.5
1	88.85	88.24	39.4	116.5	3101.5
2	89.29	**88.61**	34.3	102.1	2941.1
3	89.19	88.47	38.9	115.3	3031.6
4	89.01	88.38	41.5	123.4	3636.5

TABLE XV. RESULTS OF TYPE 2 ON THE PENDIG DATA SET BY RANDOM SAMPLING.

Rate	Train	Test	Rules	Length	Time (s)
10%	89.23	**88.67**	32.9	97.6	2575.1
20%	89.14	**88.51**	33.1	98.2	2286.3
30%	89.09	**88.58**	32.8	97.6	1980.4
40%	88.93	88.38	32.1	95.4	1673.8
50%	88.78	88.39	32.1	95.6	1381.3

TABLE XVI. RESULTS OF TYPE 2 ON THE PENDIG DATA SET BY RANDOM SAMPLING WITH 1-NN.

Rate	Train	Test	Rules	Length	Time (s)
10%	89.26	**88.72**	34.1	100.9	2590.2
20%	89.09	88.52	33.9	100.5	2287.2
30%	89.05	88.38	33.9	101.0	1992.8
40%	89.02	88.20	32.5	96.7	1694.0
50%	88.82	88.35	30.5	90.6	1391.5

TABLE XVII. RESULTS OF TYPE 2 ON THE PENDIG DATA SET BY RANDOM SAMPLING WITH RANK.

Rate	Train	Test	Rules	Length	Time (s)
10%	89.26	**88.63**	34.2	101.9	2590.0
20%	89.23	**88.61**	34.0	101.0	2296.0
30%	89.16	88.36	32.8	97.9	2004.7
40%	89.11	**88.54**	32.0	95.0	1698.9
50%	88.95	88.40	30.7	91.4	1388.4

IV. CONCLUDING REMARKS

In this paper, we examined the effects of data reduction in our parallel distributed genetic fuzzy rule selection. We used three random sampling methods for data reduction. Through computational experiments with large data sets, we showed that the computation time can be further reduced by using data reduction. We also showed that the obtained classifiers has a smaller number of fuzzy rules when we used our parallel distributed implementation and data reduction

The best data reduction method and its reduction rate depend on the data set. We may need deeper understanding of data complexity for the design of appropriate data reduction methods. We should also discuss the results using other performance measures like AUC, Cohen's Kappa, Geometric Mean from the viewpoint of the analysis on imbalanced data sets.

[1] E. Alba and M. Tomassini, "Parallelism and evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 5, pp. 443-462, 2002.

[2] J. R. Cano, F. Herrera, and M. Lozano, "Stratification for scaling up evolutionary prototype selection," *Pattern Recognition Letters*, vol. 26, no. 7, pp. 953-963, 2005.

[3] J. R. Cano, F. Herrera, and M. Lozano, "On the combination of evolutionary algorithms and stratified strategies for training set selection in data mining," *Applied Soft Computing*, vol. 6, no. 3, pp. 323-332, 2006.

[4] E. Cantu-Paz, "A survey of parallel genetic algorithms," *IlliGAL Report* No. 95003, 1997.

[5] O. Cordon, F. Herrera, F. Hoffman, and L. Magdalena, *Genetic Fuzzy Systems*, World Scientific, 2001.

[6] P. E. Hart, "The condensed nearest neighbor rule," *IEEE Trans. on Information Theory*, vol. 14, no. 5, pp. 515-516, 1968.

[7] F. Herrera, "Genetic fuzzy systems: Taxonomy, current research trends and prospects," *Evolutionary Intelligence*, vol. 1, pp. 27-46, 2008.

[8] H. Ishibuchi, "Multiobjective genetic fuzzy systems: Review and future research directions," *Proc. of 2007 IEEE International Conference on Fuzzy Systems*, pp. 923-918, 2007.

[9] H. Ishibuchi, T. Nakashima, and M. Nii, *Classification and Modeling with Linguistic Information Granules: Advanced Approaches to Linguistic Data Mining*, Springer, Berlin, 2004.

[10] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Selecting fuzzy if-then rules for classification problems using genetic algorithms," *IEEE Trans. on Fuzzy Systems*, vol. 3, no. 3, pp. 260-270, 1995.

[11] H. Liu and H. Motoda, *Instance Selection and Construction for Data Mining*, Kluwer, 1998.

[12] H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*, Kluwer, 1998.

[13] Y. Nojima and H. Ishibuchi, "Computational efficiency of parallel distributed genetic fuzzy rule selection for large data sets," *Proc. of Information Processing and Management of Uncertainty in Knowledge-based Systems*, pp. 1137-1142, 2008.

[14] Y. Nojima, H. Ishibuchi, and I. Kuwajima, "Parallel distributed genetic fuzzy rule selection," *Soft Computing*, vol. 13, no. 5, pp. 511-519, 2009.

[15] Y. Nojima, I. Kuwajima, and H. Ishibuchi, "Data set subdivision for parallel distributed implementation of genetic fuzzy rule selection," *Proc. of 2007 IEEE International Conference on Fuzzy Systems*, pp. 2006-2011, 2007.

[16] D. J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures (4th ed.), Chapman & Hall, 2007.