Parallel Learning of Large-scale Multi-Label Classification Problems with Min-Max Modular LIBLINEAR

Yangyang Chen, Bao-Liang Lu* *Senior Member, IEEE* and Hai Zhao

***Abstract*—The study on pattern classification trends to be towards large-scale, multi-label, and imbalanced problems. The amount of the data which need to be classified is typically dozens of millions and it keeps rapid increasing in recent years. Traditional pattern classification approaches are inefficient and even ineffective in this situation. In our previous work, we proposed a min-max modular (M^3) network for dealing with large-scale and imbalanced problems. M^3-network is a generalized modular learning framework and includes three main steps: decomposing a large-scale problem into several smaller independent sub-problems, learning these sub-problems in parallel, and combining the results of the sub-problems to generate a solution to the original problem. In this paper, we embed LIBLINEAR into M^3-network (M^3-liblnear) to deal with large-scale, multi-label, and imbanlanced pattern classification problems. LIBLINEAR is a fast implementation of a linear classifier. M^3-Liblinear uses LIBLINEAR as a base classifier to learn each of the sub-problems. We compare M^3-Liblinear with Liblinear-cdblock on a large-scale Japanese patent classification problem. Experimental results demonstrate that M^3-Liblinear is superior to Liblinear-cdblock in both training time and generalization performance.**

***Index Terms*—Min-max modular network, Liblinear-cdblock, multi-label problem, imbalanced problem.**

I. Introduction

Nowadays many real-world pattern classification problems involve large-scale, multi-label, and imbalanced data sets. Traditional pattern classifiers will slow down or even be useless if the scale of the data is extremely large and exceeds the limits of the currently available hardware. On one hand, during the training, the data will have to be frequently exchanged between memory and disk and the training time will rapidly increase due to the disk being frequently read and written [14]. On the other hand, the imbalance of categories in the data will cause an extra reduction of the predicting accuracy. The performance on predicting rare classes is usually very low.

In this paper, a min-max modular (M^3) network is introduced to handle the above two difficulties. M^3-network is originally proposed to be a generalized framework to deal with large-scale and imbalanced pattern classification problems [1]. The M^3-network framework is based on the divide-and-conquer strategy. By decomposing a large-scale problem into many much smaller independent sub-problems and training all of the sub-problems in a massively parallel way, we can quickly solve large-scale problems. By selecting a proper decomposition strategy which makes the sizes of sub-problems equal, influence of the imbalance can be effectively reduced.

In this study, we embed LIBLINEAR into min-max modular network (M^3-Liblinear) to deal with large-scale, multi-label, and imbalanced problems. LIBLINEAR is a fast implementation of a linear classifier for training data sets with millions of instances and features [13]. According to the work in [13], [14], LIBLINEAR may perform better than support vector machines (SVMs) when the number of instances and features in the training data is large. M^3-Liblinear uses LIBLINEAR as a base classifier to learn each of the sub-problems.

In this paper, we evaluate the performance of M^3-Liblinear in a systemic way and compare it with another extension of LIBLINEAR, Liblinear-cdblock [13], [14]. The main advantage of M^3-Liblinear over normal LIBLINEAR and Liblinear-cdblock is that a large-scale problem can be solved more efficiently.

The rest of the paper is organized as follows. In section II, M^3-Liblinear is briefly introduced. In section III, two decomposition strategies are introduced. In section IV, we perform experiments on a binary problem and a multi-label problem, and compare M^3-Liblinear with Liblinear-cdblock. In section V, the conclusions are drawn.

II. Min-max modular Liblinear

A problem with K-class can be divided into K binary problems according to the 'one versus rest' method or be divided into $K(K-1)/2$ binary problems according to the 'one versus one' method. Consequently, a K-class problem can be conveniently converted into several binary problems. Thus we will only focus on binary classification in this section.

Min-max modular Liblinear includes three steps: task decomposition, independent or parallel Liblinear training and module combination.

A. Task Decomposition

The decomposition of M^3-Liblinear adopts a "part versus part" strategy.

Let

$$D = \{(x, y) | x \in R^d, y \in \{+1, -1\}\}$$

This work was partially supported by the National Basic Research Program of China (Grant No. 2009CB320901) and the European Union Seventh Framework Programme (Grant No. 247619).

Y. Chen, B. L. Lu and H. Zhao are with the Department of Computer Science and Engineering and MOE-Microsoft Key Laboratory for Intelligent Computing and Intelligent Systems, Shanghai Jiao Tong University, 800 Dong Chuan Rd., Shanghai 200240, China. *Corresponding author (e-mail: blu@cs.sjtu.edu.cn)

be the training data set of a binary classification, where (x, y) is the sample of D, x is the feature vector in a d-dimensional space, and y is the label of x. Since this is a binary classification, y will be either $+1$ or -1.

Let

$$D^+ = \{(x, y) \in D | y = +1\}$$

denote the positive training data set of D and

$$D^- = \{(x, y) \in D | y = -1\}$$

denote the negative training data set.

The first step of task decomposition is dividing the positive training data set D^+ into N^+ data sets D_i^+ $(i = 1, ..., N^+)$ and dividing the negative training data D^- set into N^- data sets D_j^- $(j = 1, ..., N^-)$. The detailed decomposition strategies will be discussed later. To obtain balanced sub-problems, we require $|D_i^+| \approx |D_j^-|$ for $i = 1, ..., N^+$ and $j = 1, ..., N^-$.

The second step is concatenating each D_i^+ and D_j^- to obtain a sub-problem D_{ij}:

$$D_{ij} = D_i^+ \cup D_j^-, i = 1, ..., N^+, j = 1, ..., N^-$$

Hence we get $N^+ \times N^-$ sub-problems.

B. Training Liblinear in Parallel

After the task decomposition, the training data set has been divided into $N^+ \times N^-$ sub-problems. Since the sub-problems are independent, they can be trained independently. In M^3-Liblinear, LIBLINEAR is used to solve the sub-problems. After solving the sub-problems, we obtain the following $N^+ \times N^-$ trained modules:

$$D_{ij} \xrightarrow{LIBLINEAR} M_{ij}$$

Here M_{ij} is the module trained from D_{ij}.

C. Module Combination

Two module combination principles, the minimization principle and the maximization principle, are used for M^3-Liblinear to integrate the outputs of trained modules into a solution to the original problem.

After training, we get $N^+ \times N^-$ modules. Assume that (x_t, y_t) is a test sample. Predict this sample according to each module M_{ij} and get $N^+ \times N^-$ predictions (p_{ij} for $i = 1, \cdots, N^+$ and $j = 1, \cdots, N^-$).

$$(x_t, y_t) \xrightarrow{M_{ij}} p_{ij}$$

For any $j = 1, \cdots, N^-$, prediction p_{ij} has been trained on the same positive training data set D_i^+ and on different negative training data sets. These predictions are combined according to the minimization principle.

1) Minimization principle: The predictions that come from modules which have the same positive training data sets are combined by choosing the minimum prediction as the output.

$$p_i = \min_{j=1,...,N^-} p_{ij}$$

For any $i = 1, \cdots, N^+$, prediction p_i has been made on the same negative training data set D^- and on different positive training data sets. These predictions are combined according to the maximization principle.

2) Maximization principle: The predictions that come from the modules which have the same negative training data sets are combined by choosing the maximum prediction as the output.

$$p = \max_{i=1,...,N^+} p_i$$

Figure 1 shows the structure of M^3-Liblinear.

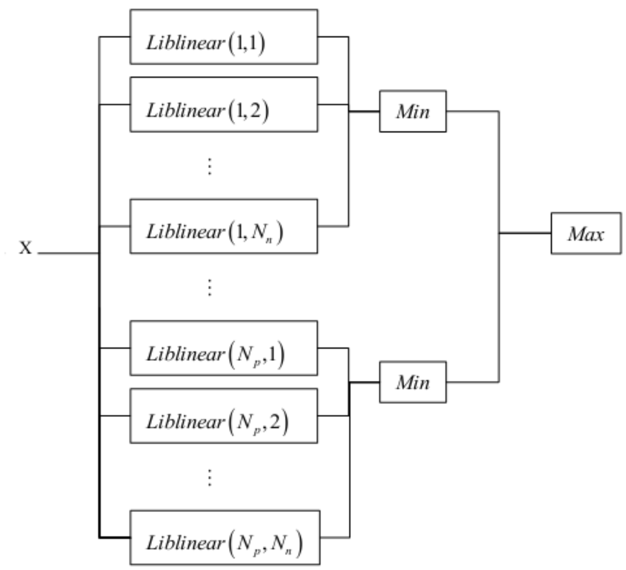


Fig. 1. The structure of M^3-Liblinear

In the situation of binary classification, the predictions belong to either positive class or negative class. Note that we have assumed that the positive label is $+1$ and the negative label is -1. The minimization principle is used to determine the negative class. If there exists a -1 in the inputs of Min unit, the output will be -1. The maximization principle is used to determine the positive class. If there exists a $+1$ in the inputs of Max unit, the output will be $+1$.

$$p_i = \begin{cases} +1 & \text{, if } p_{ij} = +1 \text{ for all } j \\ -1 & \text{, otherwise} \end{cases} \quad (1)$$

$$p = \begin{cases} +1 & \text{, if there exists a } i \text{ for which } p_i = +1 \\ -1 & \text{, otherwise} \end{cases} \quad (2)$$

Another strategy of module combination is combination using an assistant classifier [8]. This strategy is based on meta-learning [21]. The main idea is using a specified classifier instead of minimization principle and maximization principle to integrate the outputs into a solution to the original problem. In min-max combination principles, the outputs are treated as a matrix $\{p_{ij}\}$. In the strategy of module combination using an assistant classifier, the outputs are treated as a $N^+ \times N^-$-dimension vector $(p_{11}, p_{12}, p_{13}, ..., p_{N^+N^-})$. In the training phase, an assistant classifier will also be trained using the $N^+ \times N^-$ outputs of the base classifiers while training the modules. In the predicting phase, the outputs of the testing sample are treated as a $N^+ \times N^-$-dimension vector and the assistant classifier takes this vector as input to produce the final result.

III. Task Decomposition Strategies

Task decomposition strategy is important to the performance of M^3-Liblinear. Since M^3-Liblinear does not provide a specific task decomposition strategy, the decomposition strategy can be freely chosen. However, choosing different task decomposition strategies will strongly affect the performance of M^3-Liblinear. In this section, we introduce two typical task decomposition strategies.

A. Random Task Decomposition

Random task decomposition is the simplest and most straightforward strategy. Assume that the task will be divided into N sub-problems. Then for each sample in the task, assign it randomly to one of the N sub-problems. According to the law of large numbers in probability theory, the sizes of the sub-problems will be nearly equal and the sub-problems will automatically be balanced. Random task decomposition is easy to implement and it is efficient. This strategy is simple but it doesn't mean that this strategy will not perform well. In fact, if the size of data set is large enough, the distribution of the samples in each sub-problem will be similar to the distribution of the original problem.

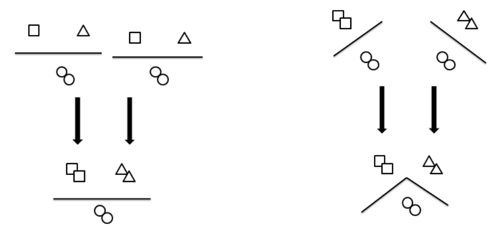
In the M^3 framework, according to the min-max combination principles, M^3-Liblinear will perform better if the samples of the same sub-problem are closer in the feature space. However, the samples of the sub-problem are usually scattered after random task decomposition so that is not always an appropriate strategy.

B. CLASS Task Decomposition

In CLASS task decomposition, the data set will be divided into subclasses. Subsets which have been divided by CLASS decomposition are closer to the reality. In our previous work [9], we have shown that the CLASS decomposition strategy is better than other methods.

Figure 2 illustrates the difference between M^3-Liblinear with random task decomposition and M^3-Liblinear with CLASS task decomposition. In the binary classification, the positive class or the negative class usually is made by several subclasses or even hierarchical subclasses. A binary problem converted from a multi-class problem by the "one versus rest" method is such an example. Generally, the distance between a subclass of the positive class and a subclass of the negative class is longer than the distance between the positive class and the negative class. So the performance of a linear classifier between a subclass of the positive class and a subclass of the negative class is better than the same linear classifier between the positive class and the negative class. In random task decomposition, the module is an epitome of the original data set. As shown in Fig. 2(a), the base classifiers are similar and the combination of them is still a linear classifier which will not perform well near the boundary of the hyperplane.

Figure 2(b) shows how M^3-Liblinear with CLASS task decomposition works. The positive class has a subclass circle



(a) M^3-Liblinear with random task decomposition (b) M^3-Liblinear with CLASS task decomposition

Fig. 2. Examples to show the difference between random task decomposition and CLASS task decomposition. Assume that the circles are the positive class and the squares and the triangles are subclasses of the negative class.

and the negative class has subclasses square and triangle. CLASS task decomposition divides this problem into two sub-problems (one between circle and square and the other one between circle and triangle). Train these sub-problems and we obtain two linear base classifiers. The functions of these base classifiers are different (one is used to separate the circles and the squares and the is other used to separate the circles and the triangles) and so are their hyper-planes. The combination is a surface which tends to encircle the positive class.

A problem of CLASS task decomposition is that it is hard to control the size of the module accurately. Hence the sub-problem may still be an imbalanced problem. By merging the small subclasses which belong to the same subclass and splitting the large subclass randomly, we reduce the rate of imbalance below 2 to 1. In this situation, the influence of imbalance is small. The CLASS task decomposition strategy is described in Algorithm 1.

Algorithm 1 Algorithm for CLASS task decomposition

Input: $modulesize$, D(data set), S(a set of the subset to be split)
Output: T(a set of the split subsets)
$S \leftarrow \{D\}$
$T \leftarrow \emptyset$
while S is not empty **do**
  for each element $E \in S$ **do**
    remove E from S
    if $|E|$ is much larger than $modulesize$ **then**
      if E has subclasses **then**
        split E by taxonomy
      else
        split E randomly
      end if
      add the subclasses of E to S
    else
      add E to T
    end if
  end for
end while
merge the sets whose size are much smaller than $modulesize$

IV. Experiments

In this section, we carry out two groups of experiments to evaluate the proposed M^3-Liblinear. The data sets used for these experiments are large-scale, multi-label, imbalanced Japanese patent classification data. The first group of experiments about binary patent classification compares M^3-Liblinear with standard LIBLINEAR and demonstrates how the module size and module combination strategies affect the performance of pattern classifiers. The second group on multi-label patent classification examines the performance of M^3-Liblinear for solving multi-label problems.

A. Data Set

The data set for the experiments is collected from the NTCIR-5 patent data set [19]. As shown in Fig. 3, NTCIR-5 is a hierarchical multi-label data set. There are four layers of labels in NTCIR-5. These four layers are SECTION, CLASS, SUBCLASS, and GROUP. In our experiments, we use the SECTION layer only. The SECTION layer contains eight different labels from A to H. The distribution of samples in the SECTION layer is listed in Table I.

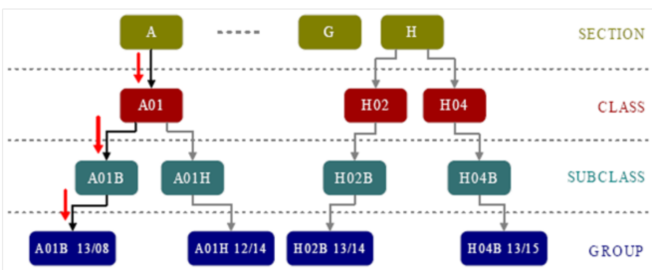


Fig. 3. Hierarchical labels of the patents

In our experiments, we use the traditional term frequency - inverse document frequency weight (tf-idf) [20] to index the text into a vector. After indexing, we have a set of vectors with 3487511 instances and 1037900 features. The data set is sparse.

B. Liblinear-cdblock

Since the size of data set is extreme large and LIBLINEAR cannot solve the problem directly, we use Liblinear-cdblock to deal with the problem instead. Liblinear-cdblock is an extension of LIBLINEAR for large data which cannot fit in memory [14]. Liblinear-cdblock is also a method based on decomposition. It splits the data set into several smaller parts and uses online learning algorithms to train each part one by one. In our experiments, Liblinear-cdblock is adopted as the baseline.

C. Binary Problem

Experiments on binary problem are divided into two parts. The first part is to demonstrate how the module size and decomposition strategies affect the performance of M^3-Liblinear. The second part is to compare the performance of M^3-Liblinear with different module combination strategies.

We use the 'one versus rest' strategy to obtain a binary data set from NTCIR-5. The samples which belong to label A are marked as positive and the other samples as negative.

We use the accuracy, precision, recall and F$_1$ score to evaluate the performance of the binary classifiers. These metrics are defined as the follows.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F_1 = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{3}$$

where TP is the true positives, FP is the false positives, TN is the true negatives, and FN is the false negatives.

We randomly take one-tenth of the samples in the binary data set as the test data set and the rest as the training data set. The training data set contains 3138356 samples and the test data set contains 349152. The number of positive samples in the training data set is 348161 and the number of negative samples is 2790195. The ratio between positive and negative samples is about 1 to 8. This problem that we examine is a large-scale and imbalanced problem. The positive class is the rare part.

In the first part of the experiments, we use the following three pattern classifiers.

1) Liblinear-cdblock: We split the training data set into 31 parts in order to make the size of each part about 100,000. As shown below, the number 100,000 gives a good tradeoff between performance and training time.

2) M^3-Liblinear with random decomposition (M^3-random)*:* We divide the data set with the module size of 100,000.

3) M^3-Liblinear with CLASS decomposition (M^3-CLASS)*:* In this method, we perform the experiments in different module sizes of 25,000, 50,000, 100,000, 400,000 and 700,000.

The experimental results are shown in Table II. From this table, we can draw the following conclusions:

- Compare the results among Liblinear-cdblock, M^3-Liblinear with random decomposition, and M^3-Liblinear with CLASS decomposition. M^3-Liblinear with CLASS decomposition gives the best performance. M^3-Liblinear with random decomposition performs almost as well as Liblinear-cdblock. M^3-Liblinear with random decomposition has a higher precision but a lower recall and their F$_1$ values are almost the same.
- Consider the training time of these three methods. Since Liblinear-cdblock trains the data set serially, it takes the longest time to finish the training process. M^3-Liblinear with random decomposition also needs more time for training than M^3-Liblinear with CLASS decomposition. This phenomenon occurs because the module is scattered after random decomposition and the classifier will take more time to reach the ending conditions in this situation.

TABLE I
THE DISTRIBUTION OF THE SECTION LAYER

	A	B	C	D	E	F	G	H
Number of samplers	387083	924773	493338	67549	206806	396991	1121361	1015251

TABLE II
RESULTS FOR BINARY PROBLEM WITH THREE METHODS

Method		Accuracy(%)	Precision(%)	Recall(%)	F_1(%)	Training Time(sec)	Predicting Time(sec)
Liblinear-cdblock (31)		96.04	82.80	81.42	82.1	25351	23
M^3-random	100000	96.38	89.79	76.22	82.45	426	63
M^3-CLASS	25000	97.44	93.39	82.87	87.81	33	116
	50000	97.30	93.32	81.64	87.09	70	74
	100000	97.09	92.91	80.01	85.98	153	54
	400000	96.67	92.08	76.72	83.70	687	51
	700000	96.47	90.57	76.32	82.84	3608	53

- As the module size decreases, the training time of M^3-Liblinear with CLASS decomposition decreases but the predicting time increases. The reason is that M^3-Liblinear needs to merge more results while predicting.

Finger 4 shows the ROC curves of Liblinear-cdblock, M^3-Liblinear with CLASS decomposition and M^3-Liblinear with random decomposition. The finger also indicates that M^3-CLASS performs best and M^3-Liblinear with random decomposition performs almost as well as Liblinear-cdblock.

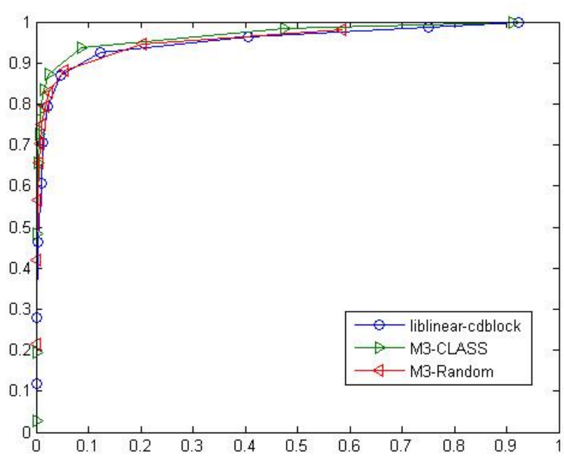


Fig. 4. The ROC curves of Liblinear-cdblock, M^3-Liblinear with CLASS decomposition and M^3-Liblinear with random decomposition

The results in Table II show that M^3-Liblinear performs better when the module size becomes smaller. So we take a small subset (one tenth) of the training data and conduct more experiments with small module size. Table III presents the results of these experiments.

From Table III, we can see that when the module size decreases from 5000 to 2500, the performance becomes worse.

The next experiment compares the performance of M^3-Liblinear with different module combination methods. Table IV shows the results. M^3-MMCP is short for M^3-Liblinear with min-max combination principle. M^3-ACMSSL is short for M^3-Liblinear with Libinear as an assistant classifier. M^3-ACMSSS is short for M^3-Liblinear with an assistant classifier using a support vector machine with a radial basis function as its kernel. The module size is 100,000. The number of positive modules is 7 and the number of negative modules is 92. Hence the input vector for the assistant classifier has 644 dimensions.

TABLE III
RESULTS FOR BINARY PROBLEM WITH DIFFERENT MODULE SIZES

Module Size	Accuracy (%)	Precision (%)	Recall (%)	F_1 (%)
2500	96.38	91.24	72.90	81.04
5000	96.60	91.47	74.97	82.40
10000	96.23	90.52	72.02	80.22
50000	95.84	89.49	68.85	77.83

As shown in Table IV, M^3-ACMSSL provides the worst result and the performance of M^3-ACMSSS is also not good. The reason for the bad performance of M^3-ACMSSL is that the classification of module combinations is not a linear classification but Liblinear is a linear classifier. So M^3-ACMSSL cannot combine the outputs correctly. As for M^3-ACMSSS, the dimension of the input vector for the assistant classifier is too large. The classification of module combinations become a very complex nonlinear classification. So M^3-ACMSSS cannot perform well.

We use a smaller training data set and try the experiments again. This time the number of positive modules is 1 and the number of negative modules is 19. Hence the input vector for assistant classifier has 19 dimensions. Table V shows the results.

TABLE V
RESULTS FOR SMALLER TRAINING DATA SET

	Acc. (%)	Precision (%)	Recall (%)	F_1 (%)
Liblinear-cdblock	87.92	45.10	63.93	52.89
M^3-MMCP	95.84	68.85	89.49	77.83
M^3-ACMSSL	81.05	35.15	93.11	51.03
M^3-ACMSSS	95.84	88.94	69.40	77.96

As the results shows, M^3-ACMSSL is also not good and M^3-ACMSSS is a little better than M^3-MMCP in the case of lower dimension.

TABLE IV
RESULTS FOR LIBLINEAR-CDBLOCK AND M^3-LIBLINEAR WITH DIFFERENT MODULE COMBINATION STRATEGIES

	Accuracy (%)	Precision (%)	Recall (%)	F_1 (%)
Liblinear-cdblock	96.04	82.80	81.42	82.1
M^3-MMCP	97.09	92.91	80.01	85.98
M^3-ACMSSL	86.63	45.08	91.34	60.36
M^3-ACMSSS	94.85	69.88	94.58	80.37

D. Multi-Label Problem

A straightforward way to deal with multi-label pattern classification problem is to train classifiers for each label [16]. We just consider the SECTION layer of NTCIR-5 and there are eight labels in the SECTION layer. So eight classifiers are necessary. Before showing the results, we introduce the evaluation metrics for multi-label problems.

In a multi-label problem, a sample can have more than one label so the output is not one label but a set of labels. Assume that the test data set is D with element (x_i, Y_i) where x_i is the sample and Y_i is the labels for x_i. Denote the output of a specified multi-label classifier as Z_i. The accuracy, precision, and recall are defined as follows.

$$Accuracy = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|} \tag{4}$$

$$Precision = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i \cap Z_i|}{|Z_i|} \tag{5}$$

$$Recall = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i \cap Z_i|}{|Y_i|} \tag{6}$$

The F_1 score is defined in the formula (3).

In this experiment, we use the accuracy, precision, recall and F_i score to evaluate the performance. The methods that we use are Liblinear-cdblock and M^3-Liblinear with CLASS decomposition. The module size is 100000. We randomly take one-tenth of the samples in the data set as testing data and the rest as training data. Then we use 'the 'one versus rest" method to convert the multi-label data set into eight binary data sets. And we train eight Liblinear-cdblock models and eight M^3-Liblinear models for the binary data sets. Table VI presents the results of these classifiers. From Table VI, we can see that the precision of M^3-Liblinear is much higher than Liblinear-cdblock and the recall is at the same level. Table VII shows the results for the multi-label problem.

TABLE VII
RESULTS OF MULTI-LABEL PROBLEM

	Acc. (%)	Precision (%)	Recall (%)	F_1 (%)
Liblinear-cdblock	58.31	71.16	76.01	73.50
M^3-Liblinear	68.86	84.34	75.77	79.82

As shown in Table VII, M^3-Liblinear outperforms Liblinear-cdblock. Generally speaking, the 'one versus rest' method results in imbalanced binary problems. So the eight binary sub-problems in this experiment are imbalanced. As shown in the binary problem experiment, the precision of M^3-Liblinear is much higher than Liblinear-cdblock in the situation of imbalanced binary problems. So in the situation where the multi-label problem is converted into binary sub-problems by 'one versus rest', the performance of M^3-Liblinear is also better.

V. CONCLUSIONS

In this paper, M^3-Liblinear is introduced to address large-scale and imbalanced multi-label problems. M^3-Liblinear provides better performance than Liblinear-cdblock for both binary problems and multi-label problems. We also compare two typical task decomposition strategies for M^3-Liblinear, CLASS task decomposition and random task decomposition. The results show that M^3 with the former decomposition strategy is more efficient and effective than the latter. By presenting M^3-Liblinear with different module sizes, we can conclude that the training time increases and the predicting time decreases as the module size increases. As future work, we will explore the method to solve the hierarchical and multi-label classification.

REFERENCES

[1] Lu, B.L. and Ito, M, "Task Decomposition Based on Class Relations: a Modular Neural N etwork Architecture for Pattern Classification," *IEEE Transactions on Neural Networks*, vol. 10, no. 5,pp 1244-1256, 1999.

[2] 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 International Joint Conference on Neural Networks*, pp. 735-740, 2004.

[3] Ken Chen, Bao-Liang Lu and Kwok, J.T, "Efficient Classification of Multi-label and Imbalanced Data using Min-Max Modular Classifiers," *Proc. of IEEE/INNS International Joint Conference on Neural Networks*, pp. 1770-1775, 2006.

[4] Fan, Z.G. and Lu, B.L, "Multi-view face recognition with min-max modular SVMs," *Lecture Notes in Computer Science*, Springer, vol. 3611, pp. 396-401, 2005.

[5] Lu B, Ma Q, Ichikawa M, et al, "Efficient Part-of-Speech Tagging with a Min-Max Modular Neural-Network Model," *Applied Intelligence*, 2003, 19(1):65-81.

[6] B. L. Lu, X. L. Wang, Y. Yang, and H. Zhao, "Learning from Imbalanced Data Sets with a Min-max Modular Support Vector Machine," *Frontiers of Electrical and Electronic Engineering in China*, vol. 6, no. 1, pp. 56-71, 2011.

[7] Z. F. Ye and B. L. Lu, "Learning Imbalanced Data Sets with a Min-Max Modular Support Vector Machine," *Proceedings of IEEE International Joint Conference on Neural Networks*, pp. 1673-1678, 2007

[8] Qi Kong, Hai Zhao and Bao-liang Lu, "Adaptive Ensemble Learning Strategy Using an Assistant Classifier for Large-Scale Imbalanced Patent Categorization," *NEURAL INFORMATION PROCESSING, THEORY AND ALGORITHMS*, Lecture Notes in Computer Science, 2010, Volume 6443/2010, 601-608.

[9] X. L. Chu, C. Ma C, J. Li, B. L. Lu, M. Utiyama and H. Isahara, "Large-scale patent classification with min-max modular support vector machines," *Proc. of IEEE International Joint Conference on Neural Networks*, vol. 1, pp. 3972-3979, HongKong, China, 2008

[10] B. L. Lu, X. L. Wang, and M. Utiyama, "Incorporating prior knowledge into learning by dividing training data," *Frontiers of Computer Science*, vol. 3, no. 1, pp. 109-122, 2009.

TABLE VI
RESULTS FOR EACH LABEL

Task	Method	Acc. (%)	Precision (%)	Recall (%)	F_1 (%)
A	Liblinear-cdblock	95.88	82.77	79.58	81.14
	M^3-Liblinear	97.09	92.91	80.01	85.98
B	Liblinear-cdblock	85.38	70.40	77.61	73.83
	M^3-Liblinear	90.52	88.28	74.17	80.61
C	Liblinear-cdblock	93.73	77.41	78.06	77.73
	M^3-Liblinear	95.60	89.47	77.78	83.22
D	Liblinear-cdblock	98.64	63.61	71.64	67.39
	M^3-Liblinear	99.22	89.56	68.17	77.41
E	Liblinear-cdblock	97.40	77.94	78.76	78.35
	M^3-Liblinear	98.18	91.44	76.58	83.35
F	Liblinear-cdblock	94.63	77.96	73.59	75.71
	M^3-Liblinear	96.07	89.19	74.44	81.15
G	Liblinear-cdblock	88.43	82.64	80.94	81.78
	M^3-Liblinear	91.90	91.63	82.27	86.70
H	Liblinear-cdblock	89.69	82.28	82.25	82.27
	M^3-Liblinear	92.55	91.84	81.59	86.42

[11] C. Ma, B. L. Lu, and M. Utiyama, "Incorporating prior knowledge into task decomposition for large-scale patent classification," *Proceedings of 6th International Symposium on Neural Networks*, LNCS 5552, pp. 784-793, 2009

[12] Haibo He and Edwardo A. Garcia, "Learning from imbalanced data," *IEEE Transactions On Knowledge And Data Engineering*, 21(9):1263C1284

[13] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *JMLR*, vol. 9, pp. 1871C1874, 2008.

[14] Hsiang-Fu Yu, Cho-Jui Hsieh, Kai-Wei Chang and Chi-Jen Lin, "Large Linear Classification When Data Cannot Fit in Memory," *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, July 25-28, 2010, Washington, DC, USA

[15] Yang Yang, Bao-Liang Lu, "Protein Subcellular Multi-Localization Prediction Using a Min-Max Modular Support Vector Machine," *International Journal of Neural Systems*, vol. 20, no. 1, pp. 13-28, 2010

[16] G. Tsoumakas and I. Katakis, Multi-label classifcation: An overview," *International Journal of Data Warehousing and Mining*, 3(3):1C13, 2007.

[17] Zhang, M.-L., Zhou,Z.-H.(2005), "A knearest neighbor based algorithm for multi-label classification," *Proceedings of the 1 st IEEE International Conference on Granular Computing*.

[18] S. Lessmann, "Solving imbalanced classifcation problems with support vector machines," *in Proc. of the Int. Conf. on Artifcial Intelligence (IC-AI'04)*, Las Vegas, Nevada, USA, June 21C24, H. Arabnia, Ed., vol. I. CSREA Press, 2004, pp. 214C220.

[19] 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.

[20] Salton, G. and M. J. McGill (1983), "Introduction to modern information retrieval," *McGraw-Hill*, ISBN 0070544840

[21] BIGGS J, "Student Approaches to Learning and Studying," *Research Mono-graph.[M].[S.l.]*: Australian Council for Educational Research Ltd., Radford House, Frederick St., Hawthorn 3122, Australia., 1987.