Fast Te t Categori ation with Min-Ma Modular Support ector Machines

Feng-Yao Liu, Ke Wu, Hai Zhao, and Bao-Liang Lu
Department of Computer Science and Engineering, Shanghai Jiao Tong University
1954 Hua Shan Rd., Shanghai 200030, China
Email: blu@cs.sjtu.edu.cn

***Abstract*— The min-max modular support vector machines (M^3-S Ms) have been proposed for solving large-scale and complex multiclass classification problems. In this paper, we apply the M^3-S Ms to multilabel text categrization and introduce a new task decomposition strategy into M^3-S Ms. A multilabel classification task can be split up into a set of two-class classification tasks. These two-class tasks are to discriminate the $\mathcal{C}$ class from non-$\mathcal{C}$ class. If these two class tasks are still hard to be learned, we can further divide them into a set of two-class tasks as small as needed and fast training of S Ms on massive multilabel texts can be easily implemented in a massively parallel way. Furthermore, we proposed a new task decomposition strategy called hyperplane task decomposition to improve generalization performance. The experimental results on the RC 1-v2 indicate that the new method has better generalization performance than traditional S Ms and previous M^3-S Ms using random task decomposition, and is much faster than traditional S Ms.**

I. Introduction

With the rapid growth of online information, text classification has become one of the key techniques for handling and organization of text data. Various pattern classification methods have been applied to text classification for the need. Due to their powerful learning ability and good generalization performance, Support Vector Machines (SVMs) [1][2] have been successfully applied to various pattern classification problems. Joachims (1997) [3] and Yang (1999) [4] made experiments on the same text data set respectively. Both experimental results showed that SVMs yield lower error rate than many other classification techniques, such as Naive Bayes and K-Nearest Neighbors. However, to train SVMs on large-scale problems is a time-consuming task, since their training time is at least quadratic to the number of training samples. Therefore, it is a hard work to learn a large-scale text data set using traditional SVMs.

On the other hand, Lu and Ito (1999) [5] proposed a min-max modular (M^3) network for solving large-scale and complex multiclass classification problems effortlessly and efficiently. And the network model has been applied to learning large-scale, real world multiclass problems such as part-of-speech tagging and classification of high-dimensional, single-trial electroencephalogram signals. Recently, Lu and his colleagues [6] have proposed a part-versus-part task decomposition method and a new modular SVM, called min-max modular support vector machine (M^3-SVM), which was developed for solving large-scale multiclass problems.

In this paper, we will apply M^3-SVMs to multilabel text classification and adopt several new strategies of dividing a large-scale sample data set into many small sample data sets to try to investigate the influence of different task decomposition methods on the generalization performance and training time.

This paper is structured as follows. In the section II, M^3-SVMs are introduced briefly. In the section III, several different task composition strategies are listed. Then in the section IV, we designed a set of experiments on a large-scale multilabel text classification. And the training time and the performance of text classification using SVMs and different M^3-SVMs will be compared. In Section V, conclusions are outlined.

II. Min-Max Modular Support Vector Machines

The min-max modular support vector machine [6] is a method that divides a complex classification problem into many small independent two-class classification problems and then integrates these small SVMs according to two module combination rules, namely the minimization principle and the maximization principle [5].

For a two-class problem $\mathcal{T}$, let $\mathcal{X}^+$ denote the positive training data set belonging to a particular category $\mathcal{C}$ and $\mathcal{X}$ denote the negative training data set not belonging to $\mathcal{C}$.

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

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

According to [6], $\mathcal{X}^+$ and $\mathcal{X}$ can be partitioned into N^+ and N subsets respectively,

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

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

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

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

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

0-7803-9048-2/05/$20.00 ©2005 IEEE

TABLE I
RESULTS ON CCAT, WHERE C=0.5

method		# SVMs	CPU time (s.) parallel	CPU time (s.) serial	Speed up parallel	Speed up serial	performance P	performance R	performance F_1
1		1	898	898	-	-	94.9	91.3	93.0
2		4	290	989	3.10	0.91	94.9	90.7	92.8
		9	125	934	7.2	0.96	94.9	90.2	92.5
		16	56	670	16.0	1.34	94.8	89.9	92.3
		30	32	667	28.1	1.35	93.6	90.9	92.2
3	a	4	373	839	2.41	1.07	94.4	91.6	93.0
		9	178	688	5.04	1.31	94.6	91.7	93.1
		16	84	540	10.7	1.66	94.5	91.7	93.1
		30	47	563	19.1	1.60	93.5	92.7	93.1
	b	4	379	930	2.37	0.97	94.5	91.6	93.1
		9	178	854	5.04	1.05	94.8	91.6	93.2
		16	85	621	10.56	1.45	94.6	91.6	93.1
		30	48	647	18.71	1.39	93.7	92.6	93.2
	c	4	379	978	2.37	0.92	94.6	91.5	93.0
		9	177	929	5.07	0.97	94.8	91.5	93.2
		16	85	677	10.56	1.33	94.7	91.5	93.1
		30	52	707	17.27	1.27	93.8	92.6	93.2
	d	4	374	1023	2.40	0.88	94.7	91.5	93.1
		9	179	1009	5.02	0.89	94.9	91.5	93.2
		16	87	781	10.32	1.15	94.7	91.6	93.1
		30	53	754	16.94	1.19	93.9	92.6	93.2
	e	4	426	1260	2.11	0.71	94.8	91.5	93.1
		9	214	1003	4.20	0.90	95.0	91.3	93.1
		16	110	975	8.16	0.92	94.9	91.5	93.2
		30	60	894	14.97	1.00	94.0	92.6	93.3

TABLE II
RESULTS ON ECAT, WHERE C=0.5

method		# SVMs	CPU time (s.) parallel	CPU time (s.) serial	Speed up parallel	Speed up serial	performance P	performance R	performance F_1
1		1	607	607	-	-	92.7	64.1	75.8
2		3	186	531	3.26	1.14	84.7	74.7	79.4
		7	53	347	11.45	1.75	73.8	82.5	77.9
		20	21	365	28.90	1.66	78.5	78.3	78.4
		26	16	365	37.94	1.66	74.5	81.1	77.7
3	a	3	234	461	2.59	1.32	87.9	71.0	78.6
		7	66	307	9.20	1.98	80.4	78.9	79.6
		20	34	300	17.85	2.02	82.6	77.5	80.0
		26	26	310	23.35	1.96	79.0	80.6	79.8
	b	3	233	501	2.61	1.21	88.7	70.3	78.4
		7	68	334	8.93	1.82	81.5	78.0	79.7
		20	34	343	17.85	1.77	83.4	77.2	80.2
		26	27	353	22.48	1.72	79.8	80.1	79.9
	c	3	233	519	2.61	1.17	89.1	70.0	78.4
		7	72	350	8.43	1.73	82.0	77.5	79.7
		20	36	363	16.86	1.67	83.8	76.9	80.2
		26	28	373	21.68	1.63	80.2	79.9	80.0
	d	3	254	579	2.39	1.05	89.4	69.6	78.3
		7	79	379	7.68	1.60	82.6	77.1	79.7
		20	38	391	15.97	1.55	83.8	76.9	80.2
		26	30	402	20.23	1.51	80.5	79.8	80.1
	e	3	239	590	2.54	1.03	89.9	68.9	78.0
		7	85	428	7.14	1.42	83.7	76.3	79.8
		20	42	451	14.45	1.35	84.1	76.7	80.3
		26	34	473	17.85	1.28	81.2	79.7	80.4

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

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

After training, the N^+ N smaller SVMs are integrated into an M^3-SVM with N^+ MIN units and one MAX unit according to two combination principles [5][6] as follows,

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

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.

III. Two Types of Task Decomposition Strategies

Task decomposition is one of the two key problems in the M^3-SVM. In this section, we will introduce two types of task decomposition methods. One is the random task decomposition strategy, and the other is the hyperplane task decomposition strategy [7].

A. Random Task Decomposition Strategy

The random task decomposition method is a simple and straightforward strategy. It means that we randomly pick up samples to form a new smaller and more balanced training data set. We refer to the M^3-SVM using random decomposition as M^3-SVM (R). Though the strategy can be implemented easily, it might lead to partial loss of statistical properties of original training data and thus result in the decrease in the performance of text classification.

B. Hyperplane Task Decomposition Strategy

An ideal decomposition method is the one doing no damage to generalization performance. In order to achieve this goal, we hope to maintain the structural properties of the smaller data sets as those of original data set after task decomposition. Based on the idea, we first introduce a specific hyperplane, and then divide original training set into smaller training set using a series of hyperplanes which are parallel with the hyperplane introduced. We refer to the M^3-SVM using the proposed hyperplane task decomposition strategy as M^3-SVM (H).

Now our problem is whether the hyperplane task decomposition strategy is the most reasonable. In order to get a more balanced training set, we also tentatively let some training samples in the neighborhood of these hyperplanes

TABLE III
RESULTS ON GCAT,WHERE C=0.5

method		# SVMs	CPU time (s.) parallel	CPU time (s.) serial	Speed up parallel	Speed up serial	performance P	performance R	performance F_1
1		1	785	785	-	-	95.5	89.1	92.2
2		3	309	805	2.54	0.98	90.8	93.9	92.3
		8	126	675	6.23	1.16	92.7	91.9	92.3
		18	44	511	17.84	1.54	92.5	91.7	92.1
		24	35	522	22.43	1.50	90.1	93.3	91.7
3	a	3	374	750	2.10	1.05	92.8	91.7	92.2
		8	141	552	5.57	1.42	93.5	91.0	92.2
		18	55	448	14.27	1.75	93.2	90.9	92.1
		24	45	474	17.44	1.66	91.1	92.4	91.8
	b	3	375	789	2.10	0.99	93.1	91.4	92.3
		8	142	658	5.53	1.19	93.7	90.9	92.3
		18	55	519	14.27	1.51	93.3	91.0	92.1
		24	45	532	17.44	1.48	91.3	92.4	91.9
	c	3	378	807	2.08	0.97	93.3	91.3	92.3
		8	141	719	5.57	1.09	93.7	90.9	92.3
		18	55	552	14.27	1.42	93.3	91.1	92.2
		24	46	572	17.07	1.37	91.3	92.4	91.9
	d	3	387	872	2.03	0.90	93.4	91.1	92.3
		8	151	825	5.20	0.95	93.7	90.9	92.3
		18	58	609	13.53	1.29	93.4	91.1	92.2
		24	47	635	16.70	1.24	91.5	92.4	91.9
	e	3	393	923	2.00	0.85	93.7	90.9	92.3
		8	182	937	4.31	0.84	93.9	90.9	92.4
		18	64	691	12.27	1.14	93.4	91.2	92.3
		24	51	708	15.39	1.11	91.7	92.4	92.0

TABLE IV
RESULTS ON MCAT,WHERE C=0.5

method		# SVMs	CPU time (s.) parallel	CPU time (s.) serial	Speed up parallel	Speed up serial	performance P	performance R	performance F_1
1		1	636	636	-	-	94.5	87.2	90.7
2		4	184	663	3.46	0.96	87.9	93.7	90.7
		12	49	411	12.98	1.55	89.6	91.7	90.6
		24	25	419	25.44	1.52	90.4	90.8	90.6
		40	15	462	42.40	1.38	90.7	89.8	90.3
3	a	4	179	571	3.55	1.11	89.5	92.1	90.7
		12	53	350	12.00	1.82	90.5	91.4	91.0
		24	31	355	20.52	1.79	91.2	90.8	91.0
		40	22	396	28.91	1.61	91.4	90.2	90.8
	b	4	194.	614	3.28	1.04	90.2	91.6	90.9
		12	58	394	10.97	1.61	90.8	91.4	91.1
		24	32	408	19.88	1.56	91.5	90.8	91.1
		40	23	449	27.65	1.42	91.6	90.4	91.0
	c	4	204	639	3.12	0.99	90.5	91.4	90.9
		12	61	418	10.43	1.52	91.0	91.4	91.2
		24	34	438	18.71	1.45	91.5	90.8	91.2
		40	23	471	27.65	1.35	91.7	90.4	91.0
	d	4	224	694	2.84	0.92	90.8	91.1	91.0
		12	67	477	9.49	1.33	91.0	91.4	91.2
		24	37	479	17.19	1.33	91.6	90.8	91.2
		40	25	537	25.44	1.18	91.8	90.4	91.1
	e	4	239	737	2.66	0.86	91.3	90.7	91.0
		12	73	561	8.71	1.13	91.3	91.1	91.2
		24	39	540	16.31	1.18	91.7	90.8	91.3
		40	28	595	22.71	1.07	91.8	90.3	91.1

simultaneously belong to two smaller and more balanced training set divided by hyperplanes, since massive data set in the real world could be fuzzy. We refer to the part of small training set simultaneously belonging to adjacent training set as overlap of the small training set.

Suppose we divide the training data set of class C_i into N_i subsets. According to the above discussions, the M^3-SVM (H) method can be described as follows.

Step 1 Compute the distance between each training sample x of class C_i and hyperplane $H : Az = 0$ as follows,

$$dist(x, H) = \frac{Ax}{||A||} \tag{6}$$

where $x = [x_1, ..., x_n]$ is sample vector, $A = [a_1, ..., a_n]$ is the normal vector of hyperplanes, and $z = [z_1, ..., z_n]$ is any point in hyperplane H.

Step 2 Sort the training data according to the value of $dist(x, H)$.

Step 3 Divide the ordered sequence of training data to N_i parts equally, and then we can get more balanced subsets whose sizes are almost the same.

Step 4 Construct M^3-SVMs according to section II

From the above decomposition procedure, we can see that the hyperplane task decomposition can be easily implemented. However, a problem remained is how to determine the normal vector A of hyperplane. Experimentally, we take $A = [1, 1, ..., 1]$. There are two reasons that we choose this value of A. Firstly, in most cases of text categorization, the dimensions of sample vectors are very sparse. Therefore, in order to validly separate those samples mostly in every coordinates axis or planes, hyperplanes we used had better not be parallel with the coordinates axis or planes. The most effective way is to set their normal vector be 45 degree against all coordinates axis or planes, which leads to the results of setting a normal vector in which all elements are 1. Secondly, a hyperplane decomposition is equally a sorting operation of all input data according to all dimensional elements under a specified weight vector, which is just the normal vector A. Since we don't often own much prior knowledge of text data sets, and it is hard to determine which dimensional element is more important, therefore, it is natural to give the same weights on each dimension of all data. An illustration of hyperplane decomposition for sparse vectors in a two-dimension space is given in Fig. 1.

IV. EXPERIMENTS

In this section, we present experimental results for a text classification problem to indicate that the proposed hyperplane task decomposition method for M^3-SVMs is effective. We

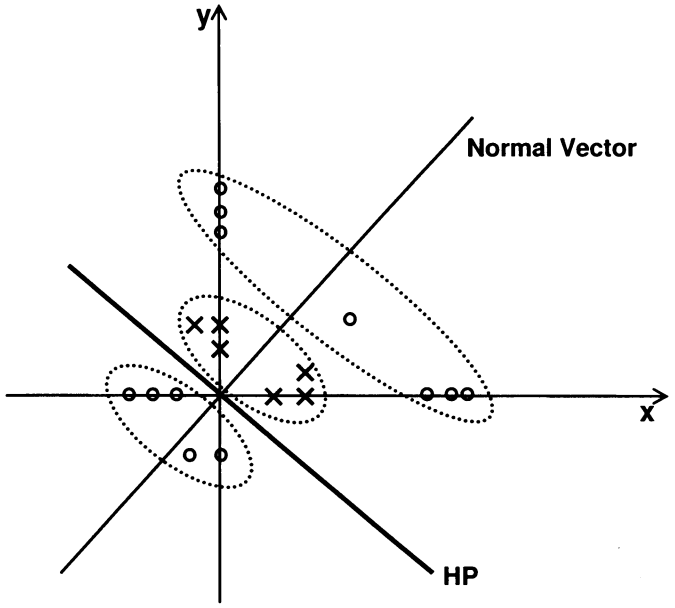

Fig. 1. Hyperplane in a two-dimension space where most sample vectors are sparse vectors. Forks and circles belong to one class. The dashed ellipses denote that the samples in it are clustered into a subproblem data set. HP is the so-called hyperplane, since this is a two-dimension space.

use the revised edition of Reuters Corpus Volume I (RCV1-v2) [8], for this study. There is an archive of over 800,000 manually categorized newswire stories in the full collections. In the simulations, we selected the top four classes, namely CCAT, ECAT, GCAT and MCAT, as given classes. CCAT denotes corporate/Industrial class, ECAT economics class, GCAT government/social class and MCAT markets class. Our training set has a size of 23,149 samples, and our testing set has a size of 199,328 samples from the first of four test data sets . The number of features for representing texts in the simulations is 47,152.

Now, the data set falls into the multilabel setting. Here we adopt the "one-versus-rest" strategy. Namely, a multilabel task [9] can be split up into a set of two-class classification tasks. Each category is treated as a separate two-class classification problem. Such a two-class problem only answers the question of whether or not a document should be assigned to a particular category. Therefore our multilabel classification task can be converted into four two-class classification tasks. Then we adopt the "part-versus-part" strategy, and each two-class classification task is decomposed into a series of two-class subproblems using different decomposition strategies.

After training all individual SVMs employed to solve two-class subproblems, we use min-max combination strategies to integrate the trained individual SVMs into a M^3-SVM. The version of the SVMs package is Libsvm2.4. We reprogram it into the MPI parallel program, namely M^3-SVMs. All the simulations were performed on an IBM p690 machine. It totally has 32 CPUs each of which is Power 4, 1.3GHZ.

To compare the performance of M^3-SVM(H) with traditional SVM and M^3-SVM(R), the text classification problem was learned by traditional SVM, M^3-SVM(R) and M^3-SVM(H) respectively. For the reason that, in many text categorization cases, linear kernel function can get a better generalization performance than other kernel functions such as polynomial and radial basis functions. So in our simulations, we take it as the main kernel function of SVMs. We have also used RBF kernel function to validate the former judgment. According to the discussion in Section III-B, we set the hyperplane $H : Az = 0$ with ones normal vector.

In Tables I through IV, '1', '2', and '3' denote that SVM, M^3-SVM(R), and M^3-SVM(H) are used in the simulation respectively. '#SVMs', 'P' and 'R' refer to the number of SVM classifiers, precision and recall. 'a', 'b', 'c', 'd', and 'e' respectively stand for no overlapping, 10% overlapping, 15% overlapping, 20% overlapping, and 30% overlapping of training data of subproblems for M^3-SVM(H) method.

We have made four groups of experiments according to different parameter C, which controls the tradeoff between complexity of the machine and the number of nonseparable points. In the simulations, we take C as 0.5, 1, 2 and 4 respectively. However, we only list the detailed results of each class with C=0.5, since the results are comparatively more representative than others for almost all methods.

For evaluating the effectiveness of category assignments by classifiers to documents, we use the standard recall, precision and F_1 measure[10]. Recall is defined to be the radio of the total number of correct assignments to correct assignments by the system. Precision is the ratio of the total number of the system's assignments to correct assignments by the system.

$$R = \frac{tp}{tp + fn}, \quad P = \frac{tp}{tp + fp} \tag{7}$$

where tp is the number of documents a system correctly assigns to the category (true positives), fp is the number of documents a system incorrectly assigns to the category (false positives), and fn is the number of documents that belong to the category but which the system does not assign to the category (false negatives).

The F_1 measure corresponds to the harmonic mean of recall and precision in the following form:

$$F_1 = \frac{2tp}{2tp + fp + fn} = \frac{2RP}{R + P} \tag{8}$$

where R is recall, and P is precision.

F_1 has been widely used in cross-method comparisons. Thus it is our main interest in the simulations.

From the experimental results shown in Tables I through IV, we can draw the following conclusions:

a. Even though all of the individual SVMs were trained in serial, M^3-SVMs including M^3-SVMs(R) and M^3-SVMs(H) is also much faster than traditional SVMs for four classes on the whole. And with the increase of classifiers, M^3-SVMs need less and less training time.

b. In most cases, the generalization performance of M^3-SVMs(R) is fluctuant. In some cases, M^3-SVMs(R) has better generalization performance than traditional SVM and in other

TABLE V

COMPARISON OF GENERALIZATION PERFORMANCE AND TRAINING TIME AMONG METHODS ,WHERE C=0.5

category	method		# SVMs	CPU time (s.)		Speed up		performance		
				parallel	serial	parallel	serial	precision	recall	F_1
CCAT	SVM		1	898	898	-	-	94.9	91.3	93.0
	M^3-SVM(R)		4	290	989	3.10	0.91	94.9	90.7	92.8
	M^3-SVM(H)	no overlap	30	47	563	19.1	1.60	93.5	92.7	93.1
		10% overlap	30	48	647	18.71	1.39	93.7	92.6	93.2
		15% overlap	30	52	707	17.27	1.27	93.8	92.6	93.2
		20% overlap	30	53	754	16.94	1.19	93.9	92.6	93.2
		30% overlap	30	60	894	14.97	1.00	94.0	92.6	**93.3**
ECAT	SVM		1	607	607	-	-	92.7	64.1	75.8
	M^3-SVM (R)		3	186	531	3.26	1.14	84.7	74.7	79.4
	M^3-SVM (H)	no overlap	20	34	300	17.85	2.02	82.6	77.5	80.0
		10% overlap	20	34	343	17.85	1.77	83.4	77.2	80.2
		15% overlap	20	36	363	16.86	1.67	83.8	76.9	80.2
		20% overlap	20	38	391	15.97	1.55	83.8	76.9	80.2
		30% overlap	26	34	473	17.85	1.28	81.2	79.7	**80.4**
GCAT	SVM		1	785	785	-	-	95.5	89.1	92.2
	M^3-SVM (R)		8	126	675	6.23	1.16	92.7	91.9	92.3
	M^3-SVM (H)	no overlap	8	141	552	5.57	1.42	93.5	91.0	92.2
		10% overlap	8	142	658	5.53	1.19	93.7	91.0	92.3
		15% overlap	8	141	719	5.57	1.09	93.7	90.9	92.3
		20% overlap	8	151	825	5.20	0.95	93.7	90.9	92.3
		30% overlap	8	182	937	4.31	0.84	93.9	90.9	**92.4**
MCAT	SVM		1	636	636	-	-	94.5	87.2	90.7
	M^3-SVM (R)		4	184	663	3.46	0.96	87.9	93.7	90.7
	M^3-SVM (H)	no overlap	24	31	355	20.52	1.79	91.2	90.8	91.0
		10% overlap	24	32	408	19.88	1.56	91.5	90.8	91.1
		15% overlap	24	34	438	18.71	1.45	91.5	90.8	91.2
		20% overlap	24	37	479	17.19	1.33	91.6	90.8	91.2
		30% overlap	24	39	540	16.31	1.18	91.7	90.8	**91.3**

cases, the reverse occurs. The experimental results support that in some case, random task decomposition might damage structual properties of original training data. However, when the generalization performance is very bad for some training set, for example for class ECAT, the generalization performance of M^3-SVM(R) can also be raised by 4.6% at the best case and still by 1.9% at the worst case.

c. In most cases, M^3-SVMs(H) shows better generalization performance than traditional SVMs and M^3-SVMs(R). And with the increasing number of classifiers, M^3-SVMs(H) has better and better generalization performance. On the other hand, while the number of the classifiers increases, training time also is on the decrease.

d. In all cases, M^3-SVMs(R) and M^3-SVMs(H) need much less training time than traditional SVMs, and compared with M^3-SVMs(H), M^3-SVMs(R) needs a little less training time. The communication expense in M^3-SVMs in the testing phase focuses on the MIN and MAX procedure, but this time cost is trivial.

For more clear comparison, we organize experimental results of different methods with all best generalization performance for each particular class into Table V. And we also give a detailed comparison on the performance of different methods corresponding to different parameter C in Table VI. The meanings of 'a', 'b', 'c', 'd' and 'e' are the same as Table I. We can see that M^3-SVMs, especially M^3-SVMs(H), is the best choice for text classification problems.

V. Conclusions

We have presented a new hyperplane task decomposition strategy for M^3-SVMs for multilabel text classification. The advantages of the proposed method over traditional SVMs are its parallelism and scalability. Experimental results proved this point. And compared with M^3-SVM (R), M^3-SVMs (H) has better generalization performance. When overlap ratio of training set is appropriate, M^3-SVMs (H) has better generalization performance. With the increase in the number of classifiers, the performance of M^3-SVMs (H) could reach a maximum. A future work is to search for breakpoint between overlapping ratio and the number of classifiers and analyze the effectiveness of the hyperplane decomposition strategy theoretically.

TABLE VI

COMPARISON OF GENERALIZATION PERFORMANCE AMONG DIFFERENT PARAMETER C FOR FOUR CLASSES

category	method		# SVMs	C=0.5			C=1			C=2			C=4		
				P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
CCAT	SVM		1	94.9	91.3	93.0	93.4	92.3	92.8	93.2	90.5	91.8	91.4	88.8	90.1
	M^3-SVM(R)		4	94.9	90.7	92.8	94.4	90.9	92.6	93.6	90.3	91.9	92.3	89.2	90.7
	M^3-SVM(H)	a	30	93.5	92.7	93.1	93.5	92.4	93.0	92.9	91.3	92.1	91.8	90.2	91.0
		b	30	93.7	92.6	93.2	93.7	92.3	93.0	93.0	91.2	92.1	91.9	90.0	90.9
		c	30	93.8	92.6	93.2	93.7	92.3	93.0	93.1	91.1	92.1	92.0	90.0	91.0
		d	30	93.9	92.6	93.2	93.7	92.3	93.0	93.0	91.1	92.1	91.9	90.0	90.9
		e	30	94.0	92.6	**93.3**	93.9	92.3	93.1	93.2	91.0	92.1	92.0	89.7	90.8
ECAT	SVM		1	92.7	64.1	75.8	90.0	67.5	77.2	86.1	69.0	76.6	80.4	68.8	74.2
	M^3-SVM (R)		3	84.7	74.7	79.4	83.8	75.4	79.1	91.4	73.7	77.4	78.5	71.0	74.6
	M^3-SVM (H)	a	20	82.6	77.5	80.0	82.0	77.7	79.8	80.8	75.8	78.2	79.6	73.1	76.2
		b	20	83.4	77.2	80.2	82.6	77.4	79.9	81.0	75.6	78.2	79.3	73.1	76.1
		c	20	83.8	76.9	80.2	82.9	77.2	80.0	81.2	75.5	78.3	78.7	73.8	76.2
		d	20	83.8	76.9	80.2	82.9	77.2	80.0	81.2	75.5	78.3	78.7	73.8	76.2
		e	26	81.2	79.7	**80.4**	81.1	78.9	80.0	81.6	75.1	78.2	80.1	72.4	76.1
GCAT	SVM		1	95.5	89.1	92.2	94.9	89.4	92.0	93.4	88.7	91.0	90.9	86.8	88.8
	M^3-SVM (R)		8	92.7	91.9	92.3	92.7	91.6	92.1	91.9	90.6	91.2	90.8	89.0	89.9
	M^3-SVM (H)	a	8	93.5	91.0	92.2	93.2	90.6	91.9	92.1	89.3	90.7	90.5	87.6	89.0
		b	8	93.7	91.0	92.3	93.3	90.5	91.9	92.4	89.2	90.8	90.8	87.5	89.1
		c	8	93.7	90.9	92.3	93.4	90.5	91.9	92.3	89.3	90.8	90.7	87.4	89.0
		d	8	93.7	90.9	92.3	93.4	90.5	91.9	92.4	89.3	90.8	90.8	87.4	89.1
		e	8	93.9	90.9	**92.4**	93.5	90.5	92.0	92.6	89.2	90.9	90.9	87.3	89.0
MCAT	SVM		1	94.5	87.2	90.7	93.7	88.0	90.7	91.9	88.3	90.1	88.5	87.5	88.0
	M^3-SVM (R)		4	87.9	93.7	90.7	88.1	93.1	90.5	87.9	91.7	89.7	87.1	89.2	88.1
	M^3-SVM (H)	a	24	91.2	90.8	91.0	91.1	91.1	91.1	90.3	90.4	90.4	89.4	88.9	89.1
		b	24	91.5	90.8	91.1	91.2	91.1	91.1	90.3	90.4	90.3	89.3	88.8	89.0
		c	24	91.5	90.8	91.2	91.2	91.1	91.2	90.3	90.4	90.3	89.3	88.8	89.1
		d	24	91.6	90.8	91.2	91.2	91.0	91.1	90.4	90.3	90.3	89.4	88.8	89.1
		e	24	91.7	90.8	**91.3**	91.3	91.0	91.2	90.5	90.3	90.4	89.4	88.9	89.2

ACKNOWLEDGMENTS

This research was partially supported 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.

REFERENCES

[1] C. Cortes and V. N. Vapnik, "Support-vector network", Machine Learning, Vol. 20 (1995) 273-297

[2] V. N. Vapnik, Statistical Learning Theory, John Wiley and Sons, New York, 1998

[3] Thorsten Joachims, "Text categorization with support vector machine: Learning with many relevant features", Technical report, University of Dortmund, Computer Science Department, 1997

[4] Yiming Yang and Xin Liu, "A re-examination of text categorization methods", In: Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval, 1999

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

[6] 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, July25-29 (2004) 735-740.

[7] Kai-An Wang, Hai Zhao and Bao-Liang Lu, "Task Decomposition Using Geometric Relation for Min-Max Modular SVMs", Advances in Neural Networks-ISNN2005, LNCS 3496, pp.887-892, Chongqing, China, May 29-June 2, 2005

[8] David D. Lewis, Yiming Yang, Tony G. Rose, Fan Li, "RCV1: a new benchmark collection for text categorization research", Journal of Machine Learning Research 5 (2004) 361-397

[9] Thorsten Joachims, Learning to classify text using support vector machine:method, theory, and algorithms. Kluwer Academic Publishers, 2002

[10] David D.Lewis, "Evaluating and optimizing autonomous text classification systems". In Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrival (SIGIR 95), pages 246-254,1995.