Computational Efficiency of Parallel Distributed Genetic Fuzzy Rule Selection for Large Data Sets

Yusuke Nojima
Osaka Prefecture University
Gakuen-cho 1-1, Naka-ku, Sakai
Osaka 599-8531, JAPAN
nojima@cs.osakafu-u.ac.jp

Hisao Ishibuchi
Osaka Prefecture University
Gakuen-cho 1-1, Naka-ku, Sakai
Osaka 599-8531, JAPAN
hisaoi@cs.osakafu-u.ac.jp

Abstract

Genetic fuzzy rule selection is a two-phase classification rule mining method. First a large number of candidate fuzzy rules are generated by an association rule mining technique. Then only a small number of generated rules are selected by a genetic algorithm. We have already proposed an idea of parallel distributed implementation of genetic fuzzy rule selection. In this paper, we examine its computational efficiency for large data sets through computational experiments using a cluster system.

Keywords: Pattern Classification, Genetic Rule Selection, Parallel Distributed Implementation.

1 Introduction

Genetic algorithms (GAs) have been successfully used in the design of fuzzy rule-based systems under the name of genetic fuzzy systems [3]. It is, however, difficult to apply genetic fuzzy systems to large data sets. This is because the evaluation of each individual needs long computational time in the application to large data sets. Thus the scalability improvement of genetic fuzzy systems is an important research issue [4, 5]. There are two well-known approaches to the decrease in computational costs for the handling of large data sets in GAs. One is parallel distributed implementation of GAs [1, 2]. The other is data reduction [9, 10, 11]. We have proposed parallel distributed genetic fuzzy rule selection using these two approaches [12]. Our idea is to divide not only population but also a data set into sub-groups. The number of the sub-groups is the same as the number of processors. We have also proposed a systematic re-assignment of sub-populations and data subsets to processors in order to keep the diversity of each sub-population. In this paper, we incorporate an idea of using Pareto and near Pareto optimal rules [6, 8] into our method [12]. Through computational experiments using a cluster system, we examine the computational efficiency of our approach.

2 Genetic Fuzzy Rule Selection

2.1 Fuzzy Rules for Pattern Classification Problems

Let us assume that we have m training (i.e., labeled) patterns $\mathbf{x}_p = (x_{p1}, ..., x_{pn})$, $p = 1, 2, ..., m$ 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 ($i = 1, 2, ..., n$). 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 pattern classification problem, we use fuzzy rules of the following type [7]:

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

where R_q is the label of the q-th fuzzy rule, $\mathbf{x} = (x_1, ..., x_n)$ is an n-dimensional pattern

L. Magdalena, M. Ojeda-Aciego, J.L. Verdegay (eds): Proceedings of IPMU'08, pp. 1137–1142
Torremolinos (Málaga), June 22–27, 2008

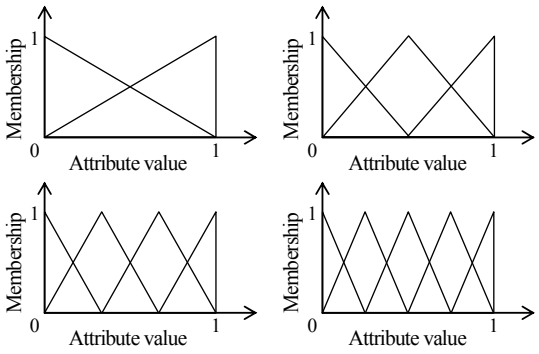

Figure 1: Four fuzzy partitions used in our computational experiments.

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 (i.e., certainty grade). We denote the antecedent part of the fuzzy rule R_q by the fuzzy vector $\mathbf{A}_q = (A_{q1}, A_{q2}, ..., A_{qn})$.

In our computational experiments, we simultaneously use four homogeneous fuzzy partitions with triangular fuzzy sets in Figure 1. Because we usually have no *a priori* information about an appropriate granularity of the fuzzy discretization for each attribute. 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 for each attribute in our computational experiments.

The consequent class C_q and the rule weight CF_q of each fuzzy rule R_q can be heuristically specified by the compatible training patterns with its antecedent part $\mathbf{A}_q$ in the following manner. First we calculate the compatibility grade of each training pattern $\mathbf{x}_p$ with the antecedent part $\mathbf{A}_q$ of the fuzzy rule R_q using the product operation as:

$$\mu_{\mathbf{A}_q}(\mathbf{x}_p) = \mu_{A_{q1}}(x_{p1}) \cdot ... \cdot \mu_{A_{qn}}(x_{pn}), \quad (2)$$

where $\mu_{A_{qi}}(\cdot)$ is the membership function of A_{qi}. Next we calculate the confidence of the fuzzy rule "$\mathbf{A}_q \Rightarrow$ Class h" for each class ($h = 1, 2, ..., M$) as follows:

$$c(\mathbf{A}_q \Rightarrow \text{Class } h) = \frac{\sum_{\mathbf{x}_p \in \text{Class } h} \mu_{\mathbf{A}_q}(\mathbf{x}_p)}{\sum_{p=1}^{m} \mu_{\mathbf{A}_q}(\mathbf{x}_p)}. \quad (3)$$

The consequent class C_q is specified by identifying the class with the maximum confidence:

$$c(\mathbf{A}_q \Rightarrow \text{Class } C_q) = \max_{h=1, 2, ..., M} \{c(\mathbf{A}_q \Rightarrow \text{Class } h)\}. \quad (4)$$

When there is no pattern in the fuzzy subspace defined by $\mathbf{A}_q$, we do not generate any fuzzy rules with $\mathbf{A}_q$ in the antecedent part. When multiple classes have the same maximum value in (4), we do not generate any fuzzy rules with $\mathbf{A}_q$, either.

The rule weight CF_q of each fuzzy rule R_q has a large effect on the performance of fuzzy rule-based classifiers. Different specifications of the rule weight have been proposed and examined in the literature. We use the following specification because good results were reported by this specification in [7]:

$$CF_q = c(\mathbf{A}_q \Rightarrow \text{Class } C_q) - \sum_{\substack{h=1 \\ h \neq C_q}}^{M} c(\mathbf{A}_q \Rightarrow \text{Class } h). \quad (5)$$

When the CF_q is negative, we do not generate any fuzzy rules with $\mathbf{A}_q$.

In the same manner as the confidence, the support of the fuzzy rule "$\mathbf{A}_q \Rightarrow$ Class h" is calculated as follows:

$$s(\mathbf{A}_q \Rightarrow \text{Class } h) = \frac{\sum_{\mathbf{x}_p \in \text{Class } h} \mu_{\mathbf{A}_q}(\mathbf{x}_p)}{m}. \quad (6)$$

The confidence and support of each rule are used for rule evaluation in rule extraction.

2.2 Fuzzy Rule Evaluation

Using the above-mentioned procedure, we can generate a large number of fuzzy rules by specifying the consequent class and the rule weight for each of the 15^n combinations of the antecedent fuzzy sets. It is, however, very difficult for human users to handle such a large number of generated fuzzy rules. It is also very difficult for human users to intuitively understand long fuzzy rules with many antecedent conditions. Thus we only generate

short fuzzy rules with a few antecedent conditions. It should be noted that *don't care* conditions with the antecedent interval $[0, 1]$ can be omitted from fuzzy rules. Thus the rule length means the number of antecedent conditions excluding *don't care* condition. In the field of data mining, two rule evaluation criteria (i.e., confidence and support) have often been used. We examine only short fuzzy rules of length $L_{\max}$ or less (e.g., $L_{\max} = 3$) which satisfy the minimum confidence and support. This restriction is to find a small number of short fuzzy rules.

In our previous work [6], we showed that most of selected fuzzy rules by genetic fuzzy rule selection are Pareto-optimal and near Pareto-optimal rules in terms of confidence and support. Thus we use those fuzzy rules as candidate rules in genetic fuzzy rule selection, which are defined by the following ε-dominance. A rule R_i is said to be ε-dominated by another rule R_j when both the two inequalities in (7) hold and at least one of the two inequalities in (8) holds:

$$\begin{aligned} &\mathit{Confidence}(R_i) + \varepsilon \leq \mathit{Confidence}(R_j), \\ &\quad \mathit{Support}(R_i) + \varepsilon \leq \mathit{Support}(R_j). \end{aligned} \tag{7}$$

$$\begin{aligned} &\mathit{Confidence}(R_i) + \varepsilon < \mathit{Confidence}(R_j), \\ &\quad \mathit{Support}(R_i) + \varepsilon < \mathit{Support}(R_j). \end{aligned} \tag{8}$$

When a rule R_i is not dominated by any other rules in the sense of ε-dominance in (7) and (8), we call R_i an ε-non-dominated rule.

In order to remove weak Pareto-optimal rules with small support values, we modify the minimum support for each class depending on Pareto-optimal rules and the value of ε as follows [8]:

$$\begin{aligned} s_{\text{Class } h} = \max\{&\mathit{Support}(R_q) \mid C_q = \text{Class } h, \\ &\mathit{Confidence}(R_q) = 1\} - \varepsilon, \end{aligned} \tag{9}$$

where $s_{\text{Class } h}$ is the modified minimum support for class h.

Figure 2 shows extracted candidate fuzzy rules for the Pendig data in the UCI machine learning repository. We specified the minimum confidence and support, and ε as 0.5, 0.02, and 0.01, respectively.

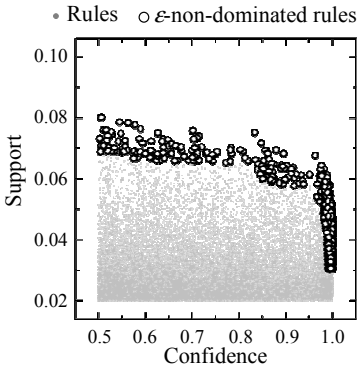

Figure 2: Candidate fuzzy rules with Class 1 consequent for the Pendig data. All rules satisfying the minimum confidence and support, and the maximum rule length are shown by grey dots. ε-non-dominated rules satisfying the modified minimum support are shown by open circles.

2.3 Combinatorial Optimization of Rule Sets by a Genetic Algorithm

Let us assume that N candidate fuzzy rules have already been extracted. The task of genetic fuzzy rule selection is to design an accurate and compact fuzzy rule-based classifier from the N candidate fuzzy rules.

Any subset S of the N candidate fuzzy rules can be denoted by a binary string of length N as $S = s_1 s_2 \cdots s_N$ where $s_i = 1$ and $s_i = 0$ mean that the i-th candidate fuzzy rule is included in and excluded from the rule set S, respectively. Such a binary string is used as an individual in genetic fuzzy rule selection.

In this paper, we use the following weighted sum fitness function:

$$\mathit{fitness}(S) = w_1 \cdot f_1(S) - w_2 \cdot f_2(S) - w_3 \cdot f_3(S), \tag{10}$$

where w_1, w_2, and w_3 are pre-specified non-negative weights, and each objective function f_i is

$f_1(S)$: The number of correctly classified training patterns by S,

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

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

When the first objective is calculated, each pattern is classified by a single winner rule

with the maximum product of the rule weight and the compatibility grade in S. We use the single winner-based method without random tiebreak to evaluate the accuracy of the rule set S. Thus only a single rule is responsible for the classification of each training pattern. As a result, some fuzzy rules may be used for the classification of no training patterns. Whereas the existence of such an unnecessary fuzzy rule in S has no effect on the first objective, it deteriorates the second and third objectives. Thus we remove from S all the unnecessary rules responsible for the classification of no training patterns before the second and third objectives are calculated. After removing all the unnecessary rules, the second and third objectives are calculated. By maximizing (10), we can obtain an accurate and compact fuzzy classifier.

2.4 Parallel Distributed Implementation

We use a cluster system with a single server and a number of clients. Concretely saying, six workstations are used as clients in our computational experiments. Each workstation has two CPUs (Xeon 3.6GHz $\times$ 2). Thus this cluster system has 12 clients.

Our idea is to divide not only a population but also a training data set. They are divided into the same number of sub-populations and training data subsets, which is also the same as the number of clients. Thus, the training data set and the population are divided into 12 training data subsets and 12 sub-populations, respectively.

Since each sub-population is likely to overfit to the corresponding training data subset, we change the assignment of the training data subsets to the clients after a pre-specified number of generations (every ten generations in this paper).

Our parallel distributed implementation of genetic fuzzy rule selection is written as follows:

Phase I: Candidate Rule Extraction

Step 1: Extract candidate fuzzy rules in the same manner as in Section 2. This phase is executed on the clients by using all the training data. Let the number of extracted candidate fuzzy rules be N.

Phase II: Genetic Optimization

Step 2: Randomly generate binary strings of length N as an initial population on the server.

Step 3: Randomly divide the current population and the training data set into sub-populations and training data subsets, respectively, on the server.

Step 4: Distribute the sub-populations and the training data subsets from the server to the clients.

Step 5: Evaluate each string in the assigned sub-population using the assigned training data subset on each client.

Step 6: Execute genetic fuzzy rule selection for a pre-specified computation load (which is specified by the total number of evaluated strings in this paper) on each client using the assigned training data subset and the assigned sub-population.

Step 7: Systematically change the assignment of the training data subsets to the clients (e.g., from the first client to the second one, from the second one to the third one, ..., and from the 12th one to the first one in the case of 12 clients).

Step 8: If a pre-specified termination condition (the total number of evaluated strings in this paper) is not satisfied, return to Step 5. Otherwise go to Step 9.

Step 9: Calculate the fitness value of each string in each sub-population using the whole training data set on the server. Choose the best string as the final solution (i.e., as the finally obtained fuzzy rule-based classifier).

Our parallel distributed implementation decreases the computational time by the magnitude of the square of the number of clients. For example, it is ideally 144 times faster than the original non-parallel algorithm when we have 12 clients. This is because both the population size and the number of training patterns at each client are 1/12 of those in the original non-parallel algorithm.

Table 1: Data sets used in our experiments.

Data set	Attributes	Patterns	Classes
Pendig	16	10992	10
Letter	16	20000	26

Table 2: Minimum confidence and support, the ε value, and the number of generated candidate rules for each data set.

Data set	Conf.	Supp.	ε	Rules
Pendig	0.5	0.02	0.01	5080.9
Letter	0.5	0.001	0.001	2268.0

3 Computational Experiments

Through computational experiments on two benchmark data sets in the UCI machine learning repository, we examined the computational efficiency of the proposed parallel distributed implementation.

Table 1 shows the two benchmark data sets used in our computational experiments. We evaluated the generalization ability of obtained fuzzy rule-based classifiers by a single ten-fold cross validation procedure (i.e., 10 runs).

We first extracted candidate fuzzy rules using the ε-dominance in terms of confidence and support. Table 2 shows the minimum support, the minimum confidence, and ε used for each data set. We also show the average number of extracted candidate rules in Table 2. The weight vector in the weighted sum fitness function in (10) was specified as $\mathbf{w} = (100, 1, 1)$. We examined four specifications of the population size and the termination condition in this paper. The first specification is for non-parallel genetic fuzzy rule selection. The population size and the termination condition were 240 and 240240 string examinations (i.e., an initial population with 240 strings plus 1000 generation updates), respectively.

The other three specifications are for parallel genetic fuzzy rule selection. We specified the population sizes as 120, 240, and 480, respectively. Thus the sub-population sizes were 10, 20, and 40, respectively. We specified the termination condition 240120, 240240, and 240480 string evaluations for each case.

Tables 3 and 4 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 (hour: minute: second) over the ten-fold cross validation procedure (i.e., over 10 runs) for each specification. The first column shows the population and sub-population size of each specification. Thus, the case of 240 (240) means non-parallel genetic fuzzy rule selection. The case of 120 (10) means that the total population size and the sub-population size are 120 and 10, respectively.

From these tables, we can see that the average training and test data accuracy of parallel distributed genetic fuzzy rule selection were worse than those of non-parallel ones. But the accuracy of parallel distributed ones became close to that of non-parallel one when we specified the sub-population size larger. We can also see that our parallel distributed implementation drastically decreased the average CPU time of the original non-parallel algorithm.

An interesting observation is that fuzzy rule-based classifiers with lower complexity (i.e., a smaller number of rules and a shorter total rule length) were obtained by our parallel distributed implementation. While the number of rules in the classifier obtained by non-parallel algorithm was about 63, that obtained by parallel distributed implementation with small sub-populations (e.g., 10) was about 25 for the Pendig data set.

4 Conclusions

In this paper, we demonstrated the computational efficiency of our parallel distributed implementation of genetic fuzzy rule selection for large data sets. Through computational experiments, it was shown that the proposed parallel distributed implementation found fuzzy rule-based classifiers with almost the same test data accuracy as the original non-parallel algorithm while the computa-

Table 3: Results on Pendig data set.

Pop. size	Training acc.	Test acc.	Number of rules	Total rule length	CPU Time
240 (240)	90.71	89.88	62.6	185.4	22:47:37
120 (10)	88.99	88.67	25.4	75.0	0:14:47
240 (20)	89.47	89.16	27.5	81.6	0:10:08
480 (40)	89.83	89.20	29.1	86.6	0:09:54

Table 4: Results on Letter data set.

Pop. size	Training acc.	Test acc.	Number of rules	Total rule length	CPU Time
240 (240)	52.97	52.36	72.8	217.5	35:34:41
120 (10)	51.20	50.86	39.2	117.0	0:17:46
240 (20)	51.29	50.78	40.3	119.2	0:13:23
480 (40)	51.52	51.16	43.2	128.9	0:14:22

tional time can be drastically decreased (i.e., more than 100 times faster than non-parallel genetic fuzzy rule selection.).

Acknowledgements

This work was partially supported by Grant-in-Aid for Young Scientists (B): KAKENHI (18700228).

References

[1] E. Alba, M. Tomassini, Parallelism and Evolutionary Algorithms, *IEEE Transactions on Evolutionary Computation*, **6** 5 (2002) 443-462.

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

[3] O. Cordon, F. Herrera, F. Hoffman, L. Magdalena, *Genetic Fuzzy Systems*, World Scientific (2001).

[4] F. Herrera, Genetic fuzzy systems: Status, critical considerations and future directions, *International Journal of Computational Intelligence Research*, **1** 1 (2005) 59-67.

[5] H. Ishibuchi, Evolutionary multiobjective design of fuzzy rule-based systems, *Proc. of 1st IEEE Symposium on Foundations of Computational Intelligence*, (2007) 9-16.

[6] H. Ishibuchi, I. Kuwajima, Y. Nojima, Use of Pareto-optimal and near Pareto-optimal rules as candidate rules in genetic fuzzy rule selection, *P. Melin et al (eds.): Analysis and Design of Intelligent Systems using Soft Computing Techniques*, Springer, Berlin (2007) 387-396.

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

[8] H. Ishibuchi, I. Kuwajima, Y. Nojima, Prescreening of candidate rules using association rule mining and Pareto-optimality in genetic rule selection, *Proc. of 11th International Conference on Knowledge Based Intelligent Information & Engineering Systems*, (2007) 509-516.

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

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

[11] J. R. Cano, F. Herrera, M. Lozano, Stratification for scaling up evolutionary prototype selection, Pattern Recognition Letters, **26** 7 (2005) 953-963.

[12] Y. Nojima, H. Ishibuchi, I. Kuwajima, Parallel Distributed Genetic Fuzzy Rule Selection, *Soft Computing*, (accepted).