2006 International Joint Conference on Neural Networks
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Efficient Classification of Multi-label and Imbalanced Data Using Min-Max Modular Classifiers

Ken Chen and Bao-Liang Lu
Department of Computer Science and Engineering
Shanghai Jiao Tong University
800 Dong Chuan Rd., Shanghai 200240, China
Email: {chenkennt, bllu}@sjtu.edu.cn

James T. Kwok
Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
Email: jamesk@cs.ust.hk

***Abstract*— Many real-world applications, such as text categorization and subcellular localization of protein sequences, involve multi-label classification with imbalanced data. In this paper, we address these problems by using the min-max modular network. The min-max modular network can decompose a multi-label problem into a series of small two-class subproblems, which can then be combined by two simple principles. We also present several decomposition strategies to improve the performance of min-max modular networks. Experimental results on subcellular localization show that our method has better generalization performance than traditional SVMs in solving the multi-label and imbalanced data problems. Moreover, it is also much faster than traditional SVMs.**

I. Introduction

Many real-world applications involve multi-label classification. For example, in text categorization, a document can belong to more than one categories; and in bioinformatics, a protein may exist in more than one subcellular locations. Unfortunately, most traditional classifiers can only handle single-label problems. In recent years, some encouraging progress have been made with the development of multi-label text categorization algorithms [1] and kernel methods [2].

On the other hand, many classification problems also involve imbalanced data. For example, in subcellular localization, the "cytoplasmic", "nuclear" and "plasma membrane" classes are often much larger than the others [10]. Most learning algorithms, like neural networks and support vector machines, are designed for well-balanced data and do not work well on imbalanced data. While a classifier can achieve very high "accuracy" by simply ignoring the minority samples, this is obviously undesirable and such a classifier is useless in practice. A number of approaches have been proposed to address this imbalanced data problem. Examples include over-sampling of the minority class samples [3] and adjusting the misclassification costs of the two classes [4].

The min-max modular (M^3) network [5] is an efficient classifier for solving large-scale complex problems. This network model decomposes a large problem into a series of smaller subproblems that are independent of each other in the training phase. These subproblems can then be processed in a parallel manner, and the outputs of the subproblems are finally combined using simple principles.

In this paper, we use the M^3 network to address the multi-label and imbalanced data problems. We also propose several task decomposition strategies to improve the performance of M^3 networks. Experiments show that our method has better generalization performance, and is also much faster than traditional classifiers.

This paper is structured as follows. In section II, the min-max modular network is briefly introduced. In section III, different decomposition strategies are proposed. In section IV, we perform experiments on subcellular localization problems and compare our methods with traditional classifiers. Finally, some conclusions are drawn in section V.

II. Min-Max Modular Network

Given a K-class multi-label problem T, the training set is described as follows:

$$\mathcal{X} = \{(x_k, y_k)\}_{k=1}^{l}, \text{ and } y_k = \{y_k^m\}_{m=1}^{t_k},$$

where $x_k \in \mathbb{R}^n$ is the kth sample in the training set, y_k is the label set of x_k, y_k^m is the mth label of x_k, l denotes the number of samples in the training set, and t_k denotes the number of labels of x_k. We use the one-versus-rest decomposition method to decompose the original problem T into K two-class problems. The positive and negative samples of subproblem T_i are defined as

$$\mathcal{X}_i^+ = \{(x_k^{i+}, +1)\}_{k=1}^{l_i^+}, \text{ and } \mathcal{X}_i^- = \{(x_k^{i-}, -1)\}_{k=1}^{l_i^-},$$

where l_i^+ denotes the number of positive samples and l_i^- denotes the number of negative ones. The positive training samples of T_i are those whose label sets contain label i, and the negative samples are the remaining ones.

Now we have K two-class subproblems, each of which considers whether a sample should be assigned to a particular label. All these subproblems can be handled by traditional methods, such as neural networks or support vector machines. However, the training data may be imbalanced because the positive samples are from one class while the negative samples are from $K-1$ classes. So we will use the part-versus-part decomposition strategy [6] to divide T_i into relatively smaller and more balanced subproblems.

The positive and negative samples of subproblem T_i can be divided into N_i^+ and N_i^- smaller subsets according to

0-7803-9490-9/06/$20.00/©2006 IEEE

the part-versus-part decomposition:

$$\mathcal{X}_{ij}^{+} = \{(x_k^{ij+}, +1)\}_{k=1}^{l_{ij}^{+}}, j = 1, \ldots, N_i^{+},$$

$$\mathcal{X}_{ij}^{-} = \{(x_k^{ij-}, -1)\}_{k=1}^{l_{ij}^{-}}, j = 1, \ldots, N_i^{-},$$

where l_{ij}^{+} and l_{ij}^{-} denote the number of samples in subset $\mathcal{X}_{ij}^{+}$ and $\mathcal{X}_{ij}^{-}$, respectively. The issue on how to divide these samples will be discussed in section III. By combining the positive and negative samples of these subsets, we obtain $N_i^{+} \times N_i^{-}$ subproblems. Each subproblem T_i^{jk} has a training set of

$$\mathcal{X}_i^{jk} = \mathcal{X}_{ij}^{+} \cup \mathcal{X}_{ik}^{-}.$$

Now, the original problem is divided into a series of smaller and more balanced subproblems. Moreover, each subproblem is independent of each other. In the learning phase, each of these subproblems can be trained by a traditional learning algorithm to obtain a classifier. In the classification phase, the outputs of these classifiers are integrated by two combination principles (minimization principle and maximization principle) [5] to produce a solution to the original problem. Let $\mathcal{T}_i^{jk}(x)$ be the output of the classifier trained by $\mathcal{X}_i^{jk}$, and $\mathcal{T}_i^{j}(x)$ be the output of the classifier integrated by N^{-} classifiers with the MIN unit. Then,

$$\mathcal{T}_i(x) = \max_{j=1}^{N^{+}} \mathcal{T}_i^{j}(x),$$

and

$$\mathcal{T}_i^{j}(x) = \min_{k=1}^{N^{-}} \mathcal{T}_i^{jk}(x).$$

III. Decomposition Strategies

A key problem in the M^3 networks is how to divide samples into smaller subsets. Obviously, better performance can be obtained if samples are decomposed properly. Here, we present four decomposition strategies.

A. Random Decomposition

This strategy is simple and straightforward. We randomly divide the samples into smaller subsets, with the only constraint that these subsets must be of about the same size. This constraint is very important because we want the subproblems to be balanced. Other strategies discussed in the sequel must also follow this basic principle. The advantage of this method is that it can be easily implemented. However, it does not make use of any statistical properties of the samples or prior knowledge on the samples.

B. Hyperplane Decomposition Strategy

Ideally, the decomposition strategy should divide the samples according to the sample distribution. For example, samples that are close together in the feature space should be partitioned into the same subset. Hyperplane decomposition [7] is one of the methods that is aimed at achieving this. An illustration is shown in Fig. 1. As can be seen, the samples are clustered into three groups (as shown by the dashed circles) by using a series of parallel hyperplanes. Obviously, this cannot be achieved by random decomposition.

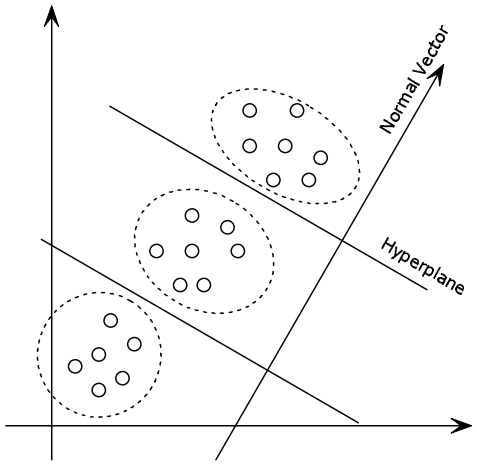

Fig. 1. Hyperplane decomposition.

On the other hand, the finding of hyperplanes that divide the samples into balanced subsets is very computationally expensive. To avoid computing the hyperplanes, we will use the normal vector of the hyperplane instead. First, we project the samples onto the direction of the normal vector by computing the dot product between each sample and the vector. Then we sort the samples according to their projections. Finally, we divide the samples equally according to the sorting result. This method can then be implemented easily.

C. PCA Hyperplane Decomposition

As an extension of the hyperplane strategy, we consider the selection of the normal vector. Since we want samples in the same subset to be close, a good idea is to choose the normal vector to be the direction of maximum variance. This direction can, in turn, be obtained by the classical method of principal component analysis (PCA) (Fig. 2).

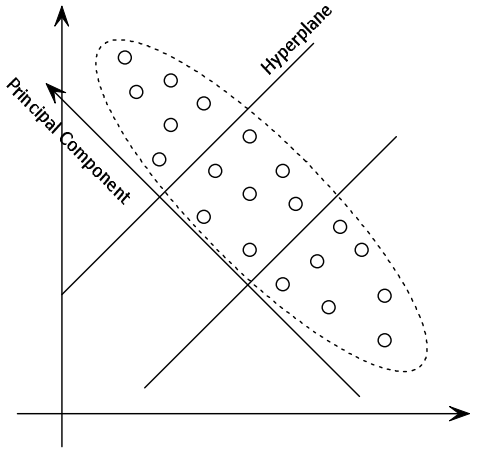

Fig. 2. PCA decomposition.

Given a sample set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ with $\mathbf{x}_i \in \mathbb{R}^n$, the sample mean is

$$\boldsymbol{\mu} = E(\mathbf{x})$$

and the sample covariance matrix is

$$\mathbf{C} = E\{(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T\}.$$

We compute the eigenvalues λ and eigenvectors $\boldsymbol{\alpha}$ of $\mathbf{C}$ by solving

$$\mathbf{C}\boldsymbol{\alpha} = \boldsymbol{\lambda}\boldsymbol{\alpha}.$$

The eigenvector with the largest eigenvalue gives the direction corresponding to the maximum variance of the samples.

D. Equal Clustering Decomposition

Another strategy is to use clustering algorithms, which group samples that are close together to the same group. However, most clustering methods cannot guarantee that the produced clusters are of about the same size, which thus violates the basic principle of task decomposition.

Equal clustering [8] is a clustering method which produces subsets of roughly the same size. Its basic idea is to adjust the size of each subset by moving the center of one subset towards (or away from) the center of another subset according to the sizes of the two subsets. So we can get the subsets with equal size even if the original data set is not balanced clustered. Its algorithm is shown in Algorithm 1.

Algorithm 1 Equal clustering.

Input:
Sample set: $\mathcal{X} = \{x_i\}_{i=1}^N$
Number of subsets: M
Learning rates: α and l
Threshold of error: ϵ
Max number of iterations: $maxiter$
Output: Clustered subsets: $\mathcal{C}_i, i = 1, \ldots, N$

```
Select M samples as the centers of the subsets C_i: C_i
for i = 1 to maxiter do
    Assign each sample to the nearest subset
    Count the number of samples in each subset: W_i
    Calculate the error function: e = max_{i=1}^{M} |W_i − W̄|
    if e < ε then
        break
    end if
    for i = 1 to M do
        Calculate δ_i = Σ_{j=1, j≠i}^{M} ((l × W_j)/(W_j + (l−1) × W_i) − 1)(C_j − C_i)
        Update C_i = C_i + α δ_i
    end for
end for
```

IV. Experiments

For performance evaluation, we apply the proposed method to an important problem in bioinformatics, namely subcellular localization of protein sequences.

A. Data Set

Protein subcellular locations are closely related to its functions. In 1994, Nakashima and Nishikawa discriminated intracellular and extracellular proteins successfully by amino acid composition and residue-pair frequencies [9]. Cai and Chou first regarded this problem as a multi-label problem [11]. The data set we use in this paper [12] is collected from the SWISS-PROT database [13]. Each protein is represented by a 20-dimensional vector by using its amino acid composition. We identified eukaryotic proteins with specific subcellular locations according to the annotation information in the database. There are 48,535 eukaryotic proteins with localization information. After using the Hobohm algorithm [14] to remove similar sequences, about 75% of the proteins are removed. The data set consists of 11,880 proteins. These proteins can be classified into 12 subcellular locations [10].

This is a typical multi-label and imbalanced data problem. About 12% of the proteins exist in more than one subcellular locations. In general, a protein can exist in at most 5 locations. Some locations, like extracellular and nuclear, are much larger than the others. The distribution of the data set is listed in Tables I and II.

TABLE I
NUMBER OF PROTEINS IN EACH LOCATION.

No.	Location	Protein No.
1	Chloroplast	823
2	Cytoplasmic	1,808
3	Cytoskeleton	126
4	Endoplasmic reticulum	255
5	Extracellular	3,429
6	Golgi apparatus	143
7	Lysosomal	62
8	Mitochondrial	1,067
9	Nuclear	3,671
10	Peroxisomal	132
11	Plasma membrane	1,697
12	Vacuolar	78
Number of labels		13,291
Number of proteins		11,880

TABLE II
NUMBER OF LOCATIONS IN EACH PROTEIN.

Number of locations	Number of proteins
1	10,570
2	1,217
3	86
4	6
5	1

Support vector machine (SVM)[1] is used as the base classifier of the M^3 network. 5-fold cross-validation is employed to measure the performance. All experiments are performed on a 3.0GHz Pentium-4 PC with 2GB RAM.

B. Evaluating the Performance of the Classifier

We use the recall (R), precision (P) and F_1 values [16] to evaluate the performance of the classifiers on a single class.

[1]We use the LIBSVM implementation [15].

Recall and precision are defined as

$$R = \frac{TP}{TP+FN},$$
$$P = \frac{TP}{TP+FP}.$$

Here TP is the true positives (correctly predicted as positive samples), FP is the false positives (incorrectly predicted as positive samples), and FN is the false negatives (incorrectly predicted as negative samples). The F_1 measure is the harmonic mean of recall and precision. It is defined as

$$F_1 = \frac{2RP}{R+P}.$$

We use the macro-average and micro-average [17] to evaluate the performance of the classifiers on all classes. Macro-average is the average of the F_1 values from all classes, while micro-average can be calculated by regarding all classes as the same class and then calculate its F_1 value.

C. Discussion on Different Module Sizes

The module size is the total number of training samples in each subproblem. It is a very important parameter in the M^3 network. To test its effect, we performed experiments on different module sizes. We compare our method with the traditional SVM. The decomposition strategy we use here is random decomposition. Each experiment is performed 5 times and the mean and standard deviation are reported. The plots of recall and precision are shown in Figs. 3 and 4 respectively. Results on the F_1 values, their micro-average, macro-average and CPU time, are shown in Table III. The number in parentheses denotes the module size.

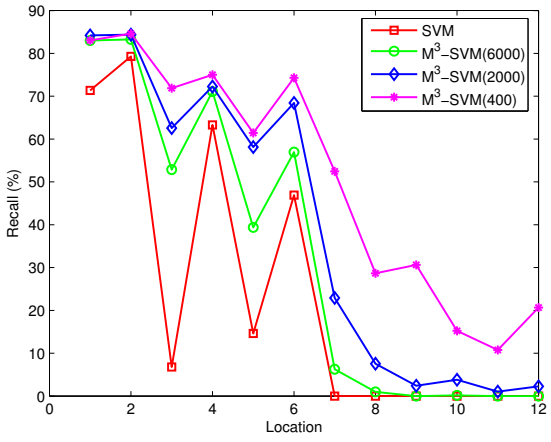

Fig. 3. Recall values for different module sizes. Different curves are for different module sizes. The x-axis shows the subcellular locations, sorted by size.

From the results, we can draw the following conclusions:

1) : Six classes have the F_1 value of zero in the traditional SVM. This means that they are ignored because they are too small. When we use task decomposition to divide the original problem into smaller subproblems, each subproblem is more balanced than the original one. So the small classes can be predicted. When the module size is 2,000, all classes can be predicted.

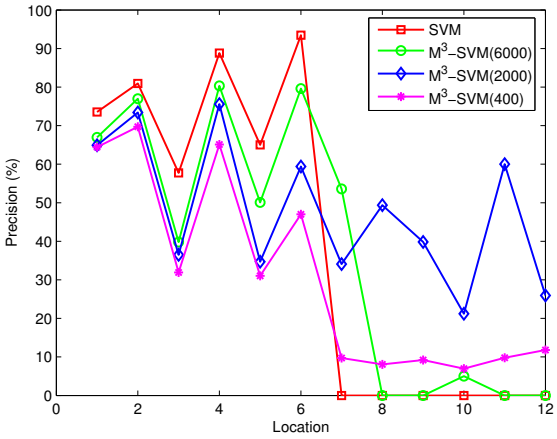

Fig. 4. Precision values for different module sizes. Different curves are for different module sizes. The x-axis shows the subcellular locations, sorted by size.

2) : The recall value of each class gets higher when the module size becomes smaller, especially for the small classes. This means that there are more true positives in the result. On the other hand, the precision value of each class decreases as the module size gets smaller, with the exception of those classes that cannot be predicted. In other words, there are more false positives in the result.

3) : The M^3-SVM is better than the traditional SVM in terms of the F_1 value, with the exception of the extracellular class. For the large classes, the best performance is usually achieved with a larger module size. For example, "nuclear" (the largest class in the sample set) has the highest F_1 value of 74.1% when the module size is 6,000. On the other hand, small classes like "vacuolar" and "lysosomal" have better performance when the module size is small. Thus, it may be a good idea to select the module size according to the size of each class.

From the macro-averaging and micro-averaging results, we can see that the M^3-SVM is better than the traditional SVM. The macro-average and micro-average of the M^3-SVM are higher than those of the traditional SVM except the micro-average at a module size of 400. The highest macro-average value can be obtained with a module size of 400, while the highest micro-average value can be obtained with a module size of 6,000.

4) : The training and testing speeds are also very important for a classifier. From the results, we can see that the M^3-SVM is much faster than the traditional SVM. This is because the original problem has been divided into a series of smaller subproblems, each of which is much simpler than the original one. It takes less time to train a simple problem than a complex one. However, as the module size gets smaller, the number of subproblems increases, and so

TABLE III

THE F_1 VALUES FOR DIFFERENT MODULE SIZES.

Location	F_1 (%) SVM	M^3-SVM (6000)	M^3-SVM (2000)	M^3-SVM (400)
Chloroplast	62.5	$\mathbf{66.4} \pm 1.1$	63.6 ± 0.2	57.6 ± 2.1
Cytoplasmic	12.2	45.4 ± 0.9	$\mathbf{46.1} \pm 0.8$	44.2 ± 0.3
Cytoskeleton	0.0	3.1 ± 1.4	6.5 ± 2.1	$\mathbf{9.5} \pm 1.1$
Endoplasmic reticulum	0.0	11.2 ± 2.8	$\mathbf{27.4} \pm 1.5$	16.4 ± 0.9
Extracellular	**80.1**	80.0 ± 0.2	78.5 ± 0.2	76.4 ± 0.4
Golgi apparatus	0.0	1.9 ± 1.5	$\mathbf{13.0} \pm 3.7$	12.6 ± 0.4
Lysosomal	0.0	0.0 ± 0.0	4.1 ± 6.6	$\mathbf{15.0} \pm 3.6$
Mitochondrial	23.9	$\mathbf{44.1} \pm 0.9$	43.5 ± 0.9	41.2 ± 0.8
Nuclear	72.4	$\mathbf{74.1} \pm 0.3$	73.3 ± 0.3	72.6 ± 0.3
Peroxisomal	0.0	0.0 ± 0.0	4.6 ± 4.1	$\mathbf{14.1} \pm 2.1$
Plasma membrane	73.9	$\mathbf{75.4} \pm 0.5$	73.9 ± 0.4	69.7 ± 0.6
Vacuolar	0.0	0.0 ± 0.0	2.0 ± 2.3	$\mathbf{10.2} \pm 2.0$
Macro-average	27.1	33.2 ± 0.3	36.4 ± 0.8	$\mathbf{36.6} \pm 0.5$
Micro-average	63.5	$\mathbf{66.7} \pm 0.3$	64.7 ± 0.2	58.4 ± 0.1
No. of modules	12	34	146	2,123
CPU time in serial (sec.)	1,624	830	561	792
CPU time in parallel (sec.)	285	118	5.76	0.75

The number in bold denotes the highest F_1 value at each location.

the CPU time in serial will also increase. But considering that these subproblems are independent of each other and can be processed in a parallel manner, the M^3-SVM will be even faster on massively parallel machines.

5) : In the above experiments, we use the same parameters (C and γ) in all the subproblems. Obviously, the classifier cannot achieve its best performance by using the same set of parameters. We perform parameter selection with a module size of 6,000 using grid search, which is simple but efficient. The parameter C is selected from the range 2^0 to 2^4, while γ is from the range 2^4 to 2^{10}. Results are shown in Table IV.

TABLE IV

RESULTS ON PARAMETER SELECTION.

Location	F_1 (%) M^3-SVM (Before)	M^3-SVM (After)
Chloroplast	66.4 ± 1.1	66.4 ± 0.7
Cytoplasmic	45.4 ± 0.9	$\mathbf{45.7} \pm 0.6$
Cytoskeleton	3.1 ± 1.4	$\mathbf{6.5} \pm 3.4$
Endoplasmic reticulum	11.2 ± 2.8	$\mathbf{25.3} \pm 1.9$
Extracellular	80.0 ± 0.2	$\mathbf{80.4} \pm 0.2$
Golgi apparatus	1.9 ± 1.5	$\mathbf{18.7} \pm 0.8$
Lysosomal	0.0 ± 0.0	$\mathbf{15.9} \pm 5.1$
Mitochondrial	44.1 ± 0.9	44.1 ± 0.9
Nuclear	74.1 ± 0.3	$\mathbf{74.4} \pm 0.3$
Peroxisomal	0.0 ± 0.0	$\mathbf{11.1} \pm 0.9$
Plasma membrane	75.4 ± 0.5	75.4 ± 0.5
Vacuolar	0.0 ± 0.0	$\mathbf{10.0} \pm 2.4$

From the results, we can see that M^3-SVM has better performance after parameter selection. For example, for the extracellular location, the traditional SVM is better than the M^3-SVM before parameter selection. Now, after parameter selection, the M^3-SVM becomes better (the traditional SVM only has a F_1 value of 80.3% after parameter selection). Thus, to achieve the best performance of M^3-SVM, it is better to use different parameters in different subproblems.

D. Comparison of Different Decomposition Strategies

We also perform experiments with different decomposition strategies. The module size is set to 6,000. The parameters for each class are adjusted for best performance. Results are shown in Table V.

From the result, we can draw the following conclusions:

1) : The random strategy has the best performance for most locations, while the other strategies are better than the random strategy for locations like "golgi apparatus", "lysosomal" and "peroxisomal". The reason may lie in the distribution of the sample set. For most situations, the random strategy is the best choice. But for others, another strategy should be used. To achieve the best performance of the M^3 network, it is better to use different strategies for different classes.

2) : Although the random strategy is better than the other strategies in most classes, it is also the slowest. Hyperplane, PCA and equal-clustering are all faster than the random strategy. The main reason is that these three strategies make use of the statistical properties of the sample set so that samples in one subset are close to each other. Thus, each subproblem becomes simpler and is easier to classify.

Unfortunately, the equal-clustering procedure is time-consuming. Hence, if we take clustering time into account, equal-clustering is slower than the other methods.

TABLE V

THE F_1 VALUES FOR DIFFERENT DECOMPOSITION STRATEGIES.

	F_1 (%)			
Location	Random	Hyperplane	PCA	Equal clustering
Chloroplast	$\mathbf{66.4} \pm 0.7$	64.0	65.7	64.5 ± 1.2
Cytoplasmic	$\mathbf{45.7} \pm 0.6$	33.9	34.1	34.0 ± 0.3
Cytoskeleton	$\mathbf{6.5} \pm 3.4$	6.1	6.3	4.0 ± 4.1
Endoplasmic reticulum	$\mathbf{25.3} \pm 1.9$	22.9	22.8	22.4 ± 4.0
Extracellular	$\mathbf{80.4} \pm 0.2$	80.1	80.2	80.0 ± 0.4
Golgi apparatus	18.7 ± 0.8	18.3	**20.5**	17.5 ± 0.5
Lysosomal	15.9 ± 5.1	12.3	**16.2**	10.2 ± 9.0
Mitochondrial	$\mathbf{44.1} \pm 0.9$	36.7	37.9	34.3 ± 2.4
Nuclear	$\mathbf{74.4} \pm 0.3$	73.5	73.7	73.0 ± 0.6
Peroxisomal	11.1 ± 0.9	**11.3**	10.8	11.1 ± 5.4
Plasma membrane	$\mathbf{75.4} \pm 0.5$	74.4	74.8	74.4 ± 0.6
Vacuolar	$\mathbf{10.0} \pm 2.4$	5.8	9.6	5.4 ± 2.2
CPU time in serial (sec.)	833	711	704	694 (167)
CPU time in parallel (sec.)	117	101	100	99 (167)

The number in parentheses denotes the clustering time.

V. CONCLUSIONS

We have used the M^3 network to address the multi-label and imbalanced data problems. We also proposed four decomposition strategies. From the results, we can see that M^3-SVM has better generalization performance than the traditional SVM. The M^3-SVM is also much faster than the traditional SVM because each subproblem is simpler, especially when the hyperplane, PCA and equal-clustering strategies are used. Moreover, in order to achieve the best performance, the module size and classifier parameters should be selected according to the properties of different classes. It is also better to use different decomposition strategies for different classes. In the future, we will analyze the performance of the M^3 network and decomposition strategies theoretically.

ACKNOWLEDGMENT

This research was partially supported by the National Natural Science Foundation of China via the grants NSFC 60375022 and NSFC 60473040. The authors thank Mr. Wei-Ming Liang for his helpful work on data preparation.

REFERENCES

[1] R. E. Schapire and Y. Singer, "Boostexter: a boosting-based system for text categorization," *Machine Learning*, vol. 39, pp. 135–168, 2000.

[2] A. Elisseeff and J. Weston, "A kernel method for multi-labelled classification," *in Advances in Neural Information Processing Systems*, 2001, pp. 681–687.

[3] N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, 16, pp. 341–378, 2002.

[4] N. Japkowicz and S. Stephen, "The class imbalance problem: a systematic study," *Intelligent Data Analysis Journal*, vol. 6, pp. 429–449, 2002.

[5] B. L. Lu and M. Ito, "Task decomposition and module combination based on class relations: a modular neural network for pattern classification," *IEEE Transactions on Neural Networks*, vol.10, pp.1244–1256, 1999.

[6] B. L. Lu, K. A. Wang, M. Utiyama and H. Isahara, "A part-versus-part method for massively parallel training of support vector machines," *Proc. IEEE International Joint Conference on Neural Networks*, Budapast, July 25-29, 2004, pp. 735–740.

[7] F. Y. Liu, K. Wu, H. Zhao and B. L. Lu, "Fast text categorization with min-max modular support vector machines," *Proc. IEEE International Joint Conference on Neural Networks*, Montreal, Quebec, Canada, July 31-Aug. 4, 2005, pp. 570–575.

[8] Y. M. Wen, B. L. Lu and H. Zhao, "Equal clustering makes min-max modular support vector machine more efficient," *Proc. 12th International Conference on Neural Information Processing*, Taipei, Taiwan, Oct. 30-Nov. 2, 2005, pp. 77–82.

[9] H. Nakashima and K. Nishikawa, "Discrimination of intracellular and extracellular proteins using amino acid composition and residue-pair frequencies," *J. Mol. Biol.*, 238, pp. 54–61, 1994.

[10] K. J. Park and M. Kanehisa, "Prediction of protein subcellular locations by support vector machines using compositions of amino acids and amino acid pairs," *Bioinformatics*, vol. 19, pp. 1656–1663, 2003.

[11] K. C. Chou and Y. D. Cai, "Prediction of protein subcellular locations by GO-FunD-PseAA predictor," *Biochemical and Biophysical Research Communications*, vol. 320, pp. 1236–1239, 2004.

[12] K. Chen, W. M. Liang and B. L. Lu, "Data analysis of SWISS-PROT database," BCMI Technical Report, BCMI-TR-0501, Shanghai Jiao Tong University, 2005.

[13] A. Bairoch and R. Apweiler, "The SWISS-PROT protein sequence data bank and its supplement TrEMBL," *Nucleic Acids Res.*, 25, pp. 31–36, 1997.

[14] U. Hobohm, M. Scharf, R. Schneider and C. Sander, "Selection of representative protein data sets," *Protein Science*, 1992 Mar, 1(3):409–417.

[15] C. C. Chang and C. J. Lin, "LIBSVM: a library for support vector machines," Software available at http://www.csie.ntu.edu.tw/˜cjlin/libsvm, 2001.

[16] D. D. Lewis, "Evaluating and optimizing autonomous text classification systems," *Proc. The 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 95)*, pp. 246–254, 1995.

[17] D. D. Lewis, "Evaluating text categorization," *Proc. Speech and Natural Language Workshop*, pp. 312–318, 1991.