A Part-Versus-Part Method for Massively Parallel Training of Support Vector Machines

Bao-Liang Lu, Kai-An Wang
Department of Computer Science and Engineering
Shanghai Jiao Tong University
1954 Hua Shan Rd., Shanghai 200030, China
blu@cs.sjtu.edu.cn

Masao Utiyama, and Hitoshi Isahara
Communications Research Laboratory
3-5 Hikaridai, Seika-cho, Soraku-gun
Kyoto, 619-0289, Japan
{utiyama;isahara}@crl.go.jp

Abstract— **This paper presents a part-versus-part decomposition method for massively parallel training of multi-class support vector machines (SVMs). By using this method, a massive multi-class classification problem is decomposed into a number of two-class subproblems as small as needed. An important advantage of the part-versus-part method over existing popular pairwise-classification approach is that a large-scale two-class subproblem *can be further divided into a number of relatively smaller* and balanced two-class subproblems, and fast training of SVMs on massive multi-class classification problems can be easily implemented in a massively parallel way. To demonstrate the effectiveness of the proposed method, we perform simulations on a large-scale text categorization problem. The experimental results show that the proposed method is faster than the existing pairwise-classification approach, better generalization performance can be achieved, and the method scales up to massive, complex multi-class classification problems.**

I. Introduction

In the last several years, support vector machines (SVMs) [2], [16] have been successfully applied to various pattern classification problems. They are clearly recognized as useful tools one might use for pattern classification. To apply SVMs to multi-class problems, one usually needs to decompose a multi-class problem into a series of two-class subproblems, since SVMs were originally designed for learning two-class classification problems. There are two popular decomposition methods in machine learning literature: one-versus-the-rest approach [1] and pairwise-classification approach [7], [4], [9], [8], in which a K-class problem is decomposed into K two-class subproblems and $K(K-1)/2$ two-class subproblems, respectively.

In some real-world multi-class problems such as text categorization and patent classification, the size of training data is usually massive. For example, the Yomiuri News Corpus used in this paper contains 2,190,512 documents collected from Yomiuri Newspapers dated 1987-2001, with 75 unique categories assigned to those documents [15]. How to learn this kind of massive multi-class problems efficiently with SVM learning techniques [16], [6], [17], [18] and new computing infrastructure such as grid computing [3] is a big challenge to researchers in both neural network and machine learning fields.

In our previous work [9], [10], we have proposed a general task decomposition method for pattern classification. An important advantage of this method over pairwise-classification approach is that a two-class problem can be further decomposed into a series of two-class subproblems as small as needed. The pairwise-classification approach can be considered as a special case of our decomposition method, when a K-class problem is just decomposed into $K(K-1)/2$ two-class subproblems and there is no further decomposition for each of the two-class subproblems. Our decomposition method has been successfully applied to neural-networks for learning large-scale, real-world multi-class problems such as part-of-speech tagging [12] and classification of high-dimensional, single-trial electroencephalogram signals [13].

In this paper, we adapt our decomposition method to multi-class SVM learning. In Section II, we introduce our decomposition method. In Section III, we describe how to integrate individual trained SVMs into a hierarchical, parallel, and modular SVM with two module combination principles. In Section IV, we perform a simulation study on a large-scale text categorization problem. Conclusions are outlined in Section V.

II. Part-Versus-Part Decomposition Method

For human beings, the only way to solve a complex problem is to divide it into smaller, more manageable subproblems. Breaking up a problem helps human beings deal with complex issues involved in its solution. This "divide-and-conquer" strategy is also helpful to neural networks and SVMs in complex learning problems. Our goal in this section is to explain a part-versus-part decomposition method for training massive multi-class SVMs.

Let $\mathcal{T}$ be the given training data set for a K-class classification problem,

$$\mathcal{T} = \{(X_l, \hat{Y}_l)\}_{l=1}^{L}, \tag{1}$$

where $X_l \in \mathcal{X} \subset \mathbf{R}^n$ is the input vector, $\mathcal{X}$ is the set of training inputs, $\hat{Y}_l \in \mathcal{Y} \subset \mathrm{R}^K$ is the desired output, $\mathcal{Y}$ is the set of desired outputs, and L is the total number of training data.

We have suggested that a K-class problem defined by (1) can be divided into $K(K-1)/2$ two-class subproblems [9], [10], each of which is given by

$$\mathcal{T}_{ij} = \{(X_l^{(i)}, +1)\}_{l=1}^{L_i} \cup \{(X_l^{(j)}, -1)\}_{l=1}^{L_j} \tag{2}$$
$$\text{for } i = 1, \cdots, K \text{ and } j = i+1, \cdots, K$$

0-7803-8359-1/04/$20.00 ©2004 IEEE

$X_l^{(i)} \in \mathcal{X}_i$ and $X_l^{(j)} \in \mathcal{X}_j$ are the training inputs belonging to class $\mathcal{C}_i$ and class $\mathcal{C}_j$, respectively, $\mathcal{X}_i$ is the set of training inputs belonging to class $\mathcal{C}_i$, L_i denotes the number of data in $\mathcal{X}_i$, $\cup_{i=1}^{K} \mathcal{X}_i = \mathcal{X}$, and $\sum_{i=1}^{K} L_i = L$. In this paper, the training data in a two-class subproblem are called *positive* training data if their desired outputs are $+1$. Otherwise, they are called *negative* training data. The two-class subproblems defined by (2) are called *pairwise classification* in the machine learning literature [4], [8]. We would like to emphasize that decomposition of a K-class problem into $K(K-1)/2$ two-class subproblems defined by (2) is unique for a given training data set because of the uniqueness of $\mathcal{X}_i$ for $i = 1, \cdots, K$.

Although the two-class subproblems defined by (2) are smaller than the original K-class problem, this partition may not be adequate for parallel computation and fast learning due to the following reasons. (a) Some of the two-class subproblems might fall into a 'load imbalance' situation. Since the speed of parallel learning is limited by the speed of the slowest subproblem, the undue burdening of even a single subproblem can dramatically degrade the overall performance of learning. (b) Some of the two-class subproblems might still be too large for learning[18]. (c) Some of the two-class subproblems might be imbalanced in terms of types of examples in the training set, i.e., the training set contains many more data of the 'dominant' class than the other 'subordinate' class. . To speed up learning, all the large and imbalanced two-class subproblems should be further divided into relatively smaller and more balanced two-class subproblems.

Assume that $\mathcal{X}_i$ is partitioned into N_i subsets in the form

$$\mathcal{X}_{ij} = \{X_l^{(ij)}\}_{l=1}^{L_i^{(j)}} \quad \text{for } j = 1, \cdots, N_i \text{ and } i = 1, \cdots, K, \tag{3}$$

where $1 \le N_i \le L_i$ and $\cup_{j=1}^{N_i} \mathcal{X}_{ij} = \mathcal{X}_i$.

Various methods can be used for partitioning $\mathcal{X}_i$ into N_i subsets [9]. A simple and straightforward approach is to divide $\mathcal{X}_i$ randomly. The subsets $\mathcal{X}_{ij}$ might be disjoint or joint. Without loss of generality and for simplicity of description, we assume throughout this paper that the random decomposition method is used and the subsets $\mathcal{X}_{ij}$ are disjoint from each other, i.e., $\mathcal{X}_{ij} \cap \mathcal{X}_{ik} = \phi$ for $i = 1, \cdots, K$, $j, k = 1, \cdots, N_i$, and $j \neq k$.

In practical applications of SVMs, an appropriate value of N_i might depend on two main factors, such as the number of training data belonging to each class and the available computational power. In the simulations presented in this paper, we randomly divide $\mathcal{X}_i$ into N_i subsets $\mathcal{X}_{ij}$, which are roughly the same in size. The number of subsets N_i for class $\mathcal{C}_i$ is determined according to the following rule:

$$N_i = \begin{cases} \lfloor \frac{2L_i}{\rho} \rfloor & \text{if } fmod\ (\frac{2L_i}{\rho}) \le \gamma \text{ and } 2L_i > \rho \\ \lceil \frac{2L_i}{\rho} \rceil & \text{otherwise} \end{cases} \tag{4}$$

where ρ is the desired number of training data for two-class subproblems, γ is a threshold parameter ($0 < \gamma < 1$) for fine-tuning the number of subsets, $\lfloor z \rfloor$ denotes the largest integer less than or equal to z, $\lceil z \rceil$ denotes the smallest integer larger than or equal to z, the function of $fmod(z_1/z_2)$ is employed to produce the decimal part of z_1/z_2, and z_1 and z_2 are two positive integers, respectively.

After partitioning $\mathcal{X}_i$ into N_i subsets, every two-class subproblem $\mathcal{T}_{ij}$ defined by (2) can be further divided into $N_i \times N_j$ relatively smaller and more balanced two-class subproblems as follows:

$$\mathcal{T}_{ij}^{(u,v)} = \{(X_l^{(iu)}, +1)\}_{l=1}^{L_i^{(u)}} \cup \{(X_l^{(jv)}, -1)\}_{l=1}^{L_j^{(v)}} \quad \text{for } u = 1, \cdots, N_i,\ v = 1, \cdots, N_j, \quad i = 1, \cdots, K, \text{ and } j = i+1, \cdots, K \tag{5}$$

where $X_l^{(iu)} \in \mathcal{X}_{iu}$ and $X_l^{(jv)} \in \mathcal{X}_{jv}$ are the training inputs belonging to class $\mathcal{C}_i$ and class $\mathcal{C}_j$, respectively, $\sum_{u=1}^{N_i} L_i^{(u)} = L_i$, and $\sum_{v=1}^{N_j} L_j^{(v)} = L_j$. It should be noted that all the two-class subproblems have the same number of input dimensions as the original K-class problem. Comparing the two-class subproblems defined by (5) with the two-class subproblems obtained by the pairwise-classification approach, we can see that each of the two-class subproblems defined by (5) contains only a part of data of each class. Hence, the decomposition method is called *part-versus-part* method.

According to the above discussion, the part-versus-part decomposition method can be described as follows.

Step 1 : Set the values of ρ and γ.

Step 2 : Divide a K-class problem $\mathcal{T}$ into $\binom{K}{2}$ two-class subproblems $\mathcal{T}_{ij}$ using (2).

Step 3 : If the sizes of all $\mathcal{T}_{ij}$ are less than ρ, then stop the procedure here. Otherwise, continue with the following steps.

Step 4 : Determine the number of training input subsets N_i for $i = 1, \cdots, K$ using (4).

Step 5 : Divide the training input set $\mathcal{X}_i$ into N_i subsets $\mathcal{X}_{ij}$ using (3).

Step 6 : Divide the two-class subproblem $\mathcal{T}_{ij}$ into $N_i \times N_j$ relatively smaller and simpler two-class subproblems $\mathcal{T}_{ij}^{(u,v)}$ using (5).

From the above decomposition procedure, we see that the part-versus-part decomposition method is simple and straightforward, and neither domain specialists nor prior knowledge of the problem is required. Therefore, any user can perform this decomposition and divide a large K-class problem into many two-class subproblems as small as needed.

After task decomposition, each of the two-class subproblems can be treated as a completely independent, non-communicating problem in the learning phase. Therefore, all the two-class subproblems can be efficiently learned in a massively parallel way.

From (2) and (5), we see that a K-class problem is divided into

$$\sum_{i=1}^{K-1} \sum_{j=i+1}^{K} N_i \times N_j \tag{6}$$

two-class subproblems. The number of training data for each

of the two-class subproblems is about

$$\lceil L_i/N_i \rceil + \lceil L_j/N_j \rceil \tag{7}$$

Since $\lceil L_i/N_i \rceil + \lceil L_j/N_j \rceil$ is independent of the number of classes K, the size of each of the two-class subproblems is much smaller than the original K-class problem for reasonable N_i and N_j.

III. Min-Max Modular Support Vector Machines

After training individual SVMs assigned to learn associated two-class subproblems, all the trained SVMs are integrated into a min-max modular (M^3) SVM with the MIN, MAX, or/and INV units according to the following two module combination laws [9], [10], [11], namely the *minimization* principle and the *maximization* principle.

Minimization Principle: Suppose a two-class problem $\mathcal{B}$ were divided into P relatively smaller two-class subproblems, $\mathcal{B}_i$ for $i = 1, \cdots, P$, and also suppose that all the two-class subproblems have the same positive training data and different negative training data. If the P two-class subproblems are correctly learned by the corresponding P individual SVMs, M_i for $i = 1, \cdots, P$, then the combination of the P trained SVMs with a MIN unit will produce the correct output for all the training inputs in $\mathcal{B}$, where the function of the MIN unit is to find a minimum value from its multiple inputs. The transfer function of the MIN unit is given by

$$q(x) = \min_{i=1}^{P} \mathrm{M}_i(x) \tag{8}$$

where x denotes the input variable.

Maximization Principle: Suppose a two-class problem $\mathcal{B}$ were divided into P relatively smaller two-class subproblems, $\mathcal{B}_i$ for $i = 1, \cdots, P$, and also suppose that all the two-class subproblems have the same negative training data and different positive training data. If the P two-class subproblems are correctly learned by the corresponding P individual SVMs, M_i for $i = 1, \cdots, P$, then the combination of the P trained SVMs with a MAX unit will produce the correct output for all the training input in $\mathcal{B}$, where the function of the MAX unit is to find a maximum value from its multiple inputs. The transfer function of the MAX unit is given by

$$q(x) = \max_{i=1}^{P} \mathrm{M}_i(x) \tag{9}$$

Following the minimization and maximization principles, the $N_i \times N_j$ smaller SVMs are integrated first with N_i MIN units and one MAX unit as follows:

$$\mathrm{M}_{ij}^{(u)}(x) = \min_{v=1}^{N_j} \mathrm{M}_{ij}^{(u,v)}(x) \tag{10}$$

and

$$\mathrm{M}_{ij}(x) = \max_{u=1}^{N_i} \mathrm{M}_{ij}^{(u)}(x) \tag{11}$$

where $\mathrm{M}_{ij}^{(u,v)}(x)$ denotes the transfer function of the trained SVM corresponding to the two-class subproblem $\mathcal{T}_{ij}^{(u,v)}$, and $\mathrm{M}_{ij}^{(u)}(x)$ denotes the transfer function of a combination of N_j SVMs integrated by the MIN unit.

Suppose that a 1-out-of-K scheme were used for output representation. Let Y denote the actual output vector of the M^3-SVM for a K-class classification problem, and let $g(x)$ denote the transfer function of the entire M^3-SVM.

We may then write

$$Y = g(x) = [g_1(x), \cdots, g_K(x)]^T \tag{12}$$

According to the minimization and maximization principles, the $\binom{K}{2}$ SVMs, $\mathrm{M}_{ij}(x)$ for $i = 1, \cdots, K$ and $j = i+1, \cdots, K$, and the corresponding $\binom{K}{2}$ inversions $\mathrm{M}_{rs}(x)$ for $r = 2, \cdots, K$ and $s = 1, \cdots, r-1$, are integrated as

$$g_i(x) = \min \left[\min_{j=i+1}^{K} \mathrm{M}_{ij}(x), \min_{r=1}^{i-1} \overline{\mathrm{M}_{ri}(x)} \right] \tag{13}$$

where $g_i(x)$ for $i = 1, \cdots, K$ denotes the *discriminant function*, which discriminates the patterns of class C_i from those of the remaining classes, and the term $\overline{\mathrm{M}_{ri}(x)}$ denotes the inversion of $\mathrm{M}_{ri}(x)$.

It is easy to implement $\overline{\mathrm{M}_{ri}(x)}$ with $\mathrm{M}_{ri}(x)$ and an INV unit. The function of the INV unit is to invert its single input; the transfer function of the INV unit is given by

$$q = \alpha + \beta - p \tag{14}$$

where α, β, p, and q are the upper and lower limits of input value, input, and output, respectively. For example, α and β are set to +1 and -1, respectively, for support vector classifiers in the simulations below.

The relationship among $\mathrm{M}_{rs}(x)$, $\overline{\mathrm{M}_{sr}(x)}$, and the INV unit can be expressed as

$$\mathrm{M}_{rs}(x) = \overline{\mathrm{M}_{sr}(x)} = \mathrm{INV}(\mathrm{M}_{sr}(x)) \quad \text{for } s = 1, \cdots, K-1;\ r = s+1, \cdots, K \tag{15}$$

Similarly, the discriminant function $g_i(x)$ of the Min-Max SVM, which consists of $\sum_{i=1}^{K-1} \sum_{j=i+1}^{K} N_i \times N_j$ network modules, and the corresponding $\binom{K}{2}$ inversions can be expressed as

$$g_i(x) = \min \left[\min_{j=i+1}^{K} \left[\max_{k=1}^{N_i} \left[\min_{l=1}^{N_j} \mathrm{M}_{ij}^{(k,l)}(x) \right] \right], \min_{r=1}^{i-1} \overline{\left[\max_{k=1}^{N_r} \left[\min_{l=1}^{N_i} \mathrm{M}_{ri}^{(k,l)}(x) \right] \right]} \right] \tag{16}$$

where the term $\overline{\max_{k=1}^{N_r} [\min_{l=1}^{N_i} \mathrm{M}_{ri}^{(k,l)}(x)]}$ denotes the inversion of $\max_{k=1}^{N_r} [\min_{l=1}^{N_i} \mathrm{M}_{ri}^{(k,l)}(x)]$. It should be noted that only the inversions of network modules $\mathrm{M}_{ij}(x)$ are used for constructing the M^3-SVMs, and there are no inversions for SVMs $\mathrm{M}_{ij}^{(u,v)}(x)$.

Summarizing the discussion above, the module combination procedure can be described as follows:

Step 1 : If no SVMs $M_{ij}^{(u,v)}(x)$ exist, go to Step 3. Otherwise, perform the following steps.

Step 2 : Integrate $N_i \times N_j$ SVMs $M_{ij}^{(u,v)}(x)$ for $u = 1, \cdots, N_i$, $v = 1, \cdots, N_j$, $i = 1, \cdots, K$, and $j = i+1, \cdots, K$, into a network module $M_{ij}(x)$

TABLE I

NUMBER OF SVMS AND INTEGRATING UNITS REQUIRED TO BUILD THE M^3-SVM FOR A K-CLASS PROBLEM ($K > 2$)

Name	#elements
SVMs	$2\sum_{i=1}^{K-1}\sum_{j=i+1}^{K} N_i \times N_j$
MIN	$K + 2\sum_{i=1}^{K-1}\sum_{j=i+1}^{K} N_i\lceil\frac{N_j-1}{N_j}\rceil$
MAX	$2\sum_{i=1}^{K-1}(K-i)\lceil\frac{N_i-1}{N_i}\rceil$
INV	$K(K-1)/2$

with N_i MIN units and one MAX unit according to (10) and (11).

Step 3 : Integrate $K(K-1)/2$ modules and the corresponding $K(K-1)/2$ inversions with K MIN units according to (13).

From the procedure above dealing with module combination, we see that individual trained SVMs can be simply integrated into a M^3-SVM with MIN, MAX and/or INV units. Since the module combination procedure is completely independent of both the structure of individual trained SVMs and their performance, we can easily replace any trained SVMs with desired ones to achieve better generalization performance. In contrast to the task decomposition procedure mentioned earlier, the module combination procedure proceeds in a bottom-up manner. The smaller trained SVMs are integrated into larger modules first, and then the larger modules are integrated into a M^3-SVM.

After finishing module combination, the solutions to the original K-class problem can be obtained from the outputs of the entire M^3-SVM as follows:

$$C = \arg\max_i \{g_i(x)\} \text{ for } i = 1, \cdots, K \qquad (17)$$

where C is the class that the M^3-SVM assigns to the input x.

Once the size of each of the SVMs is fixed, the space complexity of the entire M^3-SVM is determined according to (13) and (16). Table I shows the number of SVMs and integrating units required to construct a M^3-SVM for a K-class problem.

IV. EXPERIMENTS

In this section we present experimental results on a text categorization problem to indicate the effectiveness of the proposed part-versus-part decomposition method. We use Yomiuri News Corpus for this study.

There are 2,190,512 documents in the full collections from the years 1987 to 2001. We used 913,118 documents dated 1996-2000 as a training set and 181,863 documents dated July-December of 2001 as a test set in this study. In the simulations, we selected the top five classes as shown in Table II. A χ^2 statistic (CHI) feature selection method [14] was used for preprocessing the documents after the morphological analysis was performed with ChaSen. The number of features is 5,000. In the simulations, C and γ [5] were selected as 8 and 0.25 for training all of the standard SVMs and M^3-SVMs. All of the simulations were performed on a 3.0GHz Pentium 4 PC with 2.0GB RAM.

TABLE II

DISTRIBUTIONS OF TRAINING AND TEST DATA OF THE TOP FIVE CLASSES IN A SUBSET OF YOMIURI NEWS CORPUS

#	Category	#data	
		Training	Test
C_1	Crime News	103607	24374
C_2	Sport News	79726	17610
C_3	Asian-Pacific News	41374	5943
C_4	North-South American News	36275	6109
C_5	Health News	35932	7004
	Total	296914	61040

TABLE III

EIGHT DIFFERENT WAYS OF PARTITIONING THE TRAINING DATA SET OF THE TEXT CATEGORIZATION PROBLEM, HERE γ IS SET TO 0.5

Case	ρ	Number of subsets for each class					#classifiers
		N_1	N_2	N_3	N_4	N_5	
$\mathcal{A}_1$	10000	21	16	8	7	7	1311
$\mathcal{A}_2$	20000	10	8	4	4	4	344
$\mathcal{A}_3$	30000	7	5	3	2	2	135
$\mathcal{A}_4$	40000	5	4	2	2	2	86
$\mathcal{A}_5$	50000	4	3	2	1	1	45
$\mathcal{A}_6$	60000	3	3	1	1	1	30
$\mathcal{A}_7$	70000	3	2	1	1	1	24
$\mathcal{A}_8$	90000	2	2	1	1	1	19

A five-class text categorization problem shown in Table II is decomposed into a series of two-class subproblems following eight different ways. The number of subsets for each of the classes is shown in Table III. By using (6), we calculate the total number of classifiers (see the right column of Table III) for constructing M^3-SVMs.

After training all of the individual SVMs assigned to the two-class subproblems, we use two combination strategies to integrate the trained individual SVMs into a M^3-SVM: 1) pure min-max combination and 2) min-max-vote combination. In pure min-max combination, all of the trained SVMs are integrated according to the module combination procedure mentioned in Section III. For min-max-vote combination, the output of the $K(K-1)/2$ SVMs is determined by the highest

TABLE IV

NINETEEN TWO-CLASS SUBPROBLEMS OBTAINED BY DIVIDING THE FIVE-CLASS TEXT CATEGORIZATION PROBLEM

#	Task	#Data		#	Task	#Data	
		Positive	Negative			Positive	Negative
1	$T_{12}^{(1,1)}$	51804	39863	11	$T_{23}^{(1,1)}$	39863	41374
2	$T_{12}^{(1,2)}$	51804	39863	12	$T_{23}^{(2,1)}$	39863	41374
3	$T_{12}^{(2,1)}$	51803	39863	13	$T_{24}^{(1,1)}$	39863	36275
4	$T_{12}^{(2,2)}$	51803	39863	14	$T_{24}^{(2,1)}$	39863	36275
5	$T_{13}^{(1,1)}$	51804	41374	15	$T_{25}^{(1,1)}$	39863	35932
6	$T_{13}^{(2,1)}$	51803	41374	16	$T_{25}^{(2,1)}$	39863	35932
7	$T_{14}^{(1,1)}$	51804	36275	17	$T_{34}^{(1,1)}$	41374	36275
8	$T_{14}^{(2,1)}$	51803	36275	18	$T_{35}^{(1,1)}$	41374	35932
9	$T_{15}^{(1,1)}$	51804	35932	19	$T_{45}^{(1,1)}$	36275	35932
10	$T_{15}^{(2,1)}$	51803	35932				

number of votes, instead of the MIN unit as in the pure min-max combination. Performance comparison of the min-max-vote method with the pure min-max method is shown in Table V. Here, 'single' and 'multiple' denote that desired outputs are represented in single label and multiple label, respectively. From this table, one can see that the performance of the min-max-vote method is superior to that of the pure min-max method for all of the cases. From Table V, one can also see that case $\mathcal{A}_8$ has the best performance among eight cases. The M^3-SVM for case $\mathcal{A}_8$ is depicted in Fig. 1.

TABLE V

CORRECT RECOGNITION RATE (%) AND F_1 MEASURE ON TEST DATA OBTAINED BY THE PURE MIN-MAX AND MIN-MAX-VOTE COMBINATION STRATEGIES

Case	Min-max			Min-max-vote		
	Single	Multiple	F_1	Single	Multiple	F_1
$\mathcal{A}_1$	79.33	82.00	0.7776	79.48	82.16	0.7808
$\mathcal{A}_2$	81.27	83.87	0.7842	81.39	83.99	0.7853
$\mathcal{A}_3$	80.55	83.14	0.7938	80.64	83.24	0.7966
$\mathcal{A}_4$	81.77	84.20	0.8055	81.86	84.29	0.8046
$\mathcal{A}_5$	80.71	83.22	0.7959	80.79	83.30	0.7990
$\mathcal{A}_6$	81.19	83.39	0.7972	81.25	83.45	0.7961
$\mathcal{A}_7$	82.25	84.60	0.8095	82.30	84.65	0.8087
$\mathcal{A}_8$	**83.90**	**86.84**	**0.8450**	**83.95**	**86.89**	**0.8458**

To compare the performance of the proposed part-versus-part method with the existing pairwise-classification approach, the text categorization problem was learned by both M^3-SVMs and standard SVMs. The simulation results are shown in Table VI. From Table VI, one can see that the part-versus-part method can obtain better generalization performance than the pairwise-classification approach when the original problem is decomposed into nineteen two-class subproblems, and meanwhile the training time can be reduced. The results in

TABLE VI

PERFORMANCE COMPARISON OF M^3-SVMS WITH STANDARD SVMS

Method	# Classifiers	CPU time (h.)		Speedup		Correct rate (%)
		Max	Total	Parallel	Serial	
SVM	10	8.36	55.9	-	-	86.62
M^3-SVM	19	3.15	45.7	2.6	1.2	**86.89**
	24	2.93	44.2	2.9	1.3	84.65
	30	1.86	28.2	4.5	**2.0**	83.45
	45	1.05	47.3	8.0	1.2	83.30
	86	0.62	43.9	13.5	1.3	84.29
	135	0.53	**38.0**	15.8	1.5	83.24
	344	0.21	44.1	39.9	1.3	83.99
	1311	**0.05**	51.5	**167.4**	1.1	82.16

Table VI also indicate that even though all of the individual SVMs were trained in serial, the part-versus-part method is still faster than the pairwise-classification approach for all eight cases.

The drawback of the part-versus-part method is that more number of support vectors are required in comparison with the pairwise-classification approach. Table VII shows the number of support vectors for each of nineteen classifiers. The total number of support vectors is 587,333, while the total number of support vectors for the pairwise-classification approach is 414,955. Whether the number of support vectors for the part-versus-part method can be reduced is an open problem as to future work.

TABLE VII

NUMBER OF SUPPORT VECTORS FOR EACH OF NINETEEN CLASSIFIERS GENERATED BY THE PART-VERSUS-PART METHOD

#	Task	#Support vectors	#	Task	#Support vectors
1	$T_{12}^{(1,1)}$	36709	11	$T_{23}^{(1,1)}$	33150
2	$T_{12}^{(1,2)}$	37086	12	$T_{23}^{(2,1)}$	30998
3	$T_{12}^{(2,1)}$	31310	13	$T_{24}^{(1,1)}$	30484
4	$T_{12}^{(2,2)}$	29905	14	$T_{24}^{(2,1)}$	27826
5	$T_{13}^{(1,1)}$	24669	15	$T_{25}^{(1,1)}$	23080
6	$T_{13}^{(2,1)}$	34193	16	$T_{25}^{(2,1)}$	29098
7	$T_{14}^{(1,1)}$	29828	17	$T_{34}^{(1,1)}$	24694
8	$T_{14}^{(2,1)}$	37885	18	$T_{35}^{(1,1)}$	30427
9	$T_{15}^{(1,1)}$	32175	19	$T_{45}^{(1,1)}$	31385
10	$T_{15}^{(2,1)}$	32431		Total	587333

V. CONCLUSIONS

We have presented a general task decomposition method for training multi-class support vector machines. The advantages of the proposed method over existing approaches are its parallelism and scalability. We have demonstrated that this method is superior to the pairwise-classification approach in both training time and generalization performance. By using

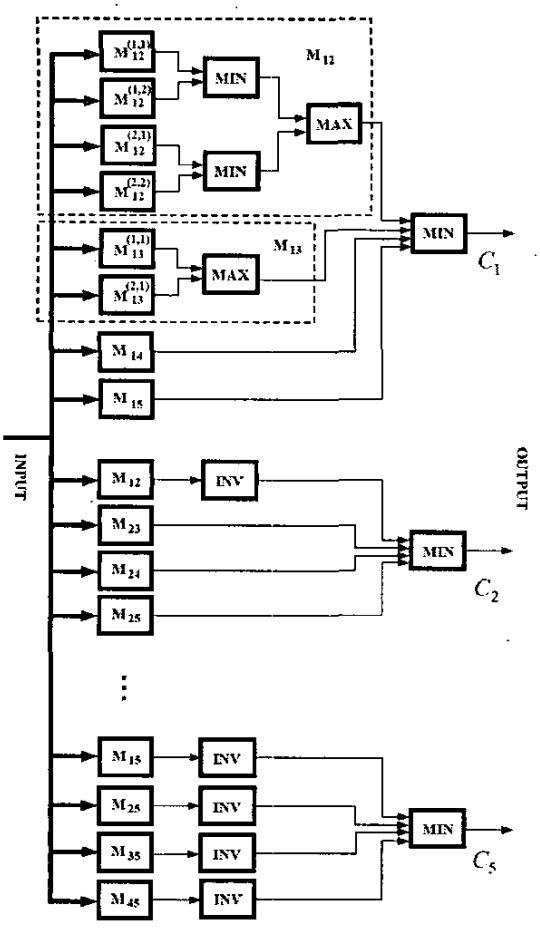

Fig. 1. The M^3-SVM for case $\mathcal{A}_8$ in which a five-class text categorization problem is decomposed into nineteen two-class subproblems. Note that $M_{ij}^{(u,v)}$ denotes the corresponding SVM module for subproblem $\mathcal{T}_{ij}^{(u,v)}$; thin lines and arrows represent scalar inputs or outputs and thick lines and arrows represent vector inputs. Due to space requirements, note that only module M_{12} and module M_{13} are plotted in detail, and the other modules are roughly illustrated.

the proposed method, we have began performing simulations on the whole Yomiuri News Corpus. A future work is to implement the method in a grid computing system and apply the method to patent classification.

ACKNOWLEDGMENT

The authors thank Ms. Hong Shen for the help on preprocessing the training and test data sets. Bao-Liang Lu was supported in part by the National Natural Science Foundation of China via the grant NSFC 60375022.

REFERENCES

[1] L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, L. D. Jackel, Y. LeCun, U. A. Muller, E. Sackinger, P. Simard, and V. Vapnik, "Comparison of classifier methods: a case study in handwritten digit recognition", *Proc. of the 12th International Conference on pattern recognition*, pp. 77-87, IEEE Computer Society Press, 1994.

[2] C. Cortes and V. Vapinik, "Support-vector network", *Machine Learning*, vol. 20, pp. 273-297, 1995.

[3] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure* , 1998, Morgan Kaufmann.

[4] J. H. Friedman, "Another approach to polychotomous classification", *Technical Report*, (ftp://stat.stanford.edu/pub/friedman/poly.ps.Z), Stanford University, 1996.

[5] C. W. Hsu and C. J. Lin, "A comparison of methods for multi-class support vector machines", *IEEE Trans. Neural Networks*, vol. 17, no. 5, pp. 14-20, 2003.

[6] T. Joachims, "Making lare-scale SVM learning practical", In B. Scholkof, C. J. C. Burges, and A. J. Smola, editors, *Advance in Kernal Methods - Support Vector Machines*, Cambridge, MA, 1998, MIT Press.

[7] S. Knerr, L. Personnaz, and G. Dreyfus, "Single-layer learning revisited: a stepwise procedure for building and training a neural network", In J. Fogelman, editor, *Neurocomputing: Algorithms, Achitectures and Applications*. Springer-Verlag, 1990.

[8] U. Kreβel, "Pairwise classification and support vector machines", In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernal Methods – Support Vector Learning*, pp. 255-268, Cambridge, MA, 1999, MIT Press.

[9] B. L. Lu and M. Ito, "Task decomposition based on class relations: a modular neural network architecture for pattern classification", *Biological and Artificial Computation: From Neuroscience to Technology, Lecture Notes in Computer Science*, J. Mira, R. Moreno-Diaz and J. Cabestany, Eds., vol. 1240, pp. 330-339, Springer, 1997.

[10] B. L. Lu and M. Ito, "Task decomposition and module combination based on class relations: a modular neural network for pattern classification", *IEEE Trans. Neural Networks*, vol. 10, no. 5, pp. 1244-1256, 1999.

[11] B. L. Lu and M. Ichikawa, "Emergence of learning: an approach to coping with NP-complete problems in learning", in *Proc. of IJCNN'2000*, vol. IV, pp. 159-164, Como, Italy, 24-27 July, 2000.

[12] B. L. Lu, Q. Ma, M. Ichikawa, and H. Isahara, "Efficient Part-of-Speech Tagging with a Min-Max Modular Neural-Network Model", *Applied Intelligence*, vol. 19, pp. 65-81, 2003.

[13] B. L. Lu, J. Shin, and M. Ichikawa, "Massively parallel classification of single-trial EEG signals using a min-max modular neural network", *IEEE Trans. Biomedical Engineering*, vol. 51, no. 3 (in press), 2004.

[14] Y. Yang and J. Pedersen, "A comparative study on feature selection in text categorization", In J. D. H. Fisher, editor, *The 14th International Conference on Machine Learning*, pp. 412-420. Morgan Kaufmann, 1997.

[15] M. Utiyama and H. Isahara, "Large-scale text categorization" (in Japanese), *Proc. of 9th Annual Meeting of the Association (Japan) for Natural Language Processing*, pp. 385-388, 2003.

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

[17] C. J. Lin and J. J. MOre, "Newton's method for large-scale bound constrained problems", in *SLAM Journal on Optimization* vol. 9 pp. 1100-1127, 1999.

[18] Y. J. Lee and O. L. Mangasarian. RSM:Reduced support vector machines. In *Proceedings of the First SIMA International Conference on Data Ming*, 2001.