Int. J. Mach. Learn. & Cyber. DOI XXXXXX
(will be inserted by the editor)

Predicting Subcellular Localization of Multi-Location Proteins by Improving Support Vector Machines with an Adaptive-Decision Scheme

Shibiao Wan · Man-Wai Mak

Received: xxx 2015 / Accepted: xxx 2015

Abstract From the perspective of machine learning, predicting subcellular localization of multi-location proteins is a multi-label classification problem. Conventional multi-label classifiers typically compare some pattern-matching scores with a fixed decision threshold to determine the number of subcellular locations in which a protein will reside. This simple strategy, however, may easily lead to over-prediction due to a large number of false positives. To address this problem, this paper proposes a more powerful multi-label predictor, namely AD-SVM, which incorporates an adaptive-decision (AD) scheme into multi-label support vector machine (SVM) classifiers. Specifically, given a query protein, a term-frequency based gene ontology vector is constructed by successively searching the gene ontology annotation database. Subsequently, the feature vector is classified by AD-SVM, which extends the binary relevance method with an adaptive decision scheme that essentially converts the linear SVMs to piecewise linear SVMs. Experimental results on two stringent benchmark datasets suggest that AD-SVM impressively outperforms existing state-of-the-art multi-location predictors. Results also show that the adaptive-decision scheme can effectively reduce over-prediction while having insignificant effect on the correctly predicted ones.

Keywords Adaptive decisions · Multi-label classification · protein subcellular localization · support vector machines

1 Introduction

Conventionally, predicting where a protein resides within a cell is a single-label classification problem, where each protein is assumed to be associated with one of the known subcellular locations only. These approaches are generally divided into two categories: (1) sequence-based methods, such as amino-acid composition methods [1,2,3], sorting-signal methods [4,5,6] and homology-based methods [7,8] and (2) knowledge-based methods, such as gene ontology (GO)[1] based methods [9,10,11,12,13], PubMed abstracts based methods [14,15] and Swiss-Prot keywords [16,17] based methods. The focus on predicting single-location proteins is probably driven by the large amount of data available in public databases such as UniProt, where a majority of proteins are assigned to a single location.

However, it is untenable to exclude the multi-location proteins or assume that multi-location proteins do not exist, because recent studies [18,19,20,21] show that there exist multi-location proteins that can simultaneously reside at, or move between, two or more different subcellular locations. Actually, proteins with multiple locations play important roles in some metabolic processes that take place in more than one compartment. For example, proteins involved in fatty acid β-oxidation are known to reside in peroxisome and mitochondria, and antioxidant defense proteins have been found in cytosol, mitochondria and peroxisome [22]. Another example is the glucose transporter GLUT4. This protein is

This work was in part supported by the RGC of Hong Kong SAR (Grant No. PolyU 152117/14E).

S. Wan (✉) · M. W. Mak (✉)
Department of Electronic and Information Engineering,
The Hong Kong Polytechnic University, Hong Kong SAR, China
E-mail: shibiao.wan@connect.polyu.hk,
enmwmak@polyu.edu.hk

[1] http://www.geneontology.org

regulated by insulin and is typically stored in the intracellular vesicles of adipocytes. However, it has also been found to translocate to the plasma membrane in response to insulin [23,24]. Thus, predicting where these proteins locate is a multi-label multi-class classification problem, where a protein may be associated with more than one subcellular location.

2 Multi-Label Classification

In the past decades, multi-label classification has received significant attention in a wide range of problem domains, such as music classification [25,26], video segmentation [27], functional genomics prediction [28,29,30], text categorization [31,32,33,34,35], and semantic annotation of images [36]. In functional genomics prediction, a gene is likely to associate with many functions. In text categorization, a document describing the politics may involve other topics, such as sports or education. Similarly, in music classification, a song may belong to more than one genre.

Multi-label classification is more complicated than single-label classification because of the large number of possible combinations of labels. Existing methods for multi-label classification can be grouped into two main categories: (1) problem transformation and (2) algorithm adaptation.

2.1 Problem-Transformation Methods

Problem transformation methods transform a multi-label learning problem into one or more single-label classification problems [36] so that traditional single-label classifiers can be applied without modification. Typical methods include binary relevance (BR) [37], ensembles of classifier chains (ECC) [38], label powerset (LP) [39] and compressive sensing [40].

Binary relevance (BR) is a popular problem-transformation method. It transforms a multi-label task into many binary classification tasks, one for each label. Given a query instance, its predicted label(s) are the union of the positive-class labels output by these binary classifiers. BR is effective, but it neglects the correlation between labels, which may carry useful information for multi-label classification.

The classifier chain method is a variant of BR but it can take the correlation between labels into account. Similar to BR, a set of one-vs-rest binary classifiers are trained. But unlike BR, the classifiers are linked in a chain and the feature vectors presented to the i-th classifier in the chain are augmented with the binary vectors representing the label(s) of the 1-st class to the $(i-1)$-th class. Therefore, label dependence is preserved through the feature space. Classification performance, however, depends on the chain order. This order-dependency can be overcome by ensembles of classifier chains [38].

Label powerset method reduces a multi-label task to a single-label task by treating each possible multi-label subset as a new class in the single-label classification task. This method is simple, but is likely to generate a large number of classes, many of which are associated with a few examples only. The compressive sensing approach is motivated by the fact that when the number of classes is large, the actual labels are often sparse. In other words, a typical query instance will belong to a few classes only, even though the total number of classes is large. This approach exploits the sparsity of the output (label) space by means of compressive sensing to obtain a more efficient output coding scheme for large-scale multi-label learning problems.

2.2 Algorithm-Adaptation Methods

Algorithm adaptation methods extend specific single-label algorithms to solve multi-label classification problems. Typical methods include multi-label C4.5 [41], AdaBoost.MH [33], and hierarchical multi-label decision trees [28].

The C4.5 algorithm [42] builds decision trees using the concept of information entropy. At each node of the tree, C4.5 chooses the feature that most effectively splits the data into two classes; in other words, the feature with the highest normalized information gain (or difference in entropy) is chosen to create a decision node. The C4.5 algorithm then recurs on the subclasses obtained by the previous step and the nodes thus obtained are added as the children of the node in the previous step. The multi-label C4.5 [41] uses the C4.5 algorithm as a baseline classifier and extends the definition of entropy to include multi-label data by estimating the number of bits needed to describe the membership or non-membership of each class. One disadvantage of this algorithm is that it only learns a set of accurate rules, not a complete classification. AdaBoost.MH is an extension of AdaBoost [43] for multi-label classification. It uses the one-vs-rest approach to convert an M-class problem into M 2-class AdaBoost problems in which an additional feature defined by the class labels is augmented to the input space.

In [28], class labels are organized in a hierarchy and for each class, a binary decision tree is learned in a hierarchical way. A sample belongs to a class means that it also belongs to the superclasses of that class. This parent-children relationship enables the decision tree to predict multi-label instances.

Several algorithms based on support vector machines (SVM) [44] have been proposed to tackle multi-label classification problems. In [45], a ranking algorithm for multi-label classification is proposed. It uses the ranking loss [33] as the cost function, which is defined as the average fraction of pairs of labels that are ordered incorrectly. Similar to SVMs, it finds the optimal hyperplane with the maximum margin of separation. One major disadvantage of this method is that it does not output a set of labels. The SVM classifiers in [46] adopt the BR method by extending the feature vectors of the original data set with some additional features indicating the relationship between classes.

2.3 Problem-Transformation vs. Algorithm-Adaptation

Compared to algorithm adaptation methods, one advantage of problem transformation methods is that any algorithm which is not capable of dealing with multi-label classification problems can be easily extended to deal with multi-label classification via transformation.

It should be pointed out that the multi-label classification methods are different from the multi-class classification methods, such as error-correcting-output-coding methods [47] and pairwise comparison methods [48]. There is probably no multi-class method that outperforms all others in all circumstances [49], so is the same case for multi-label methods.

2.4 Applications to Protein Subcellular Localization

Existing state-of-the-art multi-label predictors – including Virus-mPLoc [50], Plant-mPLoc [51], iLoc-Virus [52], iLoc-Plant [53], mGOASVM [54], HybridGO-Loc [55], R3P-Loc [56], mPLR-Loc [57] and others [58, 59, 60, 61, 62] – use the GO information as the features and apply different multi-label classifiers to tackle the multi-label problems. Among these predictors, Virus-mPLoc, Plant-mPLoc, iLoc-Virus and iLoc-Plant use algorithm adaptation methods, while mGOASVM, HybridGO-Loc, R3P-Loc and mPLR-Loc use problem transformation methods. However, to determine the number of subcellular locations of a query protein, most of the multi-label classifiers compare the pattern-matching scores with a fixed decision threshold. This simple strategy is liable to a large number of false positives and thus weaken the generalization capabilities of the multi-label classifiers.

To address this problem, this paper extends our earlier work on adaptive thresholding [63] and proposes an adaptive-decision based multi-label classifier, namely AD-SVM, to predict subcellular localization of both single- and multi-location proteins. Specifically, given a query protein, a successive-search strategy is used to search against the gene ontology annotation (GOA) database with either the accession number (AC) or the homologous AC of the query protein as the key, so that each protein will be associated with at least one GO term. A feature vector is subsequently formulated with the term-frequency based GO information, which is then classified by the proposed AD-SVM. AD-SVM extends binary relavance methods with an adaptive-decision scheme that essentially converts the linear SVMs into piecewise linear SVMs, which can effectively reduce the false positives without affecting the correctly predicted ones. Experimental results on two benchmark datasets demonstrate the superiority of AD-SVM over existing state-of-the-art predictors.

3 Feature Extraction

To extract relevant features, AD-SVM performs two steps: (1) retrieval of GO terms and (2) construction of GO vectors.

3.1 Retrieval of GO Terms

A major issue in GO-based predictors is that GO information is not always available to every protein. AD-SVM searches the GO information from the GOA database,[2] which uses standardized GO vocabularies to systematically annotate non-redundant proteins from the UniProt database. For proteins with known accession numbers (ACs), their respective GO terms are retrieved from the GOA database using the ACs as the searching keys. For a protein without an AC, its amino acid (AA) sequence is presented to BLAST [64] to find its homologs, whose ACs are then used as keys to search against the GOA database. Therefore, given a query protein, AD-SVM can handle two possible cases: (1) the AC is known and (2) the amino acid (AA) sequence is known.

While the GOA database allows us to associate the AC of a protein with a set of GO terms, for some novel proteins, neither their ACs nor the ACs of their top homologs have any entries in the GOA database; in other words, the GO vectors constructed in Section 3.2 will contain all-zero, which are meaningless for classification. In such case, AD-SVM uses a successive-search strategy as follows. The ACs of the homologous proteins, as returned from BLAST search, are successively used to search against the GOA database until a match

[2] http://www.ebi.ac.uk/GOA

is found. Specifically, for the proteins whose top homologs do not have any GO terms in the GOA database, AD-SVM uses the second-top homolog to find the GO terms; similarly, for the proteins whose top and 2-nd homologs do not have any GO terms, the third-top homolog was used; and so on until all the query proteins can correspond to at least one GO term.

With the rapid progress of the GOA database [65], it is reasonable to assume that the homologs of the query proteins have at least one GO term [66]. Thus, it is not necessary to use back-up methods to handle the situation where no GO terms can be found.

3.2 Construction of GO Vectors

Given a dataset, we used the procedure described in Section 3.1 to retrieve the GO terms of all of its proteins. Let $\mathbb{W}$ denotes a set of distinct GO terms corresponding to a data set. $\mathbb{W}$ is constructed in two steps: (1) identifying all of the GO terms in the dataset and (2) removing the repetitive GO terms. Suppose W distinct GO terms are found, i.e., $|\mathbb{W}|= W$; these GO terms form a GO Euclidean space with W dimensions. For each sequence in the dataset, a GO vector is constructed by matching its GO terms against $\mathbb{W}$, using the number of occurrences of individual GO terms in $\mathbb{W}$ as the coordinates.

Similar to our earlier works [54,67], the GO frequency information is used to construct GO feature vectors. Specifically, the GO vector $\mathbf{q}_i$ of the i-th protein $\mathbb{Q}_i$ is defined as:

$$\mathbf{q}_i = [b_{i,1}, \cdots, b_{i,j}, \cdots, b_{i,T}]^{\mathsf{T}}, b_{i,j} = \begin{cases} f_{i,j} & \text{, GO hit} \\ 0 & \text{, otherwise} \end{cases} \tag{1}$$

where $f_{i,j}$ is the number of occurrences of the j-th GO term (term-frequency) in the i-th protein sequence. Detailed information can be found in [54,67].

4 Adaptive-Decision Based Support Vector Machines

After feature extraction, the term-frequency based GO vectors are classified by our proposed classifier, AD-SVM, which is elaborated below.

4.1 Multi-label SVM Scoring

GO vectors, as computed in Eq. 1, are used for training the multi-label one-vs-rest SVMs. Specifically, for an M-class problem (here M is the number of subcellular locations), M independent binary SVMs are trained, one for each class. Denote the GO vector created by using the true AC of the i-th query protein as $\mathbf{q}_{i,0}$ and the GO vector created by using the accession number of the k-th homolog as $\mathbf{q}_{i,k}$, $k = 1, \ldots, k_{\max}$, where $k_{\max}$ is the number of homologs retrieved by BLAST with the default parameter setting. Then, given the i-th query protein $\mathbb{Q}_i$, the score of the m-th SVM is:

$$s_m(\mathbb{Q}_i) = \sum_{r \in \mathcal{S}_m} \alpha_{m,r} y_{m,r} K(\mathbf{p}_r, \mathbf{q}_{i,h}) + b_m, \tag{2}$$

where

$$h = \min \Big\{ k \in \{0, \ldots, k_{\max}\} \text{ s.t. } ||\mathbf{q}_{i,k}||_0 \neq 0 \Big\}, \tag{3}$$

and $\mathcal{S}_m$ is the set of support vector indexes corresponding to the m-th SVM, $y_{m,r} \in \{-1, +1\}$ are labels such that $y_{m,r} = 1$ if $\mathbb{Q}_i$ belongs to the m-th class, $\alpha_{m,r}$ are the Lagrange multipliers, $K(\cdot, \cdot)$ is a kernel function; here, the linear kernel is used. Note that $\mathbf{p}_r$'s in Eq. 2 represents the GO training vectors, which may include the GO vectors created by using the true AC of the training sequences or their homologous ACs.

4.2 Adaptive Decision for SVM (AD-SVM)

To predict the subcellular locations of datasets containing both single-label and multi-label proteins, an adaptive decision scheme for multi-label SVM classifiers is proposed. Unlike the single-label problem where each protein has one predicted label only, a multi-label protein could have more than one predicted labels. Thus, the predicted subcellular location(s) of the i-th query protein are given by:
If $\exists \ s_m(\mathbb{Q}_i) > 0$,

$$\mathcal{M}_l(\mathbb{Q}_t) = \bigcup_{m=1}^{M} \left(m : s_m(\mathbb{Q}_t) \geq \min\{1.0, f(s_{\max}(\mathbb{Q}_t))\} \right) \tag{4}$$

otherwise,

$$\mathcal{M}(\mathbb{Q}_i) = \arg \max_{m=1}^{M} s_m(\mathbb{Q}_i). \tag{5}$$

In Eq. 4, $f(s_{\max}(\mathbb{Q}_i))$ is a function of $s_{\max}(\mathbb{Q}_i)$, where $s_{\max}(\mathbb{Q}_i) = \max_{m=1}^{M} s_m(\mathbb{Q}_i)$. In [63], a linear function was used, i.e. ,

$$f(s_{\max}(\mathbb{Q}_i)) = \theta s_{\max}(\mathbb{Q}_i), \tag{6}$$

where $\theta \in [0.0, 1.0]$ is a parameter. Because $f(s_{\max}(\mathbb{Q}_i))$ is linear, Eq. 4 and Eq. 5 turn the linear SVMs into piecewise linear SVMs. Eq. 4 also suggests that the predicted labels depend on $s_{\max}(\mathbb{Q}_i)$, a function of the test instance (or protein). This means that the decision and the corresponding threshold are adaptive to the test protein. For ease of reference, we refer to the proposed predictor as AD-SVM.

4.3 Analysis of AD-SVM

To facilitate discussion, let's define two terms: *over-prediction* and *under-prediction*. Specifically, over (resp. under) prediction means that the number of predicted labels of a query protein is larger (resp. smaller) than the ground-truth. In this paper, both over- and under-predictions are considered as incorrect predictions, which will be reflected in the "overall actual accuracy (OAA)" to be defined in Section 5.2.

Conventional methods use a fixed threshold to determine the predicted classes. When the threshold is too small, the prediction results are liable to over-prediction; on the other hand, when the threshold is too large, the prediction results are susceptible to under-prediction. To overcome this problem, the adaptive decision scheme in the classifier uses the maximum score ($s_{\max}(\mathbb{Q}_i)$) among the one-vs-rest SVMs in the classifier as a reference. In particular, $s_{\max}(\mathbb{Q}_i)$ in Eq. 4 adaptively normalizes the scores of all one-vs-rest SVMs so that for SVMs to be considered as runner-ups, they need to have a sufficiently large score relative to the winner. This strategy effectively reduces the chance of over-prediction. The first condition in Eq. 4 ($s_m(\mathbb{Q}_i) > 1$) aims to avoid under-prediction when the winning SVM has very high confidence (i.e., $s_{\max}(\mathbb{Q}_i) \gg 1$) but the runners-up still have enough confidence ($s_m(\mathbb{Q}_i) > 1$) in making a right decision.[3] On the other hand, when the maximum score is small (say $0 < s_{\max}(\mathbb{Q}_i) \leq 1$), θ in the second term of Eq. 4 can strike a balance between over-prediction and under-prediction. When all of the SVMs have very low confidence (say $s_{\max}(\mathbb{Q}_i) < 0$), the classifier switches to single-label mode via Eq. 5.

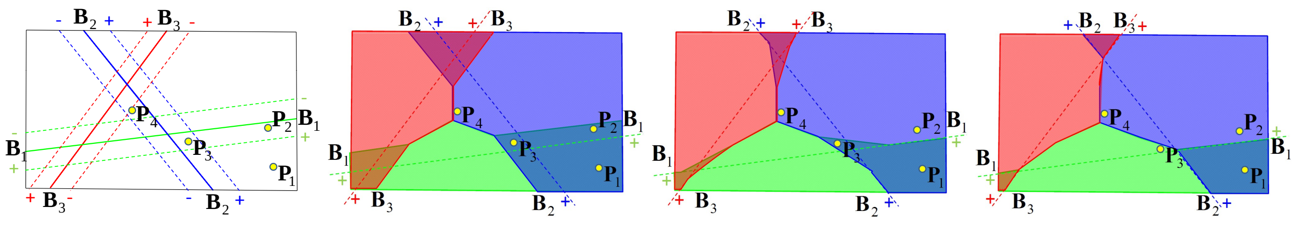
To illustrate how this decision scheme works, an example is shown in Fig. 1. Suppose there are 4 test data points ($\mathbf{P}_1, \ldots, \mathbf{P}_4$) which are possibly distributed into 3 classes: {green, blue, red}. The decision boundaries of individual SVMs and the 4 points are shown in Fig. 1(a). Suppose $s_m(\mathbf{P}_i)$ is the SVM score of $\mathbf{P}_i$ with respect to the class m, where $i = \{1, \ldots, 4\}$ and $m \in$ {green, blue, red}. Fig. 1(a) suggests the following conditions:

$$\begin{array}{ll}
s_{\text{green}}(\mathbf{P}_1) > 1, & s_{\text{blue}}(\mathbf{P}_1) > 1,\ s_{\text{red}}(\mathbf{P}_1) < 0; \\
0 < s_{\text{green}}(\mathbf{P}_2) < 1, & s_{\text{blue}}(\mathbf{P}_2) > 1,\ s_{\text{red}}(\mathbf{P}_2) < 0; \\
0 < s_{\text{green}}(\mathbf{P}_3) < 1, & 0 < s_{\text{blue}}(\mathbf{P}_3) < 1,\ s_{\text{red}}(\mathbf{P}_3) < 0; \\
s_{\text{green}}(\mathbf{P}_4) < 0, & s_{\text{blue}}(\mathbf{P}_4) < 0,\ s_{\text{red}}(\mathbf{P}_4) < 0.
\end{array}$$

Note that points whose scores lie between 0 and 1 are susceptible to over-prediction because they are very close to the decision boundaries of the corresponding SVM. The decision scheme used in Eqs. 4–6 (i.e., $\theta = 0.0$) leads to the decision boundaries shown in Fig. 1(b). Based on these boundaries, $\mathbf{P}_1$, $\mathbf{P}_2$ and $\mathbf{P}_3$ will be assigned to class green $\cap$ blue , and $\mathbf{P}_4$ will be assigned to the class with the highest SVM score (using Eq. 5). If θ increases to 0.5, the results shown in Fig. 1(c) will be obtained. The assignments of $\mathbf{P}_1$, $\mathbf{P}_3$ and $\mathbf{P}_4$ remain unchanged but $\mathbf{P}_2$ will be changed from class green $\cap$ blue to class blue. Similarly, when θ increases to 1.0 (Fig. 1(d)), then the class of $\mathbf{P}_3$ will also be determined by the SVM with the highest score. This analysis suggests that when θ increases from 0 to 1, the decision criterion becomes more stringent, which has the effect of shrinking the 2-label regions in Fig. 1, thus reducing the over-prediction. Provided that θ is not close to 1, this reduction in over-prediction will not compromise the decisions made by the high scoring SVMs.

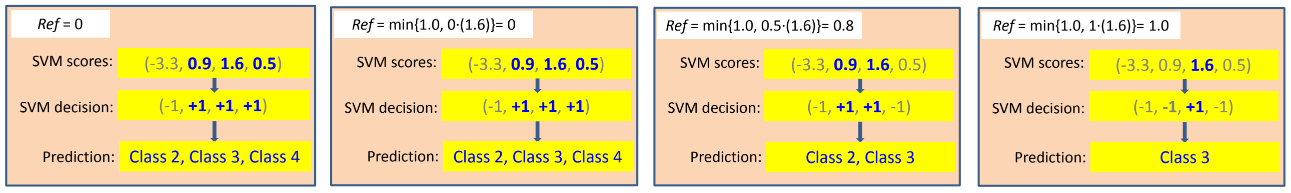
To further exemplify the strategy of the adaptive decision scheme, a four-class multi-label example demonstrating how the adaptive decision scheme works is shown in Fig. 2. In the training phase, four independent binary SVMs are first trained for the four-class problem, one for each class. The training GO vectors (not shown) participate in all of the four binary SVMs. However, contrary to the multi-class SVM classifier where each training vector has the positive label only in one binary SVM and has negative labels in the remaining binary SVMs, a training vector in the multi-label SVM classifier may have the positive label in more than one binary SVMs. Here, we only use one query protein to demonstrate the adaptive scheme. Fig. 2(a) shows the testing phase of the baseline predictor, i.e. mGOASVM, and Fig. 2(b)–(d) show the testing phases of the adaptive decision schemes with θ in Eq. 6 equal to 0, 0.5 and 1, respectively. As shown in Fig. 2(b), when $\theta = 0$, the adaptive scheme is the same as the decision scheme in mGOASVM, with the reference or SVM score threshold equal to 0. In other words, if there is any positive SVM score, the query protein will be assigned to the corresponding class. Thus, in this case, the query protein is assigned to Class 2, Class 3 and Class 4. When $\theta = 0.5$ (Fig. 2(c)), the reference score becomes $Ref = \min\{1.0, 0.5 \cdot s_{\max}(\mathbb{Q}_i)\} = \min\{1.0, 0.5 \cdot (1.6)\} = 0.8$, namely only those classes whose SVM scores are larger than 0.8 will be predicted as positive. In this

[3] SVM scores larger than one means that the test proteins fall beyond the margin of separation; therefore, the confidence is fairly high.



(a) (b) $\theta = 0.0$ (c) $\theta = 0.5$ (d) $\theta = 1.0$

Fig. 1 A 3-class example illustrating how the adaptive decision scheme changes the decision boundaries from linear to piecewise linear and how the resulting SVMs assign label(s) to test points when θ in Eq. 6 changes from 0 to 1. In (a), the solid and dashed lines respectively represent the decision boundaries and margins of individual SVMs. In (b)–(d), the input space is divided into three 1-label regions (green, blue and red) and three 2-label regions (green ∩ blue, blue ∩ red, and red ∩ green).


Ref = 0
SVM scores: (-3.3, 0.9, 1.6, 0.5)
SVM decision: (-1, +1, +1, +1)
Prediction: Class 2, Class 3, Class 4
Ref = min{1.0, 0·(1.6)}= 0
SVM scores: (-3.3, 0.9, 1.6, 0.5)
SVM decision: (-1, +1, +1, +1)
Prediction: Class 2, Class 3, Class 4
Ref = min{1.0, 0.5·(1.6)}= 0.8
SVM scores: (-3.3, 0.9, 1.6, 0.5)
SVM decision: (-1, +1, +1, -1)
Prediction: Class 2, Class 3
Ref = min{1.0, 1·(1.6)}= 1.0
SVM scores: (-3.3, 0.9, 1.6, 0.5)
SVM decision: (-1, -1, +1, -1)
Prediction: Class 3

(a) (b) $\theta = 0.0$ (c) $\theta = 0.5$ (d) $\theta = 1.0$

Fig. 2 A 4-class example showing how the adaptive decision scheme works when θ in Eq. 6 changes from 0 to 1. (a) The testing phase of the decision scheme of mGOASVM. (b)–(d) The testing phases of the adaptive decision schemes with θ in Eq. 6 equal to 0, 0.5 and 1, respectively. *Ref*: the reference or the SVM score threshold, over which the corresponding class label(s) are assigned to the query protein.

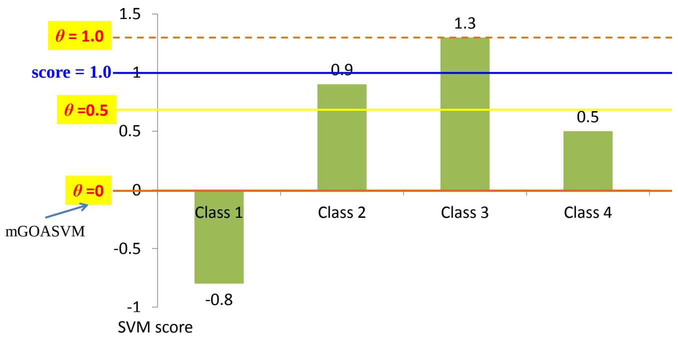


Fig. 3 An example showing how the adaptive decision scheme works when θ in Eq. 6 changes from 0 to 1. The ordinate represents the SVM score, and the abscissa represents the classes.

case, the query protein will be predicted to locate in Class 2 and Class 3. When we increase θ to 1, as shown in Fig. 2(d), the reference score will become $Ref = \min\{1.0, 1 \cdot s_{\max}(\mathbb{Q}_i)\} = \min\{1.0, 1\cdot(1.6)\} = 1.0$. In this case, only the 3-rd SVM score is larger than 1.0. Therefore, the query protein will be considered as a single-location protein and is predicted to locate in Class 3.

This four-class multi-label problem can also be shown in another way as in Fig. 3. From Fig. 3, we can clearly see that when $\theta = 0$ (the orange solid line), or the decision scheme of mGOASVM, there will be three classes (Classes 2, 3 and 4) passing the criterion; when $\theta = 0.5$ (the yellow solid line), only Class 2 and Class 3 can pass the criterion; when θ increases to 1, then the criterion becomes 1.0 (the solid blue line), and only Class 3 will be the predicted label for the query protein. This suggests that with θ increases, the decision scheme becomes stringent and the over-predictions will be reduced.

5 Experiments

5.1 Datasets

A virus dataset [50,52] and a plant dataset [53] were used to evaluate the performance of the proposed predictors. The virus and the plant datasets were created from Swiss-Prot 57.9 and 55.3, respectively. The virus dataset contains 207 viral proteins distributed in 6 locations. Of the 207 viral proteins, 165 belong to one subcellular locations, 39 to two locations, 3 to three locations and none to four or more locations. This means that about 20% of the proteins in the dataset are located in more than one subcellular location. The plant dataset contains 978 plant proteins distributed in 12 locations. Of the 978 plant proteins, 904 belong to one subcellular locations, 71 to two locations, 3 to three locations and none to four or more locations. The sequence identity of both datasets was cut off at 25%.

The breakdown of these two datasets are listed in Figs. 4(a) and 4(b). Fig. 4(a) shows that the majority (68%) of viral proteins in the virus dataset are located in host cytoplasm and host nucleus while proteins located in the rest of the subcellular locations totally account for only around one third. This means that this multi-label dataset is imbalanced across the six subcellular locations. Similar conclusions can be drawn from Fig. 4(b), where most of the plant proteins exist in chloroplast, cytoplasm, nucleus and mitochondrion, while proteins in other 8 subcellular locations totally account for less than 30%. This imbalanced property makes the prediction of these two multi-label datasets difficult. More detailed statistical properties of these two datasets are listed in Table 1.

In Table 1, M and N denote the number of actual (or distinct) subcellular locations and the number of actual (or distinct) proteins. Besides the commonly used properties for single-label classification, the following measurements [39] are used as well to explicitly quantify the multi-label properties of the datasets: label cardinality (LC), label density (LD), distinct label set (DLS), proportion of distinct label set ($PDLS$) and total locative number (TLN). The detailed definitions of these measurements can be found in [57].

Among these measurements, LC is used to measure the degree of multi-labels in a dataset. For a single-label dataset, $LC = 1$; for a multi-label dataset, $LC > 1$. And the larger the LC, the higher the degree of multi-labels. LD takes into consideration the number of classes in the classification problem. For two datasets with the same LC, the lower the LD, the more difficult the classification. DLS represents the number of possible label combinations in the dataset. The higher the DLS, the more complicated the composition. $PDLS$ represents the degree of distinct labels in a dataset. The larger the $PDLS$, the more probable the individual label-sets are different from each other. From Table 1, we notice that although the number of proteins in the virus dataset ($N = 207, TLN = 252$) is smaller than that of the plant dataset ($N = 978, TLN = 1055$), the former ($LC = 1.2174, LD = 0.2029$) is a denser multi-label dataset than the latter ($LC = 1.0787, LD = 0.0899$).

5.2 Performance Metrics

Compared to traditional single-label classification, multi-label classification requires more complicated performance metrics to better reflect the multi-label capabilities of classifiers. These measures include *Accuracy, Precision, Recall, F1-score (F1)* and *Hamming Loss (HL)*. The definitions of these five measurements can be found in [55].

Accuracy, Precision, Recall and *F1* indicate the classification performance. The higher the measures, the better the prediction performance. Among them, *Accuracy* is the most commonly used criteria. *F1-score* is the harmonic mean of *Precision* and *Recall*, which allows us to compare the performance of classification systems by taking the trade-off between *Precision* and *Recall* into account. The *Hamming Loss (HL)* [68,69] is different from other metrics. When all of the proteins are correctly predicted, $HL = 0$; whereas, other metrics will be equal to 1. On the other hand, when the predictions of all proteins are completely wrong, $HL = 1$; whereas, other metrics will be equal to 0. Therefore, the lower the *HL*, the better the prediction performance.

Two additional measurements [52,54] are often used in multi-label subcellular localization prediction. They are overall locative accuracy (OLA) and overall actual accuracy (OAA). Specifically, denote $\mathcal{L}(\mathbb{Q}_i)$ and $\mathcal{M}(\mathbb{Q}_i)$ as the true label set and the predicted label set for the i-th protein $\mathbb{Q}_i$ ($i = 1, \ldots, N$), respectively.[4], then OLA is given by:

$$OLA = \frac{1}{\sum_{i=1}^{N} |\mathcal{L}(\mathbb{Q}_i)|} \sum_{i=1}^{N} |\mathcal{M}(\mathbb{Q}_i) \cap \mathcal{L}(\mathbb{Q}_i)|, \tag{7}$$

and the overall actual accuracy (OLA) is:

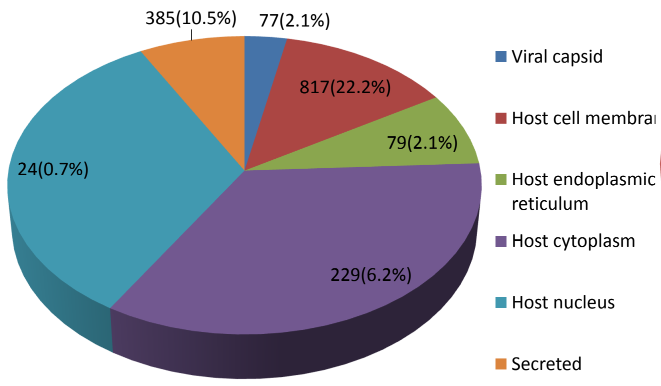
$$OAA = \frac{1}{N} \sum_{i=1}^{N} \Delta[\mathcal{M}(\mathbb{Q}_i), \mathcal{L}(\mathbb{Q}_i)] \tag{8}$$

where

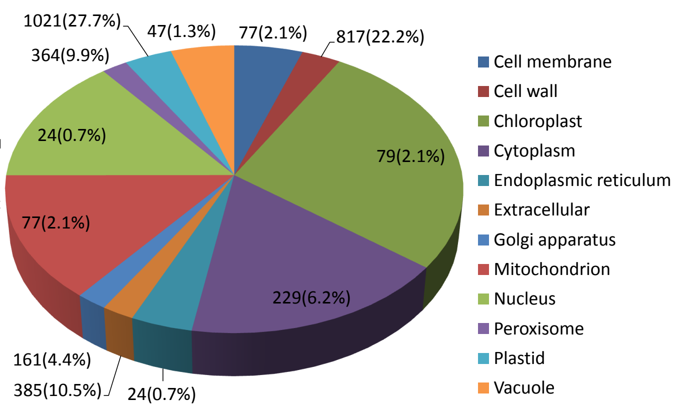
$$\Delta[\mathcal{M}(\mathbb{Q}_i), \mathcal{L}(\mathbb{Q}_i)] = \begin{cases} 1 \text{ , if } \mathcal{M}(\mathbb{Q}_i) = \mathcal{L}(\mathbb{Q}_i) \\ 0 \text{ , otherwise.} \end{cases} \tag{9}$$

According to Eq. 7, a locative protein is considered to be correctly predicted if any of the predicted labels matches any labels in the true label set. On the other hand, Eq. 8 suggests that an actual protein is considered to be correctly predicted only if *all* of the predicted labels match those in the true label set exactly. For example, for a protein coexist in, say, three subcellular locations, if only two of the three are correctly predicted, or the predicted result contains a location not belonging to the three, the prediction is considered to be incorrect. In other words, when and only when all the subcellular locations of a query protein are exactly predicted without any overprediction or underprediction, can the prediction be considered as correct. Therefore, OAA is a more stringent measure as compared to OLA. OAA is also more objective than OLA. This is because locative accuracy is liable to give biased performance measure when the predictor tends to over-predict, i.e., giving large $|\mathcal{M}(\mathbb{Q}_i)|$ for many $\mathbb{Q}_i$. In the extreme case, if every protein is predicted to have all of the M subcellular locations, according to Eq. 7, the OLA is 100%. But obviously, the predictions are wrong and meaningless. On the contrary, OAA is 0% in this extreme case, which definitely reflects the real performance.

[4] Here, $N = 207$ for the virus dataset and $N = 978$ for the plant dataset.



(a) virus dataset



(b) plant dataset

Fig. 4 Breakdown of (a) the virus dataset and (b) the plant dataset. The number of proteins shown in each subcellular location represents the number of 'locative proteins' [52,54]. Here, in (a), 207 actual proteins have 252 locative proteins; in (b), 978 actual proteins have 1055 locative proteins.

Table 1 Statistical properties of the two datasets used in our experiments. M: number of subcellular locations; N: number of actual proteins; LC: label cardinality; LD: label density; DLS: distinct label set; $PDLS$: proportion of distinct label set; and TLN: total locative number.

Dataset	M	N	LC	LD	DLS	$PDLS$	TLN
Virus	6	207	1.2174	0.2029	17	0.0821	252
Plant	12	978	1.0787	0.0899	32	0.0327	1055

Among all the metrics mentioned above, OAA is the most stringent and objective. This is because if some (but not all) of the subcellular locations of a query protein are correctly predict, the numerators of the other 4 measures are non-zero, whereas the numerator of OAA in Eq. 8 is 0 (thus contribute nothing to the frequency count). Note that OAA and HL are equivalent to *absolute-true* and *absolute-false*, respectively, used in [70].

In statistical prediction, leave-one-out cross validation (LOOCV) is considered to be the most rigorous and bias-free method [71]. Hence, LOOCV was used to examine the performance of AD-SVM.

6 Results and Analysis

6.1 Effect of Adaptive Decisions on OAA

Because OAA is the most objective and stringent criteria of all the performance metrics, we first analyze the effect of the adaptive-decision parameter θ (in Eq. 6) on OAA using the two benchmark datasets. Fig. 5 shows the OAA of AD-SVM on the virus dataset and the plant dataset with respect to the adaptive-decision parameter θ based on leave-one-out cross-validation. As can be seen, for the virus dataset, as θ increases from 0.0 to 1.0, the overall actual accuracy increases first, reaches the peak at $\theta = 0.3$ (with an actual accuracy of 93.2%), and then decreases.

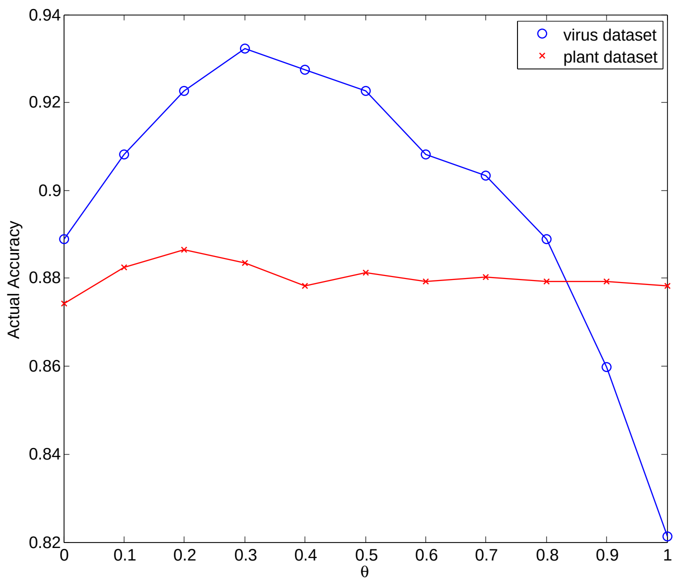


Fig. 5 OAA of AD-SVM based on leave-one-out cross-validation (LOOCV) varying with θ using the virus and plant datasets, respectively. $\theta = 0$ represents the performance of mGOASVM.

An analysis of the predicted labels $\{\mathcal{L}(\mathbf{P}_i); i = 1, \ldots, N\}$ suggests that the increases in OAA is due to the reduction in the number of over-prediction, i.e., the number of cases where $|\mathcal{M}(\mathbf{P}_i)| > |\mathcal{L}(\mathbf{P}_i)|$ has been reduced.

When $\theta > 0.3$, the benefit of reducing the over-prediction diminishes because the criterion in Eq. 4 becomes so stringent that some of the proteins were under-predicted, i.e., the number of cases where $|\mathcal{M}(\mathbf{P}_i)| < |\mathcal{L}(\mathbf{P}_i)|$ increases. Note that the performance at $\theta = 0.0$ is equivalent to the performance of mGOASVM, and that the best *OAA* (93.2% when $\theta = 0.3$) obtained by the proposed decision scheme is more than 4% (absolute) higher than mGOASVM (88.9%).

For the plant dataset, when θ increases from 0.0 to 1.0, the overall actual accuracy increases from 87.4%, and then fluctuates around 88%. If we take the same θ as that for the virus dataset, i.e., $\theta = 0.3$, the performance of AD-SVM is 88.3%, which is still better than that of mGOASVM at $\theta = 0.0$.

6.2 Effect of Adaptive Decisions on All Performance Metrics

Then, we extended the analysis from *OAA* to all of the performance metrics. Fig. 6(a) shows all of the performance metrics of AD-SVM on the virus dataset for different values of θ based on leave-one-out cross-validation (LOOCV). Note that when $\theta = 0.0$, AD-SVM is equivalent to mGOASVM [54]. As can be seen, as θ increases from 0.0 to 1.0, the *OAA* of AD-SVM increases first, reaches the peak at $\theta = 0.3$, with $OAA = 0.932$, which is more than 4% (absolute) higher than mGOASVM (0.889). The *Precision* increases until $\theta = 0.6$ and then remains almost unchanged when $\theta \geq 0.6$. On the contrary, *OLA* and *Recall* peak at $\theta = 0.0$, and these measures drop almost linearly with θ until $\theta = 1.0$. Among these metrics, no matter how θ changes, *OAA* is no higher than other five measurements.

An analysis of the predicted labels $\{\mathcal{L}(\mathbf{P}_i); i = 1, \ldots, N\}$ suggests that the increase in *OAA* is due to the reduction in the number of over-prediction, i.e., the number of cases where $|\mathcal{M}(\mathbf{P}_i)| > |\mathcal{L}(\mathbf{P}_i)|$. When $\theta > 0.3$, the benefit of reducing the over-prediction diminishes because the criterion in Eq. 4 becomes so stringent that some of the proteins were under-predicted, i.e., the number of cases where $|\mathcal{M}(\mathbf{P}_i)| < |\mathcal{L}(\mathbf{P}_i)|$. When θ increases from 0.0 to 0.3, the number of cases where $|\mathcal{M}(\mathbf{P}_i)| > |\mathcal{L}(\mathbf{P}_i)|$ decreases while at the same time $|\mathcal{M}(\mathbf{P}_i) \cap \mathcal{L}(\mathbf{P}_i)|$ remains almost unchanged. In other words, the denominators of *Accuracy* and *F1-score* decrease while the numerators for both metrics remain almost unchanged, leading to better performance for both metrics. When $\theta > 0.3$, for the similar reason mentioned above, the increase in under-prediction outweighs the benefit of the decrease in over-prediction, causing performance loss. For *Precision*, when $\theta > 0.3$, the loss due to the stringent criterion is counteracted by the gain due to the reduction in $|\mathcal{M}(\mathbf{P}_i)|$, the denominator of *Precision*. Thus, the *Precision* increases monotonically when θ increases from 0 to 1. However, *OLA* and *Recall* decrease monotonically with respect to θ because the denominator of these measures is independent of $|\mathcal{M}(\mathbf{P}_i)|$ and the number of correctly predicted labels in the numerator decreases when the decision criterion is getting stricter.

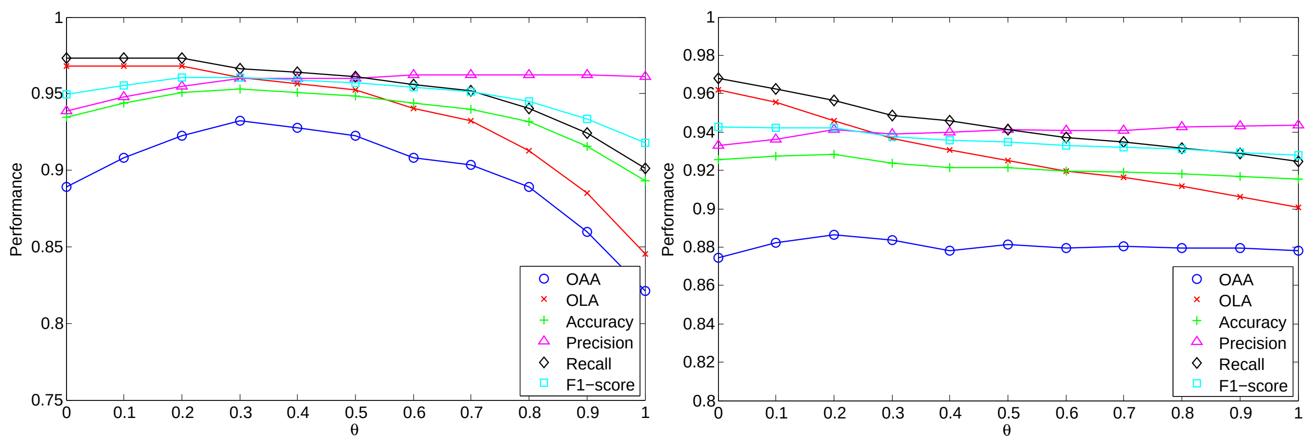
Fig. 6(b) shows the performance of AD-SVM on the plant dataset for different values of θ based on LOOCV. In Fig. 6(a), when θ increases from 0.0 to 1.0, the *OAA* of AD-SVM increases from 0.874, and then fluctuates around 0.880. The *OAA* (0.887) of AD-SVM peaks at $\theta = 0.2$. This suggests that the optimal value of θ is dataset-dependent. Other metrics show similar performance trend as those in the virus dataset when θ varies from 0.0 to 1.0. Similar analysis mentioned above can be applied to those for the plant dataset.

When comparing Fig. 6(b) with Fig. 6(a), we found that the performance metrics for the plant dataset is less sensitive to the change of θ than those for the virus dataset. But the *OAA* can be improved at a certain optimal value when θ varies from 0 to 1 for both datasets.

6.3 Comparing AD-SVM with State-of-the-art Predictors

Table 2 and Table 3 compare the performance of AD-SVM against state-of-the-art predictors on the virus and plant dataset, respectively. All of the predictors use the information of GO terms as features. From the classification perspective, both Virus-mPLoc [50] and Plant-mPLoc [51] use ensemble OET-KNN (optimized evidence-theoretic K-nearest neighbors) classifiers; both iLoc-Virus [52] and iLoc-Plant [53] use multi-label KNN classifiers; mGOASVM [54] uses a multi-label SVM classifier; and the proposed AD-SVM uses a multi-label SVM classifier incorporated with the proposed adaptive decision scheme.

As shown in Table 2, AD-SVM significantly outperforms Virus-mPLoc and iLoc-Virus. Both the *OLA* and *OAA* of AD-SVM are more than 15% (absolute) higher than iLoc-Virus. Although the *OLA* of AD-SVM is slightly smaller than that of mGOASVM, the *OAA* of AD-SVM is more than 4% (absolute) higher than that of mGOASVM. In terms of *Accuracy, Precision, F1* and *HL*, AD-SVM performs better than mGOASVM. In terms of *Recall*, mGOASVM performs the better. This is understandable because according to the analysis in the Section 4.2, the *Recall* decreases when θ increases. The results suggest that the multi-label SVM classifiers using the proposed adaptive decision scheme perform



(a) virus dataset

(b) plant dataset

Fig. 6 Performance of AD-SVM based on leave-one-out cross-validation (LOOCV) varying with θ on (a) the virus dataset and (b) the plant dataset, respectively. $\theta = 0$ represents the performance of mGOASVM.

Table 2 Comparing AD-SVM with state-of-the-art multi-label predictors based on leave-one-out cross validation (LOOCV) using the virus dataset.

Label	Subcellular Location	LOOCV Locative Accuracy (LA)			
		Virus-mPLoc [50]	iLoc-Virus [52]	mGOASVM [54]	AD-SVM
1	Viral capsid	8/8 = 100.0%	8/8 = 100.0%	8/8 = 1.000	8/8 = 1.000
2	Host cell membrane	19/33 = 57.6%	25/33 = 75.8%	32/33 = 0.970	32/33 = 0.970
3	Host ER	13/20 = 65.0%	15/20 = 75.0%	17/20 = 0.850	17/20 = 0.850
4	Host cytoplasm	52/87 = 59.8%	64/87 = 73.6%	85/87 = 0.977	83/87 = 0.954
5	Host nucleus	51/84 = 60.7%	70/84 = 83.3%	82/84 = 0.976	82/84 = 0.976
6	Secreted	9/20 = 45.0%	15/20 = 75.0%	20/20 = 1.000	20/20 = 1.000
Overall Actual Accuracy (*OAA*)		–	155/207 =74.8%	184/207 = 0.889	193/207 = **0.932**
Overall Locative Accuracy (*OLA*)		152/252 = 60.3%	197/252 = 78.2%	244/252 = **0.968**	242/252 = 0.960
	Accuracy	–	–	0.935	**0.953**
	Precision	–	–	0.939	**0.960**
	Recall	–	–	**0.973**	0.966
	F1	–	–	0.950	**0.960**
	HL	–	–	0.026	**0.019**

better than the state-of-the-art classifiers. The individual locative accuracies of AD-SVM are also comparable to mGOASVM.

Similar conclusions can be drawn from Table 3, where the superiority of AD-SVM over mGOASVM on the plant dataset seems to be not so obvious compared to that in Table 2.

7 Conclusions

This paper proposes an adaptive-decision based multi-label SVM classifier, namely AD-SVM, to predict subcellular localization of both single- and multi-location proteins. Given a query protein, by using the successive-search strategy, the GO information is extracted by using either its AC or its homologous AC as keys to search against GO annotation database, which is subsequently used to construct term-frequency based GO vectors. After scoring the GO vectors by the multi-label SVM classifier, the predicted results are determined by an adaptive-decision scheme, which can efficiently reduce the false positives while imposing little influence on the correctly predicted ones. Results on two benchmark datasets demonstrate that the adaptive threshold scheme can be readily integrated into multi-label SVM classifiers.

The advantages of AD-SVM over existing state-of-the-art predictors can be summarized as follows: (1) it incorporates an adaptive-decision based scheme to determine the number of predicted subcellular locations and thus it can improve the generalization capabilities of multi-label SVM classifiers; (2) it adopts a successive-search strategy to retrieve GO information from the GOA database to guarantee that AD-SVM is applicable to every query protein; and (3) it uses term-frequency based GO information to construct feature vectors which contains richer discriminative information than conventional 1-0 value methods.

Table 3 Comparing AD-SVM with state-of-the-art multi-label predictors based on leave-one-out cross validation (LOOCV) using the plant dataset.

Label	Subcellular Location	LOOCV Locative Accuracy (LA)			
		Plant-mPLoc [51]	iLoc-Plant [53]	mGOASVM [54]	AD-SVM
1	Cell membrane	24/56 = 42.9%	39/56 = 69.6%	53/56 = 0.946	52/56 = 0.929
2	Cell wall	8/32 = 25.0%	19/32 = 59.4%	27/32 = 0.844	27/32 = 0.844
3	Chloroplast	248/286 = 86.7%	252/286 = 88.1%	272/286 = 0.951	271/286 = 0.948
4	Cytoplasm	72/182 = 39.6%	114/182 = 62.6%	174/182 = 0.956	167/182 = 0.917
5	Endoplasmic reticulum	17/42 = 40.5%	21/42 = 50.0%	38/42 = 0.905	38/42 = 0.905
6	Extracellular	3/22 = 13.6%	2/22 = 9.1%	22/22 = 1.000	22/22 = 1.000
7	Golgi apparatus	6/21 = 28.6%	16/21 = 76.2%	19/21 = 0.905	19/21 = 0.905
8	Mitochondrion	114/150 = 76.0%	112/150 = 74.7%	150/150 = 1.000	149/150 = 0.993
9	Nucleus	136/152 = 89.5%	140/152 = 92.1%	151/152 = 0.993	148/152 = 0.974
10	Peroxisome	14/21 = 66.7%	6/21 = 28.6%	21/21 = 1.000	21/21 = 1.000
11	Plastid	4/39 = 10.3%	7/39 = 17.9%	39/39 = 1.000	36/39 = 0.923
12	Vacuole	26/52 = 50.0%	28/52 = 53.8%	49/52 = 0.942	48/52 = 0.923
Overall Actual Accuracy (*OAA*)		–	666/978 = 68.1%	855/978 = 0.874	867/978 = **0.887**
Overall Locative Accuracy (*OLA*)		672/1055 = 63.7%	756/1055 = 71.7%	1015/1055 =**0.962**	998/1055 = 0.946
Accuracy		–	–	0.926	**0.928**
Precision		–	–	0.933	**0.941**
Recall		–	–	**0.968**	0.956
F1		–	–	**0.942**	**0.942**
HL		–	–	**0.013**	**0.013**

References

1. H. Nakashima, K. Nishikawa, J. Mol. Biol. **238**, 54 (1994)
2. G.P. Zhou, K. Doctor, PROTEINS: Structure, Function, and Genetics **50**, 44 (2003)
3. K.C. Chou, Proteins: Structure, Function, and Genetics **43**, 246 (2001)
4. O. Emanuelsson, H. Nielsen, S. Brunak, G. von Heijne, J. Mol. Biol. **300**(4), 1005 (2000)
5. K. Nakai, M. Kanehisa, Proteins: Structure, Function, and Genetics **11**(2), 95 (1991)
6. H. Nielsen, J. Engelbrecht, S. Brunak, G. von Heijne, Int. J. Neural Sys. **8**, 581 (1997)
7. R. Mott, J. Schultz, P. Bork, C. Ponting, Genome research **12**(8), 1168 (2002)
8. M.W. Mak, J. Guo, S.Y. Kung, IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) **5**(3), 416 (2008)
9. S. Wan, M.W. Mak, *Machine Learning for Protein Subcellular Localization Prediction*, De Gruyter, (2015)
10. S. Wan, M.W. Mak, S.Y. Kung, in *2011 IEEE International Workshop on Machine Learning for Signal Processing (MLSP'11)* (2011), pp. 1–6
11. K.C. Chou, H.B. Shen, J. of Proteome Research **5**, 1888 (2006)
12. K.C. Chou, Y.D. Cai, Biochem. Biophys. Res. Commun. **320**, 1236 (2004)
13. S. Wan, M.W. Mak, S.Y. Kung, in *2012 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'12)* (2012), pp. 2229–2232
14. S. Brady, H. Shatkay, in *Pac. Symp. Biocomput.* (2008), pp. 604–615
15. A. Fyshe, Y. Liu, D. Szafron, R. Greiner, P. Lu, Bioinformatics **24**, 2512 (2008)
16. Z. Lu, D. Szafron, R. Greiner, P. Lu, D.S. Wishart, B. Poulin, J. Anvik, C. Macdonell, R. Eisner, Bioinformatics **20**(4), 547 (2004)
17. R. Nair, B. Rost, Protein Science **11**, 2836 (2002)
18. L.J. Foster, C.L.D. Hoog, Y. Zhang, Y. Zhang, X. Xie, V.K. Mootha, M. Mann, Cell **125**, 187 (2006)
19. S. Zhang, X.F. Xia, J.C. Shen, Y. Zhou, Z. Sun, BMC Bioinformatics **9**, 127 (2008)
20. A.H. Millar, C. Carrie, B. Pogson, J. Whelan, Plant Cell **21**(6), 1625 (2009)
21. R.F. Murphy, Cytometry **77**(7), 686 (2010)
22. J.C. Mueller, C. Andreoli, H. Prokisch, T. Meitinger, Mitochondrion **3**, 315 (2004)
23. R. Russell, R. Bergeron, G. Shulman, H. Young, American Journal of Physiology **277**, H643 (1997)
24. S. Rea, D. James, Diabetes **46**, 1667 (1997)
25. T. Li, M. Ogihara, IEEE Transactions on Multimedia **8**(3), 564 (2006)
26. K. Trohidis, G. Tsoumakas, G. Kalliris, I. Vlahavas, in *Proceedings of the 9th International Conference on Music Information Retrieval* (2006), pp. 325–330
27. C.G.M. Snoek, M. Worring, J.C. van Gemert, J.M. Geusebroek, A.W.M. Smeulders, in *Proceedings of the 14th annual ACM International Conference on Multimedia* (2006), pp. 421–430
28. C. Vens, J. Struyf, L. Schietgat, S. Dzeroski, H. Blockeel, Machine Learning **2**(73), 185 (2008)
29. Z. Barutcuoglu, R.E. Schapire, O.G. Troyanskaya, Bioinformatics **22**(7), 830 (2006)
30. M.L. Zhang, Z.H. Zhou, in *IEEE International Conference on Granular Computing* (2005), pp. 718–721
31. J. Rousu, C. Saunders, S. Szedmak, J. Shawe-Taylor, Journal of Machine Learning Research **7**, 1601 (2006)
32. I. Katakis, G. Tsoumakas, I. Vlahavas, in *Proceedings of the ECML/PKDD 2008 Discovery Challenge* (2008)
33. R.E. Schapire, Y. Singer, Machine Learning **39**(2/3), 135 (2000)
34. R. Moskovitch, S. Cohenkashi, U. Dror, I. Levy, A. Maimon, Y. Shahar, Artificial Intelligence in Medicine **37**, 177 (2006)
35. N. Ghamrawi, A. McCallum, in *Proceedings of the 2005 ACM Conference on Information and Knowledge Management (CIKM'05)* (2005), pp. 195–200
36. M. Boutell, J. Luo, X. Shen, C. Brown, Pattern Recognition **37**(9), 1757 (2004)

37. G. Tsoumakas, I. Katakis, International Journal of Data Warehousing and Mining **3**, 1 (2007)
38. J. Read, B. Pfahringer, G. Holmes, E. Frank, in *Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases* (2009), pp. 254–269
39. G. Tsoumakas, I. Katakis, I. Vlahavas, in *Data Mining and Knowledge Discovery Handbook, O. Maimon, l. Rokach (Ed.). Springer, 2nd edition* (2010), pp. 667–685
40. D. Hsu, S.M. Kakade, J. Langford, T. Zhang, in *Advances in Neural Information Processing Systems 22* (2009), pp. 772–780
41. A. Clare, R.D. King, in *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery* (2001), pp. 42–53
42. J.R. Quinlan, *C4.5: programs for machine learning*, vol. 1 (Morgan Kaufmann, 1993)
43. Y. Freund, R. Schapire, Journal of Japanese Society for Artificial Intelligence **14**(771-780), 1612 (1999)
44. V.N. Vapnik, in *John Wiley & Sons* (1998)
45. A. Elisseeff, J. Weston, in *In Advances in Neural Information Processing Systems 14* (MIT Press, 2001), pp. 681–687
46. S. Godbole, S. Sarawagi, in *Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining* (Springer, 2004), pp. 22–30
47. T.G. Dietterich, g. Bakari, Journal of Artificial Intelligence Research pp. 263–286 (1995)
48. U. Kressel, in *Advances in Kernel Methods: Support Vector Learning, Chap. 15. MIT Press* (1999)
49. B. Scholkopf, A.J. Smola, in *MIT Press* (2002)
50. H.B. Shen, K.C. Chou, J. Biomol. Struct. Dyn. **26**, 175 (2010)
51. K.C. Chou, H.B. Shen, PLoS ONE **5**, e11335 (2010)
52. X. Xiao, Z.C. Wu, K.C. Chou, Journal of Theoretical Biology **284**, 42 (2011)
53. Z.C. Wu, X. Xiao, K.C. Chou, Molecular BioSystems **7**, 3287 (2011)
54. S. Wan, M.W. Mak, S.Y. Kung, BMC Bioinformatics **13**, 290 (2012)
55. S. Wan, M.W. Mak, S.Y. Kung, PLoS ONE **9**(3), e89545 (2014)
56. S. Wan, M.W. Mak, S.Y. Kung, Journal of Theoretical Biology **360**, 34 (2014)
57. S. Wan, M.W. Mak, S.Y. Kung, Analytical Biochemistry **473**, 14 (2015)
58. J. He, H. Gu, W. Liu, PLoS ONE **7**(6), e37155 (2011)
59. L.Q. Li, Y. Zhang, L.Y. Zou, C.Q. Li, B. Yu, X.Q. Zheng, Y. Zhou, PLoS ONE **7**(1), e31057 (2012)
60. S. Wan, M.W. Mak, B. Zhang, Y. Wang, S.Y. Kung, in *2013 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* (2013), pp. 35–42. DOI 10.1109/BIBM.2013.6732715
61. S. Wan, M.W. Mak, S.Y. Kung, Engineering **5**, 68 (2013)
62. S. Wan, M.W. Mak, B. Zhang, Y. Wang, S.Y. Kung, in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'14),* (IEEE, 2014), pp. 5999–6003
63. S. Wan, M.W. Mak, S.Y. Kung, in *2013 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'13)* (2013), pp. 3547–3551
64. S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller, D.J. Lipman, Nucleic Acids Res. **25**, 3389 (1997)
65. D. Barrel, E. Dimmer, R.P. Huntley, D. Binns, C. O'Donovan, R. Apweiler, Nucl. Acids Res. **37**, D396 (2009)
66. S. Mei, PLoS ONE **7**(6), e37716 (2012)
67. S. Wan, M.W. Mak, S.Y. Kung, Journal of Theoretical Biology **323**, 40 (2013)
68. K. Dembczynski, W. Waegeman, W. Cheng, E. Hullermeier, Machine Learning **88**(1-2), 5 (2012)
69. W. Gao, Z.H. Zhou, in *Proceedings of the 24th Annual Conference on Learning Theory* (2011), pp. 341–358
70. K.C. Chou, Molecular BioSystems **9**, 1092 (2013)
71. T. Hastie, R. Tibshirani, J. Friedman, *The element of statistical learning* (Springer-Verlag, 2001)