Prediction of Protein Subcellular Multi-localization by Using a Min-Max Modular Support Vector Machine

Yang Yang[1] and Bao-Liang Lu[1,2,⋆]

[1] Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2] Laboratory for Computational Biology, Shanghai Center for Systems Biomedicine,
800 Dong Chuan Road, Shanghai 200240, China
{alayman, bllu}@sjtu.edu.cn

Abstract. Prediction of protein subcellular location is an important issue in computational biology because it provides important clues for characterization of protein function. Currently, much effort has been dedicated to developing automatic prediction tools. However, most of them focus on mono-locational proteins. It should be noted that many proteins bear multi-locational characteristics, and they carry out crucial functions in biological processes. This work aims to develop a general pattern classifier for predicting multiple subcellular locations of proteins. We used an ensemble classifier, called min-max modular support vector machine (M^3-SVM), to solve protein subcellular multi-localization problem, and proposed a task decomposition method based on gene ontology (GO) semantic information for the M^3-SVM. We applied our method to a high-quality multi-locational protein data set. The M^3-SVMs showed better performance than traditional SVMs using the same feature vectors. And the GO decomposition also helped improve the prediction accuracy with more stable performance than random decomposition.

1 Introduction

Identification of subcellular location is an important goal of protein bioinformatics. It can provide information helpful for understanding protein function, regulation and protein-protein interaction. And efficient computational tools can save costly and laborious wet-lab experiments. Therefore, prediction of protein subcellular localization has been an active research topic in bioinformatics in the last decade.

To develop automatic tools for subcellular localization, machine learning methods, such as neural networks [1], hidden Markov models (HMMs) [2] and support vector machines (SVMs) [3], have been widely used, thanks to the abundance of proteins with known locations in the public databases. The extracted features used in these classifiers fall into two types: sequence-based and annotation-based. Sequence-based methods use single-residue composition, dimer, trimer composition, or represent sequences as condensed feature vectors using pseudo-amino acid composition [4], signal-processing and

⋆ To whom correspondence should be addressed. This work was partially supported by the National Natural Science Foundation of China (Grant No. 60773090 and Grant No. 90820018), the National Basic Research Program of China (Grant No. 2009CB320901), and the National High-Tech Research Program of China (Grant No. 2008AA02Z315).

text processing techniques [5]. In addition, N-terminal signals are very effective in identifying mitochondrial, chloroplast, and secretory pathway proteins. But sometimes the leading sequences of the test proteins are missing, and for many locations, no obvious sorting signal could be detected. As annotation becomes more abundant, many studies use annotation-based methods, including motifs, function domains, or gene ontology (GO) [6] to improve the prediction accuracy.

Most of these studies focus on mono-locational proteins, i.e., they assume that proteins may exist in only one cellular compartment. This is not always the case. Many proteins are multi-locational. They may translocate into different compartments, or secret out of the cell. In most of the previous prediction systems, such proteins were discarded or treated like mono-locational ones. Cai and Chou [7] dealt with such proteins in budding yeast by unfolding the multi-label data. For example, a tri-localized protein would be unfolded into three distinct samples with different labels, and predicted respectively. In essence, their method treats the multi-locational proteins like single-locational ones. Certain strategies and evaluation measures are needed to deal with the multi-locational cases. In our previous studies [8,9], we collected the multi-locational proteins from Swiss-Prot [10]. However, the annotations on subcellular localization are incomplete for many entries. According to our statistics, around 10% proteins in Swiss-Prot are annotated with more than one location. Recently, Zhang *et al.* [11] published a high-quality database of proteins with multiple subcellular locations, called DBMLoc. Given this database, the performance of predictors for multi-locational proteins can be estimated more fairly.

Moreover, the subcellular localization prediction is usually an imbalanced classification problem. The numbers of proteins located in different compartments vary significantly, i.e., the class distribution is uneven. For example, proteins in cytoplasm, membrane and nucleus are much more numerous than those in other locations. A number of approaches have been proposed to address the class imbalance problem. Oversampling and undersampling are two typical methods [12]. Oversampling approach duplicates data from the minority class, and undersampling approach eliminates data from the majority class. Both methods aim to re-balance the classes. Obviously, oversampling increases the complexity of classification problem, while undersampling results in information loss. Although SVMs make the decision boundary based on support vectors rather than all data samples, it still can not work well in class imbalance problems because of the imbalanced support vector ratio and weakness of soft-margins [13,14].

In this paper, we used a min-max modular support vector machine (M^3-SVM) to predict protein subcellular multi-localization. The classifier is an ensemble of support vector machines (SVMs) [15]. It is suited for imbalanced classification problems. It decomposes the original problem into relatively balanced subproblems to eliminate the skew of decision boundary. The subproblems are integrated by minimization and maximization principles in the ensemble classifier.

How to decompose the data set of a class for an M^3-classifier has not been perfectly solved so far. The random decomposition is the most straightforward way, which divides the majority and minority classes randomly into nearly equal sizes. But it cannot ensure stable performance. In this paper, we propose a new decomposition method based on biological domain knowledge, i.e., GO annotation. We noticed that for many pattern classification problems, the training data are organized by some prior knowledge, which could be useful clues for the modular methods. Here, we calculated the semantic similarity of GO terms, used the similarity to cluster proteins and partitioned training data for the proposed ensemble classifier.

In addition, to obtain good classification performance, we developed a new feature extraction method that combines multiple knowledge sources, including amino acid composition, secondary structure, and solvent accessibility. All of these features are believed to be closely correlated with protein subcellular localization.

The proposed method was evaluated on the data set, DBMLoc [11]. The M^3-SVMs showed better performance than traditional SVMs using the same feature vectors. We also compared GO decomposition and random decomposition, and found that GO decomposition helped improve the prediction accuracy.

2 Methods

In this paper, we propose a modular classifier, min-max modular support vector machine (M^3-SVM) [16,17], which is an ensemble of SVMs. Each SVM classifier is trained on a subset of the original data set. When a test sample comes, each trained SVM outputs a classification result. Then all of the outputs are integrated to get a final solution to the original problem according to two module combination rules, namely the minimization and the maximization principles.

For solving a large-scale and complex multi-label problem, our method consists of three main steps: a) decompose the original problem into two-class problems; b) further decompose the two-class problems which are difficult to be learned into a number of relatively smaller and balanced two-class subproblems. c) combine all the submodules into a hierarchical, parallel, and modular pattern classifier.

2.1 Classification of Multi-label Problems

The traditional method for solving multi-label task is to split the original problem into a set of binary classification tasks using one-versus-rest decomposition strategy. For a K-class multi-label problem, let $\mathcal{T}$ denote its training set:

$$\mathcal{T} = \{(x_m, t_m)\}_{m=1}^{L}, t_m = \{t_m^k\}, k = 1, ..., \tau_m, \tag{1}$$

where $x_m \in \mathrm{R}^n$ is the mth sample in the data set, t_m is the label set of x_m, t_m^k is the kth label of x_m, τ_m denotes the total number of labels of x_m, and L is the total number of samples.

By decomposing a K-class multi-label problem $\mathcal{T}$ into K mono-label two-class problems $\mathcal{T}_i$ for $i = 1, ..., K$, we have the training set of $\mathcal{T}_i$ as follows:

$$\mathcal{T}_i = \{(x_m^{i+}, +1)\}_{m=1}^{L_i^+} \cup \{(x_m^{i-}, -1)\}_{m=1}^{L_i^-}, \tag{2}$$

where L_i^+ is the number of positive samples of the two-class problem $\mathcal{T}_i$, and L_i^- is the number of negative samples. For $\mathcal{T}_i$, positive samples are the samples whose label sets contain label i, and negative samples are the remaining ones. Thus x_m will appear τ_m times as positive training data, and $(K - \tau_m)$ times as negative training data.

Each binary classifier decides whether or not a novel sample belongs to a particular class. Obviously, each of the binary problems has the same size as the original problem. The data distribution would become more imbalanced because of the one-versus-rest strategy. For a complex and imbalanced binary classification problem, we can further divide it into a number of relatively small and balanced two-class subproblems. Each subproblem is solved by a SVM, and all the subproblems are combined using MIN and MAX principles. The functions of MIN and MAX are to find the minimum and maximum values of all the inputs, respectively.

For a two-class problem $\mathcal{T}_i$, its positive and negative training sets, $\mathcal{T}_i^+$ and $\mathcal{T}_i^-$, are further decomposed into N_i^+ and N_i^- subsets, where $1 \le N_i^+ \le L_i^+$ and $1 \le N_i^- \le L_i^-$.

$$\mathcal{T}_i^{+j} = \{(x_m^+, +1)\}_{m=1}^{L_i^{+j}}, j = 1, ..., N_i^+, \tag{3}$$

$$\mathcal{T}_i^{-j} = \{(x_m^-, -1)\}_{m=1}^{L_i^{-j}}, j = 1, ..., N_i^-, \tag{4}$$

where L_i^{+j} and L_i^{-j} are the numbers of samples in $\mathcal{T}_i^{+j}$ and $\mathcal{T}_i^{-j}$, respectively. Each two-class problem $\mathcal{T}_i$ is solved by an M^3 network shown in Fig. 1.

According to the MIN and MAX principles [16], N_i^+ MIN units and one MAX unit are required to combine all the $(N_i^+ \times N_i^-)$ modules. Each of the MIN units combines N_i^- modules. The final output is determined by the outputs of all modules.

This ensemble classifier has some advantages over other methods in dealing with imbalanced problems. Compared with under-sampling method, it makes full use of the training data without information loss. Compared with over-sampling methods, it does

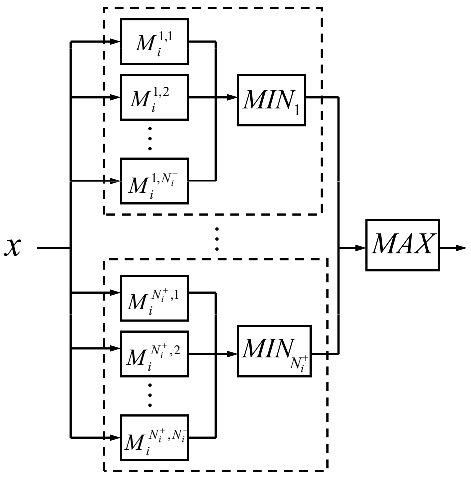

Fig. 1. Structure of M^3 network for a two-class problem $\mathcal{T}_i$, which is divided into $(N_i^+ \times N_i^-)$ two-class subproblems

not add to learning cost in each module, and speeds up the training process. The MIN and MAX rules provide an effective ensemble principle to obtain a solution to the original problem from the sub-problems, and they are easy to implement.

2.2 Task Decomposition

According to the one-versus-rest strategy, a K-class classification problem is decomposed into K two-class problems. Some of the two-class problems may have extremely imbalanced distribution of the positive and negative classes. Moreover, some of the two-class problems may be too large for fast learning. The most important advantage of the M^3 model is that it can further divide the large and imbalanced two-class problems into relatively smaller and more balanced subproblems.

Random partition is the simplest and most straightforward way. Given a specific module size, when we choose samples randomly from the training set to build a module, the samples may have no distribution relationship with each other. In such cases, although the subproblem has a reduced data size, it may be still hard to solve, and have complex decision boundary apt to overfit. Since the overall classification capability lies on the performance of all the modules, the poor boundaries learned by some modules would degrade prediction accuracy of the whole system. Therefore, the random partition can not obtain a stable performance.

Several decomposition strategies have been developed for M^3 model, such as hyperplane decomposition [17] and equal clustering [18]. Hyperplane decomposition uses a group of parallel hyperplanes to partition data into subsets. This method is fast and suitable for sparse data. Equal clustering (EC) works similarly to K-means clustering. The only difference is that EC pays more attention to load balance for the seek of parallel learning, so the clusters are kept in nearly equal size. All these methods aim to utilize the geometric distribution characteristics of data points in the high-dimension space.

In the past [9,18], we either divided the data randomly or based on the distance of sample points in the feature space, like hyperplane decomposition and equal clustering, but ignored the prior knowledge which may contribute useful information to do clustering within a big class. Here, we want to fully utilize the Gene Ontology information and achieve a better partition, such that the proteins sharing some common attributes could be grouped together. In the GO graph, GO terms are structured hierarchically and have semantic relations ('is-a' and 'part-of') with each other. A child node is more specialized than its parental nodes, and more than one parental node may exist. Here, we used similarity measure of GO terms based on their semantic relations to cluster proteins in a class which needs to be decomposed. Many methods have been developed to define the similarity between two GO terms. Given the similarity between two GO terms, the similarity between two sets of GO terms can be calculated. Suppose each protein corresponds to a set of GO terms, the similarity between two proteins can be obtained accordingly.

The data set used in our experiments was annotated with GO terms, including cellular component, biological process and molecular function. In this work, we adopted the method proposed by Wang *et al.* [19] to measure the semantic similarity of GO terms. We built the GO similarity matrix for our training data set, and used the clustering tool, CLUTO [20] to partition the data based on the similarity matrix. The program "scluster" in the tool kit was used. The number of clusters should be specified for this program. We calculated the number of clusters according to the predefined module size. Suppose that Class C_i has m samples and C_i is divided into k modules when n is the predefined module size, then m/k must be the closest value to n among all possible module sizes of C_i. Random decomposition makes each module equal size, while in GO decomposition, the actual size of each module is determined by the clustering method.

2.3 Feature Extraction

The features include amino acid composition, secondary structure and solvent accessibility informatin [21]. The former 60 dimensions are the amino acid composition of the full sequence on three secondary structure elements, i.e., strand (E), helix (H) and coil (C). The value of each dimension is calculated by

$$f_i^k = \frac{N_i^k}{L}, \tag{5}$$

where k={H, E, C}, N_i^k is the frequency of amino acid i in secondary structure element k, and L is the length of the sequence. The latter 40 dimensions are the amino acid composition on two solvent accessibility status, namely buried (B) and exposed (E), and is calculated similarly as Eq. 5 , with k = {B, E}.

The secondary structure elements were predicted by PSIPRED [22], and solvent accessibility status were predicted by ACCpro [23]. Both of them are highly accurate prediction methods. All the feature vectors were scaled in the range of $[0, 1]$ using SVM-Scale in the LibSVM package [24].

3 Results and Discussion

In order to test the performance of our methods, we applied them to a high-quality multi-locational protein database, DBMLoc, published by Zhang *et al.* [11], which was collected from multiple databases and experimentally determined localization data. All the cellular compartments were assigned into twelve categories as shown in Table 1. Some subcellular localization annotations which 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. All of the proteins in DBMLoc database have no less than two locations. The average number of labels for each protein is 2.2.

To avoid overfitting, we used the non-redundant data with sequence similarity below 25%. The training and test data sets are mutually exclusive. Test data is a high quality set including 631 proteins. Training data has a total of 2344 proteins, extracted from the complete non-redundant set (25%) by removing the overlapping data with test set. The statistics of the data sets are shown in Table 1. From the table, we can see that the data distributions are very imbalanced on different cellular compartments. Proteins of membrane, cytoplasm, and nucleus make an overwhelming portion, which adds to classification difficulty.

Although this data set is fully annotated, we did not use gene ontology as features to build our predictor based on the consideration that many test proteins are novel and do

Table 1. Training and test data distribution of DBMLoc

Location	Training	Test
Others	134	36
Extracellular	471	43
Ribosome	58	15
Virion	31	2
Membrane	1240	283
Cytoplasm	1172	417
Mitochondrion	445	123
Nucleus	844	344
Plastid	132	8
Vacuole	16	4
Cell wall	21	5
ER	322	53
Golgi	162	45
Total label #	5048	1378
Total protein #	2344	631

not have GO information. But the prior knowledge about training data could be fully utilized. Therefore, we used GO semantic similarity to guide the module decomposition of training data. We experimented three methods with the same feature vectors. One is min-max modular support vector machine with gene ontology decomposition (M^3-SVM(GO)). The second is min-max modular support vector machine with random decomposition (M^3-SVM(R)). The last one is traditional SVM. The module sizes of 100, 400 and 800 were tested for both M^3-SVM(GO) and M^3-SVM(R). And the results of M^3-SVM(R) were averaged through five repetition experiments.

Here we chose LibSVM version 2.8 [24] as the base classifier for the ensemble classifier. We experimented with polynomial, sigmoid and RBF kernels and observed that RBF kernel has the best classification accuracy. We performed ten-fold cross-validation and grid search on training data to find the optimum parameters for SVMs. The experimental results reported in the following were obtained with the kernel parameters $\gamma = 2^{-6}$ and $C = 2^4$. All experiments were conducted on a Pentium 4 double CPU (2.8GHz) PC with 2GB RAM.

Multiple measures were used to assess the performance of our proposed method, including precision (P), recall (R), F_1, total accuracy (TA), location accuracy (LA) and average F_1 ($aveF_1$). The former three measures, P, R and F_1, were used to measure the prediction quality of each location, and the last three measures, TA, LA and $aveF_1$, were used to measure the overall prediction quality across all locations.

Table 2 shows overall performance (TA, LA and $aveF_1$) of the three methods, traditional SVM, M^3-SVM(R) and M^3-SVM(GO). Three different module sizes for M^3-SVMs are compared. Column 2 shows the predefined module sizes. Column 3 shows the numbers of subproblems.

From this table, several observations could be made. First, both M^3-SVMs with random decomposition and with GO decomposition have higher TA and LA than traditional SVM. The M^3-SVMs improve not only average location accuracy but also total

Table 2. Overall accuracy of three methods on DBMLoc

Method	Module size	Module #	TA (%)	LA (%)	$aveF_1$ (%)
SVM	2344	13	64.7	41.4	42.0
M^3-SVM	100	848	65.1	48.0	40.7
(Random)	400	83	66.9	47.7	42.7
	800	38	66.2	45.2	43.4
M^3-SVM	100	848	66.2	**48.5**	42.3
(GO)	400	83	**67.1**	45.2	42.6
	800	38	66.3	43.4	**43.6**

accuracy, which indicates that they do not sacrifice majority classes for the classification of minority classes. Second, M^3-SVMs have higher $aveF_1$ than traditional SVM except M^3-SVM(R) with module size 100. And M^3-SVM(GO) with module size 800 achieved the highest $aveF_1$. As the module size goes down, LA increases, but $aveF_1$ decreases, which suggests a higher false positive rate of M^3-SVMs when the module size becomes smaller. There is a tradeoff between location accuracy and false positive rate. Finally, M^3-SVM(R) has lower TA and $aveF_1$ than M^3-SVM(GO) but higher LA, suggesting that it gives more preference to the minority classes. And its performance is relatively deteriorated than M^3-SVM(GO)'s when the module size is very small (100). Since random decomposition randomly divides each training class into equal size modules, while GO decomposition is based on the relationship between proteins according to GO, the latter method has a more stable performance.

Tables 3 and 4 list detailed recall and F_1 of traditional SVM and M^3-SVMs(GO) on each location. Obviously, the modulization helps improve recall a lot especially for the

Table 3. Recall comparison. a: SVM, b: M^3-SVM(GO) with module size 100, c: M^3-SVM(GO) with module size 400, d: M^3-SVM(GO) with module size 800. 1-13 correspond to the 13 subcellular locations listed in Table 1.

	Recall (%)												
Method	1	2	3	4	5	6	7	8	9	10	11	12	13
a	0.0	48.8	20.0	100.0	63.3	**84.7**	52.8	69.2	25.0	0.0	20.0	41.5	13.3
b	**5.6**	**62.8**	20.0	100.0	61.8	82.5	57.7	73.3	**50.0**	0.0	**60.0**	43.4	13.3
c	2.8	53.5	20.0	100.0	63.3	83.5	**59.3**	**75.0**	25.0	0.0	40.0	**45.3**	**20.0**
d	2.8	55.8	**26.7**	100.0	**66.4**	83.9	57.7	71.2	12.5	0.0	40.0	32.1	15.6

Table 4. F_1 comparison. a, b, c and d are the same as in Table 3

	F_1 (%)												
Method	1	2	3	4	5	6	7	8	9	10	11	12	13
a	0.0	44.7	31.6	**100.0**	67.5	81.1	52.2	72.7	16.7	0.0	22.2	38.9	18.5
b	**7.0**	42.9	24.0	80.0	66.7	**81.2**	50.4	76.0	**17.8**	0.0	**50.0**	34.3	20.0
c	4.5	44.2	27.3	80.0	67.4	80.7	51.0	**76.6**	12.5	0.0	44.4	**40.0**	**25.4**
d	4.8	**46.2**	**38.1**	**100.0**	**69.8**	80.6	**53.0**	74.0	8.0	0.0	40.0	30.9	20.9

minority classes, while the two biggest classes, membrane and Cytoplasm are recognized best by M^3-SVMs(GO) with module size 800 and traditional SVM, respectively.

As the F_1 values listed in Table 4 show, M^3-SVMs(GO) with module size 800 has the best performance. It wins on 5 locations, extracellular, ribosome, virion, membrane, and mitochondrion, among the four methods, and has higher F_1 than traditional SVM on 8 locations.

In this experiment, M^3-SVMs(GO) with module size 800 performs the best considering all the measures. A too small module size would result in numerous modules which increase computation cost and deteriorate the performance. However, we could specify different module sizes for different binary classification problems according to the ratio of training samples of the positive and negative classes.

In addition, we notice that all the classifiers failed to recognize vacuole protein from other proteins. One reason is that it has the least training samples (11 proteins) and only 4 test samples, thus the ratio of positive and negative data is too small to classify the positive data correctly. The other reason would be the current feature vectors do not contain features that are informative enough to discriminate vacuole proteins from others.

Response time should also be considered as an important factor when measuring the performance of a classifier. Table 5 exhibits a comparison of response time between traditional SVM and M^3-SVMs of different module sizes. We reported two categories of run times. 'Time1' is the response time (including training and test time) of the classifier running all subproblems in series. 'Time2' is the training time for a single subproblem which costs the longest time. In parallel learning, 'Time2' is more important. For traditional SVM, a subproblem means a two-class problem.

Table 5. Response time comparison

Method	$Time1$ (sec.)	$Time2$ (sec.)
SVM	23.4	3.3
M^3-SVM (100)	22.2	<0.1
M^3-SVM (400)	19.2	0.4
M^3-SVM (800)	19.7	1.3

M^3-SVMs obtained shorter response time than traditional SVM even in sequential running. Regarding 'Time1', M^3-SVM with the module size 400 is the most efficient for DBMLoc, because it achieves a tradeoff between the number and size of modules. For large-scale data, we can train modules in parallel, and choose a modules size as small as necessary.

4 Conclusion

This paper introduces an ensemble classifier for protein subcellular multi-localization. The classifier has several advantages in solving large-scale, class imbalance, multi-label problems. On the one hand, parallel and distributed training can be easily implemented because of its modularity. On the other hand, it has a balanced performance on all classes because various task decomposition strategies can be used.

Taking into account the GO information, we partitioned large classes into relatively smaller modules according to GO semantic similarity matrix. The experimental results show the effectiveness of the proposed GO decomposition method, and demonstrate that the M^3-SVM is very competent in solving such complex problems with class imbalance and multi-label characteristics.

References

1. Reinhardt, A., Hubbard, T.: Using neural networks for prediction of the subcellular location of proteins. Nucleic Acids Research 26(9), 2230–2236 (1998)
2. Fujiwara, Y., Asogawa, M., Nakai, K.: Prediction of Mitochondrial Targeting Signals Using Hidden Markov Model.. Genome Inform. Ser. Workshop Genome Inform. 8, 53–60 (1997)
3. Hua, S., Sun, Z.: Support vector machine approach for protein subcellular localization prediction. Bioinformatics 17(8), 721–728 (2001)
4. Chou, K.: Prediction of protein cellular attributes using pseudo-amino acid composition. Proteins Structure Function and Genetics 44(1), 60–60 (2001)
5. Yang, W.Y., Lu, B.L., Yang, Y.: A Comparative Study on Feature Extraction from Protein Sequences for Subcellular Localization Prediction. In: Proc. of the 2006 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, pp. 201–208 (2006)
6. Ashburner, M., Ball, C., Blake, J., Botstein, D., Butler, H., Cherry, J., Davis, A., Dolinski, K., Dwight, S., Eppig, J., et al.: Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. Nat. Genet. 25(1), 25–29 (2000)
7. Cai, Y.D., Chou, K.C.: Predicting 22 protein localizations in budding yeast. Biochem. Biophys. Res. Commun. 323(2), 425–428 (2004)
8. Yang, Y., Lu, B.-L.: Prediction of protein subcellular multi-locations with a min-max modular support vector machine. In: Wang, J., Yi, Z., Żurada, J.M., Lu, B.-L., Yin, H. (eds.) ISNN 2006. LNCS, vol. 3973, pp. 667–673. Springer, Heidelberg (2006)
9. Chen, K., Lu, B.L., Kwok, J.T.: Efficient Classification of Multi-label and Imbalanced Data using Min-Max Modular Classifiers. In: International Joint Conference on Neural Networks, pp. 1770–1775 (2006)
10. Boeckmann, B., Bairoch, A., Apweiler, R., Blatter, M., Estreicher, A., Gasteiger, E., Martin, M., Michoud, K., O'Donovan, C., Phan, I., et al.: The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. Nucleic Acids Research 31(1), 365–370 (2003)
11. Zhang, S., Xia, X., Shen, J., Zhou, Y., Sun, Z.: DBMLoc: a Database of proteins with multiple subcellular localizations. BMC Bioinformatics 9(1), 127 (2008)
12. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research 16, 321–357 (2002)
13. Wu, G., Chang, E.: Class-boundary alignment for imbalanced dataset learning. In: Proceedings of the ICML, vol. 3
14. Akbani, R., Kwek, S., Japkowicz, N.: Applying Support Vector Machines to Imbalanced Datasets. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) ECML 2004. LNCS, vol. 3201, pp. 39–50. Springer, Heidelberg (2004)
15. Vapnik, V.: Statistical learning theory. Wiley, Chichester (1998)
16. 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 10(5), 1244–1256 (1999)
17. Lu, B.L., Wang, K., Utiyama, M., Isahara, H.: A part-versus-part method for massively parallel training of support vector machines. In: Proceedings. IEEE International Joint Conference on Neural Networks, vol. 1, pp. 735–740 (2004)

18. 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, pp. 77–82 (2006)
19. Wang, J., Du, Z., Payattakool, R., Yu, P., Chen, C.: A new method to measure the semantic similarity of GO terms. Bioinformatics 23(10), 1274 (2007)
20. Karypis, G.: CLUTO-A Clustering Toolkit (2002)
21. Shamim, M., Anwaruddin, M., Nagarajaram, H.: Support Vector Machine-based classification of protein folds using the structural properties of amino acid residues and amino acid residue pairs. Bioinformatics 23(24), 3320 (2007)
22. McGuffin, L., Bryson, K., Jones, D.: The PSIPRED protein structure prediction server. Bioinformatics 16(4), 404–405 (2000)
23. Cheng, J., Randall, A., Sweredoski, M., Baldi, P.: SCRATCH: a protein structure and structural feature prediction server. Nucleic Acids Research 33, W72–W76 (2005)
24. Chang, C., Lin, C.: LIBSVM: a library for support vector machines. Software (2001), `http://www.csie.ntu.edu.tw/cjlin/libsvm`