Task Decomposition Using Geometric Relation for Min-Max Modular SVMs*

Kaian Wang, Hai Zhao, and Baoliang Lu

Department of Computer Science and Engineering, Shanghai Jiao Tong University,
1954 Hua Shan Rd., Shanghai 200030, China
blu@cs.sjtu.edu.cn

Abstract. The min-max modular support vector machine (M^3-SVM) was proposed for dealing with large-scale pattern classification problems. M^3-SVM divides training data to several sub-sets, and combine them to a series of independent sub-problems, which can be learned in a parallel way. In this paper, we explore the use of the geometric relation among training data in task decomposition. The experimental results show that the proposed task decomposition method leads to faster training and better generalization accuracy than random task decomposition and traditional SVMs.

1 Introduction

Support vector machines (SVMs)[1] have been successfully applied to various pattern classification problems, such as handwritten digit recognition, text categorization and face detection, due to their powerful learning ability and good generalization performance. However, SVMs require to solve a quadratic optimization problem and cost training time that are at least quadratic to the number of training samples. Therefore, to learn a large-scale problem by using traditional SVMs is a hard task.

In our previous work, we have proposed a part-versus-part task decomposition method[2, 3] and developed a new modular SVM for solving large-scale pattern classification problems[4], which called min-max modular support vector machines (M^3-SVMs). Two main advantage of M^3-SVMs over traditional SVM is that massively parallel training of SVMs can be easily implemented in cluster systems or grid computing systems, and large-scale pattern classification problems can be solved efficiently.

In this paper, we explore the use of the geometric relation among training data in task decomposition, and try to investigate the influence of different task decomposition methods to the generalization performance and training time.

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

J. Wang, X. Liao, and Z. Yi (Eds.): ISNN 2005, LNCS 3496, pp. 887–892, 2005.

2 Min-Max Modular Support Vector Machine

Let $\mathcal{X}^+$ and $\mathcal{X}^-$ be the given positive and negative training data set for a two-class problem $\mathcal{T}$,

$$\mathcal{X}^+ = \{(x_i^+, +1)\}_{i=1}^{l^+}, \quad \mathcal{X}^- = \{(x_i^-, -1)\}_{i=1}^{l^-} \tag{1}$$

where $x_i \in \mathbf{R}^n$ is the input vector, and l^+ and l^- are the total number of positive training data and negative training data of the two-class problem, respectively.

According to [4], $\mathcal{X}^+$ and $\mathcal{X}^-$ can be partitioned into N^+ and N^- subsets respectively,

$$\mathcal{X}_j^+ = \{(x_i^{+j}, +1)\}_{i=1}^{l_j^+}, \quad for\, j = 1, \ldots, N^+ \tag{2}$$

$$\mathcal{X}_j^- = \{(x_i^{-j}, -1)\}_{i=1}^{l_j^-}, \quad for\, j = 1, \ldots, N^- \tag{3}$$

where $\cup_{j=1}^{N^+} \mathcal{X}_j^+ = \mathcal{X}^+$, $1 \le N^+ \le l^+$, and $\cup_{j=1}^{N^-} \mathcal{X}_j^- = \mathcal{X}^-$,$1 \le N^- \le l^-$.

After decomposing the training data sets $\mathcal{X}^+$ and $\mathcal{X}^-$, the original two-class problem $\mathcal{T}$ is divided into $N^+ \times N^-$relatively smaller and more balanced two-class sub-problems $\mathcal{T}^{(i,j)}$ as follows:

$$(\mathcal{T}^{(i,j)})^+ = \mathcal{X}_i^+, \quad (\mathcal{T}^{(i,j)})^- = \mathcal{X}_j^- \tag{4}$$

where $(\mathcal{T}^{(i,j)})^+$ and $(\mathcal{T}^{(i,j)})^-$ denote the positive training data set and the negative training data set of subproblem $\mathcal{T}^{(i,j)}$ respectively.

In the learning phase, all the two-class sub-problems are independent from each other and can be efficiently learned in a massively parallel way.

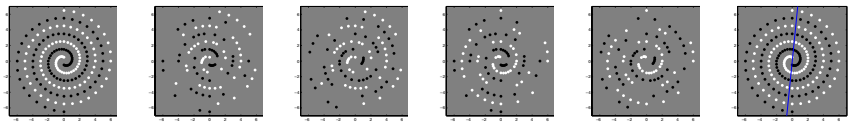
After training, the $N^+ \times N^-$ smaller SVMs are integrated into a M^3-SVM with N^+ MIN units and one MAX unit according to two combination principles [3, 4] as follows,

$$\mathcal{T}^i(x) = \min_{j=1}^{N^-} \mathcal{T}^{(i,j)}(x) \;\text{ for } i = 1, \ldots, N^+ \text{ and } \mathcal{T}(x) = \max_{i=1}^{N^+} \mathcal{T}^i(x) \tag{5}$$

where $\mathcal{T}^{(i,j)}(x)$ denotes the transfer function of the trained SVM corresponding to the two-class subproblem $\mathcal{T}^{(i,j)}$, and $\mathcal{T}^i(x)$ denotes the transfer function of a combination of N^- SVMs integrated by the MIN unit.

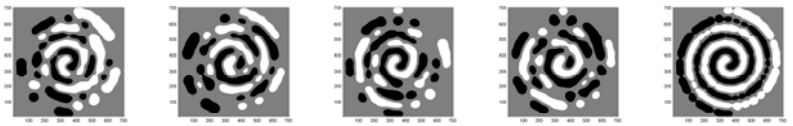
3 Task Decomposition Using Geometric Relation

M^3-SVM needs to divide training data set into several sub-sets in the first step. How to divide training data set effectively is an issue which needs to investigate furthermore. Although dividing training data set randomly is a simple and straightforward approach, the geometric relation among the original training data may be damaged[3]. The data belonging to a cluster may be separated into different sub-sets, and the data in center of a cluster may be moved to the



(a) (b) (c) (d) (e) (f)

Fig. 1. The two-spirals problem and related sub-problems



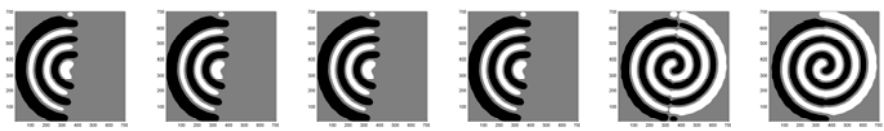
(a) (b) (c) (d) (e)

Fig. 2. The response of SVMs and M^3-SVM with random task decomposition method

boundary for the new sub-problems. From SVM's point of view, some data will become support vectors, so the boundary which is vital to SVM will be changed.

A two dimensional toy problem is depicted in Fig.1(a). The training data set of each class is randomly divided into two subsets. Four sub-problems shown in Figs.1(b) through 1(e) are generated by combining these two subsets. The response of SVMs (Rbf kernel with $\sigma = 0.5, C = 1$)corresponding to each of the four sub-problems are shown in Figs.2(a) through 2(d), and the response of the M^3-SVM is shown in Fig. 2(e). We can see that the final decision boundary is not very smooth.

On the other hand, the training data set of each class is divided along the y-axis as shown in Fig.1(f). The response of SVMs corresponding to each of the four sub-problems are shown in Figs.3(a) through 3(d), respectively. The benefit of task decomposition using geometric relation is quite obvious. The function of each SVM is quite clear. The SVMs in Fig.3(b) and Fig.3(c) determine which part of the space the test data x belongs to. Training these two SVMs is very easy because the training data are linearly separable. The SVMs in Fig.3(a) and Fig.3(d) judge which class that test data x belongs to. The response of M^3-SVM is shown in Fig.3(e), and is smoother than that in Fig.2(e). If the training data near to the y-axis are included in both the two subsets of each class, a smoother decision boundary will be obtained as shown in Fig.3(f).



(a) (b) (c) (d) (e) (f)

Fig. 3. The response of SVMs and M^3-SVM by using space-grid

For more complicated high-dimensional problems, we divide the input space using the hyperplanes which are parallel with $z_1 + z_2, \ldots, +z_n = 0$. We never need to construct hyperplane explicitly by using a trick, so we still keep the advantage of M^3-SVM that don't need any prior or domain knowledge. Suppose we divide the training data set of class C_i to N_i subsets. Firstly, We compute the distance between each training sample x of class C_i and hyperplane H: $z_1 + z_2, \ldots, +z_n = 0$ as follows,

$$dist(x, H) = \frac{1 \times x_1 + 1 \times x_2, \ldots, +1 \times x_n}{\sqrt{1^2 + 1^2, \ldots, +1^2}} = \frac{x_1 + x_2, \ldots, +x_n}{\sqrt{n}} \tag{6}$$

where x_i is the element of sample vector x. Then, we sort the training data according to the value of $dist(x, H)$, and divide the reordered sequence of training data to N_i parts equally to remain the size of sub-sets almost the same. Different from the toy problem, the hyperplanes used to divide the training data of different class is different for more general problems.

4 Experiments

In this section, we present experimental results on the Forest CoverType and banana data sets from UCI[5] to compare M^3-SVMs with traditional SVMs, as well as M^3-SVMs with different task decomposition methods.

Forest CoverType data set has seven classes, including 581012 samples, and the feature dimension is 54. We firstly normalize the original data in the range [0,1], and then randomly select 60% of the total data as training data, and the remainder as test data. Banana data set is a binary problem including 40000 training data and 490000 test data. All the simulations were done on an IBM p690.

Table 1. Three ways of decomposing the Forest CoverType problem

#	Num. of subsets			#
	C_1	C_2	others	classifier
$\mathcal{A}_1$	2	2	1	27
$\mathcal{A}_2$	3	2	1	41
$\mathcal{A}_3$	4	3	1	57

Table 2. Three ways of decomposing the Banana problem

#	Num. of subsets		#
	C_1	C_2	classifier
$\mathcal{B}_1$	2	1	2
$\mathcal{B}_2$	2	2	4
$\mathcal{B}_3$	3	2	6

We perform seven different experiments on the two data sets, respectively. The first one uses traditional SVM, the next three use M^3-SVM with random task decomposition method, called M^3-SVM(R), and the last three employe M^3-SVM with hyperplane task decomposition, called M^3-SVM(H). The original problems are decomposed according to the parameters shown in Table 1 and Table 2.

Table 3 presents simulation results on Forest CoverType problem. Although M^3-SVM(R) may sacrifice a little generalization accuracy, it reduces the training time in both serial and parallel ways. From Table 3, we can see M^3-SVM(H) can

Table 3. Results on Forest CoverType problem, where $\sigma = 0.25$ and $C = 128$

Method	# classifier	CPU Time(h.) parallel	CPU Time(h.) serial	Speed up parallel	Speed up serial	Correct rate(%)	#SV
SVM	21	122.88	133.10	-	-	93.04	82719
M^3-SVM(R)	27	61.29	128.12	2.00	1.04	92.57	98359
	41	19.94	120.75	6.16	1.10	92.48	118122
	57	10.40	117.32	11.82	1.13	92.35	133965
M^3-SVM(H)	27	33.63	72.31	3.65	1.84	93.09	84272
	41	11.17	38.55	11.00	3.45	93.19	88049
	57	5.34	24.10	23.01	5.52	93.17	90579

get a better generalization performance in comparison with traditional SVMs, costs less training time, especially in a parallel way, and has less number of support vectors than M^3-SVM(R). Table 4 lists the simulation results on banana problem. From Table 4, we can see that M^3-SVM(H) is superior to traditional SVM and M^3-SVM(R) in both generalization accuracy and training time.

Table 4. Results on Banana problem, where $\sigma = 1$ and $C = 361.2$

Method	# classifier	CPU Time(s.) parallel	CPU Time(s.) serial	Speed up parallel	Speed up serial	Correct rate(%)	#SV
SVM	1	719.63	719.63	-	-	90.64	8430
M^3-SVM(R)	2	322.34	644.36	2.23	1.12	89.79	8819
	4	155.28	594.55	4.63	1.21	90.67	9225
	6	102.58	577.80	7.02	1.25	90.20	8858
M^3-SVM(H)	2	162.78	264.88	4.42	2.72	90.87	8419
	4	127.69	193.89	5.63	3.71	90.76	8404
	6	50.77	136.17	14.17	5.28	90.67	8539

Table 5 reports the performance of six sub-problems which are obtained by dividing training data sets belonging to class C_1 and class C_2. $\mathcal{T}_{12}^{(i,j)}$ denotes a SVM corresponding to the sub-problem whose training data are from the ith sub-set of class C_1 and the jth sub-set of C_2, and $\mathcal{T}_{12}$ is combined from $\mathcal{T}_{12}^{(i,j)}$ with three MIN units and one MAX unit. From Table 5, we see that the six problems in column "Random" are very similar in correct rate, training time, and the number of SVs. However, the other six problems in the column "Hyperplane" are quite different in the aspects of correct rate, training time, and the number of SVs. We can imagine that the 1st sub-set of class C_1 and the 2nd sub-set of class C_2 are located in different parts of the input space, and therefore it is easy to train $\mathcal{T}_{12}^{(1,2)}$, and have a small number of SVs. Although the correct rate of each problem is quite low, the correct rate of whole $\mathcal{T}_{12}$ is high. This phenomenon can be explained as follows: each SVM in the "Hyperplane" column is responsible

for one part of the input space and cooperates with each other to predict the whole input space.

Table 5. Subproblems of C_{12} of Forest CoverType problem

	#data		Random			Hyperplane		
Task	#pos.	#neg.	rate(%)	#SV	time(h.)	rate(%)	#SV	time(h.)
$\mathcal{T}_{12}^{(1,1)}$	56660	63552	92.04	27027	19.00	67.04	19519	10.93
$\mathcal{T}_{12}^{(1,2)}$	56660	63552	92.03	26704	18.32	56.42	2522	0.13
$\mathcal{T}_{12}^{(2,1)}$	56660	63552	92.08	27163	19.03	66.81	14946	7.56
$\mathcal{T}_{12}^{(2,2)}$	56660	63552	92.18	26977	18.52	69.35	9557	3.72
$\mathcal{T}_{12}^{(3,1)}$	56660	63552	92.27	27406	19.34	49.47	2380	0.11
$\mathcal{T}_{12}^{(3,2)}$	56660	63552	92.29	27214	19.94	64.36	18632	11.17
$\mathcal{T}_{12}$	169980	127104	93.49	83996	114.15	94.19	61458	33.62

5 Conclusions

We have proposed a new task decomposition method using geometric relations among training data for M^3-SVM. We also have compared our method with random decomposition approach. From experiments results, we can draw the following conclusions. a) Although M^3-SVM have a few more number of support vectors, training M^3-SVM is faster than traditional SVM in both serial and parallel way. b) The proposed task decomposition method improves the performance of M^3-SVM in the aspects of training time, the number of support vectors, and generalization accuracy. It's worth noting that M^3-SVM with our proposed task decomposition method is superior to traditional SVM in both training time and generalization performance. As a future work, we will analyze the effectiveness of the decomposition method based on geometric relation theoretically.

References

1. Cortes, C., Vapinik, V.N.: Support-vector Network. Machine Learning, **20** (1995) 273-297
2. Lu, B.L., Ito, M.: Task Decomposition Based on Class Relations: a Modular Neural Network Architecture for Pattern Classification. In: Mira, J., Moreno-Diaz, R., Cabestany, J.(eds.), Biological and Artificial Computation: From Neuroscience to Technology, Lecture Notes in Computer Science. Vol. 1240. Springer (1997) 330-339
3. Lu, B.L., Ito, M.,: Task Decomposition and Module Combination Based on Class Relations: a Modular Neural Network for Pattern Classification. IEEE Transactions on Neural Networks, **10** (1999) 1244 -1256
4. 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, Budapast, **25-29** (2004) 735-740
5. Blake, C.L., Merz, C.J.: UCI Repository of Machine Learning Databases. In: ftp://ftp.ics.uci.edu/pub/machine-learning-databases(1998)