A Hierarchical and Parallel Method for Training Support Vector Machines*

Yimin Wen[1,2] and Baoliang Lu[1]

[1] Department of Computer Science and Engineering, Shanghai Jiao Tong University, 1954 Hua Shan Rd., Shanghai 200030, China
{wenyimin,blu}@cs.sjtu.edu.cn
[2] Hunan Industry Polytechnic, Changsha 410007, China

Abstract. In order to handle large-scale pattern classification problems, various sequential and parallel classification methods have been developed according to the divide-and-conquer principle. However, existing sequential methods need long training time, and some of parallel methods lead to generalization accuracy decreasing and the number of support vectors increasing. In this paper, we propose a novel hierarchical and parallel method for training support vector machines. The simulation results indicate that our method can not only speed up training but also reduce the number of support vectors while maintaining the generalization accuracy.

1 Introduction

In the last decade, there are many surges of massive data sets. It is necessary to develop efficient methods to deal with these large-scale problems. Support vector machine (SVM) [1] has been widely used in a wide variety of problems and is a candidate tool for solving large-scale classification problems. However, the essence of training SVMs is solving a quadratic convex optimization problem whose time complexity is $O(N^3)$, here N is the number of training samples.

In the literature of machine learning, the divide-and-conquer principle is always used to handle large-scale problems and can be implemented in series or in parallel. In sequential learning approach, a large-scale problem is divided into many smaller subproblems that are learned sequently. These approaches include the advanced working set algorithms, which use only a subset of the variables as a working set while freezing the others [2], [3], such as Chunking, SMO, SVM^{light}, and LibSVM. The shortcoming of these approaches is that a large number of iterations are needed and this will lead to memory thrashing when training data set is large. In parallel learning approach, a large-scale problem is divided into many smaller subproblems that are parallelly handled by many modules. After training, all the trained modules are integrated into a modular system [4], [5], [6].

* This work was supported by the National Natural Science Foundation of China under the grants NSFC 60375022 and NSFC 60473040. This work was also supported in part by Open Fund of Grid Computing Center, Shanghai Jiao Tong University.

J. Wang, X. Liao, and Z. Yi (Eds.): ISNN 2005, LNCS 3496, pp. 881–886, 2005.

This kind of method has two main advantages over traditional SVMs approaches. 1) It can dramatically reduce training time. 2) It has good scalability. However, these methods will lead to increasing of the number of support vectors and often decrease generalization accuracy slightly.

Schölkopf [7] and Syed[8] have pointed out that support vectors (SVs) summarize classification information of training data. Based on their work and our previous work [9], we propose a novel hierarchical and parallel method for training SVMs. In our method, we first divide the training data of each class into $K(>1)$ subsets, and construct K^2 classification subproblems. We train SVMs parallelly on these K^2 classification subproblems and union their support vectors to construct another K classification subproblems. After that, we train SVMs parallelly on these K subproblems and take union of all their support vectors to construct the last subproblem. At last, we train a SVM as the final classifier. All the experiments illustrate that our method can speed up training and reduce the number of support vectors while maintaining the generalization accuracy. The paper is organized as follows: Section 2 will introduce our hierarchical and parallel training method. The experiments and discussions are presented in Section 3. Finally, Section 4 is conclusions.

2 Hierarchical and Parallel Training Method

Given positive training data set $\mathcal{X}^+ = \{(X_i, +1)\}_{i=1}^{N^+}$ and negative training data set $\mathcal{X}^- = \{(X_i, -1)\}_{i=1}^{N^-}$ for a two-class classification problem, where X_i denotes the ith training sample, and N^+ and N^- denote the number of positive training samples and negative training samples, respectively. The entire training data set can be defined as $\mathcal{S} = \mathcal{X}^+ \bigcup \mathcal{X}^-$. In order to train a SVM on S to classify future samples, our method includes three steps:

In the first step, we divide $\mathcal{X}^+$ and $\mathcal{X}^-$ into K roughly equal subsets respectively, according to a given partition value K,

$$\begin{aligned} \mathcal{X}^+ &= \bigcup_{i=1}^{K} \mathcal{X}_i^+, \quad \mathcal{X}_i^+ = \{(X_j, +1)\}_{j=1}^{N_i^+}, \quad i = 1, 2, ..., K \\ \mathcal{X}^- &= \bigcup_{i=1}^{K} \mathcal{X}_i^-, \quad \mathcal{X}_i^- = \{(X_j, -1)\}_{j=1}^{N_i^-}, \quad i = 1, 2, ..., K \end{aligned} \tag{1}$$

where: $N_i^+ = \lfloor N^+/K \rfloor, i = 1, 2, ..., K-1; N_K^+ = N^+ - \sum_{i=1}^{K-1} N_i^+$; $N_i^- = \lfloor N^-/K \rfloor, i = 1, 2, ..., K-1; and\ N_K^- = N^- - \sum_{i=1}^{K-1} N_i^-$.

According to (1), the original classification problem is divided into K^2 smaller classification subproblems as follows,

$$\mathcal{S}_{i,j}^1 = \mathcal{X}_i^+ \bigcup \mathcal{X}_j^-, \quad 1 \le i, j \le K \tag{2}$$

Because these K^2 smaller subproblems need not to communicate with each other in learning phase, they can be handled simultaneously by traditional method like SVM^{light} and K^2 sets of support vectors, $\mathcal{SV}_{i,j}^1 (1 \le i, j \le K)$, are obtained.

In the second step, taking union of each of the K sets of support vectors to construct the following K classification problems,

$$\mathcal{S}_i^2 = \bigcup_{j=1}^{K} \mathcal{SV}_{j,permu(j+i-1)}^1, \quad 1 \le i \le K \tag{3}$$

where $permu(n)$ is defined as:

$$permu(n) = \begin{cases} n-K, & \text{for } n > K \\ n, & \text{otherwise} \end{cases} \tag{4}$$

The approach that takes K unions from $\mathcal{SV}_{i,j}^1 (1 \le i, j \le K)$ can be defined as "cross-combination", which can ensure all classification information are maintained. All the subproblems $\mathcal{S}_i^2 (1 \le i \le K)$ are handled parallelly and K support vector sets $\mathcal{SV}_i^2 (1 \le i \le K)$ will be gotten.

At the last step, we take a union of $\mathcal{SV}_i^2 (1 \le i \le K)$, i.e. $\mathcal{S}_{final} = \bigcup_{i=1}^{K} \mathcal{SV}_i^2$. We train a SVM on $\mathcal{S}_{final}$ to get the final classifier. The procedure of our method for constructing a full-K-tree is depicted in Fig. 1.

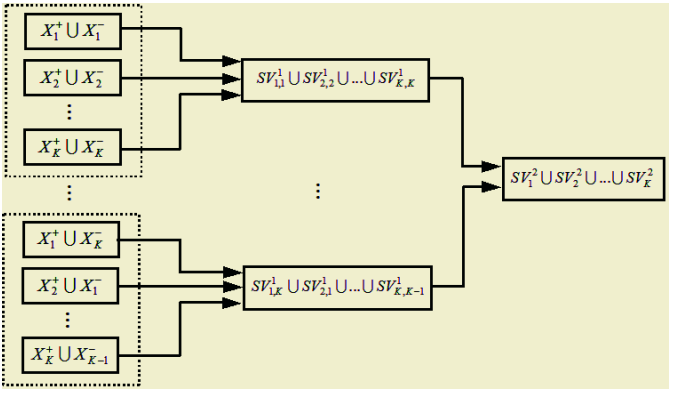

Fig. 1. Illustration of our hierarchical and parallel training method

3 Experiments

3.1 Data Sets and Simulation Environment

In order to validate our method systematically, we perform two experiments on UCI data sets [10]. The first data set is Forest coverType and the second one is Space shuttle. In this paper, multi-class classification problems are handled by one-against-one technique, i.e. a M-class classification problems is divided into $M(M-1)/2$ two-class classification subproblems by combining every two classes. The total training time is the sum of all the training time of two-class classification subproblems. Given a test instance, all the trained $M(M-1)/2$ sub-classifiers vote for the final classification. We take SVM^{light} [3] as sequential

Table 1. The problem statistics and the selections of parameters of SVMs

Problems	#attributes	#class	#training data	#test data	c	σ
Forest coverType	54	7	290504	290508	128	0.25
Space shuttle	9	5	43483	14494	1000	50

training method for its friendly interface and integrate it into our hierarchical and parallel training method. The kernel used is the radial-basis function: $\exp(-\frac{1}{2\sigma^2}\|X - X_i\|^2)$. For Forest coverType data, we take one half of it as training data and the rest one half as test data. These training and test data are normalized in the range [0, 1]. In this experiment, The sequential training is performed on a PC that has 3.0GHz CPU with 1GB RAM, while the hierarchical and parallel training is performed on a cluster IBM e1350, which has one management node and eight computation nodes. The management node has two 2.0GHz CPUs with 2GB RAM and the computation node has two 2.0GHz CPUs with 1.5GB RAM. So, the sequential training has an advantage on speed over our method. We implement our method by using MPI. Because the limitation of computing resource, the partition K is only took from 1 to 4. Here, $K = 1$ means that we take the sequential training on the entire training data.

In order to explore the performance of our method thoroughly, we take the values of K as 1,2,...,30 in Space shuttle experiment. Because of the resource limitation of the cluster IBM e1350, we simulate our hierarchical and parallel training method on a PC, i.e. we execute our hierarchical and parallel training in a sequential mode, but we count the training time in parallel mode. In this experiment, the original training and test data are used, but we exclude the samples of the sixth and seventh classes in training data and test data, because they can not be parted when $K > 6$. All the classification problems statics and the selection of the parameters of SVMs are showed in Table. 1.

3.2 Experimental Results and Discussions

From Table. 2, we can see that even though the partition K takes different values, the accuracy of our method is almost the same as the accuracy of the sequential method. Furthermore, our method can significantly reduce not only the training time but also the number of support vectors. From Table. 2, we see that the largest speedup is 5.05 when $K = 4$, and the number of support vectors is reduced 5.7% at $K = 3$. Fig. 2 shows that both the training time and the number of support vectors of all the two-class subproblems consistently decrease by using our hierarchical and parallel training method. From Table. 3, we can see that the accuracy of the classifiers with $K > 1$ is almost the same as the accuracy of the classifier with $K = 1$. From Fig. 3, we can also see that the largest speedup of our method is larger than 10, and the largest reduction of the number support vectors is 3%.

The reason of reducing training time in our method lies in the fact that a large number of non-support vectors are filtered out in the first two steps but a training instance maybe used again and again in the sequential method such

as SVM^{light}. The reason of reducing the number of support vectors is that the training data partition lead to simpler classifiers for subproblems. As a result, the split of the training data takes an affect like editing training samples [11]. When K increases, it can be expected that the number of support vectors will increase. Fig. 3 shows an increment trend of support vectors. In the worst case, no training instances are filtered out in the first step and $S_i^2 (1 \leq i \leq K)$ will be the same as S, so the number of support vectors in our method will be equal to the number of support vector in sequential method. For many large-scale classification problems, the number of support vectors always take a small proportion of the entire training data. Consequently, our method will take higher performance than sequential training method.

Table 2. The number of SVs and training time in Forest coverType problem with different values of K

Partitions	Accuracy	#SVs	Training time(s)	Speedup
$K = 1$	0.936721	47146	13037	1
$K = 2$	0.936153	44455	5281	2.47
$K = 3$	0.936105	44300	3275	3.98
$K = 4$	0.936752	44365	2582	5.05

Table 3. The accuracy variation in Space shuttle problem with different partition K

$K = 1$	$K = 2, 3, 4, ..., 30$	
	Mean	Variance
0.99897	0.99894	0.00036

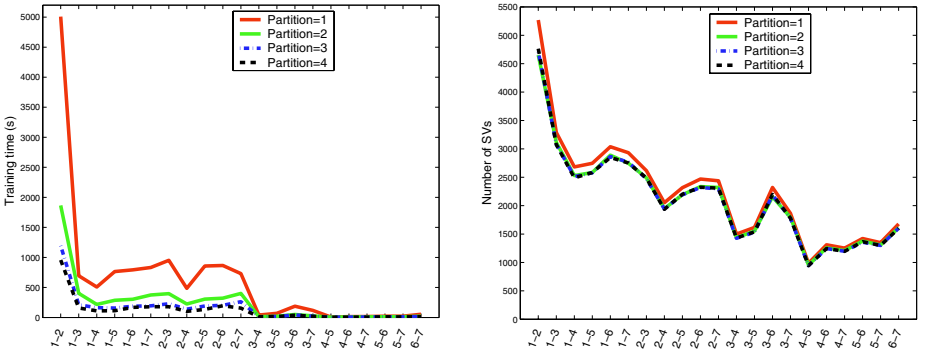

Fig. 2. Training time and the number of SVs for every two-class classification subproblems in Forest coverType classification. The digits from 1 to 7 below the horizontal axes denote the seven classes in the Forest coverType data, one pair of two digits means a two-class classification subproblem

4 Conclusions

In this paper we have presented a hierarchical and parallel methods for training SVMs. Several experimental results indicate that the proposed method has two attractive features. The first one is that it can reduce training time while keeping the generalization accuracy of the classifier. The second one is that the number of support vectors generated by our method is smaller than that of the SVMs

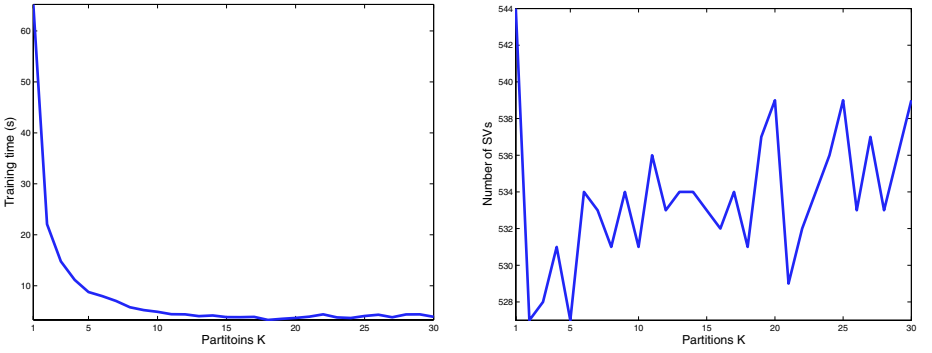

Fig. 3. Training time and the number of SVs in Space shuttle problem

trained by the traditional method. This advantage will reduce response time of the classifier and simplify implementation of the classifier in both software and hardware. We believe that our method might provide us with a promising approach to deal with large-scale pattern classification problems.

References

1. Vapnik, V.N.: Statistical Learning Theory. Wiley Interscience (1998)
2. Osuna, E., Freund, R., Girosi, F.: An Improved Training Algorithm for Support Vector Machines. In: Proceedings of IEEE NNSP'97 (1997) 276-285
3. Joachims, T.: Making Large-scale Support Vector Machine Learning Pratical. In: Schölkopf, B., Burges, C.J., Smola, A.J.(eds.), Advances in Kernel Methods-Support Vector Learning. MIT Press (2000) 169-184
4. Lu, B.L., Ito, M.: Task Decomposition and Module Combination Based on Class Relations: A Modular Neural Network for Pattern Classification. IEEE Transaction on Neural Networks, **10** (1999) 1244-1256
5. Lu, B.L., Wang, K.A., Utiyama, M., Isahara, H.: A Part-versus-part Method for Massively Parallel Training of Support Vector Machines. In: Proceedings of IJCNN'04. Budapest, Hungary (2004) 735-740
6. Schwaighofer, A., Tresp, V.: The Bayesian Committee Support Vector Machine. In: Dorffner, G., Bischof, H., and Hornik, K.(eds.), Proceedings of ICANN 2001. Lecture Notes in Computer Science, Springer Verlag, **2130** (2001) 411-417
7. Schölkopf, B., Burges, C., and Vapnik, V.N.: Extracting Support Data for a Given Task. In: Proceedings of the First International Conference on Knowledge Discovery & Data Mining. Menlo Park, CA (1995) 252-257
8. Syed, N.A., Liu, H., Sung, K.K.: Incremental Learning with Support Vector Machines. In: Proceedings of the Workshop on Support Vector Machines at the International Joint Conference on Artificial Intelligence. Stockholm, Sweden (1999)
9. Wen, Y.M. and Lu, B.L.: A Cascade Method for Reducing Training Time and the Number of Support Vectors. In: Proceedings of ISNN2004. Dalian, China. Lecture Notes in Computer Science, Springer Verlag, **3173** (2004) 480-485
10. Blake, C.L., and Merz, C. J.: UCI. In: ftp://ftp.ics.uci.edu/pub/machine-learning-databases (1998)
11. Ke, H.X. and Zhang, X.G.: Editing Support Vector Machines. In: Proceedings of IJCNN'01. Washington, USA (2001) 1464-1467