A Cascade Method for Reducing Training Time and the Number of Support Vectors

Yi-Min Wen[1,2] and Bao-Liang Lu[1]

[1] Department of Computer Science and Engineering, Shanghai Jiao Tong University, 1954 Hua Shan Rd., Shanghai 200030, China
wenyimin@sjtu.edu.cn, blu@cs.sjtu.edu.cn
[2] Hunan Industry Polytechnic, Changsha 400052, China

Abstract. A novel cascade learning strategy for training support vector machines (SVMs) is proposed to speed up the training of SVMs. The training procedure consists of three steps which are performed in a cascade way. All the subproblems are processed parallelly in each step, and non-support-vector data are filtered out step by step. The simulation results indicate that our method not only speeds up the training procedure while maintaining the generalization accuracy of SVMs but also reduces the number of support vectors.

1 Introduction

In many real-world problems such as text categorization and geography information classification, the sizes of training data sets are usually massive. For example, the Yomiuri News Corpus contains 2,190,512 documents dated 1987-2001. It is necessary to develop efficient methods to deal with these real-world large-scale problems.

Support vector machines (SVMs) [1] have become a popular tool of machine learning. There are two kinds of methods for solving large-scale pattern classification problems. The first method is the incremental learning approach, in which a large-scale problem is divided into many small subproblems that are learned sequently [2]. This approach includes the advanced working set algorithms that use only a subset of the variables as a working set while freezing the others [3], [4]. The shortcoming of this kind of method is that a large number of iterations are required. Therefore, if the training data set is large, the training time will be very long. The second method is the parallel learning method. The basic idea behind this method is to divide a large-scale problem into many subproblems and to parallelly learn these subproblems by many modules. After training, all the trained modules are integrated into a modular system [5], [6], [7]. This kind of method has two main advantages over the existing SVM approaches. 1) It can dramatically reduce training time. 2) It has good scalability and expansibility. However, this method will lead to increasing the number of support vectors.

Based on the essence of support vector (SV) [8], we proposed a novel cascade method for training SVMs for pattern classification. Our method not only speeds up training but also reduces the number of support vectors.

F. Yin, J. Wang, and C. Guo (Eds.): ISNN 2004, LNCS 3173, pp. 480–486, 2004.

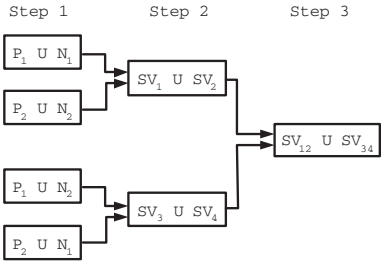

Fig. 1. Illustration of the cascade training method for producing a new smaller training set, $SV_{12} \cup SV_{34}$

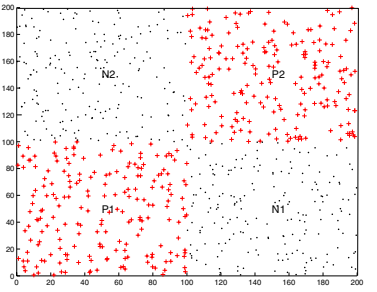

Fig. 2. Combined scatter plot of both classes in checkerboard problem. Here, N_1, N_2, P_1 and P_2 denote four subsets

The paper is organized as follows: Section 2 will introduce our cascade method. The experiments are conducted in Section 3. Finally, Section 4 is conclusions.

2 A Cascade Training Method

Because the multi-class classification problem can be transformed into a series of two-class classification problems [5], we only consider two-class classification problems here. We assume that class C_1 includes positive samples and class C_2 includes negative samples. Let $P = \{X_i\}_{i=1}^{L_p}$ be the training set of class C_1 and $N = \{Y_i\}_{i=1}^{L_n}$ be the training set of class C_2. Here, L_p and L_n denote the number of elements in class C_1 and class C_2, respectively. Therefore, the training data set for a two-class problem is given by $T = P \cup N$. The proposed cascade training method has three main steps as illustrated in Fig. 1.

In the first step, the original data sets, P and N, are divided into two subsets by the same ratio r $(0 < r \leq 0.5)$, respectively, as follows:

$$P_1 = \{X_i\}_{i=1}^{L_{p_1}}, P_2 = \{X_i\}_{i=L_{p_1}+1}^{L_p}, N_1 = \{Y_i\}_{i=1}^{L_{n_1}}, \text{ and } N_2 = \{Y_i\}_{i=L_{n_1}+1}^{L_n} \quad (1)$$

where $L_{p_1} = \lceil r * L_p \rceil$ and $L_{n_1} = \lceil r * L_n \rceil$.

According to this decomposition, the original two-class problem T is divided into four two-class subproblems as follows:

$$T_1 = P_1 \cup N_1, \; T_2 = P_2 \cup N_2, \; T_3 = P_1 \cup N_2, \text{ and } T_4 = P_2 \cup N_1 \quad (2)$$

An important feature of these subproblems is that there are no common training data between T_1 and T_2, and also between T_3 and T_4. Therefore, these subproblems can be handled by conventional SVM method [4] in a completely parallel way. After training, we can obtain four sets of support vectors, SV_1, SV_2, SV_3, and SV_4, which correspond to T_1, T_2, T_3, and T_4 respectively.

In the second step, we construct two training data sets from SV_1, SV_2, SV_3, and SV_4 as follows:

$$T_{12} = SV_1 \cup SV_2 \; and \; T_{34} = SV_3 \cup SV_4 \tag{3}$$

It should be noted that SV_1 and SV_2 are disjunctive each other and SV_3 and SV_4 are disjunctive each other too. After handling these two subproblems on T_{12} and T_{34} parallelly by conventional method [4], we obtain two new sets of support vectors, SV_{12} and SV_{34}. These two sets might include some common support vectors because there might be some common training data between T_{12} and T_{34}. From SV_{12} and SV_{34}, we obtain a new smaller training data set as follows:

$$T_{final} = SV_{12} \cup SV_{34} \tag{4}$$

Finally, we train a SVM on T_{final} by conventional SVM method [4]. The final SVM will be used in recognition phase and all the trained SVMs obtained in the first and the second steps are discarded and the occupied computing resources are released.

If the subproblems in the first step is too large to solve, we can recursively divide it into four subproblems according to the method mentioned above. In addition, we can extend our method by dividing each class into k (> 2) subsets and we will discuss this problem later on.

3 Experiments

In order to verify our method, we present three experiments. The first is a artificial problem and the other two are real world problems. We take the tool SVM^{light} [4] for its friendly interface. In all the experiments, "standard method" means that SVMs are trained with the whole training data together by conventional SVM method [4]. The kernel we used is the radial-basis function. All the experiments were performed on a 2.4GHz Pentium 4 PC with 512MB RAM.

3.1 The Checkerboard Experiment

A 2D checkerboard problem is depicted in Fig. 2. The checkerboard divides a 200×200 square into four quadrants. All the points are uniformly distributed in the square. The points labelled by plus are positive samples and the points labelled by dot are negative samples. In this experiment, we randomly generate four training data sets, each of which includes 5000 positive samples and 5000 negative samples. A common test data set, which includes 10000 positive samples and 10000 negative samples, is also generated. Therefore, we get four classification problems, A_1, A_2, A_3, and A_4. Every training data set is divided into four parts, P_1 and P_2, N_1, and N_2.

In this experiment, r is fixed on 0.5. The purpose is to see whether the generalization performance of the SVMs trained by our cascade method is robust

Table 1. Four runnings over four training data sets and one common test data set in the checkerboard experiment, where $\sigma = 31.62$, $c = 1000$, and $r = 0.5$. The row "Averaging" denotes the average results of the four problems, $A_1, A_2, A_3, and\ A_4$

Problem	Method	Accuracy		Training	Number
		Training (%)	Test (%)	time(s)	of SV
A_1	Standard method	99.84	99.81	46.39	93
	Cascade method	99.78	99.72	13.08	81
A_2	Standard method	99.89	99.72	38.00	96
	Cascade method	99.85	99.70	15.34	83
A_3	Standard method	99.93	99.84	32.44	88
	Cascade method	99.86	99.75	13.45	79
A_4	Standard method	99.89	99.81	35.50	94
	Cascade method	99.92	99.83	19.87	84
Averaging	Standard method	99.89	99.80	38.08	93
	Cascade method	99.85	99.75	15.44	82

when the training data set varies slightly. From Table 1, we can see that the generalization performance obtained by our method is almost unchanged with the different training data. The last row titled "Averaging" in Table 1 indicates that the generalization accuracy of the SVMs trained by the standard method and the SVMs trained by our method are almost the same, while our method is faster than the standard method and reduces the number of support vectors.

3.2 The Forest Covertype Experiment

In the second experiment, we use the Forest Covertype data set from UCI [9]. This data set is designed to predict the forest cover types in undisturbed forests. The attributes about one instance is 54. In this experiment, we only take all the instances of the sixth and the seventh classes to construct a two-class classification problem A_5 and all the instances of the fourth and the fifth classes to construct another two-class classification problem A_6. In each data set, we randomly take one half of instances for training and the other half for test. The data sets are shown in Table 2.

In this experiment, The training data set are decomposed by r which takes the values of 0.1 and 0.5, respectively. According to Tables 4 and 5, we can see that r does not influence classification accuracy and it only affects the training time. From these two tables, we can conclude that $r = 0.5$ is suitable for our method. Table 6 shows the details of the experiment on problem A_5. We can see that the no-support-vector data are filtered out step by step. If the number of support vectors take only a small proportion of the whole training data set, our cascade method will be much faster than the standard method. From the simulation results shown in Table 4 and 5, we can see that our cascade method can speed up training and reduce the number of support vectors.

3.3 Text Categorization

In the third experiment, we use the Yomiuri News Corpus. Here, we select the data of three classes as shown in Table 3 to construct three two-class classification problems, A_7, A_8, and A_9, by choosing every two classes from these three classes in Table 3. The number of features is 5,000 and r is set to 0.5. From Table 7,we can see that our cascade method outperforms the standard method in training time and reduce the number of support vectors. Because the number of support vectors might take only a small proportion of the whole training data in text categorization, we believe the proposed method is appropriate to dealing with text categorization problems.

Table 2. Distributions of the training and test data used in the second experiment

Problem	Training		Test	
	Positive samples	Negative samples	Positive samples	Negative samples
A_5	8684	10255	8683	10255
A_6	1374	4747	1373	4746

Table 3. The training data and test data used in the third experiment

Category	Data	
	Training	Test
Accidents	34044	8483
Health	35932	7004
By-time	33590	7702

Table 4. Experiment results on problem A_5, where $\sigma = 100$ and $c = 1000$

	Standard method	Cascade method	
		$r = 0.1$	$r = 0.5$
Training time	461.92s	508.04s	268.20s
Number of SV	3943	3827	3778
Training (%)	100	100	100
Test (%)	99.82	99.82	99.82

Table 5. Experiment results on problem A_6, where $\sigma = 100$ and $c = 1000$

	Standard method	Cascade method	
		$r = 0.1$	$r = 0.5$
Training time	27.02s	36.66s	24.69s
Number of SV	1661	1605	1571
Training (%)	100	100	100
Test (%)	96.93	96.93	96.93

4 Conclusions

In this paper we have presented a cascade method for training SVMs. Several experimental results indicate that the proposed method has two attractive features: the first is that it can speed up training while maintaining the generalization accuracy. The second is that the number of support vectors generated by our cascade method is smaller than that of the SVMs trained by the standard method, and this will reduce the time for recognition and simplify the design of classifiers. We believe the proposed method might provide us with a promising approach to deal with large-scale pattern classification problems, such as biological data mining and geography information classification.

Table 6. Details of simulation on problem A_5, where $\sigma = 100$, $c = 1000$, and $r = 0.5$

Step 1	Number of training samples	9470	9469	9470	9469
	Number of SV	3121	1630	2453	2291
	Training time (s)	131.61	56.03	82.95	84.5
Step 2	Number of training samples	4751		4744	
	Number of SV	3772		3770	
	Training time (s)	91.67		91.42	
Step 3	Number of training samples	4276			
	Number of SV	3778			
	Training time (s)	44.92			

Table 7. Simulation results on text categorization, where $\sigma = 2$, $c = 64$, and $r = 0.5$

	Methods	A_7	A_8	A_9
Training accuracy (%)	Standard method	97.74	97.93	96.67
	Cascade method	97.73	97.75	96.67
Test accuracy (%)	Standard method	95.81	96.01	93.62
	Cascade method	95.83	96.02	93.62
Training time (s)	Standard method	12664	7458	18566
	Cascade method	9519	4491	15060
Number of SV	Standard method	10933	9445	12750
	Cascade method	10553	9222	12387

Acknowledgments. This work was supported in part by the National Natural Science Foundation of China via the grant NSFC 60375022. The authors thank Kai-An Wang and Hong Shen for the help on preparing the text categorization data.

References

1. Vapnik, V.N.: Statistical Learning Theory. Wiley Interscience (1998)
2. 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)
3. Osuna, E., Freund, R., Girosi, F.: An Improved Training Algorithm for Support Vector Machines. In: Proceedings of IEEE NNSP'97 (1997) 276-285
4. 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
5. Lu, B.L. and Ito, M.: Task Decomposition and Module Combination Based on Class Relations: A Modular Neural Network for Pattern Classification. IEEE Transaction on Neural Networks, Vol.10 (1999) 1244-1256
6. 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. (to appear) Budapest, July 25-29 (2004)

7. Tresp, V.: Scaling Kernel-Based Systems to Large Data Sets. Data Mining and Knowledge Discovery, Vol.5 (2001)
8. Syed, N.A., Liu, H., Sung, K.K.: Handling Concept Drifts in Incremental Learning with Support Vector Machines. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Diego, CA, USA (1999) 317-321
9. Blake, C.L., and Merz, C. J.: UCI. In: ftp://ftp.ics.uci.edu/pub/machine-learning-databases (1998)