On Effective Decomposition of Training Data Sets for Min-Max Modular Classifier

Hai Zhao, Bao-Liang Lu , Yi-Min Wen, and Kai-An Wang
Department of Computer Science and Engineering, Shanghai Jiao Tong University
1954 Hua Shan Rd., Shanghai 200030, China
{zhaohai,blu}@cs.sjtu.edu.cn

***Abstract*— Our previous work shows that traditional randomization partition method for min-max modular (M^3) classifier can not ensure stable generalization accuracy when the number of two-class problems increases. To overcome this drawback, we consider how to effectively decompose the training data set for a two-class problem in this paper. We propose four basic clustering and anti-clustering strategies and their combinations for partitioning training data sets. These four basic strategies are hyperplane decomposition, K-means algorithm, anti-K-means algorithm, and scatter procedure. Our experimental results show that all the proposed clustering and anti-clustering strategies are superior to the traditional random partition method.**

I. Introduction

Modular classification is often aimed to process large-scale data more flexibly and concurrently. However, it is equally important for combining classifier to keep combining accuracy stable while the number of modules increases. Thus, there comes the problem of how to decompose training data set effectively and to ensure a stable combining accuracy. Clustering, or another name, unsupervised learning, is a good choice for such requirements. Actually, related fields have been paid more and more attention in recent years. Giacinto and Roli [1] consider a so-called 'overproduced' scheme for automatic base classifier production, a clustering method is used for such scheme. Frosyniotis et al [2]use Fuzzy C-means and greedy-EM algorithms to improve combining classification. Chawla et al [3] also use Fuzzy C-means algorithm with a bagging procedure to decompose training set. These studies show that clustering-based training set partition often improve classification performance and is better than the case that a single classifier is applied.

In this paper, we will consider other helpful constraints for large-scale classification . One is balanced-loading problem, which requires each base classifier with similar training or test cost in order to avoid a bottleneck in systems. The other is balanced learning for a single base classifier, which not always but often requires the number of samples in each class keep close.

Decomposition of two-class classification of min-max modular classifier (M^3) can meet these two requirements [4], which is our central object of this study. Another combining classifier technology similar to M^3 classifier is "bites" suggested by Breiman and Chawla [5] [6]. The difference between two approaches is that the former decomposition performs in all the samples, while the latter just in single class. Therefore, M^3 classifier holds more flexibility on decomposition of training data set than bites. Distributed Boosting algorithm introduced by Lazarevic et al [7] is a parallel version of boosting algorithm, which performs similar decomposition of training set in samples of all classes just like bites method. In addition, the algorithm needs much communication among every base classifiers.

Traditionally, the task decomposition used in M^3 classifier is based on a random strategy [4]. But our previous work pointed out that such partition procedure is not always effective for keeping a stable combining accuracy. For example, experimental study shows k-NN algorithm is very sensitive for small scale classification [8]. In this paper, we will propose four basic strategies and their combinations to improve partition procedure of training data sets.

II. Min-Max Modular Modular Classifier

Consider a two-class classification problem, whose output coding of class labels are denoted by C_0 and C_1(as two values, $C_1 > C_0$), or equally, 0 and 1, which will be concise and not lose any generality for our algorithm description. Suppose the training set of class C_0 is decomposed into n subsets and the training set of class C_1 is divided into m subsets. By arranging those m and n training subsets in pairs, we obtain $m \times n$ training set pairs. Each pair is learned by a single binary sub-classifier. Therefore, a larger-scale two-class problem can be decomposed into $m \times n$ smaller subproblems. We call the binary sub-classifier as base classifier.

Suppose all produced training set pairs are denoted by X_{ij}, for $i = 0, 1, ..., m-1$, and $j = 0, 1, ..., n-1$. Without misunderstanding, we also express X_{ij} as the output of the corresponding base classifier.

Min-max combination defines how the outputs of those $m \times n$ base classifiers are combined into the solution to the original problem. Before combination, a grouping operation on $m \times n$ base classifiers should be done: these base classifiers, X_{ij}, for $j = 0, ..., n-1$, are defined as one "C_1 group" and i is defined as its group label, and those base classifiers, X_{ij}, for $i = 0, ..., m-1$, are defined as one "C_0 group" and j is defined as its group label.

Min-max combination of all base classifiers includes two stages: Firstly, combination rule *Min* is applied to each C_1 group to make the output of the group. Secondly, the outputs

of all groups are integrated by combination rule *Max* to make the final output of the original two-class classification problem.

III. Clustering Algorithms for Training Set Partition

Combining accuracies under three partition states of training set will be compared. The first is randomization partition, which is the traditional strategy. The second is clustering partition. The third is so called anti-clustering partition, which means maximize distances among samples in the same cluster. Since these three states of partition are typical, it is beneficial for a comparison study of different decomposition processing.

Four basic clustering and anti-clustering algorithms are introduced.

A. K-means and Anti-K-means Algorithm

K-means algorithm is a clustering algorithm which continuously adjust K clustering centers to minimize inner-cluster distance in order to separate samples far away [10]. The algorithm focuses in finding optimized center for each cluster, that is, K clustering centers, $\mathbf{m}_j$, for $i = 1, 2, ..., k$, and assign each sample to the cluster which the nearest cluster center belongs to. For any sample $\mathbf{x}_i$ in a cluster and its center $\mathbf{m}_j$, inner-cluster distance can be written by

$$E = \sum_{j=1}^{K} \sum_{\mathbf{x}_i \in \omega_j} \|\mathbf{x}_i - \mathbf{m}_j\|^2 \tag{1}$$

K-means algorithm is described as the following.

a) Specify the number of clusters, K, and corresponding center $\mathbf{m}_j$. Let δ denote minimum iteration error, and T maximum iteration times. Let counter $t = 1$.
b) Assign the sample $\mathbf{x}_i$ to the cluster which its nearest center $\mathbf{m}_j$ belongs to.
c) Compute new center $\mathbf{m}_j^{(t+1)}$ and inner-cluster distance $E^{(t+1)}$.
d) Repeat steps b) and c) until $t = T$ or $\|E^{(t+1)} - E^{(t)}\| < \delta$.

As for anti-K-means algorithm, it is an inverse version of original K-means algorithm. To modify b) as, "Assign the sample $\mathbf{x}_i$ to the cluster which its farthest center $\mathbf{m}_j$ belongs to" will be anti-K-means algorithm, which is to maximize inner-cluster distance E defined in (1) in order to separate samples nearby, instead of minimizing E in original K-means algorithm.

B. Hyperplane Decomposition of Training Set

Hyperplane decomposition we suggest is actually a simple clustering method, which use a group of hyperplane partition training set. Two sub-space will be both convex after the whole space is divided by a hyperplane. Thus, this partition method may ensure that partitioned subset are still clustered in some degree.

Suppose hyperplane equation in linear space of n dimensions is $A(\mathbf{x}-\mathbf{b}) = 0$, where $A = [a_1, ..., a_n]$ is normal vector of hyperplane, $\mathbf{x} = [x_1, ..., x_n]^T$ is any point in hyperplane and $\mathbf{b} = [b_1, ..., b_n]^T$ is a known point in hyperplane.

Hyperplane decomposition of training set is to meet two requirements. One is to make all used hyperplane parallel, which reduces partitioned un-convex subsets, the other is that partitioned subsets have any specified number of samples, which ensures elastic decomposition at most degree. To simplify the realization of the algorithm, We firstly suppose normal vector A has been decided (Experimentally, let $A = [1, 1, ..., 1]$). Secondly, a base hyperplane, without losing generalization, hyperplane $P : A\mathbf{x} = 0$ is specified. Thirdly, vertical distance between Sample $\mathbf{x}$ and P is calculated. Fourthly, a sorting operation is performed in all samples according to calculated distances. Finally, Extract samples in turn according to the sorted order to finish partition of training set. Thus, hyperplane decomposition is equally a sorting procedure of all samples under weight vector A.

C. Scatter Procedure after Clustering

Scatter procedure is to extract equal quantity of samples from each cluster in turn after a clustering procedure has been done. It is interesting that a scatter procedure will transform decomposition of clustering into anti-clustering decomposition on the very big level, or vice versa. In addition, it is used in another aspect: Though many clustering algorithms are effective as clustering procedures themselves, they all hold a shortcoming according to our requirements for partition of training set, that is, subsets the clustering algorithm yields are often not of equal sizes. an equal-sized scatter procedure may partially overcome such difficulty.

In order to avoid the scatter procedure disordering original clustered data too much, just like hyperplane decomposition above, we also introduce a weighted sorting operation for each cluster before scatter procedure is performed. Such scatter procedure is named after hyperplane scatter. All following scatter procedures will refer to these ones.

IV. Experiments

Two algorithms, k-NN and SVM with RBF kernel function are taken as base classifiers, respectively. k varies from 1 to 40 under k-NN algorithm, the average combining accuracy is taken in all values of k. Twelve data sets from STATLOG benchmark repository [11] are chosen for this study, however, here we will only demonstrate two of them for space limitation. Data set information and kernel parameters of SVM are shown in Table I. All kernel parameters are from recommended ones in the document of STATLOG benchmark repository.

Consider K-means algorithm is sensitive for initial partition of cluster centers, we take training set processed by hyperplane sorting as another optional initial partition for K-means algorithm. Correspondingly, A hyperplane scatter initial partition means that a scatter procedure is performed after a sorting operation is applied to the training set, it will be set as one optional initial partition of anti-K-means algorithm, too.

All partition operations are carried out in three optional steps, which yields ten clustering or anti-clustering algorithms

TABLE I

DISTRIBUTIONS OF SAMPLES IN DATA SETS AND CORRESPONDING KERNEL PARAMETERS OF SVM TRAINING

Data Set	Number of Training Samples			Number of Test Samples			Kernel Parameters	
	Total	Positive	Negative	Total	Positive	Negative	γ	C
Breast cancer	20000	14118	5882	7700	5482	2218	0.2	1.519
Heart	17000	9440	7560	10000	5560	4440	0.008333	3.162

TABLE II

CLUSTERING AND ANTI-CLUSTERING ALGORITHMS TYPES

Algorithm Type	First Step		Second Step		Third Step
	Hyperplane	Hyperplane Scatter	K-means	Anti-K-means	Hyperplane Scatter
RAN					
HP	Yes				
KM			Yes		
HPKM	Yes		Yes		
KMHS			Yes		Yes
HPKMHS	Yes		Yes		Yes
HS		Yes			
AKM				Yes	
HSAKM		Yes		Yes	
AKMHS				Yes	Yes
HSAKMHS		Yes		Yes	Yes

and their combination. Their notations are shown in table II. The smaller classes in every training sets are divided into 2 to 26 equal-sized subsets, respectively, and the larger classes in every training sets are decomposed to equal-sized subsets, correspondingly. SVM program uses LibSVM 2.6 [12]. The vector $\mathbf{e}$ is computed by MATLAB 7.0. All experiments are performed in a PC with 2.8G CPU and 2G RAM.

The comparative results of combining accuracies and response performances under different clustering algorithms are shown in Fig 1-6, where data of response time are from mean values of those of all base classifiers, which reflects the effectiveness of testing procedure under parallel mode. Fig 7-8 show a ratio of CPU time between clustering procedure and total procedure with clustering and training, where data of training time are the sum of all base classifiers.

The comparative results of combining accuracies and response performances under different anti-clustering algorithms are shown in Fig 9-14. Data of response time are from mean values of those of all base classifiers, which reflects the effectiveness of testing procedure under parallel mode. Fig 15-16 show a ratio of CPU time between anti-clustering procedure and total procedure with anti-clustering and training, where data of training time are the sum of all base classifiers.

A. comments

Experimental results show that traditional randomization partition of training set can not ensure stable combining accuracies while partition number increases. Thus it can not be taken as one credible partition strategy for training set. All presented clustering or anti-clustering strategies have some effectiveness on ensuring combining accuracy in various degrees. Moreover, response performance of combining classifier

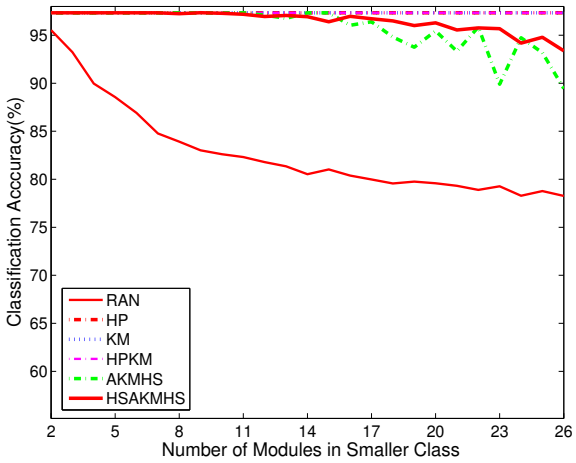

Fig. 1. Breast cancer data set: average combining accuracies after different clustering algorithms under k-NN algorithm

after clustering or anti-clustering partition is better than that after randomization partition, too.

Hyperplane decomposition method gives similar effectiveness compared to K-means method. However, it costs little on computation and is easy to realize an equally parallel version of algorithm for concurrent application [13], [14]. What's more, hyperplane decomposition processed data set can be regarded as a better initial partition of clustering algorithm.

Randomization based anti-K-means partition takes on similar effectiveness as randomization partition, while hyperplane scatter method based anti-K-means partition shows much better effectiveness. This suggests that anti-K-means algorithm is more sensitive for initial partition than K-means algorithm.

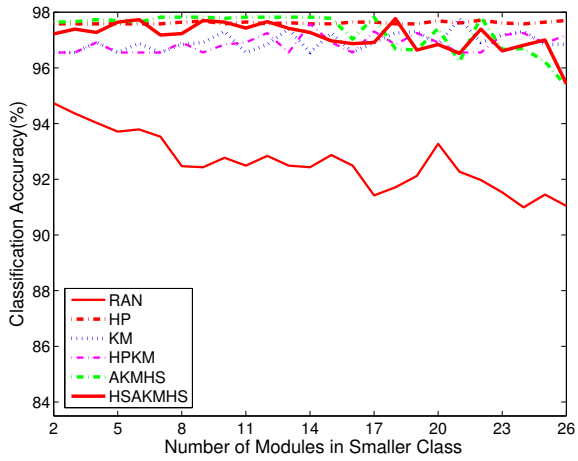

Fig. 2. Breast cancer data set: combining accuracies after different clustering algorithms under SVM algorithm

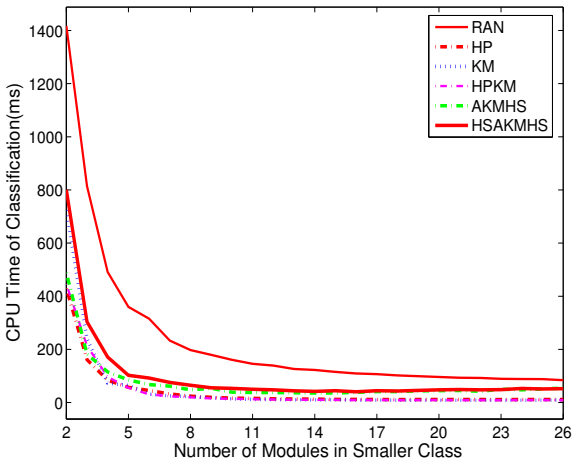

Fig. 5. Breast Cancer data set: combining performance after different clustering algorithms under SVM algorithm

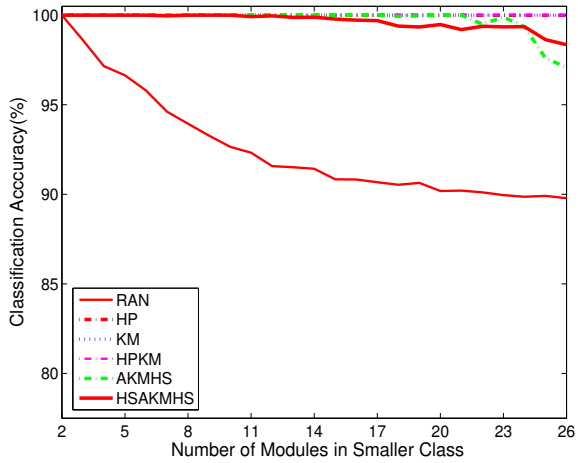

Fig. 3. Heart data set: average combining accuracies after different clustering algorithms under k-NN algorithm

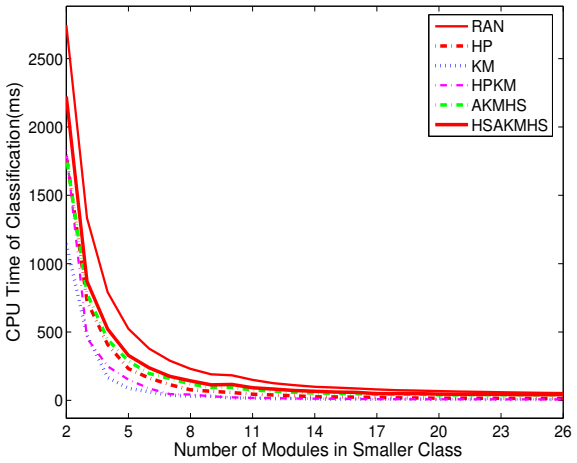

Fig. 6. Heart data set: combining performance after different clustering algorithms under SVM algorithm

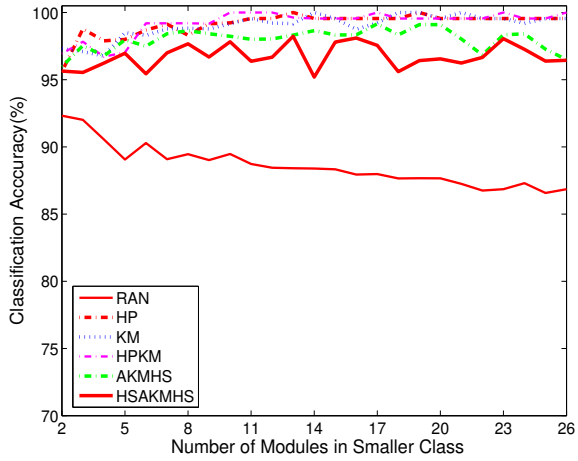

Fig. 4. Heart data set: combining accuracies after different clustering algorithms under SVM algorithm

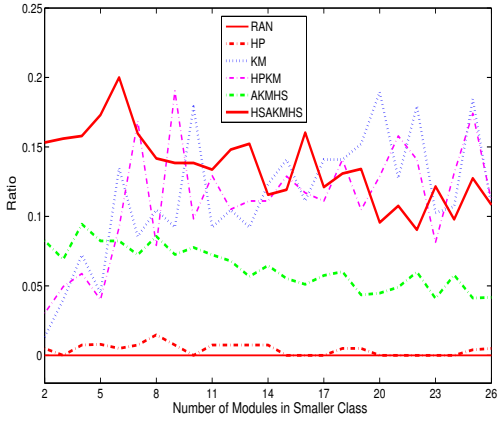

Fig. 7. Breast Cancer data set: the ratio of CPU time between clustering and total procedure with clustering and training under SVM algorithm

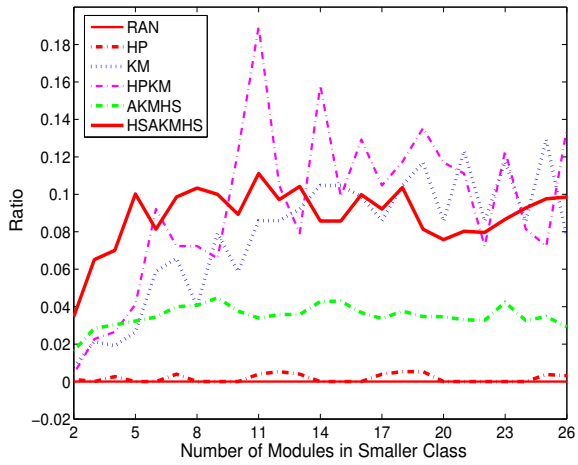

Fig. 8. Heart data set: the ratio of CPU time between clustering and total procedure with clustering and training under SVM algorithm

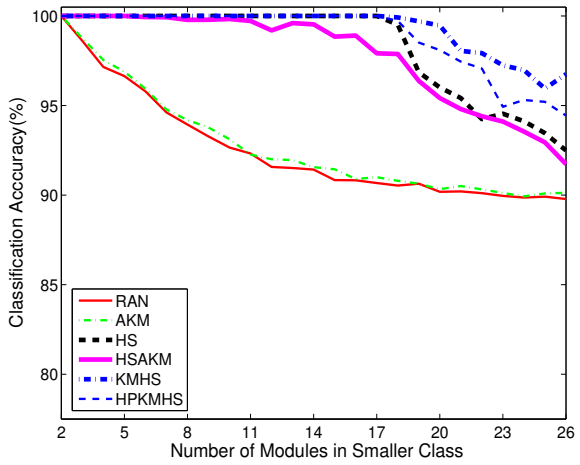

Fig. 11. Heart data set: average combining accuracies after different anti-clustering algorithms under k-NN algorithm

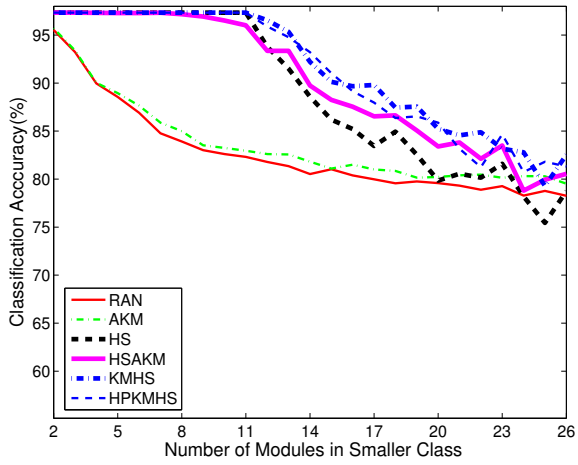

Fig. 9. Breast cancer data set: average combining accuracies after different anti-clustering algorithms under k-NN algorithm

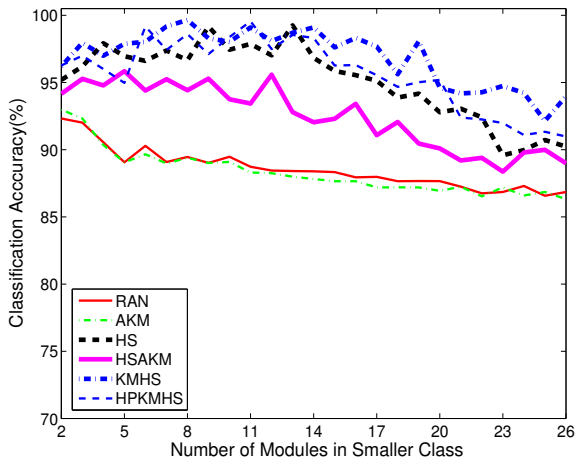

Fig. 12. Heart data set: combining accuracies after different anti-clustering algorithms under SVM algorithm

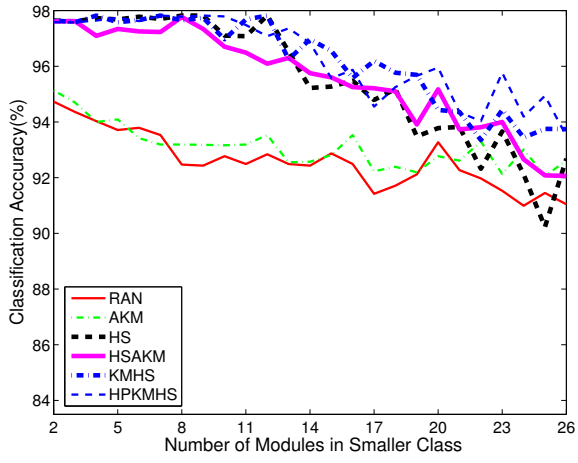

Fig. 10. Breast cancer data set: combining accuracies after different anti-clustering algorithms under SVM algorithm

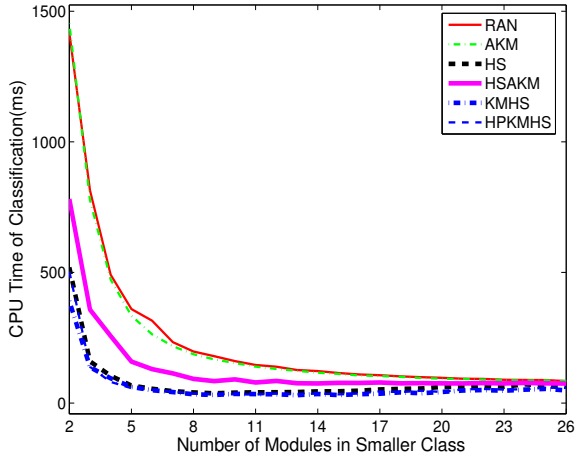

Fig. 13. Breast Cancer data set: combining performance after different anti-clustering algorithms under SVM algorithm

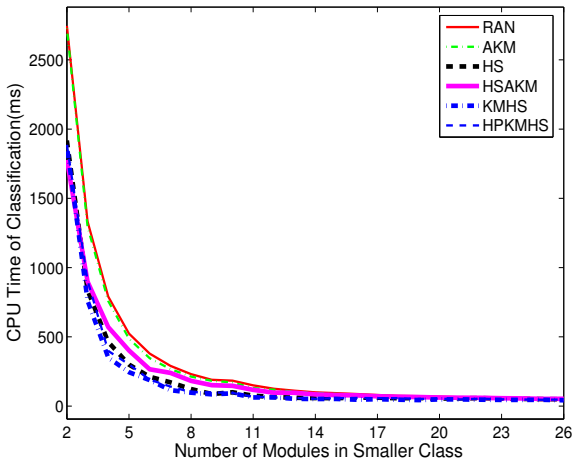

Fig. 14. Heart data set: combining performance after different anti-clustering algorithms under SVM algorithm

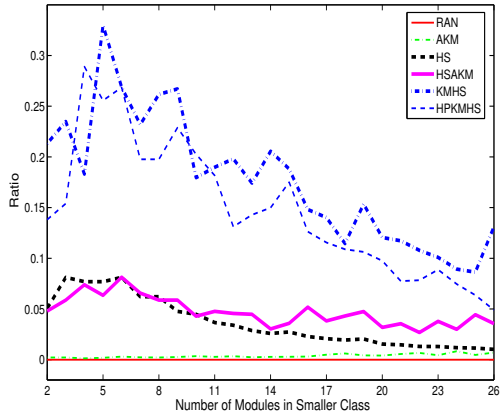

Fig. 15. Breast Cancer data set: the ratio of CPU time between anti-clustering and total procedure with anti-clustering and training under SVM algorithm

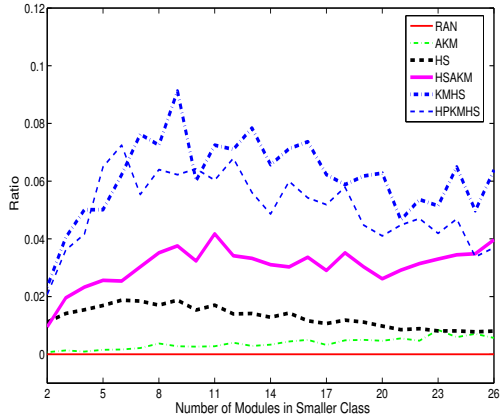

Fig. 16. Heart data set: the ratio of CPU time between anti-clustering and total procedure with anti-clustering and training under SVM algorithm

V. Conclusion

Four basic clustering and anti-clustering strategies and their combinations are proposed for improving training data set partition for min-max modular classifier.

Our study show that training data set partition can be handled from two contrary ways, clustering or anti-clustering. Experimental results show that the presented strategies may all ensure combining accuracy in various degrees, which can be seen as an improvement of traditional random partition method. A scatter procedure is presented, which can transform clustering results into anti-clustering ones to a great extent, or vice versa. Its effectiveness ulteriorly shows clustering and anti-clustering method should be both the rational directions of processing on partition of training data set. Except for improvement of combining accuracy, better response performance is also obtained after clustering or anti-clustering partition, too.

Acknowledgements

This research was partially supported by the National Natural Science Foundation of China via the grants NSFC 60375022 and NSFC 60473040. The authors would like to give their thankfulness to YANG Yang for her helpful advices.

References

[1] Giorgio Giacinto, Fabio Roli, "Automatic Design of Multiple Classifier Systems by Unsupervised Learning", Proceedings of Machine Learning and Data Mining in Pattern Recognition, First International Workshop, MLDM'99, Leipzig, Germany, September 16-18, 1999, 131-143

[2] D. Frosyniotis, A. Stafylopatis and A. Likas, "A divide-and conquer method for multi-net classifiers", Pattern Anal Applic (2003) 6: 32-40

[3] Nitesh V. Chawla, Steven Eschrich, Lawrence O. Hall, "Creating Ensembles of Classifiers", Technical Report ISL-01-01, University of South Florida, http://isl.ceee.usf.edu/report, 2001

[4] B. L. Lu, M. Ito, "Task Decomposition and Module Combination Based on Class Relations: a Modular Neural Network for Pattern Classification", IEEE Transactions on Neural Networks, Vol.10 (1999) 1244-1256

[5] L. Breiman, "Pasting bites together for prediction in large data sets", Machine Learning, 36(2) (1999) 85-103

[6] Nitesh V. Chawla, Lawrence O. Hall, Kevin W. Bowyer and W. Philip Kegelmeyer, "Learning Ensembles from Bites: A Scalable and Accurate Approach", Journal of Machine Learning Research 5(2004): 421-451

[7] Aleksandar Lazarevic, Zoran Obradovic, "Boosting Algorithms for Parallel and Distributed Learning", Distributed and Parallel Databases, 11(2): 203-229 (2002)

[8] Hai Zhao and Bao-Liang Lu, "A Modular k-Nearest Neighbor Classification Method for Massively Parallel Text Categorization", International Symposium on Computational and Information Sciences(CIS'04), LNCS, Shanghai, China, December, 2004

[9] B. L. Lu, K. A. Wang, M. Utiyama, H. Isahara, "A part-versus-part method for massively parallel training of support vector machines", In: Proceedings of IJCNN'04, Budapast, July 25-29(2004) 735-740

[10] S. Lloyd, "Least squares quantization in PCM", Technical report, Bell Laboratories. (1957) Published in 1982 in IEEE Trans. Inf. Theory, Vol. 28: 128-137

[11] G. Ratsch, T. Onoda and K. Muller, "Soft margins for AdaBoost", Machine Learning, 2000, 1-35[http://ida.first.fhg.de/projects/bench/]

[12] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM: a library for sup-port vector machines", 2001, Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm

[13] Laxmikant V. Kal, Sanjeev Krishnan, "A Comparison Based Parallel Sorting Algorithm", ICPP 1993: 196-200

[14] William Gasarch, Evan Golub, Clyde Kruskal, "Constant Time Parallel Sorting: An Empirical View", Journal of Computer and System Sciences, Volume 67, Issue 1, August 2003, 63-91