Mathematical Biosciences and Engineering

MBE, 21(1): 214–236.
DOI: 10.3934/mbe.2024010
Received: 24 September 2023
Revised: 13 November 2023
Accepted: 22 November 2023
Published: 11 December 2023

http://www.aimspress.com/journal/MBE

Research article

Improved multi-label classifiers for predicting protein subcellular localization

Lei Chen*, Ruyun Qu and Xintong Liu

College of Information Engineering, Shanghai Maritime University, Shanghai 201306, China

* **Correspondence:** Email: chen_lei1@163.com; Tel: +86-21-3828-2825; Fax: +8621-3828-2800.

Abstract: Protein functions are closely related to their subcellular locations. At present, the prediction of protein subcellular locations is one of the most important problems in protein science. The evident defects of traditional methods make it urgent to design methods with high efficiency and low costs. To date, lots of computational methods have been proposed. However, this problem is far from being completely solved. Recently, some multi-label classifiers have been proposed to identify subcellular locations of human, animal, Gram-negative bacterial and eukaryotic proteins. These classifiers adopted the protein features derived from gene ontology information. Although they provided good performance, they can be further improved by adopting more powerful machine learning algorithms. In this study, four improved multi-label classifiers were set up for identification of subcellular locations of the above four protein types. The random k-labelsets (RAKEL) algorithm was used to tackle proteins with multiple locations, and random forest was used as the basic prediction engine. All classifiers were tested by jackknife test, indicating their high performance. Comparisons with previous classifiers further confirmed the superiority of the proposed classifiers.

Keywords: protein subcellular localization; multi-label classification; random k-labelsets algorithm; random forest; gene ontology; jackknife test

1. Introduction

Proteins, as the material basis of life, play an important role in participating in various forms of life activities. Generally, the locations where these protein molecules reside in compartments or organelles are called “subcellular locations” [1]. It has been found that there is a close relationship

between the functions of proteins and their subcellular locations in cells. Thus, correct prediction of protein subcellular locations is helpful to uncover their functions. Determination of protein subcellular locations is one of the most important and essential problems in protein science.

The traditional methods for identifying protein subcellular location include fluorescence microscopy, electron microscopy, cell fractionation, etc. [2]. Although these experimental approaches can output assured results, there still exists a great gap between the number of proteins with known subcellular locations and those without determined subcellular locations [3,4] because many new proteins have been generated or found. Evidently, the above traditional methods cannot timely determine the subcellular locations of new proteins. In recent years, with the rapid development of computer science, computational methods are an important alternative way to determine protein subcellular locations, especially the machine learning based methods.

During the last two decades, plenty of efforts have been made to design computational methods for determining protein subcellular locations. As mentioned above, most of them are machine learning based methods. As most machine learning algorithms need numeric values as input, the main step to design machine learning based methods is to encode proteins into several features. According to the current research progress, the machine learning based methods can be roughly divided into the following four groups. The first group consists of sequence-based methods, which extract protein features from their sequences, such as amino acid composition [5,6], pseudo amino acid composition [7–11], position specific score matrix (PSSM) [12–15], etc. As sequence is the widely used form to represent proteins, this method always had wide applications. However, some essential properties of proteins cannot be extracted from their sequences, which affected the efficiency of this method. The second group consists of annotation-based methods. This group of methods uses annotation information of proteins for extracting essential protein features. Gene ontology (GO) [16–22] information is widely used in this group. Other information includes functional domain [23–26], Swiss-Prot keywords, PubMed abstracts, etc. The efficiency of this method is quite high because the annotation information contains several essential properties of proteins. However, for a newly designed protein, this method cannot always output the result, as its annotation information is not available. The third group consists of network-based methods. Protein-protein interaction (PPI) networks have wide applications for tackling protein-related problems, including the prediction of protein subcellular locations. Such methods use the features derived from one or more PPI networks, which are fed into downstream classification algorithms [27–30]. Although such methods also provide high performance, their application to newly designed proteins is a problem, as their locations in PPI networks are unknown. The last group consists of the recently proposed image-based methods. These methods adopt the features derived from protein images [31–33] and always employ deep learning algorithms to make predictions. The performance of such methods is very high. However, their interpretability is quite poor. In this study, we focused on annotation-based methods. In previous studies [16–19], Cheng et al. proposed a novel scheme to extract features from GO information of proteins, on which efficient multi-label classifiers were built. These features had low dimensions and further contained the label information of training samples. However, they adopted a weak prediction engine, which did not contain a strict learning procedure, thereby leaving great spaces for improvement. This study continued their work by employing a set of powerful machine learning algorithms.

In this study, four multi-label classifiers were set up for predicting subcellular locations of human, animal, Gram-negative bacterial and eukaryotic proteins, respectively. These proteins were directly retrieved from four previous studies [16–19]. Each protein was represented by compact features

derived from its GO information. To fully learn these features, a powerful multi-label algorithm, random k-labelsets (RAKEL) [34], with random forest (RF) [35] as the basic prediction engine, was adopted to construct the classifiers. The jackknife test results of the classifiers on corresponding protein datasets indicated their quite high performance to predict protein subcellular locations. The full comparisons confirmed that the proposed classifiers were superior to several previous classifiers.

2. Materials and methods

2.1. Benchmark datasets

In this study, four protein types and their subcellular locations were investigated, including human, animal, Gram-negative bacterial and eukaryotic proteins. This information was directly collected from four previous studies [16–19]. Accordingly, four protein datasets were obtained, each of which consisted of proteins in one above-mentioned protein type. For easy description, the four protein datasets were called human, animal, Gram-negative bacterial and eukaryotic datasets, respectively. Some information of these four datasets is listed in Table 1. Their brief descriptions are as follows.

Table 1. Information of four protein datasets.

Dataset	Number of proteins	Number of subcellular locations
Human dataset	3106	14
Animal dataset	3919	20
Gram-negative bacterial dataset	1392	8
Eukaryotic dataset	7766	22

The human dataset consisted of 3106 human proteins reported in [16,36,37]. The sequence identity of any two proteins in this dataset was less than 25%. Fourteen subcellular locations were involved with these proteins, as shown in Figure 1(A). According to the subcellular locations of these proteins, all 3106 proteins were classified into fourteen classes. The number of proteins in each class is also shown in Figure 1(A). By simple calculation, the sum of protein numbers in all classes was 3681, which was larger than the number of different proteins (3016), indicating that some proteins belonged to more than one class. The average number of locations for proteins in this dataset was 1.185. It was a multi-label classification problem to assign the above fourteen subcellular locations to the given proteins.

In the animal dataset, 3919 proteins were contained, which were retrieved from [17,38]. Any protein had a sequence identity of less than 40% with any other protein in this dataset. The above proteins had 20 subcellular locations, which are listed in Figure 1(B). Accordingly, these proteins were divided into 20 classes. The number of proteins in each class is listed in Figure 1(B). Likewise, the sum of protein numbers in all classes was 6519, also larger than the number of different animal proteins. Each animal protein, on average, 1.669 subcellular locations. Thus, it was also a multi-label classification problem.

As for the third dataset, the Gram-negative bacterial dataset, 1392 proteins were included. These proteins were collected from [18,39,40]. The sequence identity of any two proteins in this dataset was less than 25%. Eight subcellular locations were assigned to these proteins, which are shown in Figure 1(C). Accordingly, 1392 proteins were classified into eight classes. The size of each class is also listed in Figure 1(C). Similar to the above two datasets, the sum of protein numbers in all classes (1456)

was larger than the number of different Gram-negative bacterial proteins (1392), and the average number of locations that a protein was assigned in this dataset was 1.046. Assigning eight subcellular locations to the given proteins was also a multi-label classification problem.

The last dataset, the eukaryotic dataset, was sourced from [19,24]. Overall, 7766 proteins were contained in this dataset, which were classified into 22 subcellular locations, as shown in Figure 1(D). The number of proteins in each class is also listed in Figure 1(D). The sequence identity of any two proteins in this dataset was less than 25%. Similarly, some proteins can belong to more than one class, as the sum of protein numbers in all classes (8897) was larger than the number of different proteins (7766). The average number of locations that a protein was assigned was 1.146. A multi-label classifier should be built to assign these subcellular locations to any eukaryotic protein.

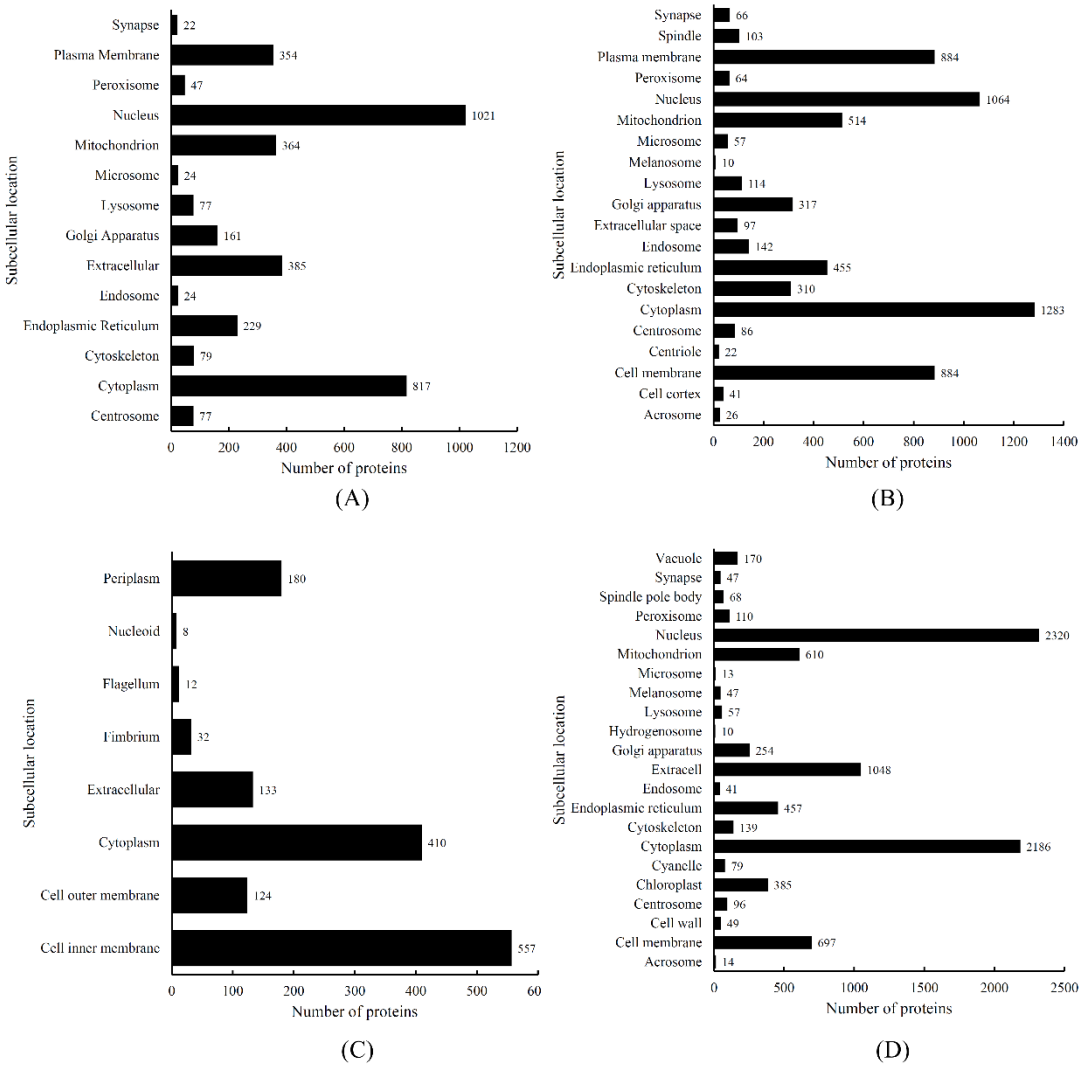

Figure 1. Bar charts to show the number of proteins in each subcellular location class for four datasets. (A) Human dataset; (B) Animal dataset; (C) Gram-negative bacterial dataset; (D) Eukaryotic dataset.

2.2. Protein representation

It is an essential step to encode each sample into a series of numeric values, which can be processed by most computer algorithms. These numeric values should contain essential properties of samples as much as possible. For proteins, GO is an important type of annotation information, which labels the essential functions of proteins. At first, the one-hot scheme was adopted for the GO annotation information of proteins, generating a high-dimensional binary feature vector for each protein. The classification models built on such representation were always of low efficiency. It is necessary to refine such information to tackle this problem. Here, we directly adopted the scheme reported in [16–19] to encode proteins for easy comparisons. This scheme not only reduced the dimensions of the final representation but also included the label information of training samples, which was deemed to be helpful for improving the classification performance. A brief description of such scheme is as follows.

Given a training dataset S with N proteins, denoted by $p_1, p_2, \cdots, p_N$, and L labels, denoted by $l_1, l_2, \cdots, l_L$, let C_i be a subset of S consisting of proteins with label l_i. In this case, $S = C_1 \cup C_2 \cup \cdots \cup C_L$. In this study, as some proteins may have more than one label, common proteins may exist for two subsets. Furthermore, the GO terms of protein p_a in S constitute the set $GO(p_a)$. The feature vector of each training and test protein p is finally formulated by

$$v(p) = [\delta_1, \delta_2, \cdots, \delta_L]^T, \tag{1}$$

where L is the number of labels. Each component δ_i in this vector indicates the relationship between p and proteins in C_i, which can be determined in the following manner:

Step 1: Determine the GO information of p. If such information is not available, find the protein in the Swiss-Prot database with the highest pairwise sequence identity to p whose GO information is available. Use this GO information to represent p. For formulation, the GO information of p thus obtained is denoted by $GO(p) = \{GO_p^1, GO_p^2, \cdots, GO_p^k\}$, where k denotes the number of GO terms annotated to p or its most similar protein.

Step 2: For each GO GO_p^j $(1 \le j \le k)$, count the number of proteins in the training dataset that are annotated by GO_p^j, denoted by $N(GO_p^j)$, i.e., $N(GO_p^j) = \left|\{p_a | p_a \in S \; and \; GO_p^j \in GO(p_a)\}\right|$. In addition, count the number of proteins in C_i which are also annotated by GO_p^j, indicated by $N(GO_p^j, i)$, i.e., $N(GO_p^j, i) = \left|\{p_a | p_a \in C_i \; and \; GO_p^j \in GO(p_a)\}\right|$. Then, calculate $R(GO_p^j, i) = \frac{N(GO_p^j, i)}{N(GO_p^j)}$.

Step 3: Take the maximum value among $R(GO_p^1, i), R(GO_p^2, i), \cdots, R(GO_p^k, i)$ and set it as δ_i, i.e., $\delta_i = \max\{R(GO_p^1, i), R(GO_p^2, i), \cdots, R(GO_p^k, i)\}$.

To understand the above procedures, an example is given. Consider a protein p annotated by three GO terms, say, GO1, GO2 and GO3. The numbers of training proteins annotated by these three GO terms are first counted (e.g., 120 for GO1, 150 for GO2 and 200 for GO3). Then, the numbers of training proteins in the first category (for label l_1) that are also annotated by GO1, GO2 and GO3 are counted (e.g., 60 for GO1, 10 for GO2 and 20 for GO3). The maximum is picked from $\{\frac{60}{120}, \frac{10}{150}, \frac{20}{200}\}$ as δ_1, that is, $\delta_1 = \frac{60}{120} = 0.5$. $\delta_2, \delta_3, \cdots, \delta_L$ can be obtained in a similar way.

With the above operation, L features were generated to represent a protein. These features contained not only the GO information of the encoded protein but also the statistical information of GO terms of proteins in the training dataset. The number of features was greatly reduced compared to the one-hot scheme.

Four datasets were investigated in this study. With the above scheme, proteins in the human, animal, Gram-negative bacterial and eukaryotic datasets were represented by 14, 20, 8 and 22 features. Based on these features, efficient multi-label classifiers can be built.

2.3. Random k-labelsets algorithm

As mentioned in the section "Benchmark datasets," some proteins in each dataset had more than one subcellular location, suggesting we should construct a multi-label classifier for each dataset for assigning subcellular locations to proteins. Generally, there are two ways to construct multi-label classifiers [41]. The first one is problem transformation, which creates some single-label problems from the original multi-label problem, builds a single-label classifier for each single-label problem and integrates the results yielded by the single-label classifiers as the output. The second way is algorithm adaption, which generalizes the single-label classifiers such that they can process samples with multiple labels. To our knowledge, problem transformation methods are more popular than algorithm adaption methods when constructing multi-label classifiers. In fact, a multi-label classifier built by a certain problem transformation method is a combination of single-label classifiers. At present, lots of single-label classification algorithms have been proposed, providing abundant sources for building such multi-label classifiers. The pure multi-label classification algorithms are much fewer than single-label classification algorithms, resulting in much fewer choices to build multi-label classifiers through algorithm adaption methods. In the previous studies [16–19], a multi-label Gaussian kernel regression (ML-GKR) classifier was built on each dataset, and it was an algorithm adaption method. Here, we employed a more powerful problem transformation method, RAKEL [34], to build the multi-label classifiers.

RAKEL can be deemed as an improved version of the label powerset (LP) [42] method. Given a training dataset containing L labels, LP uses members in the power set of the label set as new labels and assigns them to all samples. For example, if a sample has two labels, say l_1 and l_3, $\{l_1, l_3\}$ will be treated as a new label and be assigned to this sample. After this operation, all samples have exactly one label. A single-label classifier can be built on the dataset with new assigned labels based on a certain single-label classification algorithm. The LP method has an evident defect. A large number of labels results in a label disaster. So many new labels are involved, and the number of samples for some labels may small, decreasing the learning results. In view of this, RAKEL was proposed. It randomly constructs m label subsets with size k ($1 \leq k \leq L$). For each label subset, a single-label classifier is built using the LP method, called the LP classifier. RAKEL integrates m LP classifiers as the final multi-label classifier. For a test sample, each LP classifier gives its predictions (binary predictions for labels involved in this LP classifier). Then, among the LP classifiers related to one label, count the proportion of classifiers that give positive prediction on this label. If such proportion is larger than the predefined threshold, which is generally set to 0.5, RAKEL assigns this label to the test sample. In recent years, RAKEL has been used to construct lots of multi-label classifiers in bioinformatics [43–50].

The present study directly adopted the tool "RAKEL" in Meka (http://waikato.github.io/meka/) [51] to implement the above RAKEL method. Various values of k were tried to build the optimal classifier.

2.4. Random forest

When using RAKEL to construct multi-label classifiers, we need to select a base single-label classification algorithm. A proper selection may greatly improve the performance of the final classifier.

In this study, the classic classification algorithm, RF [35], was used.

RF is an ensemble learning algorithm containing several decision trees [35]. When constructing a decision tree, the same number of samples in the training dataset was randomly selected with replacement (i.e., some samples may be selected more than once), and a small proportion of features are also randomly selected. Given a test sample, each decision tree generates its prediction. RF integrates these predictions with majority voting. Although decision tree is the basis of RF, and it is deemed to be a weak algorithm, RF is much more powerful. Accordingly, it has been widely used in the field of bioinformatics [52–59].

The above RF algorithm was also implemented in Meka (http://waikato.github.io/meka/) [51] by a tool "RandomForest," which was directly used in this study. Default parameters were used, where the number of decision trees was 100.

2.5. Jackknife test

Cross-validation methods are widely used and accepted to evaluate the performance of classifiers. The jackknife test [60] is one of the commonly used cross-validation methods and has been used to evaluate many previous classifiers, including classifiers in pLoc and iLoc series [16–19,36,38,40,61]. To give a fair comparison with previous classifiers, we also adopted the jackknife test to examine all constructed multi-label classifiers. In such test, each sample is singled out as a test sample one by one, while the rest of the samples constitute a dataset to train the classifier. Evidently, such procedures are completely open, that is, the test and training samples are assured in each round. This makes the evaluation results more reliable and objective. Furthermore, such a test can always output a unique result in most cases. Thus, it has wide applications in testing the qualities of various classifiers [43,44,62–65].

2.6. Measurements for multi-label classifiers

To evaluate the performance of multi-label classifiers, several measurements have been proposed. In this study, we adopted the same measurements used in [16–19] for easy comparisons. These measurements included absolute true, absolute false, accuracy, aiming and coverage. Given a dataset with N samples and L labels, let D_i denote the set of true labels for the ith sample and D_i^* denote the set of predicted labels for the ith sample. The above five measurements can be computed as

$$\begin{cases} \text{absolute true} = \frac{1}{N}\sum_{i=1}^{N}\Delta(D_i, D_i^*) \\ \text{absolute false} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{|D_i \cup D_i^*| - |D_i \cap D_i^*|}{L}\right) \\ \text{accuracy} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{|D_i \cap D_i^*|}{|D_i \cup D_i^*|}\right) \\ \text{aiming} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{|D_i \cap D_i^*|}{|D_i^*|}\right) \\ \text{coverage} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{|D_i \cap D_i^*|}{|D_i|}\right) \end{cases} \tag{2}$$

where Δ is defined as below:

$$\Delta(D_i, D_i^*) = \begin{cases} 1, & if\ D_i = D_i^* \\ 0, & otherwise \end{cases} \tag{3}$$

According to these definitions, the higher the values of absolute true, accuracy and aiming, coverage are, the higher the performance of the classifier. As for absolute false, the relationship is the opposite. A low absolute false value indicates high performance. Among these measurements, we selected absolute true as the key measurement as used in [16–19] for optimizing parameters.

Also, some local measurements on each class were also employed, including sensitivity (SN), specificity (SP), overall accuracy (ACC) and Matthews correlation coefficient (MCC). For the j-th class, these measurements can be computed by

$$\begin{cases} \mathrm{SN}(j) = \dfrac{TP}{TP+FN} \\ \mathrm{SP}(j) = \dfrac{TN}{TN+FP} \\ \mathrm{ACC}(j) = \dfrac{TP+TN}{TP+TN+FP+FN} \\ \mathrm{MCC}(j) = \dfrac{TP\times TN-FP\times FN}{\sqrt{(TN+FN)\times(TN+FP)\times(TP+FN)\times(TP+FP)}} \end{cases} \tag{4}$$

where TP denotes the total number of correctly classified samples that belong to the j-th class; TN denotes the number of correctly classified samples that do not belong to the j-th class; FP denotes the number of samples that do not belong to the j-th class but are misclassified to the j-th class; FN denotes the number of samples that belong to the j-th label and are misclassified to other classes. The higher the above measurements are, the higher the performance of the classifiers.

3. Results and discussion

In this study, a set of multi-label classifiers was proposed to identify subcellular locations of four protein types. The entire procedures are illustrated in Figure 2. This section gives the detailed evaluation results for these classifiers and compares their performance with previous classifiers. For convenience, we named the four multi-label classifiers as PMPSL-GRAKEL-Hum (on the human dataset), PMPSL-GRAKEL-Anim (on the animal dataset), PMPSL-GRAKEL-Geng (on Gram-negative bacterial dataset) and PMPSL-GRAKEL-Euk (on the eukaryotic dataset), respectively.

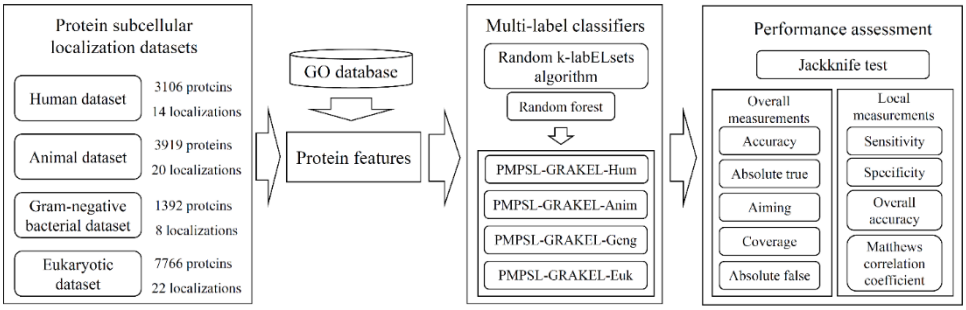

Figure 2. Entire procedures for constructing and testing multi-label classifiers on four datasets. Four protein subcellular localization datasets are retrieved from previous studies. Each protein is represented by features extracted from its gene ontology (GO) information. The Random k-labelsets (RAKEL) algorithm with random forest is adopted to build the classifier on each dataset, which is finally tested by jackknife test. The test results are as assessed with overall and local measurements.

3.1. Parameter optimization

RAKEL was adopted to build classifiers in this study. There are two main parameters, *m* and *k*, which can be tuned to optimize the classifier. The parameter *m* determines the number of LP classifiers. We used its default value (10) in Meka. As for the parameter *k*, it determines the number of labels for building each LP classifier. Here, we tried all possible values between two and the number of labels in each dataset. All classifiers under different *k* values were set up and tested by the jackknife test. In multi-label classification, accuracy and absolute true are main measurements to evaluate the quality of prediction results. Relatively, absolute true is stricter than accuracy because the sample is deemed to be correctly predicted if and only if the predicted labels are identical to true labels. If the predicted labels are partly correct, they cannot give positive contributions to absolute true, whereas they can provide this contribution to accuracy. Meanwhile, the previous studies [16–19] also selected absolute true to optimize parameters. Thus, absolute true was selected as the key measurement to optimize parameter *k*.

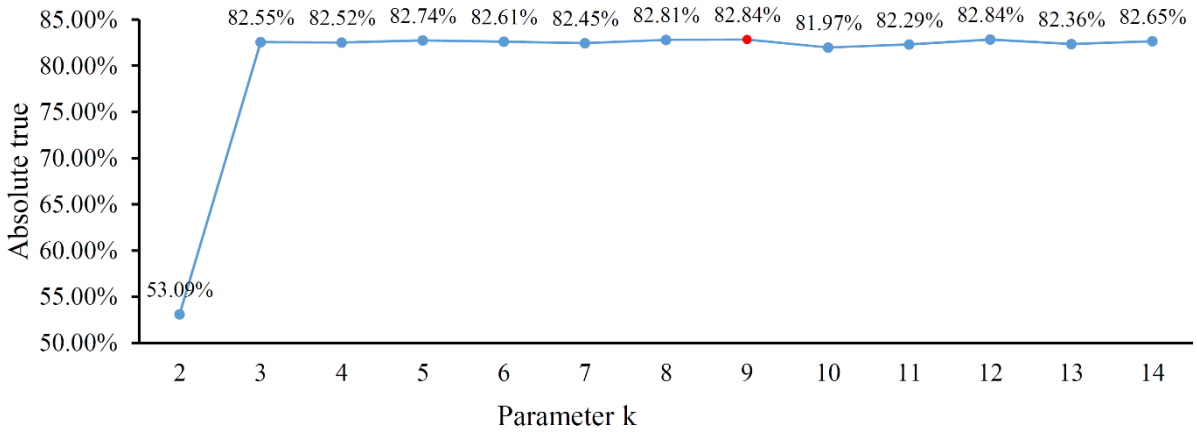

Figure 3. A curve to show the performance (absolute true) of PMPSL-GRAKEL-Hum under different values of parameter k. When k is set to nine, the classifier provides the best performance. The red dot in the curve indicates the highest absolute true.

For the classifier PMPSL-GRAKEL-Hum on the human dataset, its performance on different *k* values is shown in Figure 3. It can be observed that when $k = 2$, the classifier provided quite low performance. As *k* determines the number of selected features for constructing LP classifiers, they may not cover all labels when *k* is small, inducing low performance on some labels. This fact further decreases the overall performance of classifiers. When $k > 2$, the performances of the classifiers were almost at the same level. Relatively speaking, when $k = 9$ and 12, the classifier provided the highest performance, with absolute true of 82.84%. Considering that the complexity of the classifier follows an increasing trend with the increasing of *k*, we finally determine *k* as nine for PMPSL-GRAKEL-Hum.

For the classifier PMPSL-GRAKEL-Anim on the animal dataset, a curve was also plotted, as shown in Figure 4, to indicate its performance under different *k* values. When *k* was small (≤ 3), the performance was also low, which was same as that illustrated in Figure 3. It was caused by the same reason. When $k > 3$, the performance was quite stable. The highest absolute true was 81.86%, when *k* was set to eight. Thus, PMPSL-GRAKEL-Anim was built with $k = 8$.

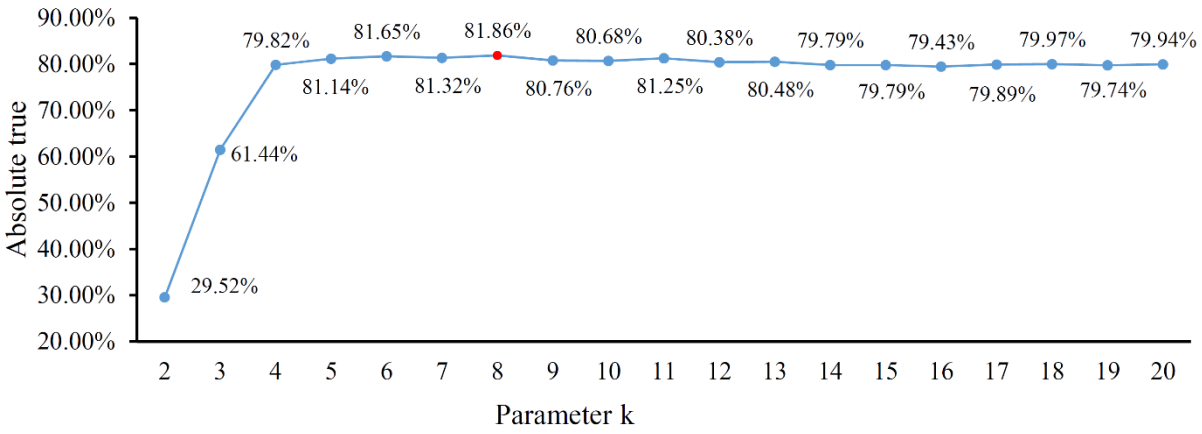

Figure 4. A curve to show the performance (absolute true) of PMPSL-GRAKEL-Anim under different values of parameter k. When k is set to eight, the classifier provides the best performance. The red dot in the curve indicates the highest absolute true.

For the classifier PMPSL-GRAKEL-Geng on the Gram-negative bacterial dataset, we also plotted a curve to illustrate its performance with different k values, as shown in Figure 5. It can be observed that all classifiers with different k values provided high performance, with absolute true higher than 94%. When $k = 6$, the highest absolute true of 95.40% was achieved. Accordingly, the PMPSL-GRAKEL-Geng using $k = 6$ was set up.

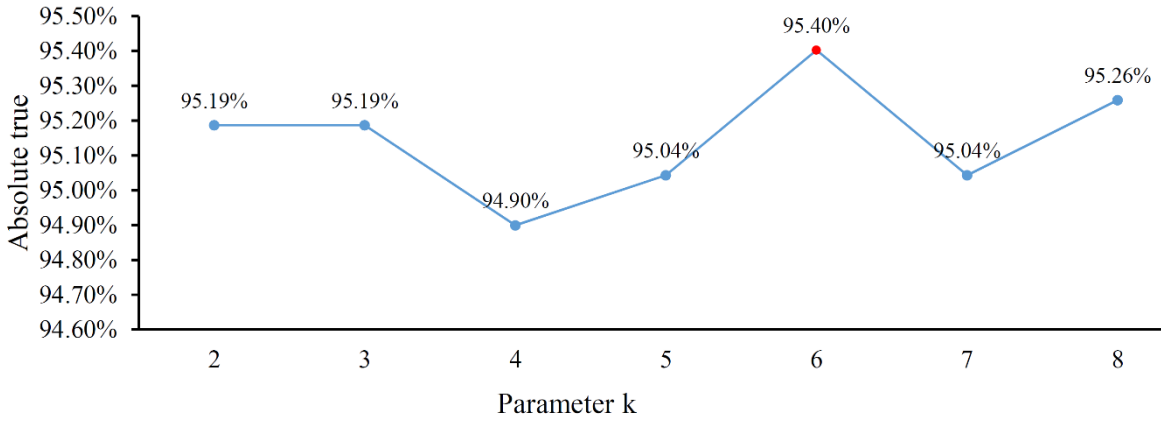

Figure 5. A curve to show the performance (absolute true) of PMPSL-GRAKEL-Geng under different values of parameter k. When k is set to six, the classifier provides the best performance. The red dot in the curve indicates the highest absolute true.

As for the last classifier, PMPSL-GRAKEL-Euk on the eukaryotic dataset, its performance under different k values is shown in Figure 6. Similar to Figures 3 and 4, the classifiers with small values still provided low performance, which was caused by the same reason elaborated above. Likewise, classifiers with large k values provided almost equal performance. When k was set to eight, the classifier yielded the highest absolute true of 82.22%. Accordingly, the PMPSL-GRAKEL-Euk with $k = 8$ was built.

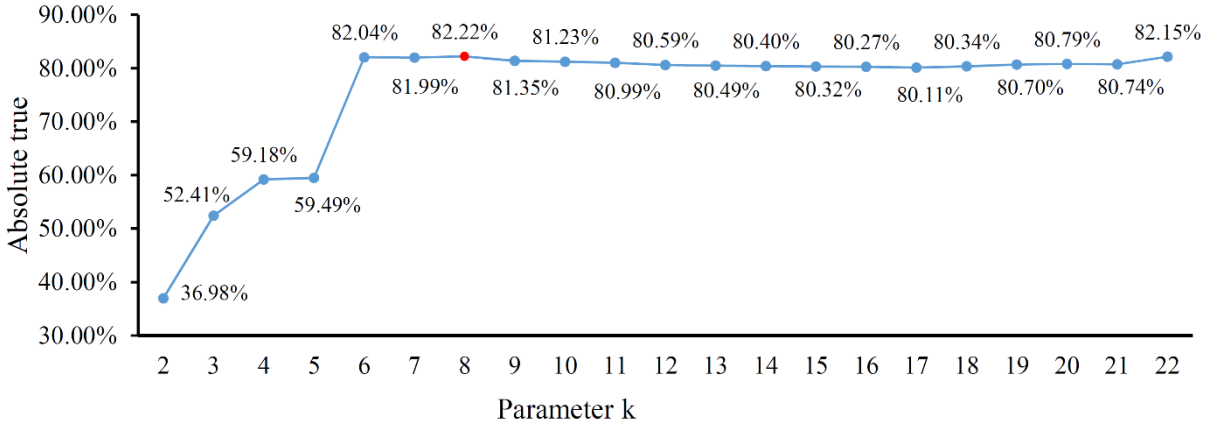

Figure 6. A curve to show the performance (absolute true) of PMPSL-GRAKEL-Euk under different values of parameter k. When k is set to eight, the classifier provides the best performance. The red dot in the curve indicates the highest absolute true.

3.2. Performance of the multi-label classifiers

In the section "Parameter optimization," the optimal *k* value was determined for each multi-label classifier. The detailed performances of these optimal classifiers are given in this section. Their overall performances, including aiming, coverage, accuracy, absolute true and absolute false, are listed in Table 2.

Table 2. Overall performances of the multi-label classifiers on four different datasets.

Classifier	Dataset	Aiming	Coverage	Accuracy	Absolute true	Absolute false
PMPSL-GRAKEL-Hum	Human dataset	90.73%	92.36%	88.60%	82.84%	1.78%
PMPSL-GRAKEL-Anim	Animal dataset	93.00%	95.21%	90.42%	81.86%	1.31%
PMPSL-GRAKEL-Geng	Gram-negative bacterial dataset	97.11%	97.99%	96.83%	95.40%	0.82%
PMPSL-GRAKEL-Euk	Eukaryotic dataset	89.92%	92.36%	88.15%	82.22%	1.06%

For PMPSL-GRAKEL-Hum, the jackknife test results indicated aiming of 90.73%, coverage of 92.36%, accuracy of 88.60%, absolute true of 82.84% and absolute false of 1.78% (Table 2). The aiming and coverage exceeded 90%, whereas accuracy and absolute true exceeded 80%, indicating the high performance of PMPSL-GRAKEL-Hum.

For PMPSL-GRAKEL-Anim on the animal dataset, it yielded the aiming of 93.00%, coverage of 95.21%, accuracy of 90.42%, absolute true of 81.86% and absolute false of 1.31% (Table 2). This classifier provided a little higher performance than PMPSL-GRAKEL-Hum, also suggesting the excellent performance of PMPSL-GRAKEL-Anim.

For PMPSL-GRAKEL-Geng on the Gram-negative bacterial dataset, its aiming, coverage, accuracy, absolute true and absolute false were 97.11%, 97.99%, 96.83%, 95.40% and 0.82% (Table 2), respectively. The first four measurements all exceeded 95%, and absolute false reduced to lower than 1%. If only considering such performance, it outperformed the above two classifiers.

As for PMPSL-GRAKEL-Euk on the eukaryotic dataset, the five measurements calculated by Eq (2) were 89.92%, 92.36%, 88.15%, 82.22% and 1.06% (Table 2). Such performance was almost equal to PMPSL-GRAKEL-Hum and PMPSL-GRAKEL-Anim and slightly lower than PMPSL-GRAKEL-Geng.

In addition to the above measurements to assess the overall performance of the four multi-label classifiers, four measurements (Eq (4)) for each subcellular location were also computed. The performances of the four classifiers for each subcellular location are listed in Tables S1–S4. It can be observed that most measurements were quite high, consistent with their overall performances mentioned above. In detail, the SP and ACC values were all higher than 90.00%. As for SN and MCC, those yielded by PMPSL-GRAKEL-Geng were also very high (Table S3), and most of them generated by the other three classifiers were also high. However, some SN and MCC values for a few locations generated by PMPSL-GRAKEL-Hum, PMPSL-GRAKEL-Anim and PMPSL-GRAKEL-Euk were quite low. For example, these include those for "Endosome" in the human dataset (12.50% for SN and 27.16% for MCC) (Table S1), "Microsome" in the animal dataset (20.00% for SN and 44.68% for MCC) (Table S2), "Endosome," "Microsome" and "Acrosome" in the eukaryotic dataset (7.32%, 7.69%, 21.43% for SN and 20.83%, 27.71%, 46.26% for MCC) (Table S4), etc. By observing Figure 1, we can find that the sizes of these subcellular locations were quite small, inducing the low performance with them. Their performances can be improved if some techniques can be applied to the datasets for solving the imbalanced problem.

3.3. Comparison with classifiers using other base prediction engines

Table 3. Performances of classifiers using different base classification algorithms.

Dataset	Base classification algorithm	Aiming	Coverage	Accuracy	Absolute true	Absolute false
Human dataset	Decision tree	63.80%	61.89%	60.99%	57.28%	3.58%
	Support vector machine	90.90%	88.82%	86.94%	81.23%	2.04%
	Random forest	**90.73%**	**92.36%**	**88.60%**	**82.84%**	**1.78%**
Animal dataset	Decision tree	72.91%	76.25%	66.91%	50.34%	4.15%
	Support vector machine	90.87%	94.8%	88.01%	76.98%	1.64%
	Random forest	**93.00%**	**95.21%**	**90.42%**	**81.86%**	**1.31%**
Gram-negative bacterial dataset	Decision tree	80.75%	80.89%	79.96%	78.23%	3.20%
	Support vector machine	97.09%	95.51%	95.47%	93.82%	1.13%
	Random forest	**97.11%**	**97.99%**	**96.83%**	**95.40%**	**0.82%**
Eukaryotic dataset	Decision tree	62.79%	74.07%	61.52%	48.65%	3.34%
	Support vector machine	86.83%	90.71%	84.88%	77.08%	1.35%
	Random forest	**89.92%**	**92.36%**	**88.15%**	**82.22%**	**1.06%**

This study adopted the problem transformation method RAKEL to build the multi-label classifiers. A certain base single-label classification algorithm is necessary for RAKEL. Here, we selected RF. To indicate this selection was reasonable, two other classic classification algorithms, decision tree (DT) [66] and support vector machine (SVM) [67], were attempted. Likewise, the parameter *k* in RAKEL was tuned for DT and SVM. The best performances, measured by absolute true on the four datasets, are listed in Table 3. For easy comparisons, the performances of our classifiers (using RF as base prediction engine) are also listed in this table. It was found that the classifiers with RF were evidently superior to those with DT and also slightly better than those with SVM. These results suggested that the selection of RF was proper.

3.4. Comparison with previous classifiers

In this study, a multi-label classifier with high performance was constructed on each of four datasets. Some previous classifiers have also been proposed for the same datasets. Here, comparisons are conducted to show the superiority of the proposed four classifiers.

On the human dataset, pLoc-mHum [16] and iLoc-Hum [36] have been proposed to identify protein subcellular locations. Table 4 lists their overall performances, including aiming, coverage, accuracy, absolute true and absolute false. The performances of these two classifiers were also evaluated by jackknife test on the same human dataset. For pLoc-mHum, its measurements were directly retrieved from [16]. As for iLoc-Hum, the coverage and absolute true were reported in [36], which were also collected in [16]. The other three measurements (aiming, accuracy and absolute false) were not provided in [36]. For easy comparison, the overall performance of PMPSL-GRAKEL-Hum is also provided in Table 4. It can be observed that PMPSL-GRAKEL-Hum provided the best performance, followed by pLoc-mHum and iLoc-Hum. In detail, PMPSL-GRAKEL-Hum yielded the highest aiming, coverage, accuracy and absolute true, whereas the absolute false was inferior to that of pLoc-mHum. For the two most important measurements, accuracy and absolute true, PMPSL-GRAKEL-Hum improved about 4.2% and 3.7% compared with those of pLoc-mHum. As for the performance of PMPSL-GRAKEL-Hum and pLoc-mHum on 14 subcellular locations, a box plot was drawn for each of SN, SP, ACC and MCC, as shown in Figure 7, from which we can see that the main parts of the boxes for PMPSL-GRAKEL-Hum were always higher than those of pLoc-mHum on the basis of each measurement. This implied that for most subcellular locations, PMPSL-GRAKEL-Hum provided better performance than pLoc-mHum.

On the animal dataset, pLoc-mAnimal [17] and iLoc-Animal [38] were two previous classifiers that were also constructed on such dataset and evaluated by jackknife test. Their performances are listed in Table 4, in which the overall performance of PMPSL-GRAKEL-Anim is also provided. In this table, the measurements for pLoc-mAnimal were directly sourced from [17]. As for iLoc-Animal, its original performance was assessed on a similar dataset, which contained proteins with sequence identity no less than 40% in some locations. Its measurements listed in Table 4 are also obtained from [17], and they were produced by re-executing iLoc-Animal on the same animal dataset. It can be observed from Table 4 that PMPSL-GRAKEL-Anim provided the highest performance on the basis of all measurements, whereas pLoc-mAnimal yielded the second highest performance, and iLoc-Animal generated the lowest performance. For accuracy and absolute true, PMPSL-GRAKEL-Anim increased about 5.7% and 8.7%, which were great improvements. Likewise, a box plot was drawn for each measurement on 22 subcellular locations yielded by PMPSL-GRAKEL-Anim and pLoc-

mAnimal, as shown in Figure 8. It can be concluded that PMPSL-GRAKEL-Anim outperformed pLoc-mAnimal on most locations no matter which measurements were adopted.

Table 4. Overall performances of the proposed and previous classifiers.

Dataset	Classifier	Aiming (rank)	Coverage (rank)	Accuracy (rank)	Absolute true (rank)	Absolute false (rank)
Human dataset	PMPSL-GRAKEL-Hum	**90.73%** (1)	**92.36%** (1)	**88.60%** (1)	**82.84%** (1)	1.78% (2)
	pLoc-mHum [16]	90.57% (2)	82.75% (2)	84.39% (2)	79.14% (2)	**1.20%** (1)
	iLoc-Hum [36]	N/A (/)	76.31% (3)	N/A (/)	68.19% (3)	N/A (/)
Animal dataset	PMPSL-GRAKEL-Anim	**93.00%** (1)	**95.21%** (1)	**90.42%** (1)	**81.86%** (1)	**1.31%** (1)
	pLoc-mAnimal [17]	87.96% (2)	85.33% (2)	84.64% (2)	73.11% (2)	1.65% (2)
	iLoc-Animal [38]	72.45% (3)	34.18% (3)	42.76% (3)	35.93% (3)	6.33% (3)
Gram-negative bacterial dataset	PMPSL-GRAKEL-Geng	**97.11%** (1)	**97.99%** (1)	**96.83%** (1)	**95.40%** (1)	0.82% (2)
	pLoc-mGeng [18]	96.61% (2)	95.81% (2)	96.05% (2)	94.68% (2)	**0.36%** (1)
	iLoc-Gneg [40]	N/A (/)	91.40% (3)	N/A (/)	89.90% (3)	N/A (/)
	Gneg-mPLoc [39]	N/A (/)	85.70% (4)	N/A (/)	N/A (/)	N/A (/)
Eukaryotic dataset	PMPSL-GRAKEL-Euk	**89.92%** (1)	**92.36%** (1)	**88.15%** (1)	**82.22%** (1)	1.06% (2)
	pLoc-mEuk [19]	88.31% (2)	85.06% (2)	84.34% (2)	78.78% (2)	**0.07%** (1)
	iLoc-Euk [61]	N/A (/)	79.06% (3)	N/A (/)	71.27% (3)	N/A (/)

On the Gram-negative bacterial dataset, three previous classifiers have been proposed, including pLoc-mGeng [18], iLoc-Gneg [40] and Gneg-mPLoc [39]. Their overall performances are listed in Table 4. Such performances of three classifiers were obtained by jackknife test on the same Gram-negative bacterial dataset. The measurements for pLoc-mGeng were directly picked up from [18]. As for iLoc-Gneg and Gneg-mPLoc, only some measurements (coverage and absolute true for iLoc-Gneg, coverage for Gneg-mPLoc) were reported in their original studies, which were also collected in [18]. The overall performance of PMPSL-GRAKEL-Geng is also listed in Table 4 for easy comparisons. Again, PMPSL-GRAKEL-Geng provided highest performance on four measurements, indicating this

classifier was superior to previous classifiers. The pLoc-mGeng gave the performance only inferior to PMPSL-GRAKEL-Geng. However, the superiority of PMPSL-GRAKEL-Geng was not very evident. The accuracy and absolute true were only 0.8% and 0.7% higher. According to their performances (SN, SP, ACC and MCC) on 8 subcellular locations (Figure 9), we can also conclude that PMPSL-GRAKEL-Geng was slightly better than pLoc-mGeng.

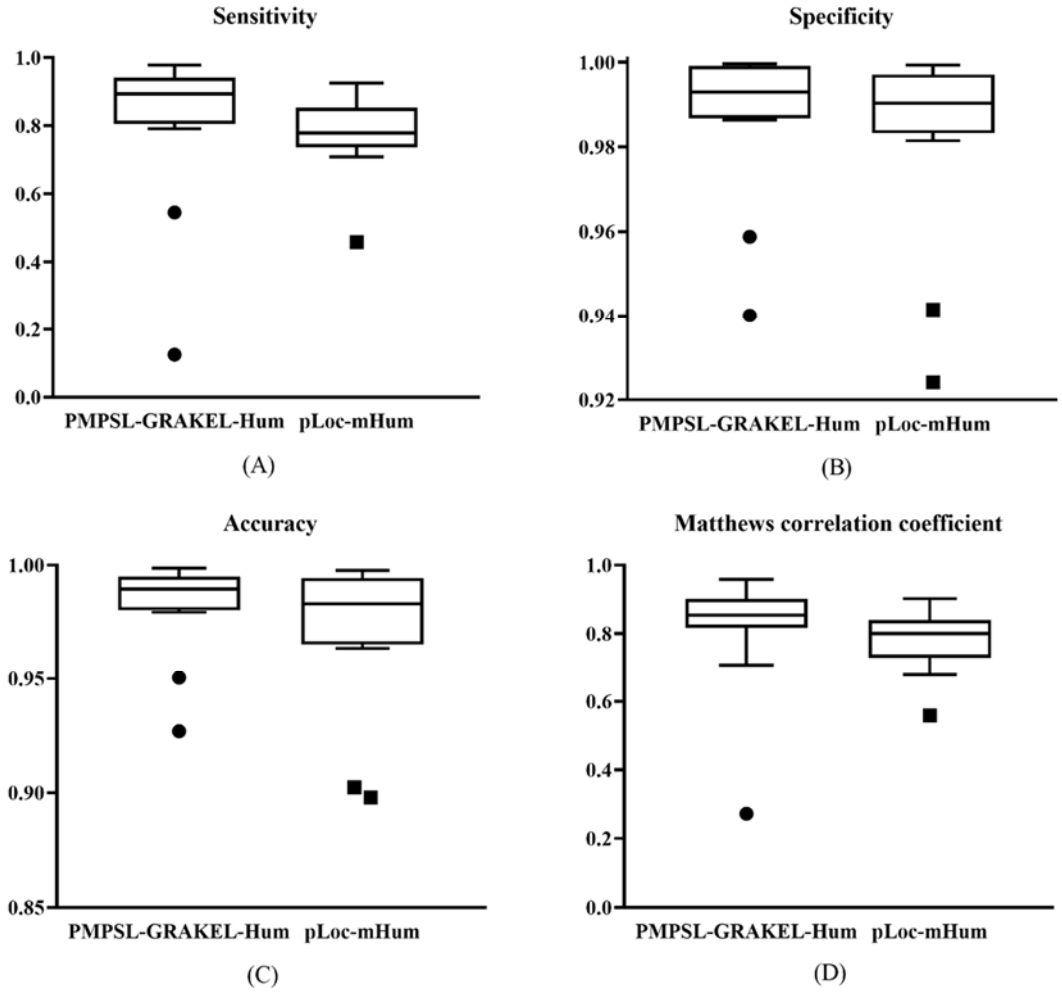

Figure 7. Box plot to show the performance of PMPSL-GRAKEL-Hum and pLoc-mHum on 14 subcellular locations. (A) Sensitivity; (B) Specificity; (C) Accuracy; (D) Matthews Correlation Coefficient. The performance of PMPSL-GRAKEL-Hum on 14 subcellular locations is generally better than that of pLoc-mHum.

On the eukaryotic dataset, two classifiers (pLoc-mEuk [19] and iLoc-Euk [61]) have been previously reported. Their overall performances are listed in Table 4. This performance was also obtained by jackknife test on the same eukaryotic dataset. The measurements for pLoc-mEuk were directly collected from [19]. For iLoc-Euk, its original study only reported its coverage and absolute true [61], which were also used in [19]. By comparing the performance of PMPSL-GRAKEL-Euk, also listed in Table 4, it was easy to find that on four measurements, PMPSL-GRAKEL-Euk provided the best performance. Again, PMPSL-GRAKEL-Euk yielded a higher absolute false than pLoc-mEuk.

On accuracy and absolute true, the improvement of PMPSL-GRAKEL-Euk was about 3.8% and 3.4%, respectively, which was quite similar to those on the human dataset. As for the performance of PMPSL-GRAKEL-Euk and pLoc-mEuk on 22 subcellular locations, a box plot was drawn for each measurement, as shown in Figure 10. It can be observed that for SN, ACC and MCC, the boxes of PMPSL-GRAKEL-Euk were at a higher level than those of pLoc-mEuk. For SP, the two boxes were almost at the same level. These suggested that PMPSL-GRAKEL-Euk can yield higher performance than pLoc-mEuk for most locations.

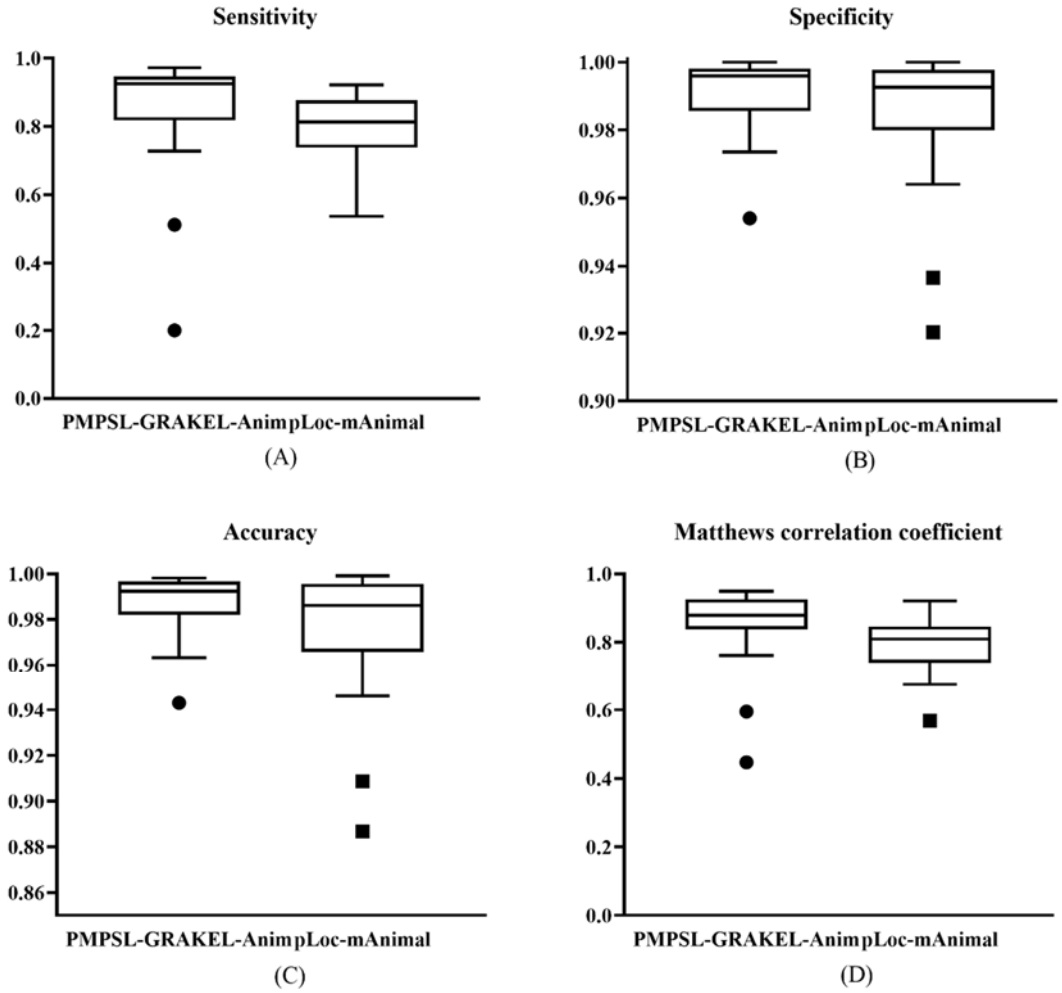

Figure 8. Box plot to show the performances of PMPSL-GRAKEL-Anim and pLoc-mAnimal on 20 subcellular locations. (A) Sensitivity; (B) Specificity; (C) Accuracy; (D) Matthews Correlation Coefficient. The performance of PMPSL-GRAKEL-Anim on 20 subcellular locations is generally better than that of pLoc-mAnimal.

With the above arguments, the proposed classifiers (PMPSL-GRAKEL-Hum, PMPSL-GRAKEL-Anim, PMPSL-GRAKEL-Geng, and PMPSL-GRAKEL-Euk) were superior to previous classifiers (pLoc-mHum, pLoc-mAnimal, pLoc-mGen and pLoc-mEuk), respectively. These previous classifiers adopted ML-GKR as the prediction engine. As ML-GKR is not a strict machine learning

algorithm, which directly makes decisions according to the distribution of data, it cannot mine hidden key patterns to identify different subcellular locations. In this study, the basic prediction engine was RF, which is one of the most classic and powerful machine learning algorithms. On basis of RF, correct patterns for different subcellular locations can be extracted, thereby improving the quality of the classifiers.

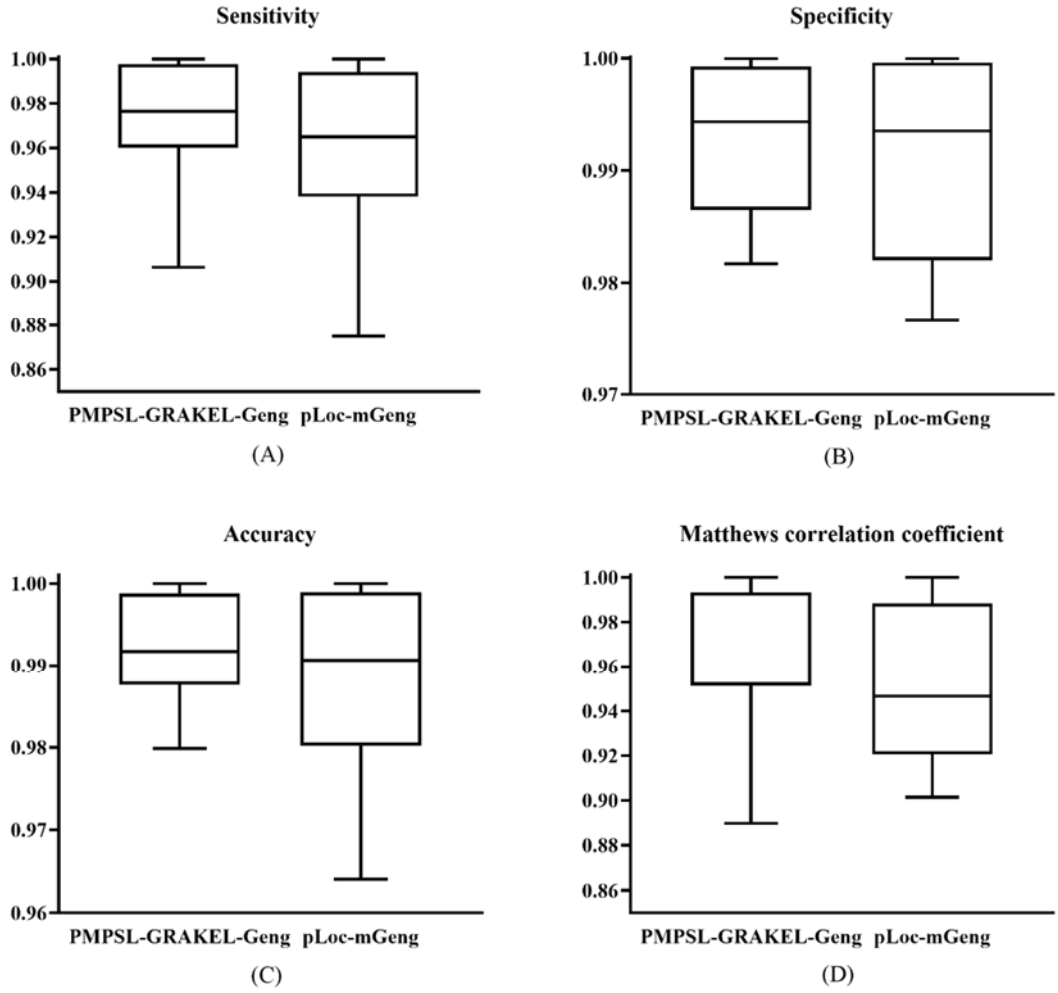

Figure 9. Box plot to show the performances of PMPSL-GRAKEL-Geng and pLoc-mGeng on 8 subcellular locations. (A) Sensitivity; (B) Specificity; (C) Accuracy; (D) Matthews Correlation Coefficient. The performance of PMPSL-GRAKEL-Geng on 8 subcellular locations is slightly better than that of pLoc-mGeng.

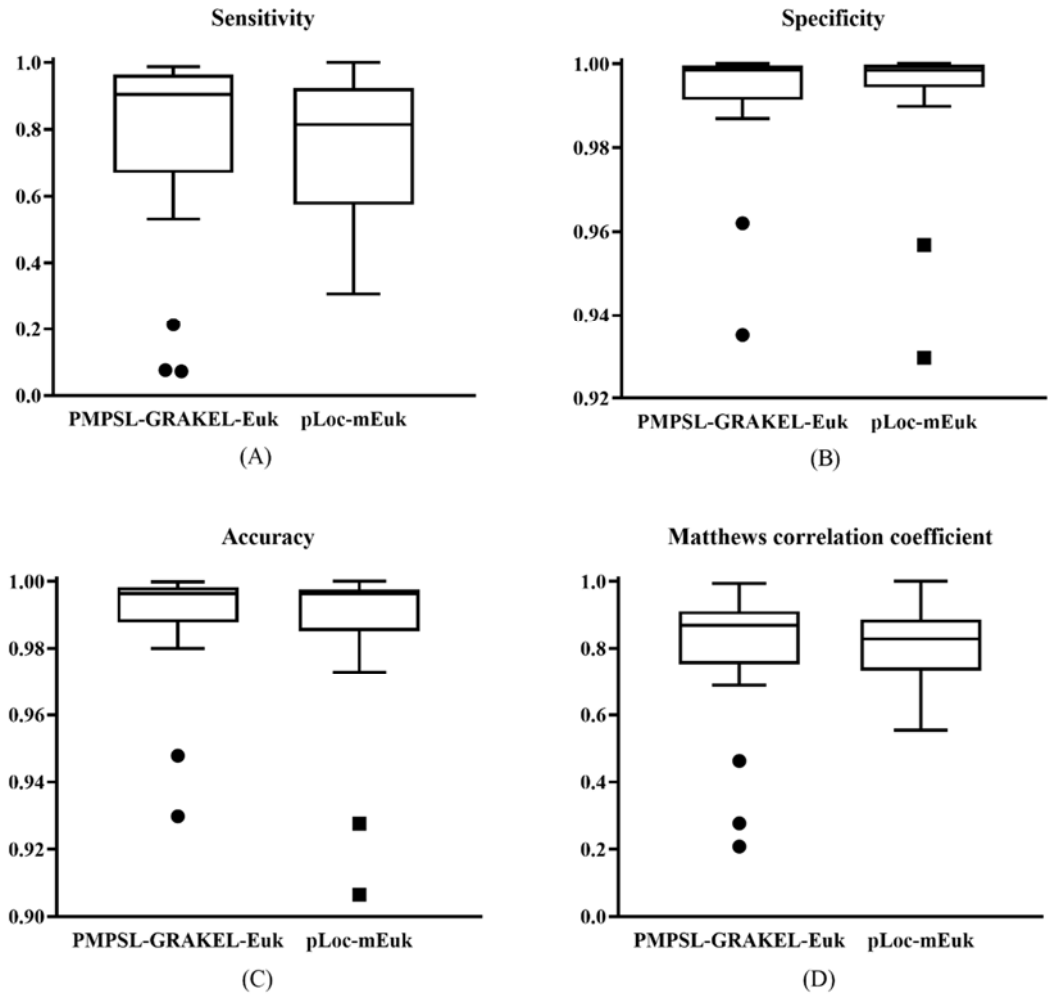

Figure 10. Box plot to show the performances of PMPSL-GRAKEL-Euk and pLoc-mEuk on 22 subcellular locations. (A) Sensitivity; (B) Specificity; (C) Accuracy; (D) Matthews Correlation Coefficient. The performance of PMPSL-GRAKEL-Euk on 22 subcellular locations is generally better than that of pLoc-mEuk.

3.5. Limitations of the classifiers

Although the proposed classifiers provided good performances to identify subcellular locations of proteins of four different types, they still had some weaknesses. The first weakness was the absolute false. The proposed classifiers yielded higher absolute false on three datasets compared with the previous classifiers. The second weakness was the low performance for some subcellular locations. The SN values for some subcellular locations were lower than 10%. Such results may be caused by the imbalanced problem. The proposed classifier was quite sensitive to the sizes of classes.

The above weaknesses may be caused by the limitations of the classifiers. First, for the base prediction engine, RF, in RAKEL, the default parameters were adopted. By tuning its main parameter, number of decision trees, the performances of the four classifiers can be improved. Second, according to Figure 1, the class sizes in the four datasets were not balanced. We did not consider this problem

when constructing the classifiers. Some imbalanced data processing methods can be added to tackle this problem, thereby enhancing the performances of the four classifiers. In the future, we will continue this work to deal with these limitations and build more efficient classifiers.

4. Conclusions

In this study, four multi-label classifiers were built to identify subcellular locations of human, animal, Gram-negative bacterial and eukaryotic proteins. These classifiers adopted the features extracted from GO information of proteins, which also included the label information in the training dataset. Furthermore, a powerful multi-label algorithm, RAKEL, with RF as the basic prediction engine was employed to construct the classifiers. These classifiers yielded quite high performance and were superior to some previous classifiers. We believe that these classifiers can be efficient tools to determine protein subcellular locations. Code and datasets are available at https://github.com/SummerXinTong/PMPSL-GRAKEL.

Use of AI tools declaration

The authors declare they have not used artificial intelligence (AI) tools in the creation of this article.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. K. C. Chou, H. B. Shen, Recent progress in protein subcellular location prediction, *Anal. Biochem.*, **370** (2007), 1–16. https://doi.org/10.1016/j.ab.2007.07.006
2. R. F. Murphy, M. V. Boland, M. Velliste, Towards a systematics for protein subcellular location: quantitative description of protein localization patterns and automated analysis of fluorescence microscope images, in *Proceedings International Conference on Intelligent System Molecular Biology*, **8** (2000), 251–259.
3. J. Cao, W. Liu, J. He, H. Gu, Mining proteins with non-experimental annotations based on an active sample selection strategy for predicting protein subcellular localization, *PLoS One*, **8** (2013), e67343. https://doi.org/10.1371/journal.pone.0067343
4. H. B. Shen, J. Yang, K. C. Chou, Methodology development for predicting subcellular localization and other attributes of proteins, *Expert Rev. Proteomics*, **4** (2007), 453–463. https://doi.org/10.1586/14789450.4.4.453
5. A. Reinhardt, T. Hubbard, Using neural networks for prediction of the subcellular location of proteins, *Nucleic Acids Res.*, **26** (1998), 2230–2236. https://doi.org/10.1093/nar/26.9.2230
6. J. Cedano, P. Aloy, J. A. Perez-Pons, E. Querol, Relation between amino acid composition and cellular location of proteins, *J. Mol. Biol.*, **266** (1997), 594–600. https://doi.org/10.1006/jmbi.1996.0804

7. Y. X. Pan, Z. Z. Zhang, Z. M. Guo, G. Y. Feng, Z. D. Huang, L. He, Application of pseudo amino acid composition for predicting protein subcellular location: stochastic signal processing approach, *J. Protein Chem.*, **22** (2003), 395–402. https://doi.org/10.1023/a:1025350409648
8. J. Y. Shi, S. Zhang, Q. Pan, G. Zhou, Using pseudo amino acid composition to predict protein subcellular location: approached with amino acid composition distribution, *Amino Acids*, **35** (2008), 321–327. https://doi.org/10.1007/s00726-007-0623-z
9. H. Lin, H. Ding, F. Guo, A. Zhang, J. Huang, Predicting subcellular localization of mycobacterial proteins by using Chou's pseudo amino acid composition, *Protein Pept. Lett.*, **15** (2008), 739–744. https://doi.org/10.2174/092986608785133681
10. K. Chou, Prediction of protein cellular attributes using pseudo-amino acid composition, *Proteins*, **43** (2001), 246–255. https://doi.org/10.1002/prot.1035
11. T. Liu, X. Zheng, C. Wang, J. Wang, Prediction of subcellular location of apoptosis proteins using pseudo amino acid composition: an approach from auto covariance transformation, *Protein Pept. Lett.*, **17** (2010), 1263–1269. https://doi.org/10.2174/092986610792231528
12. Y. Shen, J. Tang, F. Guo, Identification of protein subcellular localization via integrating evolutionary and physicochemical information into Chou's general PseAAC, *J. Theor. Biol.*, **462** (2019), 230–239. https://doi.org/10.1016/j.jtbi.2018.11.012
13. Y. H. Yao, Z. X. Shi, Q. Dai, Apoptosis protein subcellular location prediction based on position-specific scoring matrix, *J. Comput. Theor. Nanos.*, **11** (2014), 2073–2078. https://doi.org/10.1166/jctn.2014.3607
14. T. Liu, P. Tao, X. Li, Y. Qin, C. Wang, Prediction of subcellular location of apoptosis proteins combining tri-gram encoding based on PSSM and recursive feature elimination, *J. Theor. Biol.*, **366** (2015), 8–12. https://doi.org/10.1016/j.jtbi.2014.11.010
15. S. Wang, W. Li, Y. Fei, An improved process for generating uniform PSSMs and its application in protein subcellular localization via various global dimension reduction techniques, *IEEE Access*, **7** (2019), 42384–42395. https://doi.org/ 10.1109/ACCESS.2019.2907642
16. X. Cheng, X. Xiao, K. C. Chou, pLoc-mHum: predict subcellular localization of multi-location human proteins via general PseAAC to winnow out the crucial GO information. *Bioinformatics*, **34** (2018), 1448–1456. https://doi.org/10.1093/bioinformatics/btx711
17. X. Cheng, S. Zhao, W. Lin, X. Xiao, K. Chou, pLoc-mAnimal: predict subcellular localization of animal proteins with both single and multiple sites, *Bioinformatics*, **33** (2017), 3524–3531. https://doi.org/10.1093/bioinformatics/btx476
18. X. Cheng, X. Xiao, K.C. Chou, pLoc-mGneg: Predict subcellular localization of Gram-negative bacterial proteins by deep gene ontology learning via general PseAAC, *Genomics*, **110** (2017), 231–239. https://doi.org/10.1016/j.ygeno.2017.10.002
19. X. Cheng, X. Xiao, K. C. Chou, pLoc-mEuk: Predict subcellular localization of multi-label eukaryotic proteins by extracting the key GO information into general PseAAC, *Genomics*, **110** (2018), 50–58. https://doi.org/10.1016/j.ygeno.2017.08.005
20. K. Chou, Y. Cai, A new hybrid approach to predict subcellular localization of proteins by incorporating gene ontology, *Biochem. Biophys. Res. Commun.*, **311** (2003), 743–747. https://doi.org/10.1016/j.bbrc.2003.10.062
21. S. Wan, M. Mak, S. Kung, GOASVM: A subcellular location predictor by incorporating term-frequency gene ontology into the general form of Chou's pseudo-amino acid composition, *J. Theor. Biol.*, **323** (2013), 40–48. https://doi.org/10.1016/j.jtbi.2013.01.012

22. S. Wan, M. Mak, S. Kung, mGOASVM: Multi-label protein subcellular localization based on gene ontology and support vector machines, *BMC Bioinf.*, **13** (2012), 290. https://doi.org/10.1186/1471-2105-13-290
23. K. C. Chou, Y. D. Cai, Using functional domain composition and support vector machines for prediction of protein subcellular location, *J. Biol. Chem.*, **277** (2002), 45765–45769. https://doi.org/10.1074/jbc.M204161200
24. K. Chou, H. Shen, A new method for predicting the subcellular localization of eukaryotic proteins with both single and multiple sites: Euk-mPLoc 2.0, *PLoS One*, **5** (2010), e9931. https://doi.org/10.1371/journal.pone.0009931
25. Y. Cai, K. Chou, Nearest neighbour algorithm for predicting protein subcellular location by combining functional domain composition and pseudo-amino acid composition, *Biochem. Biophys. Res. Commun.*, **305** (2003), 407–411. https://doi.org/10.1016/s0006-291x(03)00775-7
26. K. Chou, Y. Cai, Predicting subcellular localization of proteins by hybridizing functional domain composition and pseudo-amino acid composition, *J. Cell. Biochem.*, **91** (2004), 1197–1203. https://doi.org/10.1002/jcb.10790
27. X. Pan, L. Chen, M. Liu, Z. Niu, T. Huang, Y. Cai, Identifying protein subcellular locations with embeddings-based node2loc, *IEEE/ACM Trans. Comput. Biol. Bioinf.*, **19** (2022), 666–675. https://doi.org/ 10.1109/TCBB.2021.3080386
28. X. Pan, H. Li, T. Zeng, Z. Li, L. Chen, T. Huang, et al., Identification of protein subcellular localization with network and functional embeddings, *Front. Genet.*, **11** (2021), 626500. https://doi.org/10.3389/fgene.2020.626500
29. H. Liu, B. Hu, L. Chen, Identifying protein subcellular location with embedding features learned from networks, *Curr. Proteomics*, **18** (2021), 646–660. https://doi.org/10.2174/1570164617999201124142950
30. R. Wang, L. Chen, Identification of human protein subcellular location with multiple networks, *Curr. Proteomics*, **19** (2022), 344–356.
31. R. Su, L. He, T. Liu, X. Liu, L. Wei, Protein subcellular localization based on deep image features and criterion learning strategy, *Briefings Bioinf.*, **22** (2020), bbaa313. https://doi.org/10.1093/bib/bbaa313
32. M. Ullah, F. Hadi, J. Song, D. Yu, PScL-DDCFPred: an ensemble deep learning-based approach for characterizing multiclass subcellular localization of human proteins from bioimage data, *Bioinformatics*, **38** (2022), 4019–4026. https://doi.org/10.1093/bioinformatics/btac432
33. M. Ullah, K. Han, F. Hadi, J. Xu, J. Song, D. Yu, PScL-HDeep: image-based prediction of protein subcellular location in human tissue using ensemble learning of handcrafted and deep learned features with two-layer feature selection, *Briefings Bioinf.*, **22** (2021), bbab278. https://doi.org/10.1093/bib/bbab278
34. G. Tsoumakas, I. Vlahavas, Random k-Labelsets: An ensemble method for multilabel classification, in *Machine Learning: ECML 2007*, (2007), 406–417. https://doi.org/10.1007/978-3-540-74958-5_38
35. L. Breiman, Random forests, *Mach. Learn.*, **45** (2001), 5–32. https://doi.org/10.1023/A:1010933404324
36. K. C. Chou, Z. C. Wu, X. Xiao, iLoc-Hum: using the accumulation-label scale to predict subcellular locations of human proteins with both single and multiple sites, *Mol. Biosyst.*, **8** (2012), 629–641. https://doi.org/10.1039/c1mb05420a

37. H. B. Shen, K. C. Chou, A top-down approach to enhance the power of predicting human protein subcellular localization: Hum-mPLoc 2.0, *Anal. Biochem.*, **394** (2009), 269–274. https://doi.org/10.1016/j.ab.2009.07.046
38. W. Z. Lin, J. Fang, X. Xiao, K. Chou, iLoc-Animal: a multi-label learning classifier for predicting subcellular localization of animal proteins, *Mol. Biosyst.*, **9** (2013), 634–644. https://doi.org/10.1039/c3mb25466f
39. H. B. Shen, K. C. Chou, Gneg-mPLoc: a top-down strategy to enhance the quality of predicting subcellular localization of Gram-negative bacterial proteins, *J. Theor. Biol.*, **264** (2010), 326–333. https://doi.org/10.1016/j.jtbi.2010.01.018
40. X. Xiao, Z. C. Wu, K. C. Chou, A multi-label classifier for predicting the subcellular localization of gram-negative bacterial proteins with both single and multiple sites, *PLoS One*, **6** (2011), e20592. https://doi.org/10.1371/journal.pone.0020592
41. G. Tsoumakas, I. Katakis, Multi-label classification: An overview, *Int. J. Data Warehouse. Min.*, **3** (2007), 1–13. https://doi.org/10.4018/jdwm.2007070101
42. S. Al-Maadeed, Kernel collaborative label power set system for multi-label classification, in *Qatar Foundation Annual Research Forum Volume 2013 Issue 1*, Hamad bin Khalifa University Press, **2013** (2013). https://doi.org/10.5339/qfarf.2013.ICTP-028
43. J. P. Zhou, L. Chen, Z. H. Guo, iATC-NRAKEL: An efficient multi-label classifier for recognizing anatomical therapeutic chemical classes of drugs, *Bioinformatics*, **36** (2020), 1391–1396. https://doi.org/10.1093/bioinformatics/btz757
44. J. P. Zhou, L. Chen, T. Wang, M. Liu, iATC-FRAKEL: A simple multi-label web-server for recognizing anatomical therapeutic chemical classes of drugs with their fingerprints only, *Bioinformatics*, **36** (2020), 3568–3569. https://doi.org/10.1093/bioinformatics/btaa166
45. X. Li, L. Lu, L. Chen, Identification of protein functions in mouse with a label space partition method, *Math. Biosci. Eng.*, **19** (2022), 3820–3842. https://doi.org/ 10.3934/mbe.2022176
46. H. Li, S. Zhang, L. Chen, X. Pan, Z. Li, T. Huang, et al., Identifying functions of proteins in mice with functional embedding features, *Front. Genet.*, **13** (2022), 909040. https://doi.org/10.3389/fgene.2022.909040
47. L. Chen, Z. Li, T. Zeng, Y. Zhang, H. Li, T. Huang, et al., Predicting gene phenotype by multi-label multi-class model based on essential functional features, *Mol. Genet. Genomics*, **296** (2021), 905–918. https://doi.org/10.1007/s00438-021-01789-8
48. Y. Zhu, B. Hu, L. Chen, Q. Dai, iMPTCE-Hnetwork: a multi-label classifier for identifying metabolic pathway types of chemicals and enzymes with a heterogeneous network, *Comput. Math. Methods Med.*, **2021** (2021), 6683051. https://doi.org/10.1155/2021/6683051
49. J. Che, L. Chen, Z. Guo, S. Wang, Aorigele, Drug target group prediction with multiple drug networks, *Comb. Chem. High Throughput Screen.*, **23** (2020), 274–284. https://doi.org/10.2174/1386207322666190702103927
50. H. Wang, L. Chen, PMPTCE-HNEA: Predicting metabolic pathway types of chemicals and enzymes with a heterogeneous network embedding algorithm, *Curr. Bioinf.*, **18** (2023), 748–759. https://doi.org/10.2174/1574893618666230224121633
51. J. Read, P. Reutemann, B. Pfahringer, MEKA: A multi-label/multi-target extension to WEKA, *J. Mach. Learn. Res.*, **17** (2016), 1–5.
52. B. Ran, L. Chen, M. Li, Y. Han, Q. Dai, Drug-Drug interactions prediction using fingerprint only, *Comput. Math. Methods Med.*, **2022** (2022), 7818480. https://doi.org/10.1155/2022/7818480

53. M. Onesime, Z. Yang, Q. Dai, Genomic island prediction via chi-square test and random forest algorithm, *Comput. Math. Methods Med.*, **2021** (2021), 9969751. https://doi.org/10.1155/2021/9969751
54. L. Chen, K. Chen, B. Zhou, Inferring drug-disease associations by a deep analysis on drug and disease networks, *Math. Biosci. Eng.*, **20** (2023), 14136–14157. https://doi.org/10.3934/mbe.2023632
55. P. Chen, T. Shen, Y. Zhang, B. Wang, A sequence-segment neighbor encoding schema for protein hotspot residue prediction, *Curr. Bioinf.*, **15** (2020), 445–454. https://doi.org/10.2174/1574893615666200106115421
56. Z. B. Lv, J. Zhang, H. Ding, Q. Zou, RF-PseU: A random forest predictor for rna pseudouridine sites, *Front. Bioeng. Biotechnol.*, **8** (2020), 134. https://doi.org/10.3389/fbioe.2020.00134
57. F. Huang, Q. Ma, J. Ren, J. Li, F. Wang, T. Huang, et al., Identification of smoking associated transcriptome aberration in blood with machine learning methods, *Biomed. Res. Int.*, **2023** (2023), 445–454. https://doi.org/10.1155/2023/5333361
58. F. Huang, M. Fu, J. Li, L. Chen, K. Feng, T. Huang, et al., Analysis and prediction of protein stability based on interaction network, gene ontology, and kegg pathway enrichment scores, *Biochim. Biophys. Acta. Proteins Proteom.*, **1871** (2023), 140889. https://doi.org/10.1016/j.bbapap.2023.140889
59. J. Ren, Y. Zhang, W. Guo, K. Feng, Y. Yuan, T. Huang,et al., Identification of genes associated with the impairment of olfactory and gustatory functions in COVID-19 via machine-learning methods, *Life (Basel)*, **13** (2023), 798. https://doi.org/10.3390/life13030798
60. K. C. Chou, C. T. Zhang, Prediction of protein structural classes, *Crit. Rev. Biochem. Mol. Biol.*, **30** (1995), 275–349. https://doi.org/10.3109/10409239509083488
61. K. C. Chou, Z. C. Wu, X. Xiao, iLoc-Euk: A multi-label classifier for predicting the subcellular localization of singleplex and multiplex eukaryotic proteins, *PLoS One*, **6** (2011), e18258. https://doi.org/10.1371/journal.pone.0018258
62. S. Tang, L. Chen, iATC-NFMLP: Identifying classes of anatomical therapeutic chemicals based on drug networks, fingerprints and multilayer perceptron. *Curr. Bioinf.*, **17** (2022), 814–824.
63. H. Zhao, Y. Li, J. Wang, A convolutional neural network and graph convolutional network-based method for predicting the classification of anatomical therapeutic chemicals, *Bioinformatics*, **37** (2021), 2841–2847. https://doi.org/10.1093/bioinformatics/btab204
64. W. Chen, H. Yang, P. Feng, H. Ding, H. Lin, iDNA4mC: identifying DNA N4-methylcytosine sites based on nucleotide chemical properties, *Bioinformatics*, **33** (2017), 3518–3523. https://doi.org/10.1093/bioinformatics/btx479
65. L. Wei, P. Xing, R. Su, G. Shi, Z. S. Ma, Q. Zou, CPPred-RF: A sequence-based predictor for identifying cell-penetrating peptides and their uptake efficiency, *J. Proteome Res.*, **16** (2017), 2044–2053. https://doi.org/ 10.1021/acs.jproteome.7b00019
66. S. R. Safavian, D. Landgrebe, A survey of decision tree classifier methodology, *T-SMCA*, **21** (1991), 660–674. https://doi.org/ 10.1109/21.97458
67. C. Cortes, V. Vapnik, Support-vector networks, *Mach. Learn.*, **20** (1995), 273–297. https://doi.org/10.1007/BF00994018