Efficient Text Categorization Using a Min-Max Modular Support Vector Machine

Feng-Yao Liu[1], Kai-An Wang[1], Bao-Liang Lu[1*], Masao Utiyama [2], and Hitoshi Isahara [2]

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

[2] NICT Computational Linguistics Group
3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto, Japan
{mutiyama;isahara}@nict.go.jp

Abstract. The min-max modular support vector machine (M^3-SVM) has been proposed for solving large-scale and complex multiclass classification problems. In this paper, we apply the M^3-SVM to multilabel text categorization and introduce two task decomposition strategies into M^3-SVMs. A multilabel classification task can be split up into a set of two-class classification tasks. These two-class tasks are to discriminate class *C* from non-class *C*. 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 SVMs 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 indicate that the new method has better generalization performance than traditional SVMs and previous M^3-SVMs using random task decomposition, and is much faster than traditional SVMs.

1 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. Due to their powerful learning ability and good generation performance,

* To whom the respondence should be addressed. 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.

G. Hommel and Sheng Huanye (eds.), Human Interaction with Machines, 13-21.

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) neural 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 multi-class 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 a new strategy 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 Section 2, M^3-SVM is introduced briefly. In Section 3, several different task composition strategies are listed. Then in Section 4, we designed a set of experiments on a large-scale multilabel text classification. In Section 5, conclusions are outlined.

2 Min-Max Modular Support Vector Machine

Min-max modular support vector machine [6] is a method that divides a complex classification problem into many small independent two-class classification problems, learns these two-class problems in a parallel way, 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 T, let χ^+ denote the positive training data set belonging to a particular category $\boldsymbol{C}$ and χ^- denote the negative training data set not belonging to $\boldsymbol{C}$.

$$\chi^+ = \left\{\left(x_i^+, +1\right)\right\}_{i=1}^{l^+}, \chi^- = \left\{\left(x_i^-, -1\right)\right\}_{i=1}^{l^-} \quad (1)$$

where $x_i \in R^n$ 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 [6], χ^+ and χ^- can be partitioned into N^+ and N^- subsets respectively,

$$\chi_j^+ = \left\{\left(x_i^{+j}, +1\right)\right\}_{i=1}^{l_j^+}, \text{ for j=1, ... , } N^+ \quad (2)$$

$$\chi_j^- = \left\{\left(x_i^{-j}, -1\right)\right\}_{i=1}^{l_j^-}, \text{ for j=1, ... , } N^- \quad (3)$$

where $\bigcup_{j=1}^{N^+} \chi_j^+ = \chi^+, 1 \le N^+ \le l^+$, and $\bigcup_{j=1}^{N^-} \chi_j^- = \chi^-, 1 \le N^- \le l^-$.

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

$$\left(T^{(i,j)}\right)^+ = \chi_i^+, \left(T^{(i,j)}\right)^- = \chi_j^- \quad (4)$$

where $\left(T^{(i,j)}\right)^+$ and $\left(T^{(i,j)}\right)^-$ denote the positive and negative data set of subproblem $T^{(i,j)}$ 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 $N^+ \times N^-$ smaller SVMs are integrated into a M^3-SVM with N^+ MIN units and one MAX unit according to two combination principles [5][6] as follows,

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

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

3 Two Types of Task Decomposition Strategies

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

3.1 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, so in some multiclass classification cases, it could result in a little decrease in the performance of text classification. Whereas, in the multilabel classifications, our M^3-SVM using random task decomposition method can also get better performance than SVMs.

3.2 Hyperplane decomposition strategy

An ideal decomposition strategy is the one without loss of the generalization performance. In order to achieve this goal, we hope to maintain the structural properties of the smaller data set as those of original data set after task decomposition. So we introduce a hyperplane strategy which divides original training set into smaller training sets using a series of hyperplanes. We regard the M^3-SVM using the hyperplane decomposition strategy as M^3-SVM (H). In some cases, we also let those training samples in these hyperplanes neighborhood simultaneously belong to two smaller and more balanced training set divided by hyperplanes which can be regarded 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: $a_1 z_1 + a_2 z_2, ..., + a_n z_n = 0$ as follows,

$$dist(x, H) = \frac{a_1 x_1 + a_2 x_2, ..., + a_n x_n}{\sqrt{a_1^2 + a_2^2, ..., + a_n^2}} \tag{6}$$

Where x_i is the element of sample vector x.

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 to remain the size of sub-sets almost the same.

Step 4. Construct M^3-SVM according to Section 2.

From the above decomposition procedure, we can see that the hyperplane task decomposition strategy can be easily implemented.

Now we construct an artificial data set illustrated in **Fig. 1.** Suppose that we want to divide the positive samples into two parts. The best way is that we take the hyperplane $X+Y=0$. Consequently, positive samples is partitioned into two comparatively smaller and more balanced data sets, and the structure of the data sets still remains unchanged after the combination according to the module combination principles of M^3-SVMs.

4 Experiments

In this section, we present experimental results for two mutillabel text classification problems to indicate that both the random task decomposition and the hyperplane task decomposition methods for M^3-SVM are effective. We use the Yomiuri News Corpus [7] and the revised edition of Reuters Corpus Volume I(RCV-v2) [8] for our study respectively. For the former, there are 913,118 documents in the full collections from the year 1996 to 2000. And we only use 274,030 documents which is 30 percent of the original collections as our training data set and 181,863 documents dated July-December of 2001 as our test set. There are totally 75 classes in this collection. For the RCV1-v2, we selected the four top classes, namely CCAT, ECAT, GCAT and MCAT, as the given classes. The training set we adopted here has a size of 23,149 samples, and the testing set 199,328 samples. The number of features for representing texts in the simulations is 5000 and 47,152 respectively. For our convenience, we regard the simulation using Yomiuri News collection as SIM(Y) and RCVI-v2 as SIM(R).

A multi-label 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 multi-label classification task can be converted into many two-class classification tasks. And each two-class classification task is decomposed into a series of two-class subproblems using different decomposition strategies. After training all two-class subproblems, we use min-max

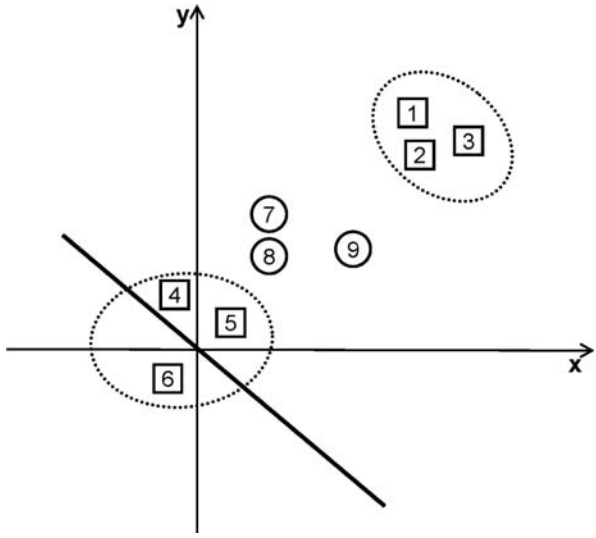

Fig. 1. An artificial data set in two-dimension space for making the explanation about effect if hyperplane task decomposition is used in M^3-SVMs. Here, square and circle denote positive and negative sample respectively. The solid line is so-called hyperplane.

combination strategies to integrate the trained individual SVMs into a M^3-SVM. All of the simulations were performed on an IBM p690 machine.

To compare the performance of M^3-SVM(H) with traditional SVM and M^3-SVM(R), the text classification in SIM(Y) is learned by traditional SVM and M^3-SVM(R) respectively, and in SIM(R), it is learned by traditional SVM, M^3-SVM(R) and M^3-SVM(H). In all the simulations, the kernel function of SVMs is linear kernel function and the hyperplane used in the task decomposition of SIM(R) is $z_1 + z_2, ..., + z_n = 0$.

In SIM(Y), we only list the results of top 10 classes and the parameter C we take is 8. The result is shown in **Table 1**. In SIM(R), we made four groups of experiments according to different tradeoffs C between training error and margin to the reciprocal of the average Euclidean norm of training examples. In each group, we take C as 0.5, 1, 2, 4, respectively, and also perform the simulations with 15% and 30% overlapping of the training samples. However, we only list the results on ECAT with C=0.5, since the results are comparatively best. The result is shown in **Table 2**. In this Table, 'o(0)', 'o(15)', 'o(30)' respectively stands for no overlap, 15% overlap, and 30% overlap of training data for M^3-SVM(H) method.

For more clear comparison, we organize experimental results of different methods in SIM(R) with all best generalization performance for each

particular class into **Table 3**. From this table, we can see that M^3-SVM is a good choice for text classification problems.

Table 1. The simulation results for SIM(Y), where C=8

Class Num	SVM			M^3-SVM		
	$F_{1.0}$	Time (h)	#SVM	$F_{1.0}$	Serial (h)	Parallel (h)
1	84.00	9.47	2	84.86	7.49	3.79
2	94.36	2.89	2	94.55	2.91	1.57
3	66.13	4.48	3	70.62	2.99	1.11
4	62.54	5.01	3	68.61	3.22	1.25
5	64.68	5.38	3	69.57	4.00	1.53
6	85.01	3.09	3	85.11	2.15	0.76
7	40.21	6.94	3	50.82	4.82	1.72
8	15.34	64.99	2	24.54	28.40	16.22
9	76.79	5.62	3	77.22	4.03	1.38
10	60.15	13.21	2	62.22	9.94	5.62

For evaluating the effectiveness of category assignments by classifiers to documents, we use the standard recall, precision and $F_{1.0}$ measure. 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. The $F_{1.0}$ measure combines recall (r) and precision (p) with an equal weight in the following form:

$$F_{1.0} = \frac{2rp}{r+p} \tag{7}$$

From the experimental results shown in Tables 2 ans 3, we can draw the following conclusions:

a. Even though all of the individual SVMs were trained in serial, M^3-SVM including M^3-SVM(R) and M^3-SVM(H) is faster than traditional SVMs in our simulations. 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-SVM(R) is fluctuant. In some cases, M^3-SVM(R) has better generalization performance than traditional SVM and in other cases, the reverse occurs. Especially in SIM(Y), all the 10 classes we selected had a better performance by M^3-SVM(R) than by traditional SVMs. When the generalization performance is bad for some training set, for example for class ECAT, the generalization performance of M^3-SVM (R) can be raised by 4.6% at the

best case and still by 1.9% at the worst case. And in SIM(Y), for class 7, the performance is raised from 40.21% to 50.82%.

Table 2. Results on ECAT in SIM(R), where C=0.5

Method		#SVM	CPU time (s.)		Performance		
			parallel	serial	P	R	$F_{1.0}$
SVM		1	607	607	92.7	64.1	75.8
M^3-SVM(R)		3	186	531	84.7	74.7	79.4
		7	53	347	73.8	82.5	77.9
		20	21	365	78.5	78.3	78.4
		26	16	365	74.5	81.1	77.7
M^3-SVM(H)	O(0)	3	234	461	87.9	71.0	78.6
		7	66	307	80.4	78.9	79.6
		20	34	300	82.6	77.5	80.0
		26	26	310	79.0	80.6	79.8
	O(15)	3	233	519	89.1	70.0	78.4
		7	72	350	82.0	77.5	79.7
		20	36	363	83.8	76.9	80.2
		26	28	373	80.2	79.9	80.0
	O(30)	3	239	590	89.9	68.9	78.0
		7	85	428	83.7	76.3	79.8
		20	42	451	84.1	76.7	80.3
		26	34	473	81.2	79.9	80.4

c. In all the cases in SIM(R), M^3-SVM(H) show better generalization performance than traditional SVMs and M^3-SVM(R). And with the increasing number of classifiers, M^3-SVM(H) has better and better generalization performance.

d. In all the cases, M^3-SVM(R) need less training time than traditional SVMs and M^3-SVM(H).

5 Conclusions

We have presented two task decomposition strategies for multilabel text classification. The advantages of the proposed methods over traditional SVMs are its parallelism and scalability. And compared with M^3-SVM(R), M^3-SVM(H) has better generalization performance. When overlap ratio of training set become high, M^3-SVM(H) have better generalization performance. With the increase of the number of classifiers, the performance of M^3-SVM(H) could reach maximum. A future work is to search for breakpoint between overlap ratio and the number of classifiers and analysis the effectiveness of the hyperplane decomposition strategy theoretically.

Table 3. Comparison of generalization performance and training time among methods in SIM(R), where C=0.5 and CAT stands for category

CAT	Method	#SVM	CPU time (s.)		Performance (%)		
			parallel	serial	P	R	$F_{1.0}$
CCAT	SVM	1	898	898	94.9	91.3	93.0
	M^3-SVM(R)	4	290	989	94.9	90.7	92.8
	M^3-SVM(H) O(0)	30	47	563	93.5	92.7	93.1
	O(30)	30	60	894	94.0	92.6	**93.3**
ECAT	SVM	1	607	607	92.7	64.1	75.8
	M^3-SVM(R)	3	186	531	84.7	74.7	79.4
	M^3-SVM(H) O(0)	20	34	300	82.6	77.5	80.8
	O(30)	26	34	473	81.2	79.7	**80.4**
GCAT	SVM	1	785	785	85.5	89.1	92.2
	M^3-SVM(R)	8	126	675	92.7	91.9	92.3
	M^3-SVM(H) O(0)	8	141	552	93.5	91.0	92.2
	O(30)	8	182	937	93.9	90.9	**92.4**
MCAT	SVM	1	636	636	94.5	87.2	90.7
	M^3-SVM(R)	4	184	663	87.9	93.7	90.7
	M^3-SVM(H) O(0)	24	31	355	91.2	90.8	91.0
	O(30)	24	39	540	91.7	90.8	**91.3**

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. M. Utiyama and H. Ichapire. Experiments with a new boosting algorithm. In Proceedings of 13th International Conference on Machine Learning, 1996, 148-156
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.