Large-Scale Patent Classification with Min-Max Modular Support Vector Machines

Xiao-Lei Chu, Chao Ma, Jing Li, Bao-Liang Lu* *Senior Member, IEEE*, Masao Utiyama, and Hitoshi Isahara

Abstract— **Patent classification is a large-scale, hierarchical, imbalanced, multi-label problem. The number of samples in a real-world patent classification typically exceeds one million, and this number increases every year. An effective patent classifier must be able to deal with this situation. This paper discusses the use of min-max modular support vector machine (M^3-SVM) to deal with large-scale patent classification problems. The method includes three steps: decomposing a large-scale and imbalanced patent classification problem into a group of relatively smaller and more balanced two-class subproblems which are independent of each other, learning these subproblems using support vector machines (SVMs) in parallel, and combining all of the trained SVMs according to the minimization and the maximization rules. M^3-SVM has two attractive features which are urgently needed to deal with large-scale patent classification problems. First, it can be realized in a massively parallel form. Second, it can be built up incrementally. Results from experiments using the NTCIR-5 patent data set, which contains more than two million patents, have confirmed these two attractive features, and demonstrate that M^3-SVM outperforms conventional SVMs in terms of both training time and generalization performance.**

I. Introduction

CURRENT patent classification mainly relies on human experts. The whole process is inefficient and imprecise. Automatic classification systems based on machine learning techniques can greatly reduce the workload, and human experts could provide a further breakdown, if needed, of automatic classification. Moreover, patent classification is a fundamental of patent analysis. Through the analysis of competitors' patents, valuable information pertaining to market strategy, product development direction, and so on can be obtained. Related patents can also be mined for specific information such as the identity of technical leaders and the key technologies in a field.

Because of its great importance, automatic patent classification has received much attention [1], [2], [3], [4], [5], [6]. The European Patent Office (EPO) tried a variety of preprocessing methods on patent data, such as assigning different weights to patent sections and, utilizing the co-citation between patents. They also redefined the evaluation standards for patent classification, and pointed out that precision levels of the order of 80% are required for practical usage [1]. Larkey [2] [3] designed a system for searching and classifying U.S. patent documents based on query, and used a K-nearest-neighbor algorithm in this system because of its scalability to larger data sets. The Japan Patent Office [4] developed a patent classification system based on keywords, and used about 310,000 patents in the training procedure. They achieved 96% classification accuracy in 38 categories and 82.5% classification accuracy in 2,815 subcategories. Their experiments also indicated that in order to ensure higher classification accuracy, each subcategory should contain at least 1,000 training samples. Fall et. al. compared the most commonly used classifiers, such as naive Bayes, k-NN, support vector machines (SVM), neural networks, and decision rules, and reported that SVM provides the best performance [5], [6].

The above research mainly dealt with small data sets, but real-world patent classification problems may contain more than a million samples. The time and space complexities of SVM limit its application in such large-scale patent classification.

In this paper, we use min-max modular support vector machines (M^3-SVM) [7], [8], [9], [10] to tackle large-scale patent classification problems. M^3-SVM includes three steps: 1) a large-scale pattern classification problem is divided into a number of smaller and more balanced subproblems according to decomposition strategies; 2) all of the subproblems are solved by support vector machines in a massively parallel way; 3) all of the trained support vector machines are combined according to minimization and maximization rules [11]. To speed up training and maintain the generalization performance of M^3-SVM for patent classification, a prior-knowledge-based task decomposition strategy is used. We also analyze the incremental learning ability of M^3-SVM. Results from experiments using the NTCIR-5 database [12] demonstrate that M^3-SVM could surpass traditional SVM in terms of both classification accuracy and training time. We have also found that if a newly introduced training data set is incrementally learned, the training time can be greatly shortened.

The rest of the paper is organized as follows. Section II provides background knowledge regarding patents and their classification. In Section III, the min-max modular network model is briefly introduced, and the incremental learning ability of M^3-SVM is analyzed. In Section IV, the experiments using a large-scale patent data set are described and experimental results are presented in detail. The conclusion and future work are presented in Section V.

Xiao-Lei Chu, Chao Ma, Jing Li and Bao-Liang Lu are with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, 800 Dong Chuan Rd., Shanghai, 200240 China. Masao Utiyama and Hitoshi Isahara are with the Knowledge Creating Communication Research Center, National Institute of Information and Communications Technology, 3-5 Hilaridai, Seika-cho, Soraku-gun, Kyoto, 619-0289 Japan. *Corresponding author (email: blu@cs.sjtu.edu.cn).

978-1-4244-1821-3/08/$25.00©2008 IEEE

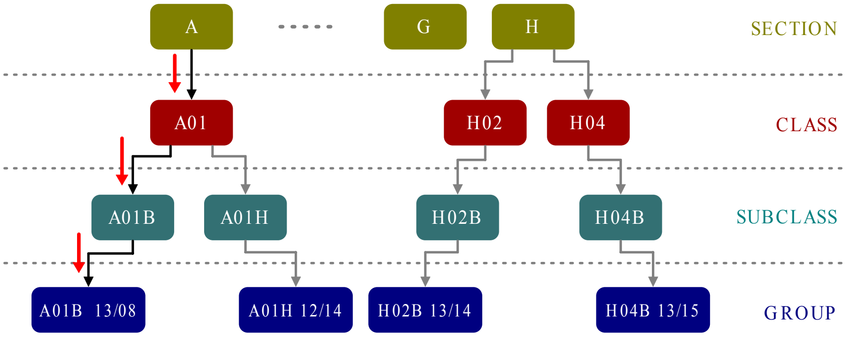
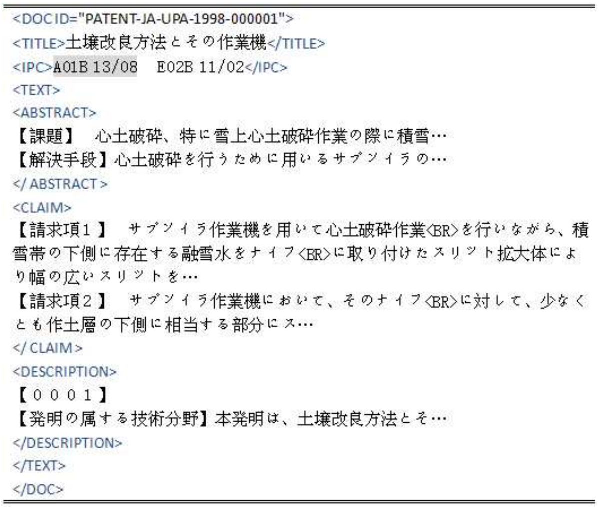


Fig. 1. A sample of IPC taxonomy. Here 'A' is the SECTION category label, 'A01' is the CLASS category label, 'A01B' is the SUBCLASS category label, and 'A01B 13/08' is the GROUP category label.

II. Patent Classification Background

The International Patent Classification (IPC) provides a common classification for patents and inventions, including published patent applications, utility models and certifications. The IPC is a hierarchically structured system including SECTION, CLASS, SUBCLASS, and GROUP layers as illustrated in Fig. 1. The SECTION layer is the top layer, and it has eight categories denoted by letters from A to H. The CLASS layer is under the SECTION layer and has 120 categories. The CLASS layer is expressed as a SECTION label followed by two digits; for example, 'A01'. The third layer is the SUBCLASS layer, which is represented as a CLASS label followed by a capital letter; for example, 'A01B'. The number of categories in the SUBCLASS layer is 630. SUBCLASS can be further divided into GROUP, but in general current research has mainly concentrated on the top three layers since the definitions of layers below SUBCLASS are still frequently changed.

Patent documents are generally stored in XML format as shown in Fig. 2. A patent is composed of three main sections: *Abstract*, *Claim*, and *Description*, and other descriptive information such as *Title* and *IPC*.



```
<DOCID="PATENT-JA-UPA-1998-000001">
<TITLE>土壌改良方法とその作業機</TITLE>
<IPC>A01B 13/08   E02B 11/02</IPC>
<TEXT>
<ABSTRACT>
【課題】　心土破砕、特に雪上心土破砕作業の際に積雪…
【解決手段】心土破砕を行うために用いるサブソイラの…
</ ABSTRACT>
<CLAIM>
【請求項１】　サブソイラ作業機を用いて心土破砕作業<BR>を行いながら、積
雪帯の下側に存在する融雪水をナイフ<BR>に取り付けたスリット拡大体によ
り幅の広いスリットを…
【請求項２】　サブソイラ作業機において、そのナイフ<BR>に対して、少なく
とも作土層の下側に相当する部分にス…
</ CLAIM>
<DESCRIPTION>
【０００１】
【発明の属する技術分野】本発明は、土壌改良方法とそ…
</DESCRIPTION>
</TEXT>
</DOC>
```

Fig. 2. A sample Japanese patent document selected from NTCIR-5. Here, this patent has two IPC labels, and the main IPC label is represented in Fig. 1.

III. Min-Max Modular Support Vector Machine

Before using M^3-SVM, we should divide a K-class problem into $K(K-1)/2$ two-class subproblems according to a one-against-one strategy. The work procedure of M^3-SVM consists of three steps: task decomposition, SVM training, and module combination. Figure 3 shows this idea of fine decomposition and module combination for a two-class problem. Min^{i,*} expresses the MIN unit for the ith subset of the positive class.

A. Task Decomposition

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

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

where $X_l^{(i)} \in \mathcal{X}_i$ 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.

Although the two-class subproblems defined by Eq. (1) are smaller than the original K-class problem, this partition may not be adequate for parallel learning. To speed up training, all the large and imbalanced two-class subproblems should be further divided into smaller and more balanced two-class subproblems.

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

$$\mathcal{X}_{ij} = \{X_l^{(ij)}\}_{l=1}^{L_i^{(j)}} \quad (2)$$
$$\text{for } j = 1, \cdots, N_i \text{ and } i = 1, \cdots, K,$$

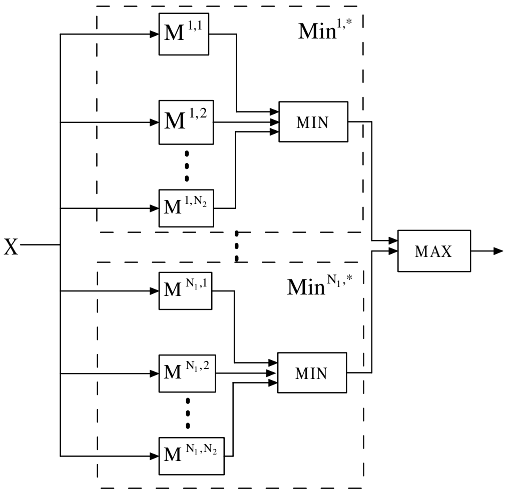


Fig. 3. Fine Decomposition and Module Composition of a Two-Class Problem. Here the two-class problem is further decomposed into $N_1 \times N_2$ subproblems. Therefore, the M^3 network consists of $N_1 \times N_2$ individual network modules, N_1 MIN units, and one MAX unit.

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

After partitioning $\mathcal{X}_i$ into N_i subsets, every two-class subproblem $\mathcal{T}_{ij}$ defined by Eq. (1) can be further divided into $N_i \times N_j$ smaller and more balanced two-class subproblems as follows:

$$\begin{aligned} \mathcal{T}_{ij}^{(u,v)} &= \{(X_l^{(iu)}, +1)\}_{l=1}^{L_i^{(u)}} \cup \{(X_l^{(jv)}, -1)\}_{l=1}^{L_j^{(v)}} \\ &\text{for } u = 1, \cdots, N_i, \ v = 1, \cdots, N_j, \\ &i = 1, \cdots, K, \text{ and } j = i+1, \cdots, K \end{aligned} \tag{3}$$

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 $\sum_{v=1}^{N_j} L_j^{(v)} = L_j$.

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 defined by Eq. (3) can be efficiently learned in a massively parallel way.

From Eqs. (1) and (3), we see that a K-class problem is divided into

$$\sum_{i=1}^{K-1} \sum_{j=i+1}^{K} N_i \times N_j \tag{4}$$

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

$$\lceil L_i/N_i \rceil + \lceil L_j/N_j \rceil \tag{5}$$

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 values of N_i and N_j.

B. Module Combination

After training, all the individual SVMs are integrated into a M^3-SVM with MIN and MAX units according to two combination principles: the minimization principle and the maximization principle [11].

Minimization Principle: Suppose a two-class problem $\mathcal{B}$ is divided into P 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

$$q(x) = \min_{i=1}^{P} \mathrm{M}_i(x) \tag{6}$$

where x denotes the input variable.

Maximization Principle: Suppose a two-class problem $\mathcal{B}$ is divided into P 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

$$q(x) = \max_{i=1}^{P} \mathrm{M}_i(x) \tag{7}$$

C. Incremental Learning

Patent classification is a typical incremental learning problem, because new patents are issued continuously. Incremental learning could be used to learn the novel knowledge from in-coming patents, and models which have been trained can be reused. M^3-SVM has such incremental learning ability if a decomposition strategy, such as prior-knowledge-based decomposition, can be used to divide each new data set independently of the learned datasets whereas random decomposition can-not be accommodated within the incremental learning architecture. Figure 4 shows M^3-SVM models after addition of a new data set. Previously trained models are reused in M^3-SVM.

First, the number of training models in the original data set can be easily calculated using

$$M^{(1)} = \sum_{0<i<j\leq K} (\lceil \frac{L_i}{a_i} \rceil \times \lceil \frac{L_j}{a_j} \rceil) \tag{8}$$

where L_i is the sample number in the *i*th category, and a_i is the sample number in each subset of the *i*th category.

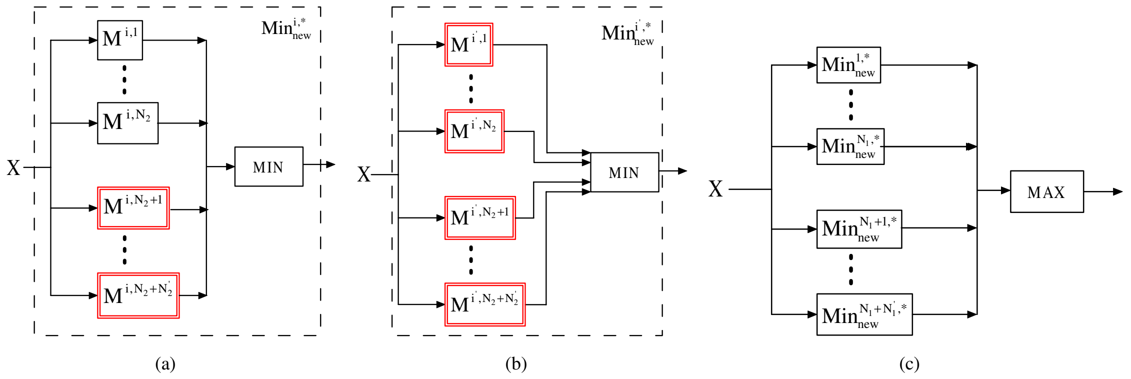


Fig. 4. Modification of modules in the case where new training data is added to existing classes. Here $N_1^{'}$ and $N_2^{'}$ are the module number of newly introduced positive and negative data sets, respectively. (a) modified old ith MIN units(Min$_{new}^{i,*}$),$i = 1, 2, \ldots, N_1$. (b) newly created $i^{'}$th MIN units(Min$_{new}^{i^{'},*}$), $i^{'} = N_1 + 1, \ldots, N_1 + N_1^{'}$ (c) Fig. 3 after incremental learning.

K denotes the category number. After the introduction of a new data set, the number of models increases to $M^{(2)}$:

$$M^{(2)} = \sum_{0<i<j\le K}((\lceil\frac{L_i}{a_i}\rceil + \lceil\frac{L_i^{'}}{a_i^{'}}\rceil) \times (\lceil\frac{L_j}{a_j}\rceil + \lceil\frac{L_j^{'}}{a_j^{'}}\rceil)) \tag{9}$$

where $L_i^{'}$ is the sample number in the *i*th category in the newly introduced data set, and $a_i^{'}$ is the sample number in each subset of the *i*th category in the newly introduced data set. Assuming the newly introduced data set is equal in sizes to the original data set, we let all a_i and $a_i^{'}$ be equal to a and all L_i and $L_i^{'}$ be equal to L. Equations (8) and (9) then become

$$M^{(1)} = \frac{K(K-1)}{2}\lceil\frac{L}{a}\rceil^2 \tag{10}$$

$$M^{(2)} = \frac{4K(K-1)}{2}\lceil\frac{L}{a}\rceil^2 \tag{11}$$

The number of models which should be incrementally learned is

$$M^{(2)} - M^{(1)} = \frac{3K(K-1)}{2}\lceil\frac{L}{a}\rceil^2 \tag{12}$$

After we introduce another data set (equal in size to the first data set),

$$M^{(3)} = \frac{9K(K-1)}{2}\lceil\frac{L}{a}\rceil^2 \tag{13}$$

The number of models which should be incrementally learned becomes

$$M^{(3)} - M^{(2)} = \frac{5K(K-1)}{2}\lceil\frac{L}{a}\rceil^2 \tag{14}$$

Thus, we can infer that after the introduction of the *i*th new data set (equal in size to the first data set),

$$M^{(i)} = i^2 W \tag{15}$$

where $W = \frac{K(K-1)}{2}\lceil\frac{L}{a}\rceil^2$ is a constant. The number of models which should be incrementally learned is

$$M^{(i)} - M^{(i-1)} = i^2 W - (i-1)^2 W = (2i-1)W \tag{16}$$

Equations (15) and (16) show that when incremental learning is used, the number of models which should be incrementally learned decreases from $O(i^2)$ to $O(i)$

This theoretical analysis has been verified by our experimental results, which are given in Section IV.

IV. Patent Classification Experiments

A. Data Set

The data set used in our experiment was collected from the NTCIR-5 patent data set [12] which follows the IPC taxonomy. There are about 350,000 new patents in per year, and new patents from a 7-year period were used in our experiment. The total number of patents is 2,399,884. We use the patents from the first five years' as training data, and those from the final two years' as test data. The number of patents in each year and in each class are listed in Table I.

We used the hierarchical text classification model to solve the patent classification problem, and focused on using M^3-SVMs to solve the large-scale problem in the SECTION layer. The SECTION layer has eight categories from A to H, and the distribution of patents in eight categories is listed in Table I. Note that a patent has one main category label and may also have several compensatory labels. Here, we simplify the multi-label problem into a unique label problem by only considering the main label of the patent.

B. Feature Selection Method

First, the raw Japanese patent was segmented using *chasen* [13], a Japanese morphological analyzer, and all auxiliary words and punctuation were removed. A patent consists mainly of three sections - *Abstract*, *Claim*, and *Description*. In our experiment, the three sections were weighted equally and wholly indexed into a single vector using the *TFIDF* method. Other researchers have proved that χ_{avg}, χ_{max} and Information Gain [14] are the top feature selection methods for text classification. We compared these three methods in our experiment. The results showed that the three methods are approximately equal in performance. We

TABLE I
NUMBER OF PATENTS PER YEAR IN EIGHT SECTIONS

SECTION	1993	1994	1995	1996	1997	1998	1999	TOTAL
A	30,583	31,316	28,357	25,444	22,475	32,427	33,126	203,728
B	65,538	68,474	68,130	68,278	62,436	68,148	69,648	470,652
C	30,747	31,834	34,163	37,996	35,700	31,198	31,494	233,132
D	4,904	5,228	5,794	6,127	5,604	4,642	4,968	37,267
E	18,605	18,000	16,114	13,690	11,099	18,604	18,810	114,922
F	30,296	31,188	29,358	28,258	26,671	31,403	32,938	210,112
G	77,692	81,691	81,677	88,716	95,679	79,158	83,942	588,555
H	72,589	72,164	72,544	81,486	86,834	75,305	80,594	541,516
TOTAL	330,954	339,895	336,137	349,995	346,498	340,885	355,520	2,399,884

also tuned the dimension parameter from 2,500 to 160,000, and found that 5,000 was the smallest value that enabled close to the top performance. Thus, we used the χ_{avg} method and a dimension size of 5000 in the experiments discussed in this paper.

C. Classifier

The M^3 framework can use support vector machine, K nearest neighbor, or neural network classifiers as the base classifier. In the experiments of other researchers, the support vector machine has proved to be the optimal classifier for most text classification problems, so we used SVM^{light} [15], [16] in our experiment as the baseline algorithm and base classifier of M^3. SVM^{light} is designed to solve large-scale text classification problems. It optimizes the speed of both the training and the testing, while also ensuring classification accuracy. Because of the outstanding capability of SVM^{light}, we considered it the most appropriate classifier for our experiment. We also used the linear kernel for SVM, because the linear kernel enables the shortest training and test times. We attempted to use other types of kernel, but the training seemed endless in the same experimental environment.

D. Decomposition Strategies

Effective task decomposition is vital for M^3-SVMs to achieve satisfactory performance [7]. We use three decomposition strategies in our experiments.

1) Random task decomposition is quite straightforward. After the subset sizes are decided, patents are randomly selected to form subsets. In our experiment, we set the subset size to 2000 based on our experience.
2) Year&Random task decomposition first divides patents into subsets according to year, and then each year's patents are further divided into subsets using the *Random Algorithm* according to the predefined subset size(2000 in our experiments).
3) Year&CLASS task decomposition first divides patents into subsets according to year, and then each year's patents are further divided into subsets according to taxonomy. The CLASS layer is used in our experiments. The CLASS number in each SECTION is shown in Table II.

TABLE II
CLASS NUMBER IN EACH IPC SECTION

SECTION	A	B	C	D	E	F	G	H
CLASS number	15	36	19	8	7	17	13	5

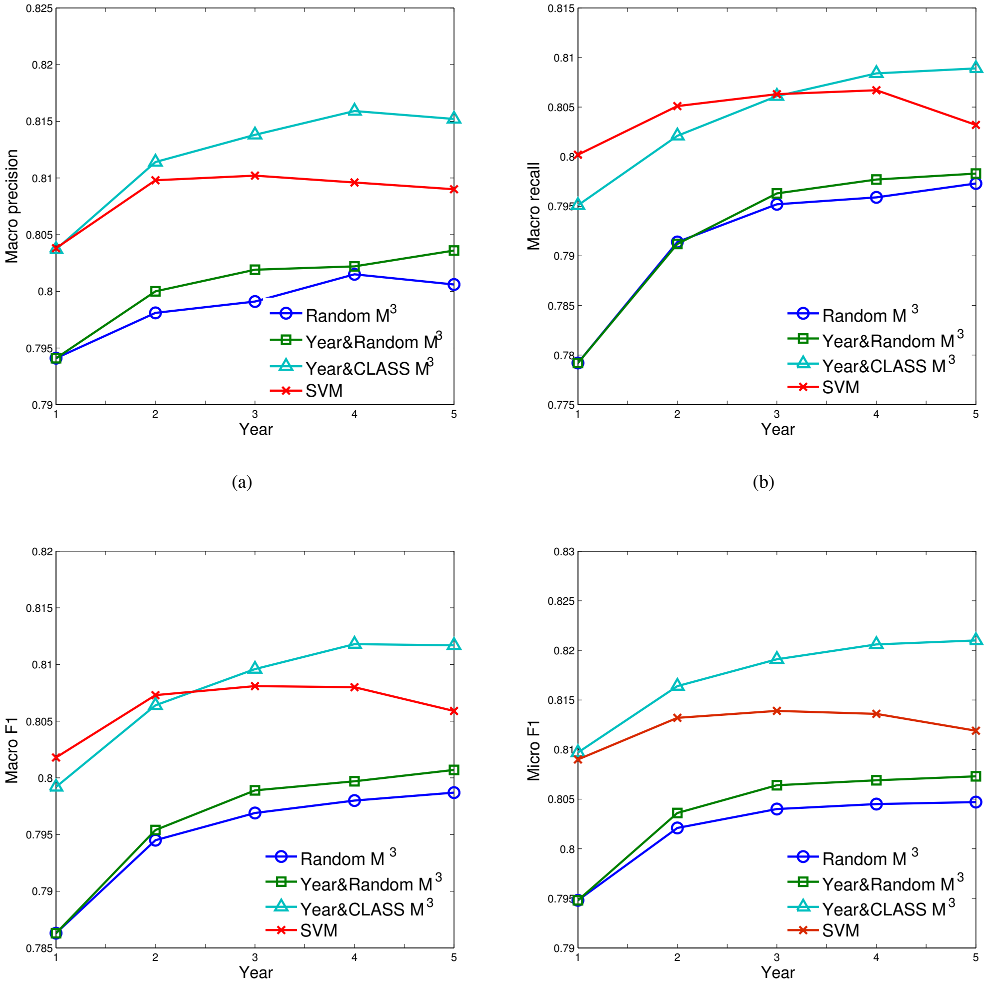
Among the three decomposition strategies, Year&CLASS decomposition provided the best performance, as shown in Fig. 5. In the following discussion, we therefore only consider M^3-SVM based on Year&CLASS decomposition.

E. Baseline Comparison

As noted, SVM^{light} performed well for most of the text classification problems, including our patent data set, in terms of both speed and classification accuracy. We compared M^3-SVM based on Year&CLASS decomposition with SVM^{light}, and found that our method could surpass even SVM^{light} in the following four regards:

1) Classification Accuracy: We use the 1998-1999 patents as a test set, and the 1997, 1996-1997, 1995-1997, 1994-1997, and 1993-1997 sets in turn as the training set . Figure 5 shows the classification results with M^3-SVM and SVM^{light}. As the training data number increased, the performance of M^3-SVM surpassed that of SVM^{light}. In addition, the performance of SVM^{light} dropped sharply with the introduction of distant years' patents, which would complicate the decision boundary. M^3-SVM can simplify the decision boundary by dividing the original complex problem into a group of much simpler subproblems.

2) Parameters Independence: The support vector machine algorithm can achieve advanced performance only when the training parameters are properly set. For large-scale classification problems, it could require several days or more to finish one round of training, so parameter tuning is unrealistic for this kind of problem. Although we can select part of the training data to use for parameter tuning, whether this subset will accurately represent the whole data set is not guaranteed. In our experiment using the SVM^{light} classifier, the performance of SVM^{light} changed dramatically along with the parameter tuning. In contrast, M^3-SVM could provide satisfactory performance regardless of parameter changes.



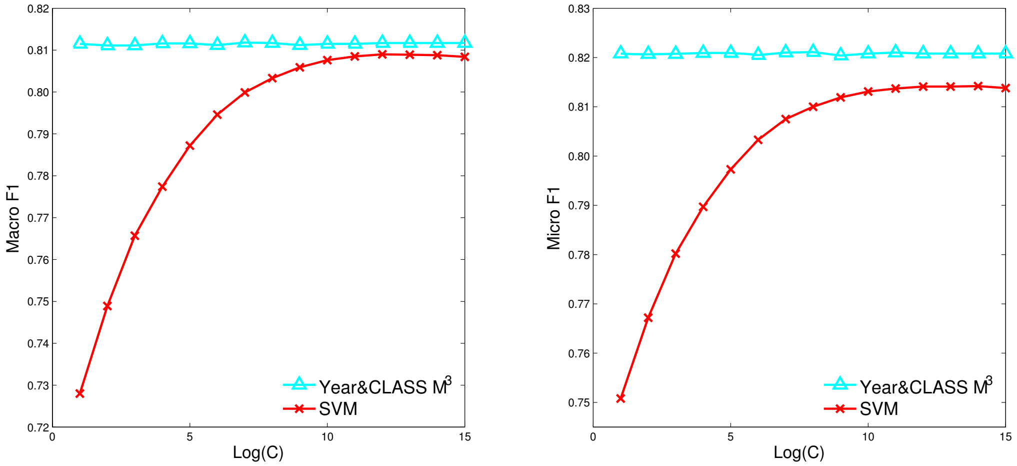
(a) (b)

(c) (d)

Fig. 5. Performance of different classifiers, (a) Macro precision, (b) Macro recall, (c) Macro F1, (d) Micro F1.

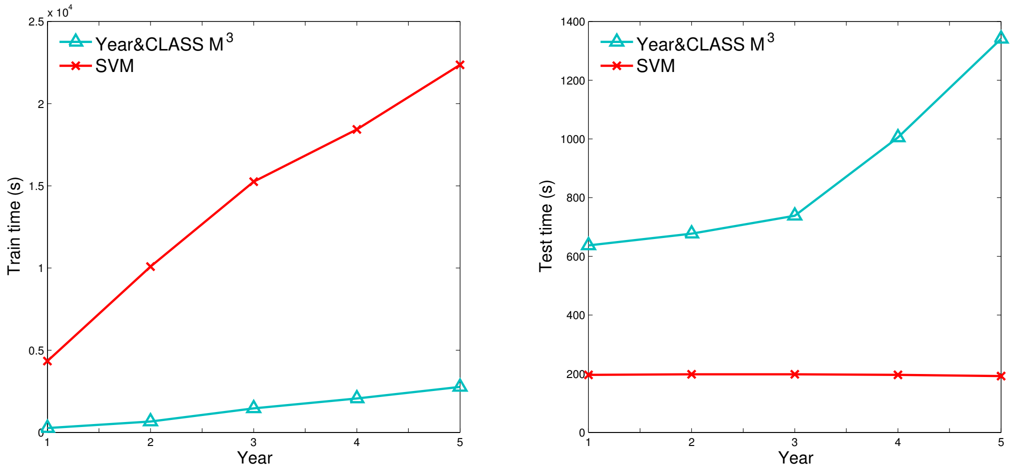
This is shown in Fig. 6, and calculated mean and standard deviation values are shown in Table III. In general, M^3-SVM provided the best performance without parameter selection, and this capability is important for large-scale problems.

3) Time Cost: We also compared the training and test times between M^3-SVM and SVM^{light}. SVM^{light} provides greatly improved training and test times compared with those of traditional SVM classifiers. Our experiments were done on a Lenovo cluster composed of three fat nodes and thirty thin nodes. Each fat node had 32G RAM and two 3.2GHz Intel(R) Xeon(TM) CPUs, while each thin node had 8G RAM and two 2.0GHz Intel(R) Xeon(TM) CPUs with each CPU having four cores. The SVM^{light} experiments were done on the fat nodes, and the M^3-SVM experiments were done on the thin nodes. The results are shown in Fig. 7. We found that although running on slower CPUs, M^3-SVM can greatly reduce the training time. However, M^3-SVM needed more testing time than SVM^{light}. For a fixed subproblem size, the M^3-SVM model number will grow with the training set size, so the test time will also increase. In the SVM^{light} test process with a linear kernel, though, only the distance between the test vectors and the dividing hyperplane needs to be calculated, which is independent of the training set size. Although the test time was greater, M^3-SVM needed only 2ms to test a patent, and can be used in real-time. Moreover, we used just 240 CPU cores to do the computation in parallel in the M^3-SVM experiment, but all the training of subproblems could be done in parallel. Using the Year&CLASS decomposition method, the original five years' of patents were divided into almost 150,000 subsets according to CLASS as shown in Table II. With enough computation nodes, the M^3-SVM training and test times



(a) (b)

Fig. 6. Performance under different training parameters: (a) Macro F1, (b) Micro F1.



(a) (b)

Fig. 7. Training and test times for M^3-SVM and SVM^{light}. (a) Training time, (b) Test time

TABLE III

MEAN AND STANDARD DEVIATION VALUES FOR M^3-SVM AND SVM^{light}

	classifier	mean	SD
Macro precision	M^3-SVMs	0.8137	0.0035
	SVM^{light}	0.7917	0.0180
Macro recall	$M^3 - SVMs$	0.8068	0.0039
	SVM^{light}	0.7747	0.0320
Macro F1	M^3-SVMs	0.8097	0.0040
	SVM^{light}	0.7818	0.0268
Micro F1	M^3-SVMs	0.8178	0.0073
	SVM^{light}	0.7906	0.0219

could be greatly reduced to $240/150,000 \approx 1/625$, much shorter than those of SVM^{light}. The scalability of M^3-SVMs is illustrated in Fig. 8.

4) Incremental Learning: We also tested the M^3-SVM incremental learning ability by successively adding one year's patents to the training set. The results are shown in Fig. 9. As we expected, the training time of incremental learning grew linearly, while the training time of no incremental learning grew quadratically.

V. CONCLUSION

We have used M^3-SVM to address a large-scale patent classification problem on the IPC top layer. M^3-SVM provides better generation performance than conventional SVM,

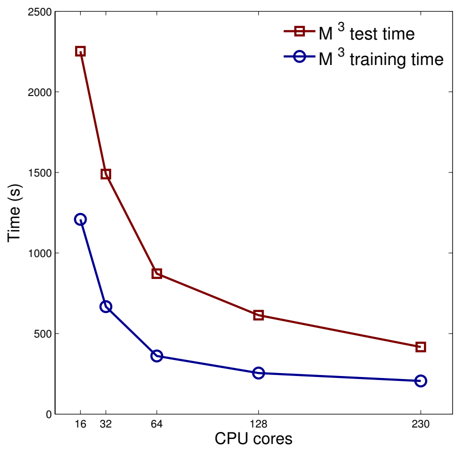


Fig. 8. Training and test times are inversely proportional to the number of CPU cores.

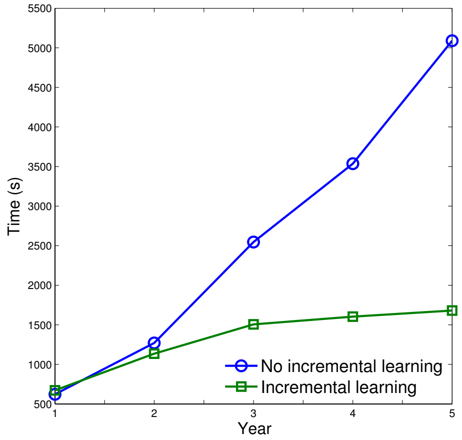


Fig. 9. Incremental learning ability of M^3-SVMs.

and more stable performance than conventional SVM with regard to parameter tuning. The training time of M^3-SVM can be greatly reduced. Experimental results showing that training and test times are inversely proportion to the number of CPU cores demonstrate the scalability of the parallel M^3-SVM system. We also verified the incremental learning ability of M^3-SVM, which is a valuable property for large-scale patent Consequently, the training time is reduced from quadratic complexity to linear complexity. In the future, the multi-label and hierarchical structure of patents will be considered with the goal of solving the patent classification problem integrally.

VI. Acknowledgements

We sincerely thank Zhifei Ye and Feng Zhou for their helpful advices, and Cong Hui for his coding help.

This work was supported in part by the National Natural Science Foundation of China via the grant NSFC 60773090, the Microsoft Laboratory for Intelligent Computing and Intelligent Systems of Shanghai Jiao Tong University, and the Okawa Foundation Research Grant.

References

[1] M. Krier and F. Zaccà, “Automatic categorisation applications at the European patent office,” *World Patent Information* Elsevier, vol. 24, pp. 187-196, 2002.

[2] L. Larkey, “Some Issues in the Automatic Classification of US Patents,” *In Learning for Text Categorization. Papers from the 1998 Workshop.* AAAI Press, Technical Report WS-98-05, pp. 87-90, 1998.

[3] L. S. Larkey, “A patent search and classification system,” *Proceedings of the fourth ACM conference on Digital libraries*, pp. 179-187, 1999.

[4] H. Mase, H. Tsuji, H. Kinukawa, M. Ishihara, “ Automatic Patents Categorization and Its Evaluation,” *Information Processing Society of Japan Journal*, vol. 39, 1998.

[5] C. J. Fall and K. Benzined, “Literature survey: Issues to be considered in the automatic classification of patents,” *World Intellectual Property Organization*, vol. 29, 2002.

[6] C. J. Fall, A. Törcsvári, K. Benzineb and G. Karetka, “Automated categorization in the international patent classification,” *ACM SIGIR Forum* ACM Press New York, NY, USA, vol. 37, pp. 10-25, 2003.

[7] B. L. Lu, K. A. Wang, M. Utiyama, and H. Isahara, “A part-versus-part method for massively parallel training of support vector machines,” *Proceedings of IEEE/INNS Int. Joint Conf. on Neural Networks (IJCNN2004)*, pp. 735-740, 2004.

[8] B. L. Lu, J. H. Shin, M. Ichikawa, “Massively parallel classification of single-trial EEG signals using a min-max modular neural network,” *IEEE Transactions on biomedical engineering*, pp. 551-558, 2004.

[9] Y. Yang, B. L. Lu, “Prediction of protein subcellular multi-locations with a min-max modular support vector machine,” *Proceedings of the Third International Symposium on Neural Networks*, pp. 667-673, 2006.

[10] H. C. Lian and B. L. Lu, “Gender Recognition Using a Min-Max Modular Support Vector Machine,” *Proceedings of ICNC’05-FSKD’05, LNCS 3611, Changshai, China*, pp. 433-436, 2005.

[11] 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*, vol. 10, pp. 1244–1256, 1999.

[12] M. Iwayama and A. Fujii and N. Kando, “Overview of Classification Subtask at NTCIR-5 Patent Retrieval Task,” *Proceedings of NTCIR-5 Workshop Meeting*, 2005.

[13] Y. Matsumoto, A. Kitauchi, T. Yamashita, Y. Hirano, O. Imaichi and T. Imamura, “Japanese morphological analysis system ChaSen manual,” *Nara Institute of Science and Technology Technical Report NAIST-IS-TR*, vol. 97007, pp. 232–237, 1997.

[14] Y. Yang and J. O. Pedersen, “A comparative study on feature selection in text categorization,” *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pp. 412-420, 1997.

[15] T. Joachims, “Making large-Scale SVM Learning Practical,” *In B. Scoelkopf, C. Burges, A. Smola, editor, Advances in Kernel Methods-Support Vector Learning.* MIT Press, 1999.

[16] T. Joachims, “SVMLight: Support Vector Machine,” *Software available from http://svmlight. joachims. org*, 1999.