Front. Electr. Electron. Eng. China 2011, 6(1): 56–71
DOI 10.1007/s11460-011-0127-1

RESEARCH ARTICLE

Bao-Liang LU, Xiao-Lin WANG, Yang YANG, Hai ZHAO

Learning from imbalanced data sets with a Min-Max modular support vector machine

© Higher Education Press and Springer-Verlag Berlin Heidelberg 2011

Abstract Imbalanced data sets have significantly unequal distributions between classes. This between-class imbalance causes conventional classification methods to favor majority classes, resulting in very low or even no detection of minority classes. A Min-Max modular support vector machine (M^3-SVM) approaches this problem by decomposing the training input sets of the majority classes into subsets of similar size and pairing them into balanced two-class classification subproblems. This approach has the merits of using general classifiers, incorporating prior knowledge into task decomposition and parallel learning. Experiments on two real-world pattern classification problems, international patent classification and protein subcellular localization, demonstrate the effectiveness of the proposed approach.

Keywords imbalanced data, Min-Max modular network (M^3-network), prior knowledge, parallel learning, support vector machine (SVM)

1 Introduction

Imbalanced data sets exist in many real-world applications, where the sizes of majority classes severely exceed those of the minor classes. For example, in international patent classification, some major classes have up to hundreds of thousands of samples while some minor classes have less than ten samples. A fundamental issue of learning from imbalanced data sets is serious performance degradation of standard learning algorithms, such as back-propagation algorithm for multi-layer perceptrons and support vector machines. Most standard algorithms assume or expect balanced class distributions or equal misclassification costs. Therefore, when serious imbalanced data sets are presented, these algorithms fail to properly represent the distribution characteristics of the data and provide predictions favorable to the majority classes.

In the last decade, imbalanced learning problem has attracted an influx of attention in the research community [1]. Several major workshops, conferences, and special issues reflect this attention, such as Association for the Advancement of Artificial Intelligence (AAAI'00) Workshop on Learning from Imbalanced Data Sets [2], the Twentieth International Conference on Machine Learning (ICML'03) Workshop on Learning from Imbalanced Data Sets [3], and ACM SIGKDD Explorations'04 [4]. Besides, the explosively growing number of publications on this topic also reflects this attention [1].

The frequently used approaches to imbalanced learning in the research community are sampling methods and cost-sensitive methods. Sampling methods handle imbalanced learning by constructing balanced training sets, while cost-sensitive methods handle imbalanced learning by making adaptive changes on extent classifiers. However, these methods are no longer adequate for large-scale data sets. In this paper, we adopt Min-Max modular support vector machine (M^3-SVM) [5], which can naturally handle imbalanced problems well and is suitable to deal with large-scale training data sets.

M^3-SVM is an extension of Min-Max modular network (M^3-network), which was proposed by Lu and Ito initially as an effort to solve complex classification problems [6,7]. The M^3-network approach consists of three

Received July 22, 2010; accepted October 18, 2010

Bao-Liang LU (✉), Hai ZHAO
Center for Brain-Like Computing and Machine Intelligence, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
MOE-Microsoft Key Laboratory for Intelligent Computing and Intelligent Systems, Shanghai Jiao Tong University, Shanghai 200240, China
E-mail: bllu@sjtu.edu.cn

Xiao-Lin WANG
Center for Brain-Like Computing and Machine Intelligence, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

Yang YANG
Department of Computer Science and Engineering, Shanghai Maritime University, Shanghai 201306, China

steps: 1) decompose the training input sets of majority classes into subsets of desired size; 2) learn the two-class subproblems formed by the subsets in parallel; 3) combine the trained component classifiers with the minimization and maximization principles. The decomposition procedure of M^3-network naturally converts the imbalanced data sets into a group of balanced binary data sets, which makes M^3-network especially suitable for imbalanced learning.

The rest of this paper is organized as follows. We first review the state-of-the-art imbalanced learning methods and compare M^3-network with them in Sect. 2. Then we formally describe M^3-network in Sect. 3. After that we apply M^3-network to two real-world tasks in Sect. 4. In the end we present the conclusions in Sect. 5.

2 Related work

In this section, we briefly review the state-of-the-art methods for imbalanced learning. A detailed review can be found at Refs. [1,8].

2.1 Sampling methods

These methods are concerned with converting an imbalanced data set into a balanced one by manipulating the samples. Two fundamental ideas are under-sampling the majority classes, which removes majority samples from the origin data set, and over-sampling the minority classes, which adds minority samples to the original data set. Studies have shown that a balanced data set provides improved overall classification performance compared to an imbalanced data set for various base classifiers [9–11].

Many researches on under-sampling have been reported. Most of them follow the strategy of selecting a representative subset of the majority class, and then combining it with the minor class to form a balanced data set. The representative subset is usually made by removing the overlapping via certain data cleaning techniques. Some typical approaches include the condensed nearest neighbor rule and Tomek links (CNN+Tomek links) integration method [12], the one-sided selection (OSS) method [13], the neighborhood cleaning rule (NCL) based on the edited nearest neighbor (ENN) [10], and the NearMiss method based on K-nearst neighbor (KNN) [14].

As for other kinds of under-sampling methods, the EasyEnsemble algorithm is to independently sample several subsets from the majority class and to use multiple classifiers based on the combination of each subset with the minority class [15].

Researches on over-sampling are relative small in numbers. The synthetic minority over-sampling technique (SMOTE) algorithm creates artificial minority samples based on the feature space similarities between the existing minority samples [16]. The cluster-based over-sampling (CBO) makes use of the K-means clustering technique [17]. The integrations of under-sampling and oversampling methods are studied by Batista *et al.*, including the integrations of SMOTE with ENN (SMOTE+ENN) and SMOTE with Tomek links (SMOTE+Tomek) [12].

In addition, the integration of sampling methods with ensemble learning techniques is widely studied in the community. Most reported work focused on the ensemble method of Adaboost. The SMOTEBoost algorithm integrates SMOTE with Adaboost.M2, which introduces synthetic sampling at each boosting iteration [18]. The DataBoost-IM algorithm combines data generation techniques with Adaboost.M1 [19]. The JOUS-Boost reduces the computational complexity of data generation in DataBoost-IM, by replacing it with adding independently and identically distributed noise to minority samples [20].

2.2 Cost-sensitive methods

This kind of method uses different cost metrics that describe the costs for misclassifying any particular sample. They can be naturally applied to imbalanced learning by associating high misclassifying cost to the minority classes [4,21,22]. Various empirical studies have shown that in some applications, cost-sensitive methods are superior to sampling methods [23–25].

Various cost-sensitive methods have been proposed by integrating cost-sensitive fitting into existent classifiers. There are lots of researches about cost-sensitive SVMs. The cost-sensitive boosting methods introduce cost items into the weight updating strategy of Adaboost, such as AdaCost [26], AdaC1, AdaC2, AdaC3 [27], CSB1, and CSB2 [28]. There are also cost-sensitive variants of neural networks [29,30], Bayesian classifiers [31–34], and decision tree [21,35].

2.3 SVM methods

Lots of studies on enhancing SVMs for imbalanced learning have been reported, since SVMs are the widely recognized start-of-the-art classifiers.

Even without any imbalanced (or cost-sensitive) fitting, SVMs can provide relatively robust classification performance on imbalanced data sets, compared to other classifiers such as linear discriminant analysis (LDA), naive Bayes and neural networks. It is because SVMs make the decision boundary by the samples near concept boundaries only rather than all the samples. Under such framework, though the majority classes have large number of samples, these samples may not actually affect the decision boundary if they are far away from the concept boundaries [36]. However, when the imbalance is severe, SVMs will possibly fail and classify all examples as the majority classes.

The direct imbalanced fitting on SVMs can take three forms. First, SVM's optimization problem (or SVM's max margin model) can directly integrate cost functions, which is adopted by the two popular implements of SVMs, SVM$^{\text{light}}$ [37,38] and LibSVM [39]. Second, the raw outputs of SVMs can be biased towards the minority class through threshold strategies such as lowering the threshold value for the minority class [40,41]. Third, the SVMs' class boundary can be adjusted by boundary alignment techniques [42–44].

The indirect imbalanced cost-sensitive fitting are those made on the ensembles of SVMs without modifying the base classifiers of SVMs. The ensemble of SVMs trained by over/undersampled methods is studied [45–48]. Besides, the aforementioned integration of sampling methods with ensemble learning such as cost-sensitive Adaboost, usually take SVMs as component classifiers in order to achieve good classification performance.

2.4 Active and one-class learning approaches

Active learning, unlike supervised learning which assumes that all of the training data are given at the start, interactively collect new examples, typically by making queries to a human user. They are traditionally used to solve problems related to unlabeled training data. Issues on active learning from imbalanced data sets have been discussed [49–52]. These variant active learning methods take the imbalance ratio of data into consideration when selecting the most informative samples from the unseen training data.

One-class learning aims to recognize instances of a class by using mainly (or only) samples of that class, rather than differentiating between positive samples and negative samples. It is mainly used for novelty detection. However, it is found to be particularly useful in dealing with extremely imbalanced data sets with high feature space dimensionality [18].

2.5 Remarks

Sampling methods and cost-sensitive methods handle, respectively, the imbalanced learning from the two basic aspects of machine learning, data sets and classifiers. Sampling methods approach the imbalanced learning by constructing balanced training sets. The advantage of this approach is the simplicity that no changes are made on the classifiers, thus general classification techniques such as SVMs including all their recent improvements can be directly incorporated. However, under-sampling methods usually explore a part of the original training set, thus the result classifier may not be trained completely and cannot obtain high classification accuracy.

On the contrary, cost-sensitive methods approach the imbalanced learning by adapting the existent classifiers or ensembling methods such as Adaboost. The advantage of this approach is the high classification performance as the training set is unchanged and fully learned. The disadvantage of this approach is that it highly depends on the original classification method and lack of generality.

Active learning and one-class learning are not supposed to handle imbalanced learning in the first place, but they do provide new perspectives to this problem. In particular, for some extremely imbalanced problems where sampling methods and cost-sensitive methods fail, one-class learning can still work well as it is totally unaffected by the sample ratio between the classes.

3 M^3-SVM

Lu and Ito proposed the Min-Max modular neural network in order to solve hard classification problems in 1997 [6]. M^3-network is a sort of ensemble learning methods that employ a group of classifiers for one classification problem. The M^3-network follows the principle of divide and conquer, dividing the whole problem into small pieces and solving them one by one, and adopts a three-step work flow (see Fig. 1): task decomposition, training component classifiers, and module combination. In 2004, Lu et al. extended M^3-network to SVM and proposed M^3-SVM [5]. Their main contribution to SVM is that a novel task decomposition strategy called part-versus-part method was developed.

3.1 An illustration

Before formally describing the M^3-network, we would like to present an illustration. Suppose that a two-class classification problem as depicted in Fig. 2(a) needs to be learned, where small red disks represent positive samples and small blue rectangles represent negative samples [53]. Though simple at first sight, it is actually a non-linearly separable problem. To demonstrate M^3-network learning, we divide the samples of each class into two subsets, surrounded by dashed lines in Fig. 2(a). According to this partition of the training data sets, we generate four smaller classification subproblems shown in Figs. 2(b) to (e). Then we train four component classifiers on these subproblems, where dashed lines represent the discriminant surfaces. After training, we

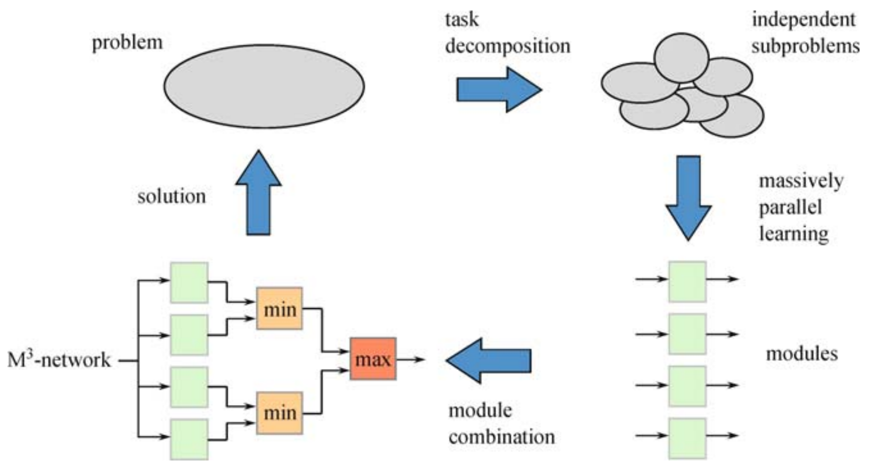

Fig. 1 Working flow of M^3-network

separately obtain two middle combination results with the *minimization* operators. One of the middle results (see Fig. 2(f)) is the combination of Figs. 2(b) and (c), and the other (see Fig. 2(g)) is the combination of Figs. 2(d) and (e). From Figs. 2(f) and (g), we can see that this kind of classifier combination results in negative zone expanding. Finally, we obtain the complete result (see Fig. 2(h)) from Figs. 2(f) and (g) with the *maximization* operator, resulting in positive zone expanding. From Fig. 2(h), we can see that the correct decision boundary for the original two-class classification problem is achieved.

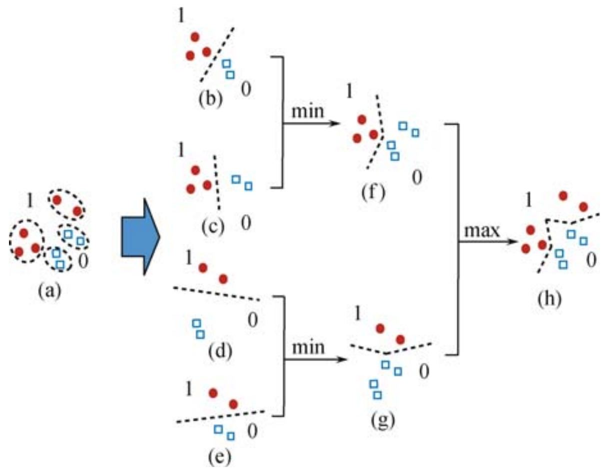

Fig. 2 Illustration of M^3-network learning

3.2 Unique task decomposition

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

$$\mathcal{T} = \{(X_i, Y_i)\}_{i=1}^{L}, \tag{1}$$

where $X_i \in R^d$ is the training input, $Y_i \in R^K$ is the corresponding desired output, and L is the total number of training samples. Without loss of generality and for simplicity of description, we consider only mono-label classification problems in this paper, in which any training input has one and only one desired output. It should be noted that our proposed method can be used to deal with multi-label classification problems.

At the beginning of the task decomposition process, a large-scale K-class classification problem can be divided into a series of relatively smaller two-class subproblems by using a one-versus-one or one-versus-rest task decomposition strategy. Suppose a K-class classification problem is divided into the following K two-class subproblems with the one-versus-rest strategy,

$$\mathcal{T}_i = \{(X_l^i,\ 1)\}_{l=1}^{L_i} \cup_{j=1, j\neq i}^{K} \{(X_l^j,\ 0)\}_{l=1}^{L_j}, \quad \text{for } i = 1, 2, \ldots, K, \tag{2}$$

where $X_l^i \in \mathcal{X}_i$ is the training input belonging to class $\mathcal{C}_i$, $\mathcal{X}_i = \{X_l^i\}_{l=1}^{L_i}$ is the set of training inputs belonging to class $\mathcal{C}_i$, L_i denotes the number of data in $\mathcal{X}_i$, $\mathcal{X} = \cup_{i=1}^{K} \mathcal{X}_i$, and $L = \sum_{i=1}^{K} L_i$. In this paper, the training data in a two-class problem are called *positive* training data if their desired outputs are 1, and they are called *negative* training data if their desired outputs are 0.

From Eq. (2), we see that the number of training data for each of the two-class subproblems is the same as that of the original K-class problem. The main weakness of the one-versus-rest strategy is that this strategy may cause the training data sets of the two-class subproblems defined by Eq. (2) very imbalanced, even through the number of data among the classes in the original K-class problem is quite balanced.

Suppose a K-class classification problem is decomposed into the following $K(K-1)$ two-class subproblems with the one-versus-rest strategy,

$$\mathcal{T}_{ij} = \{(X_l^i,\ 1)\}_{l=1}^{L_i} \cup \{(X_l^j,\ 0)\}_{l=1}^{L_j}, \quad \text{for } i = 1, 2, \ldots, K \text{ and } j \neq i, \tag{3}$$

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.

From pattern classification's point view, only $K(K-1)/2$ two-class subproblems need to be learned, because the other $K(K-1)/2$ two-class subproblems defined by Eq. (3) have just opposite desired outputs and the same training inputs. Thus, only the following $K(K-1)/2$ two-class subproblems are considered usually in the training phase,

$$\mathcal{T}_{ij} = \{(X_l^i,\ 1)\}_{l=1}^{L_i} \cup \{(X_l^j,\ 0)\}_{l=1}^{L_j}, \\ \text{for } i = 1,2,\ldots,\ K \text{ and } j = i+1, i+2, \ldots,\ K. \tag{4}$$

Examining Eqs. (2) and (4), we can see that the two-class subproblems generated by both one-versus-rest and one-versus-one task decomposition strategies are unique to a given training data set $\mathcal{T}$. The reason is that the partition of training input sets in multi-class level has already determined by Eq. (1).

3.3 Various ways of fine task decomposition

Even though the two-class problems defined by Eq. (4) are smaller than the original K-class problem, this partition may be not adequate for parallel learning. The reason is that the number of samples in each class varies largely and the sizes of these two-class problems can be quite different, especially for large-scale real-world classification problems. Consequently, the training period can be delayed by some larger two-class subproblems. In addition, a large class and a small class together form an imbalanced classification problem, which increases the difficulty in training component classifiers. To speed up training and to improve classification accuracy, all the large and imbalanced two-class subproblems should be further divided into relatively smaller and more balanced two-class subproblems.

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

$$\mathcal{X}_{ij} = \{X_l^{ij}\}_{l=1}^{L_i^j}, \\ \text{for } j = 1,2,\ldots,\ N_i \text{ and } i = 1,2,\ldots,\ K, \tag{5}$$

where $1 \leqslant N_i \leqslant L_i$ and $\cup_{j=1}^{N_i}\mathcal{X}_{ij} = \mathcal{X}_i$. It should be noted that these subsets, $\mathcal{X}_{ij}$, can be either mutually exclusive or overlapping.

After partitioning $\mathcal{X}_i$ into N_i subsets, every two-class problem $\mathcal{T}_{ij}$ defined by Eq. (4) is further divided into the following $N_i \times N_j$ smaller and more balanced two-class subproblems,

$$\mathcal{T}_{ij}^{(u,\,v)} = \{(X_l^{(iu)},\ +1)\}_{l=1}^{L_i^{(u)}} \cup \{(X_l^{(jv)},\ -1)\}_{l=1}^{L_j^{(v)}}, \\ \text{for } u = 1,2,\ldots,\ N_i,\ v = 1,2,\cdots,\ N_j, \\ i = 1,2,\ldots,\ K, \text{ and } j = i+1, i+2, \ldots,\ K, \tag{6}$$

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, $L_i = \sum_{u=1}^{N_i} L_i^{(u)}$, and $L_j = \sum_{v=1}^{N_j} L_j^{(v)}$. This partition is called part-versus-part task decomposition strategy [5].

Suppose the training set of each two-class subproblem defined in Eq. (6) has only two different samples. Then they can be easily separated by a hyperplane or a Gaussian zero-crossing discriminate function [54,55] as follows,

$$f_{ij}(x) = \exp\left[-\left(\frac{\|x-c_i\|}{\sigma}\right)^2\right] \\ -\exp\left[-\left(\frac{\|x-c_j\|}{\sigma}\right)^2\right], \tag{7}$$

where x is the input vector, c_i and c_j are the given training inputs belonging to class C_i and class C_j $(i \neq j)$, respectively, $\sigma = \lambda\|c_i - c_j\|$, and λ is a user-defined constant.

The output of M^3-network with Gaussian zero-crossing discriminate function is defined as follows,

$$g_i(x) = \begin{cases} 1, & \text{if } y_i(x) > \theta_i, \\ \text{unknown}, & \text{if } -\theta_j \leqslant y_i(x) \leqslant \theta_i, \\ -1, & \text{if } y_i(x) < -\theta_j, \end{cases} \tag{8}$$

where θ_i and θ_j are the threshold limits of class C_i and C_j, respectively. y_i denotes the transfer function of the M^3-network for class C_i, which discriminates the pattern of the M^3-network for class C_i from those of the rest of the classes. The M^3-networks with Gaussian zero-crossing discriminate function have a useful ability of saying 'unknown' to unfamiliar inputs [56].

How to divide $\mathcal{X}_i$ into N_i subsets is critical for M^3-network. In the last several years, we have proposed various approaches for partitioning $\mathcal{X}_i$. These approaches fall into four main categories: 1) random partition [6]; 2) geometrical partition [57]; 3) learning-based partition [58,59]; and 4) knowledge-based partition [60]. It should be noted that decomposition of $\mathcal{X}_i$ into N_i subsets is not unique to a given training data set $\mathcal{T}$, no matter which kind of partition strategy is used.

The random partition strategy is simple and straightforward. We simply randomly divide the training input set $\mathcal{X}_i$ into N_i smaller subsets, with the constraint that these subsets must be of equal size roughly. This constraint is very important because we want the subproblems to be balanced. The advantage of this method is that it can be easily implemented. However, it does not make use of any statistical properties of the training data or prior knowledge of the training data.

In geometrical partition, the geometrical relationship among training inputs is considered. Hyperplane method is a typical geometrical partition strategy [57].

The hyperplane method consists of the following three main steps. First, we project the training inputs onto the direction of the normal vector by computing the dot product between each training input and the vector. Then we sort the training inputs according to their projections. Finally, we divide the training inputs equally according to the sorting result. An illustration of the hyperplane method is depicted in Fig. 3.

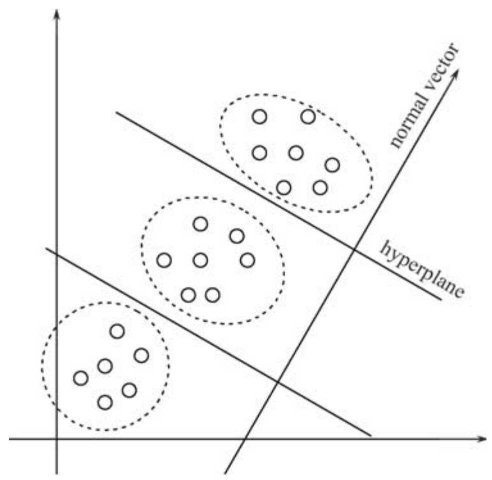

Fig. 3 Illustration of hyperplane method, where 20 training inputs are divided into three subsets

In learning-based partition, both unsupervised and supervised learning techniques are used to find suitable partition of training inputs. We have developed perceptron-based partition [59] and clustering-based partition [58]. Our experimental results indicate that these two methods are superior to random partition in generalization accuracy. But, the main deficiency of the learning-based partition is that extra training time is required. About knowledge-based partition, we will describe its mechanism and advantages through two real-world classification problems in Sect. 4.

3.4 Training component classifiers

After task decomposition, all of the two-class subproblems are treated as completely independent, non-communicating tasks in the learning phase. Therefore, all the two-class subproblems defined by Eq. (6) are efficiently learned in a massively parallel way.

From Eqs. (4) and (6), we see that a K-class problem is divided into

$$\sum_{i=1}^{K-1}\sum_{j=i+1}^{K} N_i \times N_j \tag{9}$$

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

$$\lceil L_i/N_i\rceil + \lceil L_j/N_j\rceil, \tag{10}$$

where $\lceil z\rceil$ denotes the smallest integer than or equal to z. 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 two-class subproblems is much smaller than the original K-class problem for the reasonable values of N_i and N_j.

Traditional machine learning algorithms and pattern classification approaches such as multilayer perceptrons [7], SVMs [5], and KNN algorithm [61], can be used to learn the two-class subproblems defined by Eq. (6). In this paper, the conventional SVMs are selected as component classifiers and used to learn all of the two-class subproblems. These component classifiers can be trained in parallel as they are totally independent. More precisely, the component classifiers are distributed to a group of CPUs at the beginning of the training phrase, and afterwards each CPU performs training by its own.

3.5 Module combination

After all of the two-class subproblems defined by Eq. (6) have been learned by SVMs, all the trained SVMs are integrated into a M^3-SVM with the Min and Max units according to the minimization principle and maximization principle [5].

Minimization principle Suppose a two-class problem $\mathcal{B}$ is divided into P smaller two-class subproblems, $\mathcal{B}_i$ for $i = 1, 2, \ldots, P$, 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 $i = 1, 2, \ldots, P$, then the combination of the P trained SVMs with a Min unit will produce the correct output for all the training inputs in $\mathcal{B}$, that is,

$$\mathcal{B}(x) = \min_{i=1}^{P} \mathrm{M}_i(x), \tag{11}$$

where x denotes an input instance.

Maximization principle Suppose a two-class problem $\mathcal{B}$ is divided into P smaller two-class subproblems, $\mathcal{B}_i$ for $i = 1, 2, \ldots, P$, 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 $i = 1, 2. \ldots, P$, then the combination of the P trained SVMs with a Max unit will produce the correct output for all the training input in $\mathcal{B}$, that is,

$$\mathcal{B}(x) = \max_{i=1}^{P} \mathrm{M}_i(x), \tag{12}$$

where x denotes an input instance.

In this paper, the module combination strategy based on the minimization and the maximization principles is called the Min-Max module combination in this paper. Figure 4 illustrates the Min-Max module combination for a two-class problem. Firstly the original two-class problem is decomposed into $N_i \times N_j$ two-class subproblems. Then each component classifier is trained on the corresponding two-class subproblem. Finally a M^3-network is constructed by combining these component classifiers. In this case, the M^3-network consists of $N_i \times N_j$ component classifiers, N_i Min units, and one Max unit.

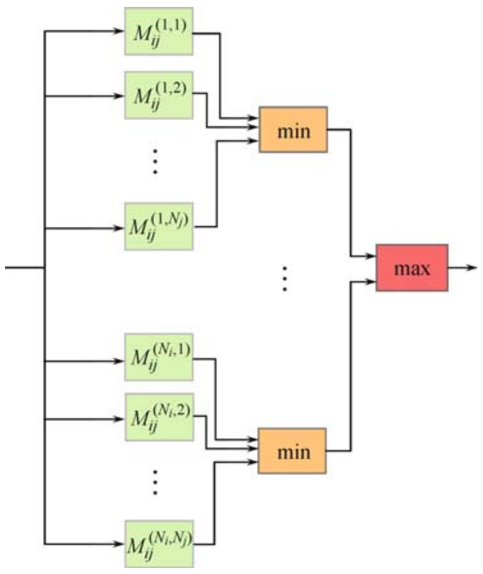

Fig. 4 Illustration of Min-Max module combination for a two-class problem

3.6 Asymmetric and symmetric module combination strategies

When the outputs of the component classifiers are simplified to binary value, i.e., either 0 or 1, the above standard Min-Max module combination method can be greatly simplified by the means of logistics. Though the logical semantics before and after the simplification are a little different, that is, produce different outputs on some rare cases, yet they are the same on most cases, and thus work well both in practical applications. According to this difference on semantics, we name the standard Min-Max combination asymmetric module combination, and name the simplified method symmetric module combination. The symmetric module combination method employs much less modules in classifying an instance than the asymmetric one does, which results in a great improvement on the classification speed. One disadvantage of the symmetric strategy, however, is that it only works on specific classifiers that output either 0 or 1.

Let us review the standard Min-Max module combination method, and explain why we call it the asymmetric module selection strategy. Suppose the outputs of all of the modules on a test sample are positive 1 or negative 0 (see Fig. 5(a)). Following the Min-Max module combination method, we first perform *minimization* operation along each row, then perform *maximization* operation among the rows' results. The final result is 1 in this example. Note that this result is caused by the fourth row, full of 1's as its elements. The translation of this process in feature space is that the test sample input is located within the subset $\mathcal{X}_{i,4}$, so that it shows positive in all the subproblems $\mathcal{T}_{ij}^{(4,k)} (k = 1, 2, \ldots, 5)$. The logical definition of the Min-Max module combination for M^3-network is

1) If there exists a row of all 1 in module's prediction matrix, the output is 1.

2) Otherwise, the output is 0.

The Min-Max module combination method has a bias towards the output of 0, and that is why we call it asymmetric module selection.

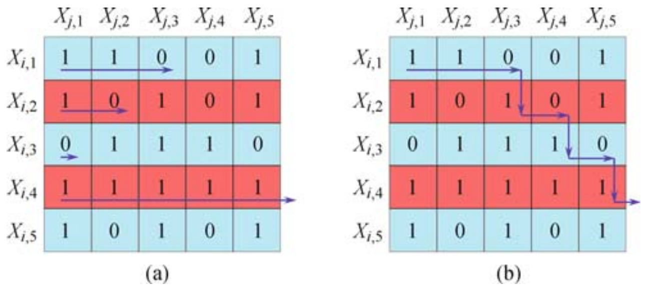

Fig. 5 Two module selection strategies. (a) Asymmetric module selection; (b) symmetric module selection

The other combination method, the symmetric module selection strategy, is illustrated in Fig. 5(b) [62]. In this method, we start from the top left element. If the element is 1, we go one step right; otherwise, we go one step down. The process continues until we go beyound the boundaries of the matrix. If we pass through the right edge of the matrix, the method outputs 1; otherwise, it outputs 0. The logical definition of this method is

1) If there exists a row of all 1 in module's prediction matrix, the output is 1.

2) If there exists a column of all 0, the output is 0.

3) Otherwise, the output can be either 1 or 0.

The third case is rare in solving regular pattern classification problems with M^3-network, and it doesn't need to be considered [62]. From the viewpoint of feature space, the first case means that the test sample input is located within some positive subsets, and the second case for some negative subsets, so this method makes sense. Moreover, this method has no bias towards either 1 or 0, that is why we call it a symmetric module selection. Indeed, this characteristic makes it likely to perform better than the asymmetric module selection method.

The symmetric module selection strategy has a great advantage on computation complexity, compared with the asymmetric one. Suppose the training data sets for two classes are divided into m and n subsets, respectively. The complexity of the asymmetric module selection is $O(mn)$, and that of symmetric module selection is

$O(m+n)$; so it is clear the symmetric module selection is one order faster than the asymmetric module selection. Further, the component classifiers that are not visited do not need to be actually computed (see Fig. 5), that is, the module combination method determines the complexity of whole test process of M^3-network. For these two reasons, the symmetric module selection method can greatly accelerate the response speed of M^3-network, which is our main purpose.

3.7 Features of M^3-network approach

M^3-network is most similar to the integration of sampling methods with ensemble learning among all the methods described above. However, both the sampling and the ensemble of M^3-network are quit different from others. The sampling of M^3-network is task decomposition, instead of under-sampling the majority classes or over-sampling the minority classes. And the ensemble of M^3-network is through a specially designed Min-Max network, different from the commonly used weighted voting such as the Adaboost.

Both M^3-network and other integration methods of sampling and ensemble can convert imbalanced data into balanced, thus eventually solve the imbalanced learning problem. However, M^3-network has a notable merit of parallel learning, which is precise in facing large-scale data sets. The modules in M^3-network are independent thus they can be learned individual, while the modules in Adaboost must be generated sequentially.

4 Applications

In this section, we choose two representative real-world imbalanced tasks to demonstrate the effectiveness of M^3-network for learning from imbalanced data sets. The first task is patent document classification from the domain of data mining and information retrieval. The experiment on this very large-scale task, which contains about seventy thousands classes and three hundred thousands samples, demonstrates M^3-network's ability to handle large-scale imbalanced data sets. The second task is protein subcellar localization from the domain of bioinformatics. The experiment on this multi-label task, where each sample has average 2.16 labels, demonstrates M^3-network's ability to handle multi-label imbalanced data sets.

4.1 Patent documents classification

In this section, we apply M^3-network to patent documents classification [63]. This is a real-world application, and many organizations and companies have been working on this topic such as the European and Japan Patent Office and Fujitsu company [64,65].

4.1.1 Introduction to international patent classification

Patent documents classification takes the standard of the international patent classification (IPC) as a label system [65]. IPC is a complex hierarchical symbol system, where all the technological fields are divided into eight sections, 120 classes, 630 subclasses and approximately 69000 groups (see Fig. 6). In this paper, we use M^3-network to classify Japanese patent documents on the Section level of IPC taxonomy.

The corpus that we work on is from the NTCIR workshop, which is publicly available for research purpose[1]. The corpus consists of about 3500000 documents of Japanese patent applications from 1993 to 2002. A patent document is a structured text with four main fields: title, abstract, claim, and description (see Table 1).

4.1.2 Task decomposition with years and hierarchies

We incorporate the prior knowledge of publish dates and

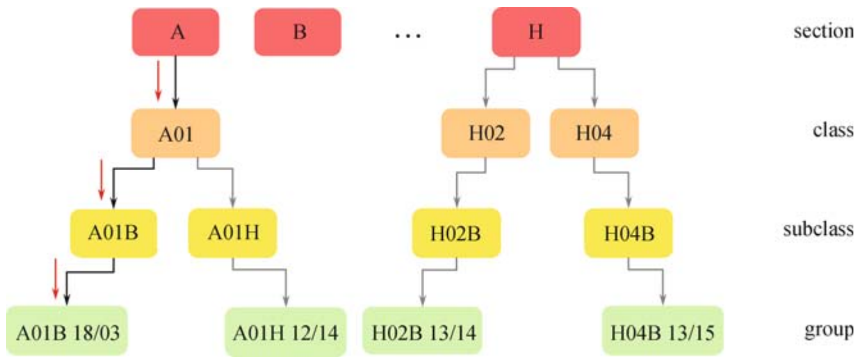

Fig. 6 Illustration of IPC taxonomy. Here, 'A', 'A01', 'A01B', and 'A01B 13/08' are the section, class, subclass, and group labels, respectively

1) http://research.nii.ac.jp/index-en.html

Table 1 Structure of Japanese patent documents

	PATENT-JA-UPA-1998-000001
<Bibliography>	
[publication date]	(43)【公開日】平成 10 年 (1998) 1 月 6 日
[title of invenction]	(54)【発明の名称】土壌改良方法とその作業機
<Abstract>	
[purpose]	【課題】心土破砕、特に雪上心土破砕作業の際に積雪...
[solution]	【解決手段】心土破砕を行うために用いるサブソイラの...
<Claims>	
[claim1]	【請求項 1】サブソイラ作業機を用いて心土破砕作業...
[claim2]	【請求項 2】サブソイラ作業機において、そのナイフ...
<Description>	
[technique field]	【発明の属する技術分野】本発明は、土壌改良方法とそ...
[prior art]	【従来の技術】圃場の表面がまだ積雪に覆われている状...
[problem to be solved]	【発明が解決しようとする課題】心土破砕は通常春先に...
[means of solving problems]	【課題を解決するための手段】述のような目的達成す...
[effects of invention]	【発明の効果】以上の説明から明らかなように、本発明...
...	...
< Explanation of Drawing >	
[figure1]	【図 1】本発明を施す圃場断面図である
...	...

hierarchical labels of patent documents into task decomposition for M^3-network. The decomposition process, namely M^3 Year-Class Decomposition (M^3-YC), consists of the following three steps (see Fig. 7).

Step 1 Divide the training data set of each section by publish dates, each subset for one year.

Step 2 Divide the subsets by class, thus each subset for one class published in one year.

Step 3 Randomly divide the remaining large subsets into smaller fix-sized subsets.

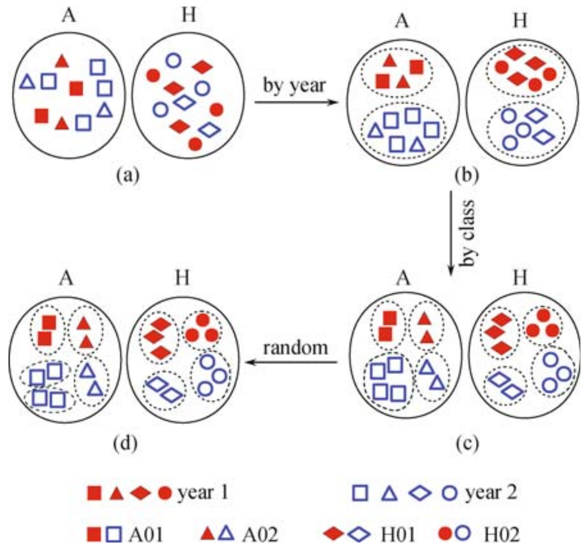

Fig. 7 Illustration of task decomposition of Japanese patent classification with prior knowledge. Different attributes among the samples are shown by colors and shapes. The task decomposition consists of three steps. The first step is to use prior knowledge of the publish year (b), the second step is to use class category information (c), and the last step is to randomly divide remaining large subsets (d)

Note that with Steps 1 and 2 removed, the above process becomes the standard M^3-network, namely M^3-Rand. In this research, we compare M^3-Rand and M^3-YC to plain SVMs, which are taken as the baseline methods.

In both M^3-Rand and M^3-YC, we can control the size of subproblems through the step of random decomposition, for example, setting the maximum number of samples in a subset. We must keep the subproblems in moderate size, neither too large for a single classifier to learn, nor so small that there are too many little subproblems. We finally decide on the maximum size of a subset to be 2000 samples, according to some pilot experiments [66].

4.1.3 Experiment settings

Data sets split The corpus that we use consists of all the patent applications published by the Japanese Patent Office from 1993 to 2002. We make eight pairs of training and test data sets, creating a real-world context to test the classification methods (see Table 2). The data distribution of the largest training set and the test

Table 2 Description of training and test data sets

years		set size	
training	test	training	test
2000	2001,2002	358072	733570
1999,2000	2001,2002	714004	733570
1998–2000	2001,2002	1055391	733570
1997–2000	2001,2002	1386850	733570
1996–2000	2001,2002	1727356	733570
1995–2000	2001,2002	2064325	733570
1994–2000	2001,2002	2415236	733570
1993–2000	2001,2002	2762563	733570

set is shown by Table 3, which is quite imbalanced.

Feature extraction and filtering We follow a standard chain of preprocess for text categorization, in order to go the vector representation of the Japanese. The preprocesses include tokenization, term filtering, and term indexing.

Table 3 Training and test data distribution of patent document corpus

lable	description	training	test
1	human necessities	228832	74483
2	performing operations and transporting	532529	134350
3	chemistry and metallurgy	290417	69246
4	textiles and paper	41563	8977
5	fixed constructions	132817	34395
6	mechanical engineering, lighting and so on	241624	62855
7	physics	671487	186100
8	electricity	623297	163165
total No. of documents		2762566	733571

Tokenization generates a list of clean and informative words from raw text. We first extract the raw text from the four patent fields: title, abstract, claim, and description (see Table 1), and segment these texts into isolated words using the software Chasen[1]. After that, we remove the stop words (or empty words) from the results.

Term Filtering removes the useless terms in the classification task. We take χ^2_{avg} as filtering criterion [41]), and pick up 5000 top terms as features, according to our pilot experiments.

Term Indexing generates the weights of feature terms for a sample with the real numerical vectors. We adopt the dominant methods of Term Frequency-Inverse Document Frequency (TFIDF) [67].

Component classifiers M^3-network is a general framework for pattern classification. The user can select suitable component classifiers according to the classification task to be solved and available computing resource. There are several alternative choices, such as multilayer neural networks [7], KNN algorithm [61], and SVMs [5]. Here we select conventional SVMs with linear kernel and use SVM$^{\text{light}}$, an implement of SVM by Joachims [37]. SVM$^{\text{light}}$ actually plays two roles in this paper, the baseline method and the component classifiers for M^3-network.

Performance measure We choose the commonly used micro-F_1 and macro-F_1 as the performance measure [67]. They are defined as follows,

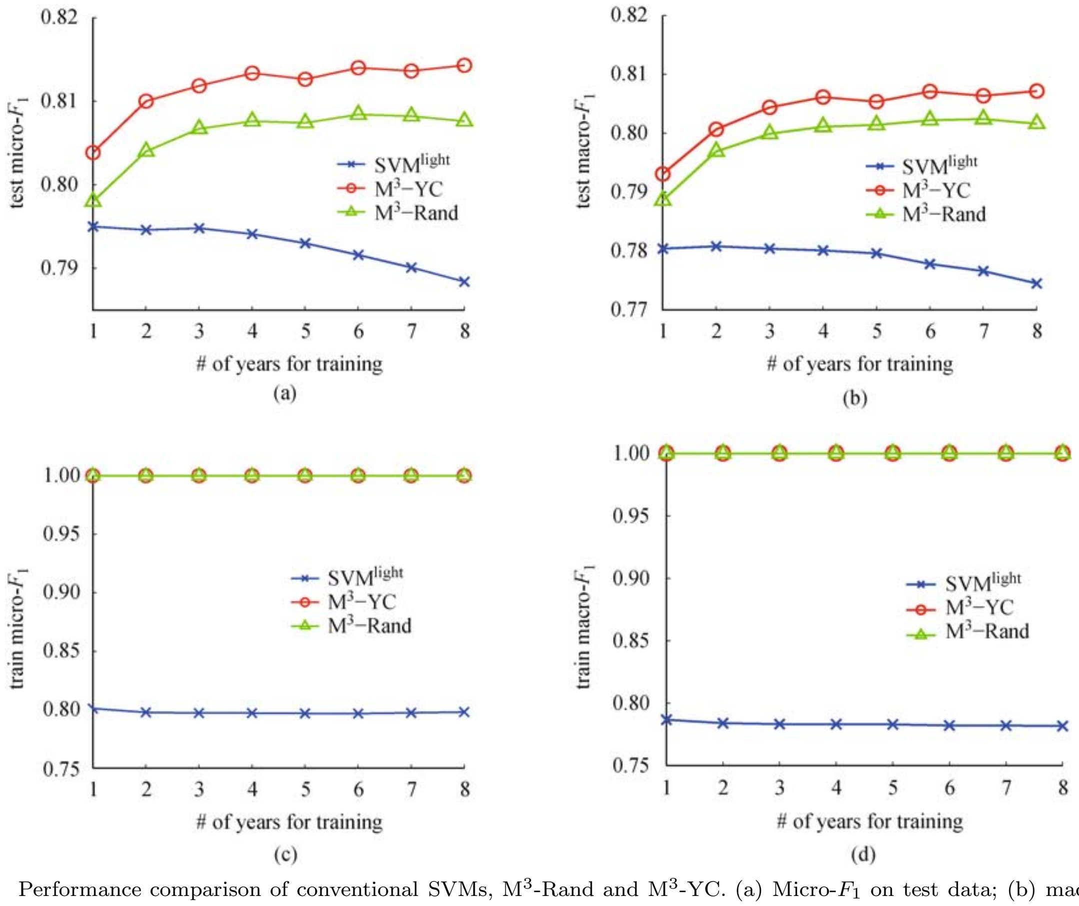

Fig. 8 Performance comparison of conventional SVMs, M^3-Rand and M^3-YC. (a) Micro-F_1 on test data; (b) macro-F_1 on test data; (c) micro-F_1 on training data; (d) macro-F_1 on training data

1) http://chasen.naist.jp/hiki/ChaSen/

$$\text{micro} - F_1 = \frac{2PR}{P+R} , \tag{13}$$

$$\text{macro} - F_1 = \operatorname*{avg}_{c \in C} \frac{2P_c R_c}{P_c + R_c} , \tag{14}$$

where P and R are overall precision and recall, and P_c and R_c are the local precision and recall of the class c.

4.1.4 Experimental results

Figure 8 shows the experimental results with micro-F_1 and macro-F_1. The following points can be drawn from the results:

1) On the aspect of test accuracy, the two M^3-SVM methods, M^3-Rand and M^3-YC, are both superior to conventional SVMs. We can learn from the training scores that conventional SVMs with linear kernel are unable to learn the training set completely, because its micro-F_1 and macro-F_1 are only about 80%; while M^3-SVMs have learned all of the training data with accuracy of nearly 100%.

2) M^3-Rand and M^3-YC show superior robustness to conventional SVMs over dated samples. Along the starting time point of training set moving back, more and more dated samples are added; the performance of flat SVMs decreases, while the performance of two M^3-SVMs increases at the same time.

3) M^3-YC outperforms M^3-Rand on each year point, which indicates that incorporating prior knowledge of the publishing date and the class category into task decomposition will undoubtedly improve classification performance.

4.2 Protein subcellar localization

Protein subcellular localization has a close relationship with protein function as proteins need to be in the right compartments in order to carry out their cellular functions. Automatic tools for predicting protein subcellular locations are very helpful because they can save a lot of wet-bench experimental work. Therefore, protein subcellular localization has been an active research topic in bioinformatics since last decade [68].

Till now, a lot of methods have been proposed to predict protein subcellular location [69]. However, few of them addressed the imbalanced data distribution problem, as the numbers of proteins located in different compartments vary significantly. And, many proteins bear multi-localization characteristics [70], i.e., a protein may have multiple locations. Therefore, we propose to use the Min-Max modular support vector machine to deal with these problems [71].

4.2.1 Data set

We worked on the DBMLoc database which collects multi-locational proteins from animal, plant, virus and bacteria [72]. All the cellular compartments were assigned into twelve categories: extracellular, cell wall, membrane, cytoplasm, mitochondrion, nucleus, ribosome, plastid, endoplasmic reticulum (ER), Golgi apparatus, vacuole, and virion. Some subcellular location annotations that can not be classified into the twelve categories are assigned to 'others'. As a result, the number of classes used in our prediction system is 13. The detailed data distribution is shown in Table 4. There is a total of 2975 protein samples, and 6426 labels. Thus each protein has 2.2 labels in average. We can observe that this data set is extremely imbalanced. The nucleus proteins take nearly a third of the whole set, while the number of vacuole proteins is only 20.

Table 4 Training and test data distribution of DBMLoc

lable	location	training	test
1	others	134	36
2	extracellular	471	43
3	ribosome	58	15
4	virion	31	2
5	membrane	1240	283
6	cytoplasm	1172	417
7	mitochondrion	445	123
8	nucleus	844	344
9	plastid	132	8
10	vacuole	16	4
11	cell wall	21	5
12	ER	322	53
13	Golgi	162	45
total No. of labels		5048	1378
total No. of proteins		2344	631

4.2.2 Task decomposition with gene ontology

During the past decades, various annotation data of proteins have grown dramatically in the public databases. Thus, plenty of prior knowledge, such as motif, function domain, and gene ontology (GO), are available in the study of protein function, regulation and other characteristics.

GO [73] has been demonstrated to be very useful in improving the prediction accuracy for protein subcellular localization [74]. However, many proteins do not have GO annotation, especially the proteins to be tested. Therefore, we did not incorporate GO for feature vectors to build our predictor. Instead, we use GO annotation to guide the task decomposition as the DBMLoc data is fully annotated with GO.

GO terms have semantic relations with each other, e.g., 'is-a' and 'part-of'. Here, we measure the similarity of GO terms based on their semantic relations [75]. Because each training data has a set of GO terms, the similarity of two proteins can be obtained by calculating the similarity between their corresponding GO term sets. Thus, the similarity matrix is constructed for each class to be decomposed, and then we use the clustering tool of CLUTO [76], to partition the data based on the similarity matrix. The numbers of clusters are determined by the predefined module size. We also test random decomposition with M^3-SVMs on this data set. These two methods are denoted as M^3-GO and M^3-Rand, respectively.

4.2.3 Experimental settings

Component classifier We chose LibSVM version 2.8 [39] to train component classifiers for M^3-SVMs, and used multi-class SVMs with one-versus-rest strategy. We experimented with polynomial, sigmoid and RBF kernels and observed that RBF kernel has the best classification accuracy. Thus the results reported in the following were obtained by using RBF kernel. Both M^3-SVMs and traditional SVMs have the same experimental settings.

Performance measure Multiple measures are used to assess the overall classification performance, including macro-F_1, total accuracy (TA) and location accuracy (LA). Total accuracy and location accuracy were first defined in Ref. [77]. We redefine them as the following equations because of the multi-label context on DBMLoc database.

$$\mathrm{TA} = \frac{\sum_{l=1}^{K} \mathrm{TP}_l}{N}, \tag{15}$$

$$\mathrm{LA} = \frac{\sum_{l=1}^{K} R_l}{K}, \tag{16}$$

where N is the total number of labels, K is the number of subcellular locations, TP_l is the number of true predictions at location l, and R_l is the recall of location l.

4.2.4 Experimental results

Table 5 shows the overall performance (TA, LA and macro-F_1) of the three methods, where traditional SVMs, M^3-Rand, and M^3-GO with three different module sizes are compared. Column 2 shows the predefined module sizes; column 3 shows the numbers of subproblems. From Table 5, several observations can be made.

1) Both M^3-SVMs with random decomposition and GO decomposition have higher TA and LA than traditional SVMs. The M^3-SVMs improve not only average location accuracy but also total accuracy, which indicates that they do not sacrifice minority classes for the classification of majority classes.

2) M^3-SVMs have higher macro-F_1 than traditional SVMs except M^3-Rand with module size 100. And M^3-GO with module size 800 achieved the highest macro-F_1. As the module size decreases, LA increases, but macro-F_1 decreases, which suggests a higher false positive rate of M^3-SVMs when the module size becomes small. So there is a tradeoff between location accuracy and false positive rate.

3) M^3-GO generally performs a little higher than M^3-Rand, and the superiority is obvious when the module size is small (100). Since random decomposition divides each training class into equal size modules randomly, while GO decomposition bases on the real relationship between proteins, the GO decomposition has a more stable performance.

In order to observe the classification accuracy on each subcellular location, we draw the recall and F_1 values of the traditional SVMs and M^3-SVMs with GO decomposition on 13 classes in Fig. 9. Obviously, the modulization helps to improve recall a lot, especially for the minority classes, e.g., cell wall. And, M^3-SVMs generally have higher F_1 values than traditional SVMs.

5 Conclusions

In this paper we study the problem of learning from imbalanced data sets. A brief review of the state-of-the-art methods for learning from imbalanced data is presented, where sampling methods and cost-sensitive methods are the two dominate sorts of approaches. In order to deal with large-scale imbalanced data sets, we introduce Min-Max modular support vector machine,

Table 5 Overall accuracy of three methods on DBMLoc

method	module size	module No.	TA/%	LA/%	macro-F_1/%
SVM	/	13	64.7	41.4	42.0
M^3-Rand	100	848	65.1	48.0	40.7
	400	83	66.9	47.7	42.7
	800	38	66.2	45.2	43.4
M^3-GO	100	848	66.2	**48.5**	42.3
	400	83	**67.1**	45.2	42.6
	800	38	66.3	43.4	**43.6**

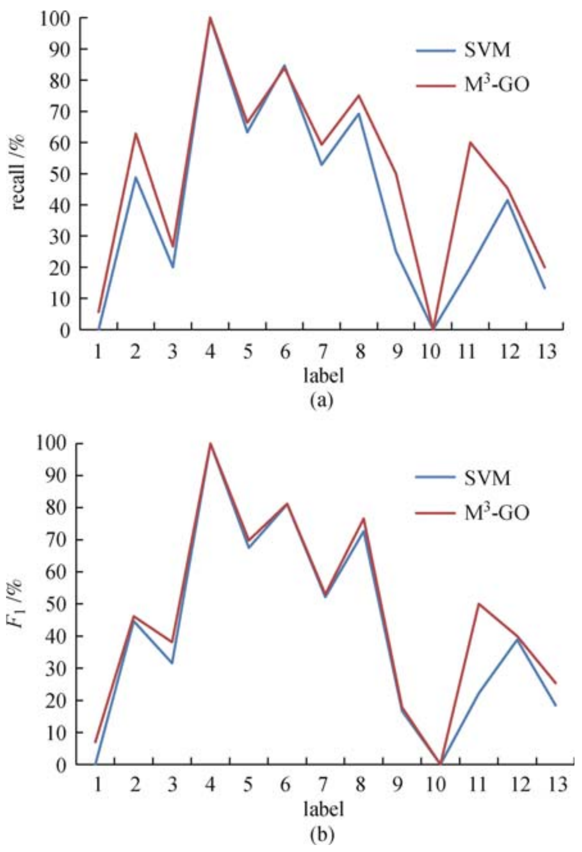

Fig. 9 Comparison of recall (a) and F_1 (b)

which is an integration of sampling method and ensemble learning approach. To show the effectiveness of our proposed method, we apply M^3-SVMs to two real-world imbalanced data sets and compare their performance with that of standard SVMs. The experimental results demonstrate that the imbalanced data sets do cause performance degradation of standard SVMs, which confirms the observations in related literature. However, the performance can be improved by M^3-SVMs, which achieve high classification accuracy by handling imbalanced data with task decomposition strategy. Moreover, the experimental results indicate that incorporating prior knowledge into task decomposition of M^3-SVMs can obviously improve their classification performance.

Acknowledgements This work was partially supported by the National Natural Science Foundation of China (Grant Nos. 60773090 and 90820018), the National Basic Research Program of China (No. 2009CB320901), the National High-Tech Research Program of China (No. 2008AA02Z315), and the Science and Technology Commission of Shanghai Municipality (No. 09511502400).

References

1. He H B, Garcia E A. Learning from imbalanced data. IEEE Transaction on Knowledge and Data Engineering, 2009, 21(9): 1263–1284
2. Japkowicz N. Learning from imbalanced data sets. In: Proceedings of Workshops at the 17th National Conference on Artificial Intelligence, 2000
3. Chawla N V, Japkowicz N, Kolcz A. Workshop Learning from Imbalanced Data Sets II, Machine Learning, 2003
4. Chawla N V, Japkowicz N, Kolcz A. Editorial: special issue on learning from imbalanced data sets. ACM SIGKDD Explorations Newsletter, 2004, 6(1): 1–6
5. Lu B L, Wang K A, Utiyama M, Isahara H. A part-versus-part method for massively parallel training of support vector machines. In: Proceedings of IEEE/INNS International Joint Conference on Neural Networks. 2004, 735–740
6. Lu B L, Ito M. Task decomposition based on class relations: a modular neural network architecture for pattern classification. Lecture Notes in Computer Science, 1997, 1240: 330–339
7. Lu B L, Ito M. Task decomposition and module combination based on class relations: a modular neural network for pattern classification. IEEE Transactions on Neural Networks, 1999, 10(5): 1244–1256
8. Ye Z F, Wen Y M, Lu B L. A survey of imbalanced pattern classification problems. CAAI Transactions on Intelligent Systems, 2009: 148–156 (in Chinese)
9. Estabrooks A, Jo T, Japkowicz N. A multiple resampling method for learning from imbalanced data sets. Computational Intelligence, 2004, 20(1): 18–36
10. Laurikkala J. Improving identification of difficult small classes by balancing class distribution. In: Proceedings of the Conference on Artificial Intelligence in Medicine in Europe. 2001, 63–66
11. Weiss G M, Provost F. The effect of class distribution on classifier learning: an empirical study. Technical Report MLTR-43. 2001
12. Batista G E A P A, Prati R C, Monard M C. A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explorations Newsletter, 2004, 6(1): 20–29
13. Kubat M, Matwin S. Addressing the curse of imbalanced training sets: one-sided selection. In: Proceedings of International Conference on Machine Learning. 1997, 179–186
14. Zhang J, Mani I. KNN approach to unbalanced data distributions: a case study involving information extraction. In: Prceedings of International Conference on Machine Learning, Workshop Learning from Imbalanced Data Sets. 2003, 1–7
15. Liu X Y, Wu J, Zhou Z H. Exploratory under sampling for class imbalance learning. In: Proceedings of International Conference on Data Mining. 2006, 965–969
16. Chawla N V, Bowyer K W, Hall L O, Kegelmeyer W P. SMOTE: synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 2002, 16(3): 321–357
17. Jo T, Japkowicz N. Class imbalances versus small disjuncts. ACM SIGKDD Explorations Newsletter, 2004, 6(1): 40–49
18. Chawla N V, Lazarevic A, Hall L O, Bowyer K W. SMOTE-Boost: improving prediction of the minority class in boosting. In: Proceedings of the 7th European Conference

on Principles and Practice of Knowledge Discovery in Databases. 2003, 107–119

19. Guo H, Viktor H L. Learning from imbalanced data sets with boosting and data generation: the dataBoost IM approach. ACM SIGKDD Explorations Newsletter, 2004, 6(1): 30–39
20. Mease D, Wyner A J, Buja A. Boosted classification trees and class probability/quantile estimation. Machine Learning Research, 2007, 8: 409–439
21. Maloof M A. Learning when data sets are imbalanced and when costs are unequal and unknown. In: Proceedings of International Conference on Machine Learning, Workshop Learning from Imbalanced Data Sets II. 2003, 1–8
22. Weiss G M. Mining with rarity: a unifying framework. ACM SIGKDD Explorations Newsletter, 2004, 6(1): 7–19
23. Liu X Y, Zhou Z H. The influence of class imbalance on cost-sensitive learning: An empirical study. In: Proceedings of International Conference on Data Mining. 2006, 970–974
24. Liu X Y, Zhou Z H. Training cost-sensitive neural networks with methods addressing the class imbalance problem. IEEE Transactions on Knowledge and Data Engineering, 2006, 18(1): 63–77
25. McCarthy K, Zabar B, Weiss G M. Does cost-sensitive learning beat sampling for classifying rare classes? In: Proceedings of International Workshop Utility-Based Data Mining. 2005, 69–77
26. Fan W, Stolfo S J, Zhang J, Chan P K. AdaCost: misclassification cost-sensitive boosting. In: Proceedings of International Conference on Machine Learning. 1999, 97–105
27. Sun Y, Kamel M S, Wong A K C, Wang Y. Cost-sensitive boosting for classification of imbalanced data. Pattern Recognition, 2007, 40(12): 3358–3378
28. Ting K M. A comparative study of cost-sensitive boosting algorithms. In: Proceedings of International Conference on Machine Learning. 2000, 983–990
29. Haykin S. Neural Networks: A Comprehensive Foundation. 2nd ed. New Jersey: Prentice-Hall, 1999
30. Kukar M Z, Kononenko I. Cost-sensitive learning with neural networks. In: Proceedings of the 13th European Conference on Artificial Intelligence. 1998, 445–449
31. Domingos P, Pazzani M. Beyond independence: Conditions for the optimality of the simple Bayesian classifier. In: Proceedings of the International Conference on Machine Learning. 1996, 105–112
32. Gama J. Iterative bayes. Theoretical Computer Science, 2003, 292(2): 417–430
33. Kohavi R, Wolpert D. Bias plus variance decomposition for zero-one loss functions. In: Proceedings of International Conference on Machine Learning. 1996, 275–283
34. Webb G R I, Pazzani M J. Adjusted probability naive Bayesian induction. In: Proceedings of the 11th Australian Joint Conference on Artificial Intelligence. 1998, 285–295
35. Drummond C, Holte R C. Exploiting the cost (in)sensitivity of decision tree splitting criteria. In: Proceedings of the International Conference on Machine Learning. 2000, 239–246
36. Vapnik V N. The Nature of Statistical Learning Theory. Berlin: Springer, 1995
37. Joachims T. Making large-scale support vector machine learning practical. Advances in Kernel Methods: Support Vector Learning. Cambridge: MIT Press, 1998, 169–184
38. Joachims T. A support vector method for multivariate performance measures. In: Proceedings of the International Conference on Machine Learning. 2005, 377–384
39. Fan R E, Chen P H, Lin C J. LIBSVM: A library for support vector machines. http://www.csie.ntu.edu.tw/cjlin/libsvm/
40. Liu T Y, Yang Y M, Wan H, Zeng H J, Chen Z, Ma W Y. Support vector machines classification with a very large-scale taxonomy. Journal of ACM Special Interest Group on Discovery and Data Mining Explorations, 2005, 7(1): 36–43
41. Yang Y M, Pedersen J O. A comparattive study on feature selection in text categorization. In: Proceedings of International Conference on Machine Learning. 1997, 187–196
42. Wu G, Chang E. Class-boundary alignment for imbalanced data set learning. In: Proceedings of International Conference on Data Mining, Workshop Learning from Imbalanced Data Sets II. 2003, 1–8
43. Wu G, Chang E Y. Aligning boundary in kernel space for learning imbalanced data set. In: Proceedings of International Conference on Data Mining. 2004, 265–272
44. Wu G, Chang E Y. KBA: kernel boundary alignment considering imbalanced data distribution. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(6): 786–795
45. Kang P, Cho S. EUS SVMs: Ensemble of under-sampled SVMs for data imbalance problems. Lecture Notes in Computer Science, 2006, 4232: 837–846
46. Liu Y, An A, Huang X. Boosting prediction accuracy on imbalanced data sets with SVM ensembles. Lecture Notes in Artificial Intelligence, 2006, 3918: 107–118
47. Vilarino F, Spyridonos P, Radeva P, Vitria J. Experiments with SVM and stratified sampling with an imbalanced problem: detection of intestinal contractions. Lecture Notes in Computer Science, 2005, 3687: 783–791
48. Wang B X, Japkowicz N. Boosting support vector mMachines for imbalanced data sets. Lecture Notes in Artificial Intelligence, 2008, 4994: 38–47
49. Abe N. Sampling approaches to learning from imbalanced data sets: active learning, cost sensitive learning and deyond. In: Proceedings of International Conference on Machine Learning, Workshop Learning from Imbalanced Data Sets II. 2003
50. Ertekin S, Huang J, Bottou L, Giles L. Learning on the border: active learning in imbalanced data classification. In: Proceedings of the 16th ACM Conference on Information and Knowledge Management. 2007, 127–136
51. Ertekin S, Huang J, Giles C L. Active learning for class imbalance problem. In: Proceedings of International SIGIR

Conference on Research and Development in Information Retrieval. 2007, 823–824

52. Provost F. Machine learning from imbalanced data sets 101. In: Proceedings of American Association Artificial Intelligence Workshop on Imbalanced Data Sets. 2000, 1–3
53. Lu B L, Wang X L, Utiyama M. Incorporating prior knowledge into learning by dividing training data. Frontiers of Computer Science in China, 2009, 3(1): 109–122
54. Lu B L, Ichikawa M. A Gaussian zero-crossing discriminant function for Min-Max modular neural networks. In: Proceedings of the 5th International Conference on Knowledge-Based Intelligent Information Engineering Systems and Allied Technologies. 2001, 298–302
55. Lu B L, Ichikawa M. Emergent on-line learning with a Gaussian zero-crossing discriminant function. In: Proceedings of IEEE/INNS International Joint Conference on Neural Networks. 2002, 2: 1263–1268
56. Lu B L, Li J. A Min-Max modular network with Gaussian-zero-crossing function. In: Chen K, Wang L, eds. Trends in Neural Computation. Berlin: Springer, 2007, 285–313
57. Wang K A, Zhao H, Lu B L. Task decomposition using geometric relation for Min-Max modular SVMs. Lecture Notes in Computer Science, 2005, 3496: 887–892
58. Wen Y M, Lu B L, Zhao H. Equal clustering makes Min-Max modular support vector machine more efficient. In: Proceedings of the 12th International Conference on Neural Information Processing. 2005, 77–82
59. Cong C, Lu B L. Partition of sample space with perceptrons. Computer simulation, 2008, 25(2): 96–99 (in Chinese)
60. Ma C, Lu B L, Utiyama M. Incorporating prior knowledge into task decomposition for large-scale patent classification. In: Proceedings of 6th International Symposium on Neural Networks: Advances in Neural Network-Part II. 2009, 784–793
61. Zhao H, Lu B L. A modular k-nearest neighbor classification method for massively parallel text categorization. Lecture Notes in Computer Science, 2004, 3314: 867–872
62. Zhao H, Lu B L. Improvement on response performance of Min-Max modular classifier by symmetric module selection. Lecture Notes in Computer Science, 2005, 3497: 39–44
63. Lu B L, Wang X L. A parallel and modular pattern classification framework for large-scale problems. In: Chen C H, ed. Handbook of Pattern Recognition and Computer Vision. 4th ed. Singapore: World Scientific, 2009, 725–746
64. Fall C J, Törcsvári A, Benzineb K, Karetka G. Automated categorization in the international patent classification. ACM SIGIR Forum, 2003, 37(1): 10–25
65. Fujii A, Iwayama M, Kando N. Introduction to the special issue on patent processing. Information Processing and Management, 2007, 43(5): 1149–1153
66. Chu X L, Ma C, Li J, Lu B L, Utiyama M, Isahara H. Large-scale patent classification with Min-Max modular support vector machines. In: Proceedings of IEEE/INNS International Joint Conference on Neural Networks. 2008, 3973–3980
67. Sebastiani F. Machine learning in automated text categorization. ACM Computing Surveys, 2002, 34(1): 1–47
68. Cedano J, Aloy P, Pérez-Pons J A, Querol E. Relation between amino acid composition and cellular location of proteins. Journal of Molecular Biology, 1997, 266(3): 594–600
69. Chou K C, Shen H B. Review: recent progresses in protein subcellular location prediction. Analytical Biochemistry, 2007, 370(1): 1–16
70. Cai Y D, Chou K C. Predicting 22 protein localizations in budding yeast. Biochemical and Biophysical Research Communications, 2004, 323(2): 425–428
71. Yang Y, Lu B L. Prediction of protein subcellular multi-localization by using a Min-Max modular support vector machine. Advances in Computational Intelligence, Ascvances in Soft Computing, 2009, 116: 133–143
72. Zhang S, Xia X, Shen J, Zhou Y, Sun Z. DBMLoc: a data base of protein swith multiple subcellular localizations. BMC Bioinformatics, 2008, 9(1): 127
73. Gene Ontology Consortium. gene ontology: tool for the unification of biology. Nature Genetics, 2000, 25(1): 25–29
74. Chou K C, Cai Y D. Predicting protein localization in budding yeast. Bioinformatics, 2005, 21(7): 944–950
75. Wang J Z, Du Z, Payattakool R, Yu P S, Chen C F. A new method to measure the semantic similarity of GO terms. Bioinformatics, 2007, 23(10): 1274–1281
76. Karypis G. CLUTO-A Clustering Toolkit, Technical Report 02-017. 2002
77. Huh W K, Falvo J V, Gerke L C, Carroll A S, Howson R W, Weissman J S, O'Shea E K. Global analysis of protein localization in budding yeast. Nature, 2003, 425(6959): 686–691

Bao-Liang LU is a professor and vice-chair of the Department of Computer Science and Engineering at Shanghai Jiao Tong University (SJTU). He is the founding Director of Center for Brain-Like Computing and Machine Intelligence at Shanghai Jiao Tong University.

Bao-Liang LU received his B.S. degree from Qingdao University of Science and Technology, China, in 1982, his M. S. degree from Northwestern Polytechnical University, China, in 1989, and Dr. Eng. degree from Kyoto University, Japan, in 1994. From 1982 to 1986, he was with the Qingdao University of Science and Technology. From April 1994 to March 1999, he was a Frontier Researcher at the Bio-Mimetic Control Research Center, the Institute of Physical and Chemical

Research (RIKEN), Japan. From April 1999 to August 2002, he was a Research Scientist at the RIKEN Brain Science Institute. Since August 2002, he has been a full-time Professor at the Department of Computer Science and Engineering, Shanghai Jiao Tong University. He has been an adjunct professor at Shanghai Center for Systems Biomedicine since 2005. He was the general co-chair of ISNN2010 and program chairs of ISNN2006 and CJNLP2006. He served as a Guest Editor for special issues of Neurocomputing and International Journal of Neural Systems. His research interests includes brain-like computing, neural networks, machine learning, brain-computer interface, and bioinformatics. He is the elected-president (2011) of Asian-Pacific Neural Networks Assembly (APNNA) and a senior member of the IEEE.

Xiao-Lin WANG received his B.A. of computer science from Shanghai Jiao Tong University in 2004, and now is working for his PhD in the same university. His research interests include machine learning, large-scale text classification and information retrieval.

Yang YANG received her B.S. and Ph.D. in computer science from Shanghai Jiao Tong University in 2003 and 2009, respectively. She has been a lecturer in the Department of Computer Science, Shanghai Maritime University since July, 2009. Her research interests include machine learning, bioinformatics and data mining.

Hai ZHAO is an assistant professor of computer science and engineering at Shanghai Jiao Tong University. He obtained his B. Eng. in sensor and instrument, and M. Phil. in control theory and engineering from Yanshan University, and Ph.D. in computer science from Shanghai Jiao Tong University. He was a visiting researcher in natural language computing group of Microsoft Research Asia. He was a Postdoctoral Research Fellow at City University of Hong Kong from 2006-2009. His research areas include machine learning, natural language processing, and data mining.