Mathematical Biosciences and Engineering

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

MBE, 19(4): 3820–3842.
DOI: 10.3934/mbe.2022176
Received: 29 September 2021
Revised: 20 January 2022
Accepted: 25 January 2022
Published: 10 February 2022

Research article

Identification of protein functions in mouse with a label space partition method

Xuan Li[1], **Lin Lu**[2] **and Lei Chen**[1,*]

[1] College of Information Engineering, Shanghai Maritime University, Shanghai 201306, China
[2] Department of Radiology, Columbia University Medical Center, New York 10032, USA

* **Correspondence:** Email: chen_lei1@163.com; Tel: +862138282825; Fax: +862138282800.

Abstract: Protein is very important for almost all living creatures because it participates in most complicated and essential biological processes. Determining the functions of given proteins is one of the most essential problems in protein science. Such determination can be conducted through traditional experiments. However, the experimental methods are always time-consuming and of high costs. In recent years, computational methods give useful aids for identification of protein functions. This study presented a new multi-label classifier for identifying functions of mouse proteins. Due to the number of functional types, which were termed as labels in the classification procedure, a label space partition method was employed to divide labels into some partitions. On each partition, a multi-label classifier was constructed. The classifiers based on all partitions were integrated in the proposed classifier. The cross-validation results proved that the proposed classifier was of good performance. Classifiers with label partition were superior to those without label partition or with random label partition.

Keywords: mouse protein, protein function, multi-label classification, label space partition, RAKEL

1. Introduction

Protein is a major component for almost all living creatures. It is highly related to the maintenance of normal physical functions in cells [1]. Several complicated and essential biological processes need proteins to participate in, such as cell proliferation [2], DNA replication [3], enzyme-mediated metabolic processes [4], etc. Furthermore, protein provides important contributions to construct basic cellular structure, maintain cellular microenvironment and form complex macrostructures. Thus, the research of protein-related problems is quite hot in recent years. Determination of the functions of proteins is one of the essential problems. Experimental determination is a solid method. However, it also has some evident shortcomings, such as high cost and low efficiency. Thus, it is of great urgency to design novel methods with low cost and high efficiency.

In recent years, several computational methods have been designed to identify protein functions. Most of them are data-driven methods. Based on lots of proteins with annotated functions, which can be obtained from some public databases, models were set up by using some existing or newly designed computer algorithms. The basic computational method to identify protein functions is based on protein sequence similarity measured by BLAST [5]. Other methods, such as sequence motif based methods (PROSITE) [6], profile-based methods (PFAM) [7], structure-based methods (FATCAT and ProCAT) [8], were also proposed to identify protein functions. In recent years, network-based methods become more and more popular to tackle some protein-related problems. Two previous studies employed protein network information to design hybrid approaches for the identification of protein functions. The method that adopted protein network information is an important step to identify protein functions [9,10]. Other steps used methods based on protein sequence similarity or biochemical and physicochemical description of proteins. Most established methods always focused on proteins, analyzing their sequences, properties, etc. Few studies considered function labels. As inspired by some studies on drug-related problems [11,12], which considered label information and improved the performance of classifiers, the associations of function labels may also be important information for protein function identification.

In this study, we constructed a multi-label classifier with a label space partition to identify protein functions. To conduct this investigation, we selected proteins of mouse, one of the most extensively studied organisms, as the research object. Proteins and their function annotations were retrieved from MfunGD [13]. 24 functional types were reported in such database. A label space partition method, incorporating Louvain method [14], was applied to analyze the associations of 24 functional types, resulting in some subsets of types. To prove such partition can improve the performance of classifiers, we set up several classifiers with RAndom k-labELsets (RAKEL) [15], with support vector machine (SVM) [16] or random forest (RF) [17] as the base classifier. On each type subset, a multi-label classifier was set up and they were integrated in the proposed classifiers. The results indicated that classifiers with a label space partition were always superior to those without considering the partition of functional types. Furthermore, these classifiers also provided better performance than those with a random partition of functional types.

2. Materials and methods

2.1. Datasets

We sourced the mouse proteins and their functional types from one previous study [9]. This information was retrieved from MfunGD (http://mips.gsf.de/genre/proj/mfungd/) [13], a public database collecting annotated mouse proteins and their occurrence in protein networks. In such database, mouse proteins were classified into 24 types, which are illustrated in Figure 1. The types of each mouse protein were determined by manually checking its annotation in the literature and GO annotation [18,19]. Because we encoded mouse proteins according to their functional domain or interaction information, those without these two types of information were excluded. Finally, a dataset consisting of 9655 mouse proteins were constructed. These proteins were also classified into above mentioned 24 functional types. The number of proteins in each functional type is also shown in Figure 1. It was easy to obtain that the sum of protein numbers in all 24 types were 29850, which was much larger than the number of different mouse proteins (9655). This fact implied that several proteins belonged to two or more functional types. Determination of functional types of mouse proteins was evidently a multi-label classification problem if functional types were deemed as labels.

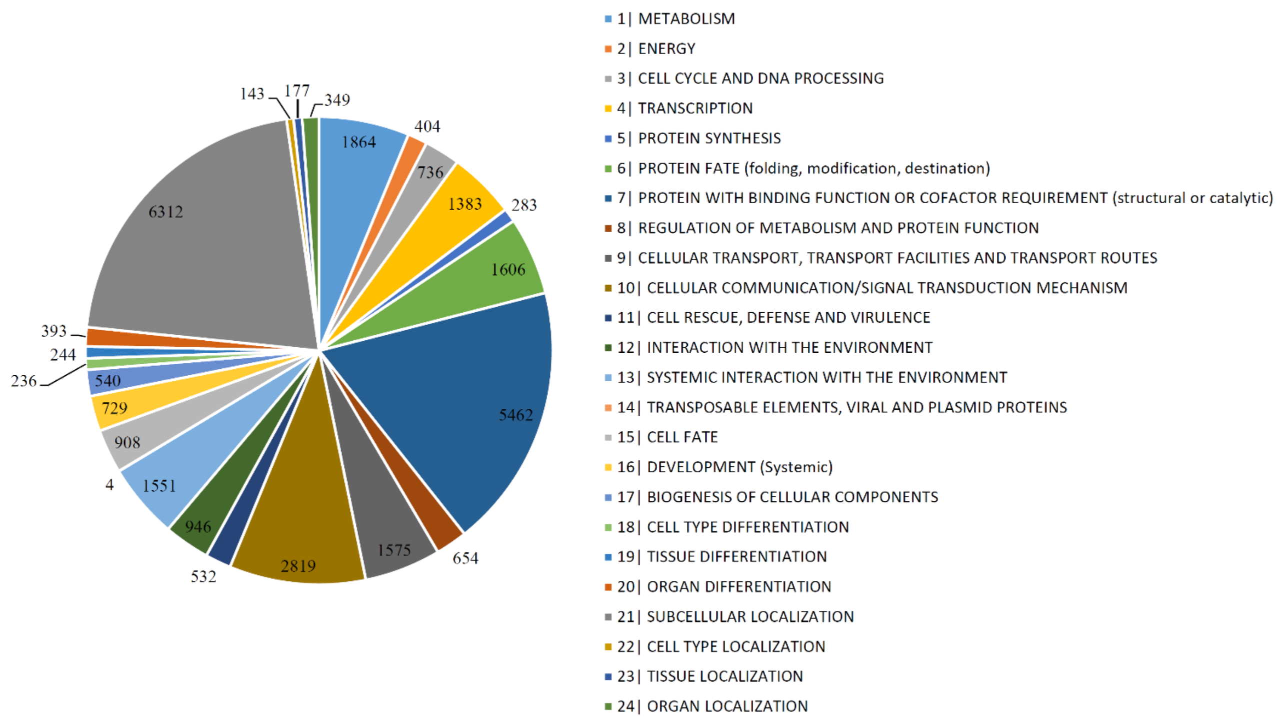

Figure 1. Pie chart to show the number of mouse proteins in each functional type.

2.2. Label space partition

As mentioned above, mouse proteins in MfunGD were classified into 24 functional types and assigning these types to given proteins was a multi-label classification problem, where types were termed as labels. Due to the number of labels, it was difficult to directly build powerful multi-label classifiers. The partition of label set may be helpful to optimize classifiers as inspired by some studies on drug-related problems [11,12]. Thus, this section proposed a label space partition method to divide labels into some label subsets.

To implement this method, a label network was constructed first. Given a training dataset D with h labels (h = 24 in this study), denoted by $l_1, l_2, \dots, l_h$, the label set for one sample s was defined as $L(s)$. For each label l_i $(1 \leq i \leq h)$, samples having such label constituted a sample subset, denoted as $SL(l_i)$, that is

$$SL(l_i) = \{s: s \in D \text{ } and \text{ } l_i \in L(s)\} \tag{1}$$

The label network defined labels as nodes and two nodes were connected by an edge if and only if their corresponding labels, say l_i and l_j, had common samples, that is $SL(l_i) \cap SL(l_j) \neq \emptyset$. Furthermore, a weight was assigned to each edge for indicating the different association strength of labels. For an edge *e*, its weight was defined by

$$w(e) = \left|SL(l_i) \cap SL(l_j)\right| \tag{2}$$

where l_i and l_j were the endpoints of edge *e*. For an easy description, let us denoted such label network by N_L.

The Louvain method [14], a community detection algorithm, was performed on the label network N_L to classify labels into some subsets. Such method adopts a greedy aggregation scheme to detect communities such that nodes in each detected community have strong associations. Initially, each node in the network constitutes a community. A loop procedure is executed. In each round, two communities are selected and merged when such merging can provide highest contribution to modularity. For a node *n* and community *C*, the gain in modularity, denoted by ΔQ, by merging *n* and *C* is defined as

$$\Delta Q = \left[\frac{\Sigma_{in} + k_{n,in}}{2m} - \left(\frac{\Sigma_{tot} + k_n}{2m}\right)^2\right] - \left[\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 - \left(\frac{k_n}{2m}\right)^2\right] \tag{3}$$

where Σ_{in} stands for the overall weights of edges inside *C*, Σ_{tot} stands for the overall weights of edges adjacent to nodes in *C*, $k_{n,in}$ represents the overall weights of edges connecting *n* and nodes in *C*, k_n denotes the overall weights of edges adjacent to *n*, *m* is the overall weights of edges in the network. For each node *n*, the gain in modularity by merging it and each of its neighbor is computed. The merging producing the highest gain in modularity is selected and a new network is constructed. In details, if such merging involves node *n* and community *C*, the new network combines *n* and community *C*, producing a new node n'. The weight of an edge connecting n' and another node n'' in the network is updated as the overall weights of edges connecting *n* (*C*) and n''. In the next round, above procedure is executed on the new network. The loop stops until the gain in modularity cannot be positive. The remaining communities in the network indicate a label partition.

In this study, the Louvain method was performed on the label network N_L. By refining its outcome, we can access a label partition. Let us denote the label partition as $L_1, L_2, \ldots, L_t$.

2.3. Feature engineering

Efficient classifiers always adopt informative features of samples, which contain essential properties of samples as much as possible. This study employed two schemes to encode each mouse protein. The first scheme extracted features derived from functional domain information of proteins through a natural language processing approach, whereas the second one generated features from several protein-protein interaction (PPI) networks. Their descriptions were as below.

2.3.1. Domain embedding features

Functional domain information is deemed to be useful to investigate various protein-related problems [20–24]. Here, we also adopted such information to encode each mouse protein.

We retrieved the functional domain information of all mouse proteins from InterPro database (http://www.ebi.ac.uk/interpro/, accessed in October 2020) [25]. This information contained 48739 mouse proteins, covering 16797 domains. Each domain was termed as words, whereas mouse proteins, annotated by domains, were deemed as sentences. Then, such above information was fed into the well-known natural language processing approach, word2vec [26,27], to learn embedding features of domains. As a result, each domain was encoded by a 256-D feature vector. Here, the word2vec program retrieved from https://github.com/RaRe-Technologies/gensim was adopted. It was executed with its default parameters.

The feature vectors of domains were further refined to represent each mouse protein. For each mouse protein, it was encoded by a vector, which was defined as the average of vectors of domains that were annotated on such protein. Thus, each protein was also represented by 256 features. For convenience, such obtained features were called domain embedding features.

2.3.2. Network embedding features

Network has been deemed to be a popular research form because it can organize objects at a system level. However, a gap exists between network and traditional machine learning algorithms. This gap promotes the process of network embedding algorithms, which can abstract linkage in one or more networks and learn features for each node in the network(s). In recent years, several network embedding algorithms, such as DeepWalk [28], Node2vec [29], and Mashup [30], etc. have been proposed. Some of them have been applied to tackle different protein-related problems [30−34]. Features obtained by network embedding algorithms are quite different from those extracted from inherent properties of samples and can reflect different aspects of samples. Here, we adopted Mashup to extract features of mouse proteins from several PPI networks.

We used the mouse PPI information collected in STRING (https://www.string-db.org/, Version 10.0) [35], a public database containing interaction of 9,643,763 proteins from 2031 organisms. Interactions in this database are derived from five main sources: Genomic context predictions, High-throughput lab experiments, (Conserved) Co-expression, Automated textmining, Previous knowledge in databases. Accordingly, they can widely evaluate the associations of proteins. The mouse PPI information involves 20648 mouse proteins and 5,109,107 interactions. Each interaction is assigned eight scores, where the first seven scores measure the association of proteins from some aspect of proteins and they are integrated in the last score. For each of first seven scores, a PPI network was constructed, where proteins were defined as nodes and two nodes were connected by an edge when their corresponding proteins can constitute a PPI with such score larger than zero. In addition, this score was assigned to the edge as its weight. Accordingly, seven PPI networks were built, which can be used to extract informative features of mouse proteins.

The network embedding algorithm, Mashup [30], was executed on above constructed seven PPI networks. To our knowledge, it is the only network embedding algorithm that can process multiple networks. This method contains two stages to extract features for each node. In the first stage, each node in each network is assigned a raw feature vector on the basis of random walk with restart algorithm [36,37]. In this way, several raw feature vectors are produced for the same node. It is necessary to combine them into one vector. At the same time, the dimensionality reduction is also inevitable because of the high dimension of raw feature vectors, which is equal to the node number in the network. All these are done in the second stage. It supposes a uniform vector for each node and a context vector for any node in any network. Based on them, it produces an approximate vector for any node in any network. The optimal components in above two types of vectors were determined by solving an optimized problem such that the produced approximate vectors based on them should be approximate to raw feature vectors as much as possible. For details, please refer to reference [30].

This study adopted the Mashup program downloaded from http://cb.csail.mit.edu/cb/mashup/. Likewise, it was executed with the default parameters. For the dimension of feature vectors, we tried various values between 100 and 300. For convenience, features produced by Mashup were called network embedding features.

Accordingly, each mouse protein can be represented by three forms: (1) domain embedding features; (2) network embedding features; (3) domain and network embedding features.

2.4. Multi-label classifier

As mentioned in Section 2.1, several mouse proteins belonged to two or more functional types. A natural way to assign types to given proteins is to design a multi-label classifier. Generally, there are two schemes to construct multi-label classifiers: problem transformation and algorithm adaption [38]. The former one transforms the original multi-label classification problem into some single-label classification problems. The later one generalizes the single-label classification algorithm so that it can process samples with more than one labels. Here, we adopted a widely used problem transformation method, called RAKEL [15], to construct the multi-label classifier.

RAKEL is a generalized method of label powerset (LP) algorithm. Given a dataset with h labels, say $l_1, l_2, \ldots, l_h$, randomly construct m label subsets, each of which consists of k labels. For each of these label subsets, new labels are defined as the members in its power set. These new labels are assigned to samples based on their original labels. After such operation, each sample is assigned only one new label. Samples with their new labels constitute a new dataset. A classifier is set up by training some single-label classification algorithm on such new dataset. Accordingly, m classifiers can be set up, which are integrated in RAKEL. For a query sample x, each classifier gives a binary prediction result (0 or 1) for each label l_i. RAKEL calculates the average vote rate for each label l_i. When the average vote rate is greater than a given threshold (Generally, it is set to 0.5), l_i is assigned to x. For an easy description, classifiers built by RAKEL were termed RAKEL classifiers in this study. To quickly implement RAKEL, the tool “RAKEL” in Meka (http://waikato.github.io/meka/) [39] was directly employed. The main parameters of RAKEL, m and k, were tuned in this study.

As mentioned in Section 2.2, all labels can be divided into t partitions, say $L_1, L_2, \ldots, L_t$. For each partition, a new dataset is constructed by restricting labels of each sample into this partition. For instance, if one sample is assigned three labels, say l_1, l_2, l_3 and l_1, l_3 belongs to one partition, this sample is assigned l_1, l_3 as its labels in the new dataset. Accordingly, a RAKEL classifier is built on the new constructed dataset. The final classifier integrates these RAKEL classifiers by collecting their results. In detail, for a query sample, each RAKEL classifier yields its prediction (i.e., a label subset). The final prediction is the union of label subsets yielded by all RAKEL classifiers.

2.5. Base classifier

When building the RAKEL classifiers, a single-label classification algorithm is needed. In this study, two powerful classification algorithms were employed: SVM [16] and RF [17].

SVM is a popular classification algorithm based on statistical learning theory [31,34,40−46]. Its principle is to use a kernel function to map samples from the original space to a higher-dimensional feature space so that samples are linearly separable in the new space. So far, several types of SVM have been designed to process different problems. Here, one type of SVM was adopted. The sequential minimal optimization (SMO) algorithm [47] was employed to optimize the training procedures of this type of SVM. A polynomial kernel or an RBF kernel was set as its kernel.

RF is another powerful classification algorithm, which has been widely applied to tackle various biological problems [48−54]. In fact, it is an ensemble algorithm, integrating several decision trees. To set up each decision tree, it randomly selects samples from the given dataset, with replacement, and features to extend the tree at each node. For a query sample, all decision trees provide their predictions. These predictions are integrated in RF by majority voting. It is widely accepted that decision tree is a relative weak classifier. However, RF is much more powerful [55].

The above SVM and RF algorithms are all implemented by corresponding tools in Meka [39]. These tools were directly employed in this study.

2.6. Performance assessment

All multi-label classifiers constructed in this study were assessed by ten-fold cross-validation [56]. Such method first divides the original dataset, denoted by D, into 10 mutually exclusive subsets with similar size, i.e., $D = D_1 \cup D_2 \cup \dots D_{10}, D_i \cap D_j = \emptyset(i \neq j, 1 \leq i,j \leq 10)$. Each subset, say D_i, is picked up as test dataset and remaining nine subsets constitute the training dataset. The classifier built on the training dataset is applied to the test dataset. Thus, each sample is exactly tested once.

For the results of ten-fold cross-validation, we can compute some measurements to assess the quality of results. In this study, we employed three widely used measurements in multi-label classification: accuracy, exact matching and hamming loss. To list their formulas, some notations are necessary. Given a dataset with n samples and m labels, suppose that L_i and L'_i are the sets of true labels and predicted labels, respectively, of the ith sample. Above three measurements can be computed by

$$\begin{cases} \text{Accuracy} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\|L_i \cap L'_i\|}{\|L_i \cup L'_i\|}\right) \\ \text{Exact match} = \frac{1}{n}\sum_{i=1}^{n}\nabla(L_i, L'_i) \\ \text{Hamming loss} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\|L_i \cup L'_i - L_i \cap L'_i\|}{m}\right) \end{cases} \tag{4}$$

where ∇ is defined as below:

$$\nabla(L_i, L'_i) = \begin{cases} 1 & \text{if } \mathrm{L}_i \text{ is identifical to } \mathrm{L}'_i \\ 0 & otherwise \end{cases} \tag{5}$$

Evidently, the high accuracy and exact match indicate the good performance of the classifier, whereas it is on the contrary for hamming loss.

When comparing the performance of different classifiers, different results may be concluded according to different measurements. The ranges of accuracy, exact match and hamming loss are all between 0 and 1. Accuracy and exact match have the same trend to represent the performance of classifiers, that is, higher value represents higher performance; whereas hamming loss suggest the contrary trend, that is, lower value suggests higher performance. Thus, we refined hamming loss as 1-

hamming loss to make it having the same trend as accuracy and exact match. In this case, accuracy, exact match and 1-hamming loss can multiply together to define a new measurement, called integrated score in this study, formulated by

$$Integrated\ score = Accuracy * Exact\ match * (1 - hamming\ loss) \tag{6}$$

The higher the integrated score, the higher the performance of the classifier. This measurement has also used in some previous studies [45,57].

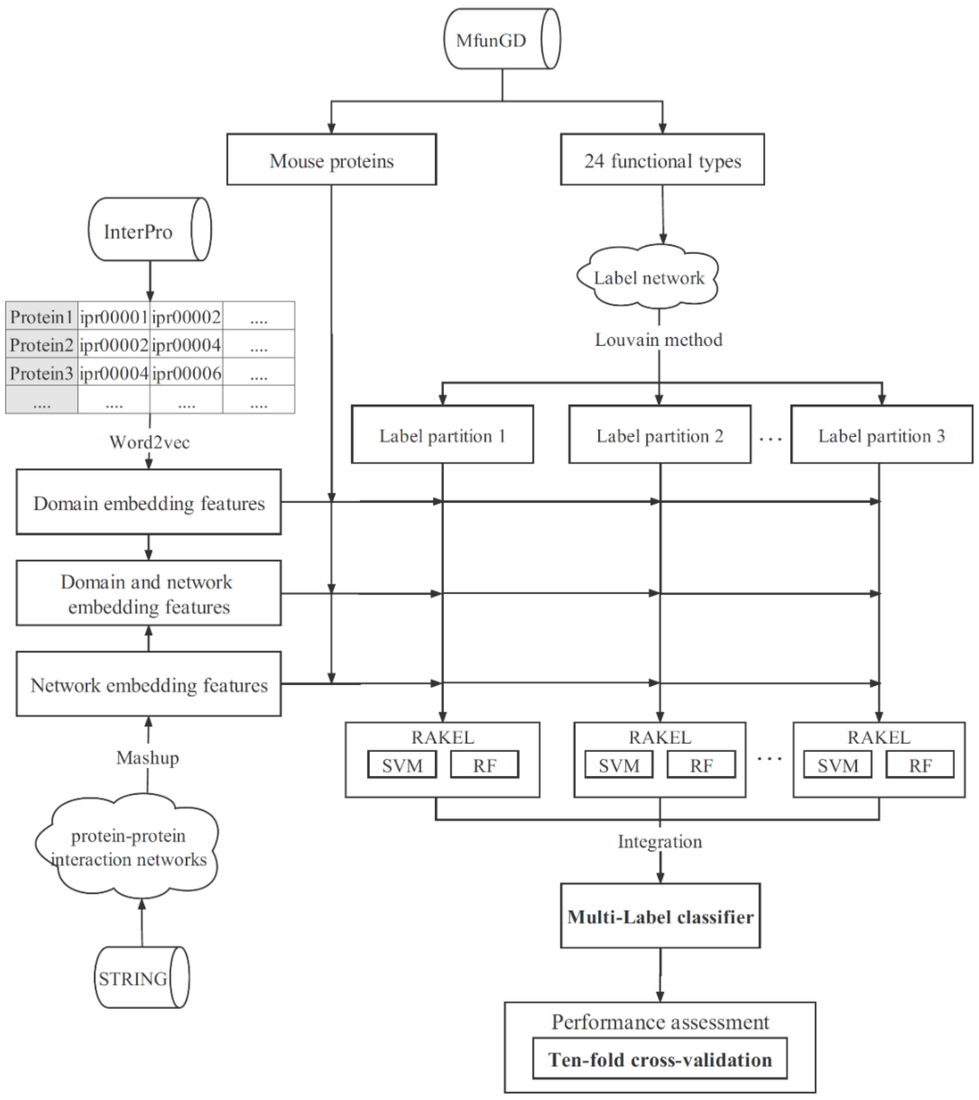
MfunGD
Mouse proteins
24 functional types
InterPro
Label network
Louvain method
Protein1 ipr00001 ipr00002 ....
Protein2 ipr00002 ipr00004 ....
Protein3 ipr00004 ipr00006 ....
Word2vec
Label partition 1
Label partition 2
Label partition 3
Domain embedding features
Domain and network embedding features
Network embedding features
Mashup
RAKEL
SVM
RF
protein-protein interaction networks
Integration
Multi-Label classifier
STRING
Performance assessment
Ten-fold cross-validation

Figure 2. Entire procedures for constructing and evaluating the multi-label classifier. Mouse proteins and their functional annotations (types) are retrieved from MfunGD. The types are analyzed by Louvain method, generating some label partitions. Proteins are represented by two feature types, where one is derived from functional domains via Word2vec and the other is derived from protein-protein interaction networks via Mashup. For each label partition, a classifier is built by RAKEL with support vector machine (SVM) or random forest (RF) as base classifier based on each type of features or both of them. The final multi-label classifier integrates above classifiers and it is assessed by ten-fold cross-validation.

3. Results and Discussion

In this study, we proposed a multi-label classifier to identify mouse protein functions, incorporating the procedure of analyzing the associations of functional types. Two types of features (domain and network embedding features) were adopted to encode proteins. RAKEL was employed to construct classifiers. The entire procedures are illustrated in Figure 2. In this section, the detailed evaluation results would be given and some comparisons were conducted.

3.1. Performance of classifiers with domain embedding features

For the protein features derived from its functional domain information, we adopted RAKEL with a certain base classifier to construct multi-label classifiers. Three base classifiers were tried in this study: (1) SVM with polynomial kernel, (2) SVM with RBF kernel, (3) RF. For two types of SVM, the regularization parameter C was set to 0.5, 1 and 2, the exponent of polynomial kernel was set to its default value (one) and the parameter γ of RBF kernel was also set to its default value (0.01). As for RF, its main parameter, number of decision trees, was tuned, including various values between 10 and 300. The main parameter m for RAKEL was set to its default value 10, the other parameter k of RAKEL was wet to 2, 3, 4, 5. The grid search was adopted to set up all RAKEL classifiers, which were assessed by ten-fold cross-validation, and extract the optimum parameters for each base classifier. The best performance, measured by integrated score, for each base classifier is provided in Table 1, in which the best parameters for each base classifier are also provided. The integrated scores for three base classifiers were 0.1026, 0.0611 and 0.1574. Evidently, the RAKEL classifier with RF provided the best performance. Its accuracy, exact match and hamming loss were 0.6025, 0.2806 and 0.0687, respectively, which were all best compared with those of RAKEL classifiers with other two base classifiers. At a glance, these three RAKEL classifiers were not good enough. However, they were better than classifiers without label partition, which would be elaborated in Section 3.4.

In addition, to fully evaluate the best RAKEL classifier with a certain base classifier, it was further assessed by ten-fold cross-validation for ten times. The performance under ten-fold cross-validation for ten times is shown in Figure 3, from which we can see that all four measurements yielded by each RAKEL classifier varied in a small range, indicating the classifiers with label partition were quite stable no matter how samples were divided.

Table 1. Performance of RAKEL classifiers with different base classifiers on domain embedding features.

Base classifier	**Parameter**	**Accuracy**	**Exact match**	**Hamming loss**	**Integrated score**
Support vector machine (Polynomial kernel)	$m = 10$, $k = 5$, $C = 2$, exponent = 1	0.5329	0.2087	0.0777	0.1026
Support vector machine (RBF kernel)	$m = 10$, $k = 5$, $C = 2$, $\gamma = 0.01$	0.4643	0.1441	0.0872	0.0611
Random forest	$m = 10$, $k = 3$, number of decision trees = 250	0.6025	0.2806	0.0687	0.1574

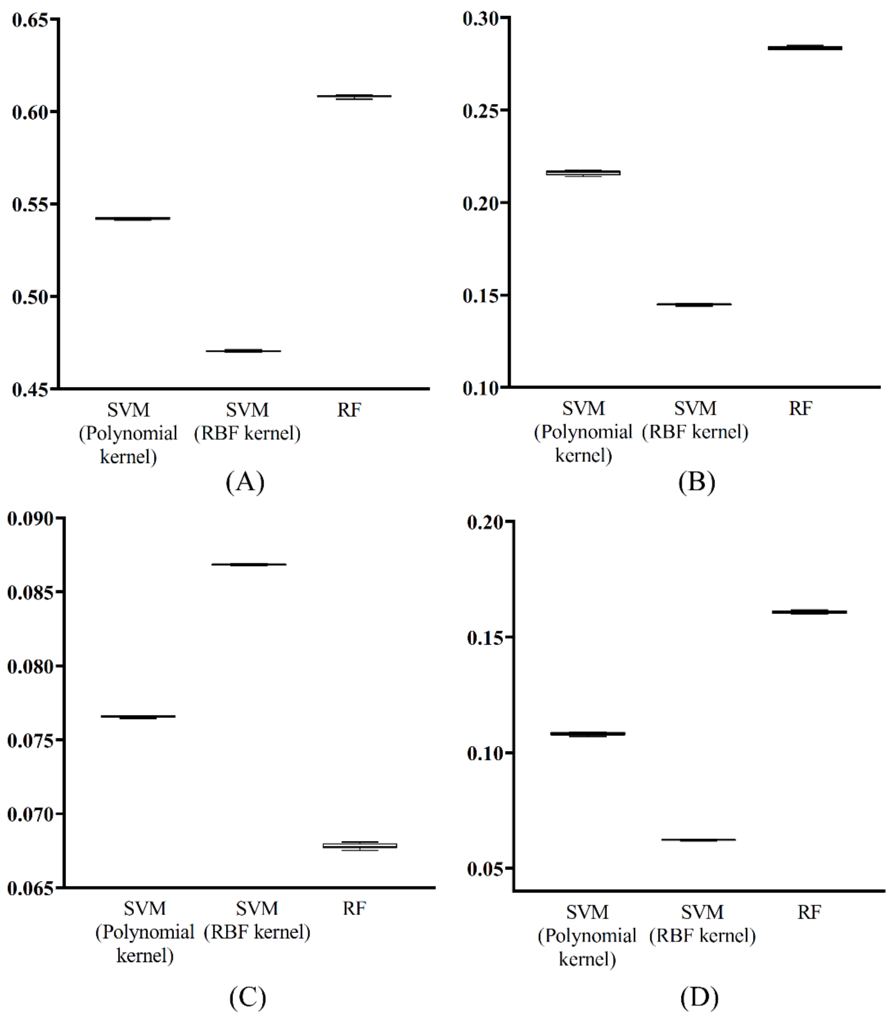

Figure 3. Box plot to show the performance of three RAKEL classifiers using domain embedding features. (A) Accuracy; (B) Exact match; (C) Hamming loss; (D) Integrated score.

3.2. Performance of classifiers with network embedding features

For the network embedding features derived from seven protein networks, a similar procedure was conducted. The same parameters were tried for three base classifiers and RAKEL. Furthermore, the dimension of features was also tuned, including 100, 150, 200, 250 and 300. The grid search was also used to build all RAKEL classifiers, which were further assessed by ten-fold cross-validation. The best RAKEL classifier with a certain base classifier was found and its performance is listed in Table 2. The optimum parameters for each base classifier are also provided in this table. The integrated scores for three base classifiers were 0.1308, 0.0714 and 0.1269, respectively. Clearly, the RAKEL classifier with SVM (polynomial kernel) generated the best performance, where the accuracy, exact match and hamming loss were 0.5853, 0.2407 and 0.0713. These measurements were best among those yielded by three RAKEL classifiers listed in Table 2. Compared with the performance of RAKEL classifiers based on domain embedding features, the superiority of RAKEL classifiers with network embedding features depended on the base classifier. The SVM base classifier gave better performance, whereas

RF base classifier yielded lower performance.

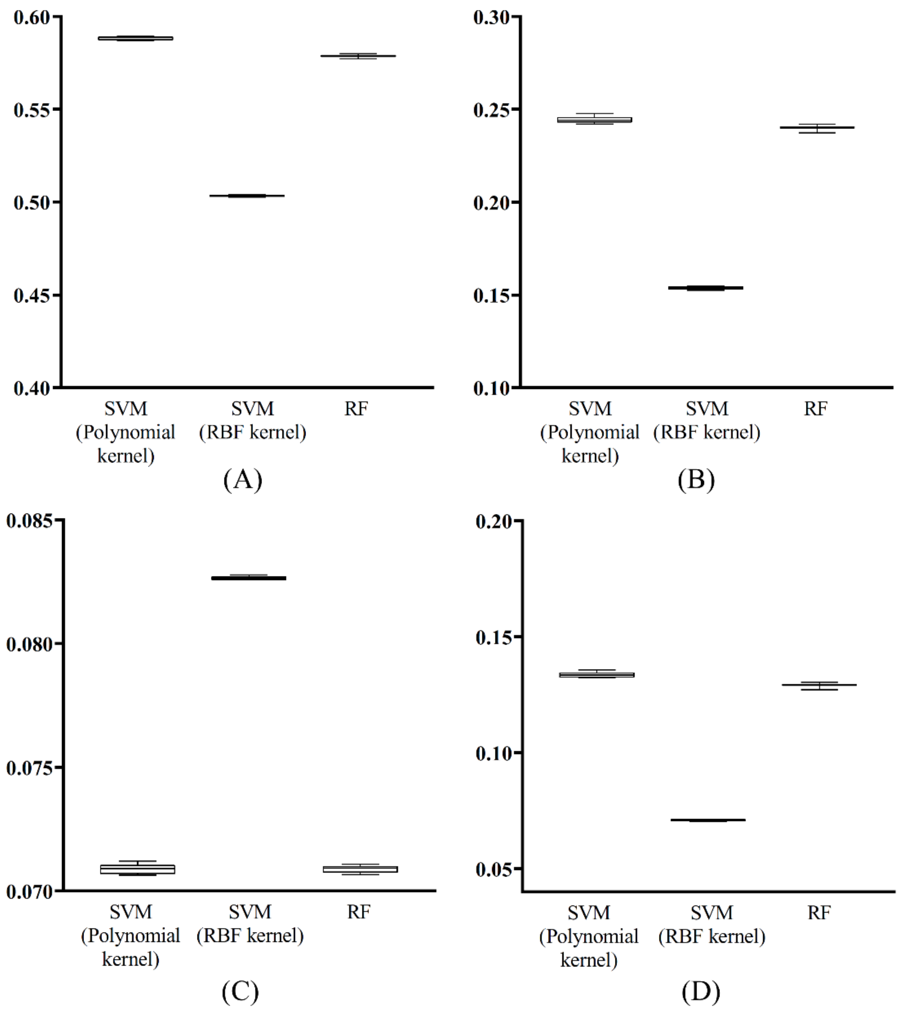

Figure 4. Box plot to show the performance of three RAKEL classifiers using network embedding features. (A) Accuracy; (B) Exact match; (C) Hamming loss; (D) Integrated score.

Table 2. Performance of RAKEL classifiers with different base classifiers on network embedding features.

Base classifier	**Parameter**	**Accuracy**	**Exact match**	**Hamming loss**	**Integrated score**
Support vector machine (Polynomial kernel)	m = 10, k = 5, C = 2, exponent = 1, feature dimension = 300	0.5853	0.2407	0.0713	0.1308
Support vector machine (RBF kernel)	m = 10, k = 3, C = 2, γ = 0.01, feature dimension=300	0.5020	0.1551	0.0824	0.0714
Random forest	m = 10, k = 5, number of decision trees = 250, feature dimension = 150	0.5727	0.2385	0.0714	0.1269

Likewise, for the best RAKEL classifiers with different base classifiers, they were further evaluated by additional ten-fold cross-validation for ten times. A box plot was shown in Figure 4 for each measurement. It is easy to see that each measurement of each classifier was changed in a small range, suggesting the stability of three RAKEL classifiers. This result was almost same as those based on the domain embedding features.

3.3. Performance of classifiers with domain and network embedding features

Two types of features were adopted in this study to represent mouse proteins. They indicated essential properties of proteins from different aspects. The combination of these two types of features can be helpful to construct more efficient classifiers. Thus, we constructed RAKEL classifiers using both domain and network embedding features. To save time, we only tried the parameters listed in Tables 1 and 2. The best performance of RAKEL classifiers with different base classifiers are provided in Table 3. The integrated scores for three base classifiers were 0.1619, 0.1096 and 0.1731, respectively. Each of them was higher than the RAKEL classifiers with the same base classifier and domain or network embedding features. Furthermore, it can be observed from Tables 1−3 that given the same base classifier, the classifier with domain and network embedding features always generated higher accuracy, exact match and lower hamming loss than that with only domain or network embedding features. Therefore, the domain and network embedding features can complement each other so that their combination can improve the performance of classifiers.

Table 3. Performance of RAKEL classifiers with different base classifiers on domain and network embedding features.

Base classifier	**Parameter**	**Accuracy**	**Exact match**	**Hamming loss**	**Integrated score**
Support vector machine (Polynomial kernel)	m = 10, k = 5, C = 2, exponent = 1, network embedding feature dimension = 300	0.6242	0.2777	0.0660	0.1619
Support vector machine (RBF kernel)	m = 10, k = 5, C = 2, γ = 0.01, network embedding feature dimension = 300	0.5439	0.2177	0.0743	0.1096
Random forest	m = 10, k = 5, number of decision trees = 250, network embedding feature dimension = 150	0.6235	0.2963	0.0633	0.1731

3.4. Comparison of classifiers without label partition

In this study, the label partition was employed to construct multi-label classifiers for identifying functions of mouse proteins. To elaborate the merits of label partition, we also built RAKEL classifiers that did not adopt the label partition. All parameters for three base classifiers and RAKEL were tried for each feature type. All such classifiers were also assessed by ten-fold cross-validation.

For the classifiers with each base classifier and domain embedding features, we plotted a violin to show their performance on each measurement under different parameters, as shown in Figure 5. For an easy comparison, those yielded by classifiers that employed the label partition were also provided in this figure. It can be observed that the accuracy, exact match and integrated score yielded by classifiers with label partition were all higher than those obtained by classifiers without label partition. As for the hamming loss, it was on the contrary. All these indicated that the employment of label partition can improve the performance of classifiers. For the other feature type, network embedding features, same tests were conducted. The violins of four measurements are illustrated in Figure 6. The same conclusion can be concluded, that is, the classifiers with label partition were generally superior to those without label partition.

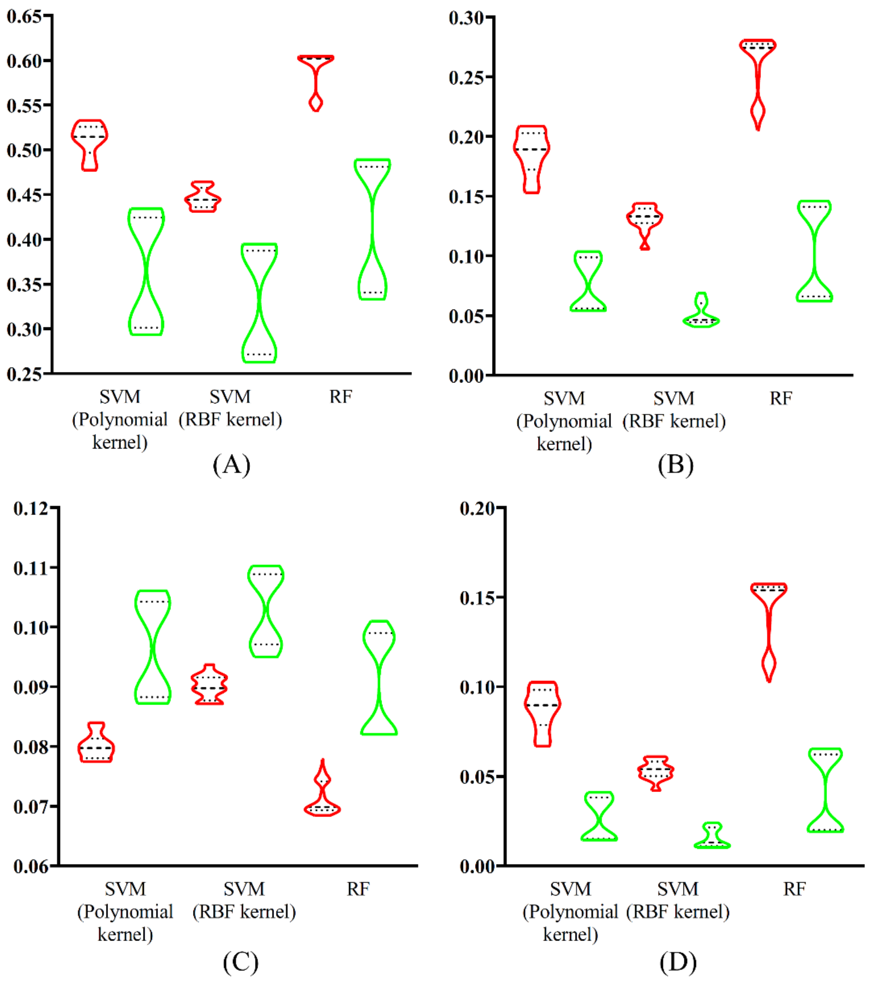

Figure 5. Violin plot to compare RAKEL classifiers using domain embedding features with or without label partition. Red violins are for RAKEL classifiers with label partition and green violins are for RAKEL classifiers without label partition. (A) Accuracy; (B) Exact match; (C) Hamming loss; (D) Integrated score.

For the classifiers using both domain and network embedding features, we tested them with parameters listed in Table 3 when the label space partition procedure was not used. The results of ten-fold cross-validation are listed in Table 4. Evidently, classifiers without label partition were much inferior to those with label partition, suggesting the effectiveness of the label partition.

Table 4. Performance of RAKEL classifiers using domain and network embedding features but without label partition.

Base classifier	Parameter	Accuracy	Exact match	Hamming loss	Integrated score
Support vector machine (Polynomial kernel)	m = 10, k = 5, C = 2, exponent = 1, network embedding feature dimension = 300	0.5059	0.1507	0.0781	0.0703
Support vector machine (RBF kernel)	m = 10, k = 5, C = 2, γ = 0.01, network embedding feature dimension=300	0.4485	0.1112	0.0848	0.0456
Random forest	m = 10, k = 5, number of decision trees = 250, network embedding feature dimension = 150	0.5069	0.1608	0.0762	0.0753

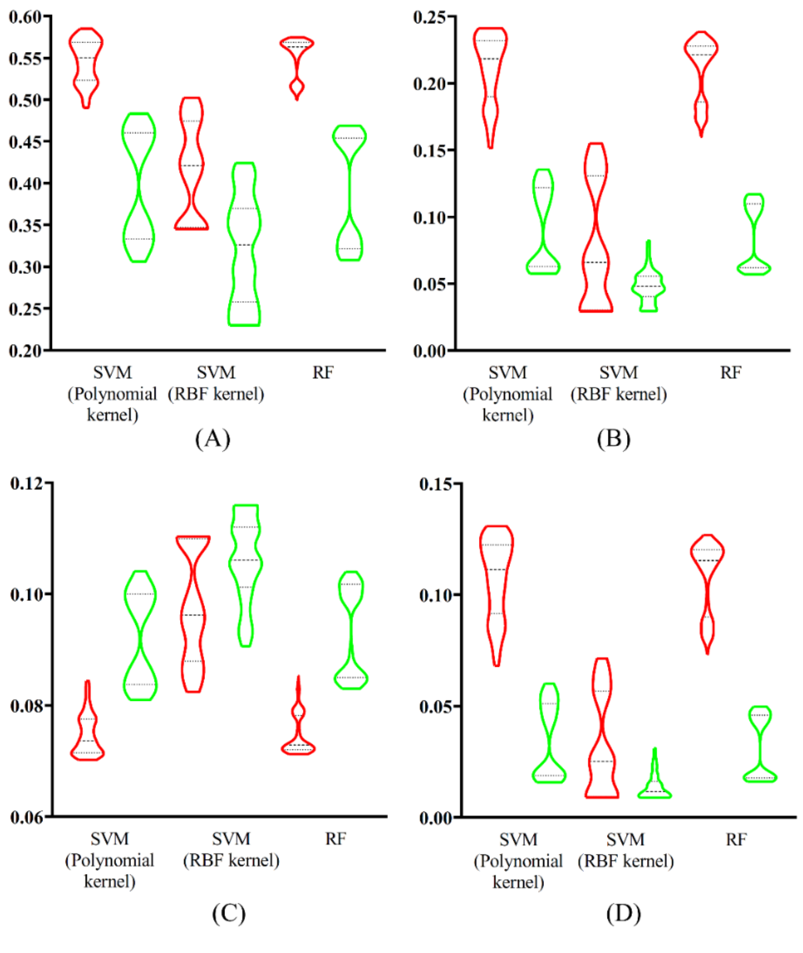

Figure 6. Violin plot to compare RAKEL classifiers using network embedding features with or without label partition. Red violins are for RAKEL classifiers with label partition and green violins are for RAKEL classifiers without label partition. (A) Accuracy; (B) Exact match; (C) Hamming loss; (D) Integrated score.

3.5. Comparison of classifiers with random label partition

The classifiers proposed in this study adopted the label partition yielded by Louvain method. To confirm such obtained partition was really helpful to improve the performance of classifiers, we employed the random label partition, which randomly divided class labels into some partitions. To give a far comparison, the distribution of partition sizes in random partition was same as that in the partition yielded by Louvain method. On each random partition, the best RAKEL classifier with each base classifier and each feature type was built and assessed by ten-fold cross-validation. Such procedures executed ten times for different random partitions. The performance (integrated score) of each RAKEL classifier on two feature types is shown in Figures 7 and 8, respectively. For easy comparisons, the performance of RAKEL classifiers with partition yielded by Louvain method under ten-fold cross-validation for ten times was also listed in these two figures. It can be observed that when the base classifier was SVM (polynomial kernel) or RF, the RAKEL classifiers with partition yielded by Louvain method always generated better performance. As for the base classifier, SVM (RBF kernel), its superiority was not very obvious. It provided relatively better performance using domain embedding features. However, for network embedding features, classifiers with partition yielded by Louvain method were not always better than those with random partition. As a whole, classifiers with partition yielded by Louvain method were superior to those with random partition. The reasonable partition of class labels can further improve the performance of classifiers.

For the classifiers with both domain and network embedding features, we also compared them with those using random partition. The performance of classifier with each base classifier and random partition is listed in Table 5. Compared with results listed in Table 3, classifiers with partition yielded by Louvain method always produced higher accuracy, exact match and integrated score. As for hamming loss, classifiers with random partition yielded lower values when SVM was the base classifier. However, this cannot change the fact that classifiers with partition yielded by Louvain method were superior to the classifiers with random partition.

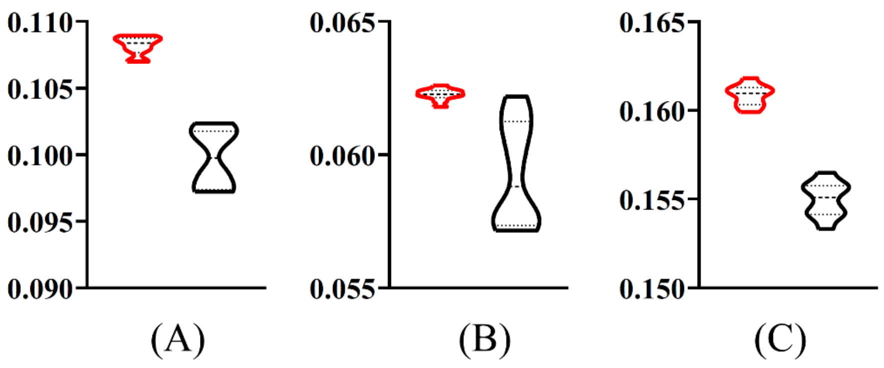

Figure 7. Violin plot to compare RAKEL classifiers using domain embedding features with partition yielded by Louvain method and random partition. Red violins indicate integrated scores yielded by classifiers with partition yielded by Louvain method, black violins represent integrated scores yielded by classifiers with random partition. (A) SVM (polynomial kernel) is the base classifier; (B) SVM (RBF kernel) is the base classifier; (C) RF is the base classifier.

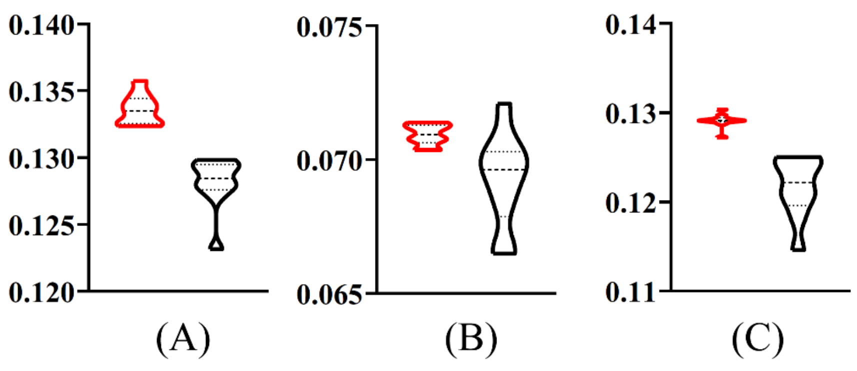

Figure 8. Violin plot to compare RAKEL classifiers using network embedding features with partition yielded by Louvain method and random partition. Red violins indicate integrated scores yielded by classifiers with partition yielded by Louvain method, black violins represent integrated scores yielded by classifiers with random partition. (A) SVM (polynomial kernel) is the base classifier; (B) SVM (RBF kernel) is the base classifier; (C) RF is the base classifier.

Table 5. Performance of RAKEL classifiers using domain and network embedding features but with random label partition.

Base classifier	**Parameter**	**Accuracy**	**Exact match**	**Hamming loss**	**Integrated score**
Support vector machine (Polynomial kernel)	*m* = 10, *k* = 5, *C* = 2, exponent = 1, network embedding feature dimension = 300	0.6177	0.2705	0.0654	0.1562
Support vector machine (RBF kernel)	*m* = 10, *k* = 5, *C* = 2, *γ* = 0.01, network embedding feature dimension=300	0.5427	0.2138	0.0737	0.1075
Random forest	*m* = 10, *k* = 5, number of decision trees = 250, network embedding feature dimension = 150	0.6195	0.2952	0.0635	0.1713

3.6. Comparison of the previous classifier

In references [9,10], two hybrid classifiers were proposed to identify functions of mouse proteins. They contained one network-based classifier, which was constructed based on PPI information reported in STRING. For a query protein, this classifier assigned a score to each of 24 functional types. Then, 24 types were sorted by the decreasing order of corresponding scores. Evidently, this classifier cannot determine which types were the predicted types. To compare with our classifiers, we employed a threshold for such score so that this classifier can determine the predicted types. Various thresholds were tried for this classifier, which was assessed by ten-fold cross-validation for ten times. The highest integrated score was only 0.0160, which was much lower than those listed in Tables 1−3. The accuracy was 0.2532, exact match was 0.0706 and hamming loss was 0.1059. Clearly, such performance was much lower than that of any above-mentioned classifier. This result indicated that the classifiers proposed in this study were superior to this previous classifier.

Table 6. Three communities obtained by using Louvain method.

Index	**Functional type**
Partition 1	PROTEIN WITH BINDING FUNCTION OR COFACTOR REQUIREMENT (structural or catalytic) REGULATION OF METABOLISM AND PROTEIN FUNCTION CELLULAR COMMUNICATION/SIGNAL TRANSDUCTION MECHANISM SUBCELLULAR LOCALIZATION CELLULAR TRANSPORT, TRANSPORT FACILITIES AND TRANSPORT ROUTES TRANSCRIPTION ENERGY METABOLISM CELL CYCLE AND DNA PROCESSING PROTEIN FATE (folding, modification, destination) BIOGENESIS OF CELLULAR COMPONENTS SYSTEMIC INTERACTION WITH THE ENVIRONMENT PROTEIN SYNTHESIS CELL RESCUE, DEFENSE AND VIRULENCE
Partition 2	INTERACTION WITH THE ENVIRONMENT CELL TYPE LOCALIZATION TISSUE LOCALIZATION ORGAN LOCALIZATION TRANSPOSABLE ELEMENTS, VIRAL AND PLASMID PROTEINS
Partition 3	CELL FATE DEVELOPMENT (Systemic) TISSUE DIFFERENTIATION ORGAN DIFFERENTIATION CELL TYPE DIFFERENTIATION

3.7. Functional type analysis

As mentioned above, the usage of label partition improved the performance of multi-label classifiers. The final classifier should use the label partition on the whole dataset. This section gave analyses on 24 functional types (labels).

First, we constructed a protein subset for each label, which consisted of all proteins having this label. For any two labels, their associations were evaluated by the Tanimoto coefficient of their corresponding protein subsets. A heat map was plotted to show Tanimoto coefficients for any two functional types, as illustrated in Figure 9. It can be observed that class 14 (TRANSPOSABLE ELEMENTS, VIRAL AND PLASMID PROTEINS) has weak associations with almost all other classes. On the contrary, class 7 (PROTEIN WITH BINDING FUNCTION OR COFACTOR REQUIREMENT (structural or catalytic)) and class 21 (SUBCELLULAR LOCALIZATION) were highly related to other classes. By using the Louvain method, 24 functional types were divided into three partitions, which are listed in Table 6. There were 14 functional types in Partition 1, whereas other two partitions all contained five functional types. Not surprisingly, class 7 and class 21 were classified into the same partition. Given a protein representation, a multi-label classifier can be built on each partition. Classifiers on all three partitions were integrated in the final multi-label classifier.

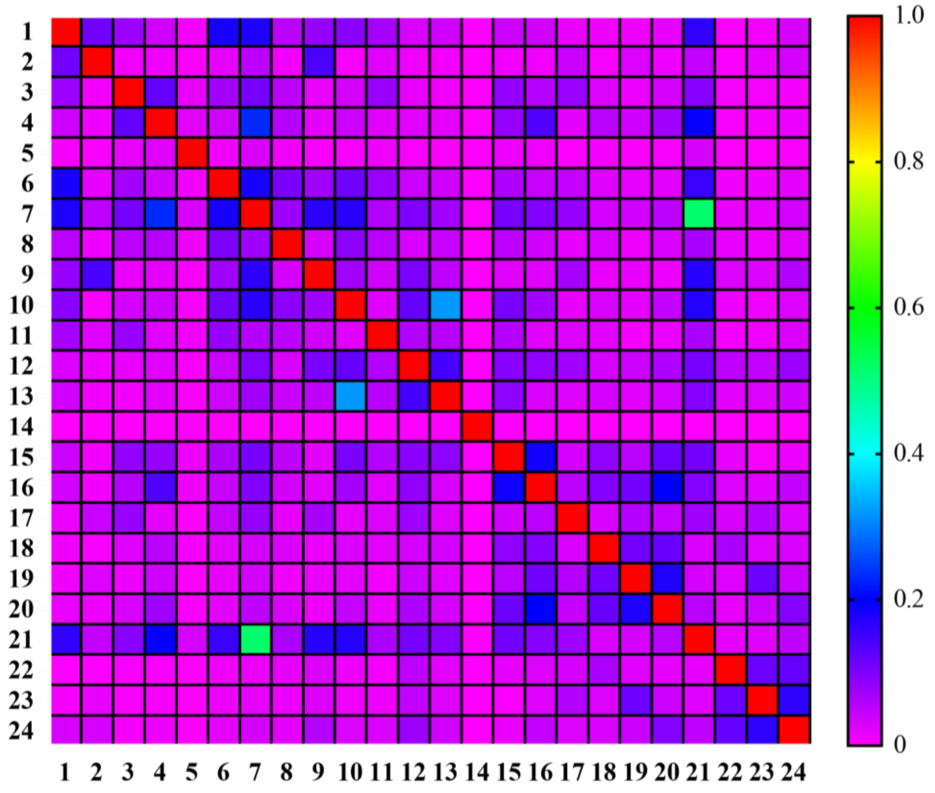

Figure 9. Heat map to show the associations of functional types. The corresponding functional types of class index 1−24 can be found in Figure 1.

3.8. *Further study*

By employing the association information of functional types, the performance of the multi-label classifiers for identification of mouse protein functions was improved. However, there still exist rooms for improvement. First, protein features are key factors that can influence the performance of classifiers. Some novel and efficient protein features, such as motif embedding features [58], can be adopted to further improve the classifiers. Second, only one community detection algorithm, Louvain method, was employed to cluster functional types in this study. It was not clear whether this algorithm was optimum to deal with this problem. Some novel community detection algorithms may deeply investigate the associations between functional types, thereby producing a more optimum label partition. Finally, we adopted traditional machine learning algorithms (RAKEL, SVM, RF) to construct classifiers. They can be replaced with more powerful algorithms, such as deep learning algorithms, so that more efficient classifiers can be built. In future, we will continue our study in these aspects.

4. Conclusions

This study proposed a novel multi-label classifier for identification of functions of mouse proteins. Such classifier considered the associations of functional types (labels) and divided labels into some partitions. By employing the label partition, the performance of classifiers was improved. This classifier can be easily extended to other organisms. It is hopeful that this classifier can be helpful to identify novel functions of mouse proteins. All codes and data are available at https://github.com/LiXuuuu/Mouse-Protein.

Conflict of interest

The authors declare no conflict of interest.

References

1. R. Milo, What is the total number of protein molecules per cell volume? A call to rethink some published values, *Bioessays*, **35** (2013), 1050−1055. https://doi.org/10.1002/bies.201300066
2. Z. C. Üretmen Kagıalı, A. Şentürk, N. E. Özkan Küçük, M. H. Qureshi, N. Özlü, Proteomics in cell division, *Proteomics*, **17** (2017), 1600100. https://doi.org/10.1002/pmic.201600100
3. M. J. Mughal, R. Mahadevappa, H. F. Kwok, DNA replication licensing proteins: Saints and sinners in cancer, *Semin. Cancer Biol.*, **58** (2019), 11−21. https://doi.org/10.1016/j.semcancer.2018.11.009
4. D. Davidi, R. Milo, Lessons on enzyme kinetics from quantitative proteomics, *Curr. Opin. Biotechnol.*, **46** (2017), 81−89. https://doi.org/10.1016/j.copbio.2017.02.007
5. S. F. Altschul, W. Gish, W. Miller, E. W. Myers, D. J. Lipman, Basic local alignment search tool, *J. Mol. Biol.*, **215** (1990), 403−410. https://doi.org/10.1016/S0022-2836(05)80360-2
6. C. J. Sigrist, L. Cerutti, E. De Castro, P. S. Langendijk-Genevaux, V. Bulliard, A. Bairoch, et al., PROSITE, a protein domain database for functional characterization and annotation, *Nucleic Acids Res.*, **38** (2010), D161−D166. https://doi.org/10.1093/nar/gkp885
7. R. D. Finn, J. Mistry, B. Schuster-Böckler, S. Griffiths-Jones, V. Hollich, T. Lassmann, et al., Pfam: clans, web tools and services, *Nucleic Acids Res.*, **34** (2006), D247−D251. https://doi.org/10.1093/nar/gkj149
8. Y. Ye, A. Godzik, FATCAT: a web server for flexible structure comparison and structure similarity searching, *Nucleic Acids Res.*, **32** (2004), W582−W585. https://doi.org/10.1093/nar/gkh430
9. L. Hu, T. Huang, X. Shi, W. C. Lu, Y. D. Cai, K. C. Chou, Predicting functions of proteins in mouse based on weighted protein-protein interaction network and protein hybrid properties, *PLoS One*, **6** (2011), e14556. https://doi.org/10.1371/journal.pone.0014556
10. G. Huang, C. Chu, T. Huang, X. Kong, Y. Zhang, N. Zhang, et al., Exploring mouse protein function via multiple approaches, *PLoS One*, **11** (2016), e0166580. https://doi.org/10.1371/journal.pone.0166580
11. X. Wang, Y. Wang, Z. Xu, Y. Xiong, D. Q. Wei, ATC-NLSP: Prediction of the classes of anatomical therapeutic chemicals using a network-based label space partition method, *Front. Pharmacol.*, **10** (2019), 971. https://doi.org/10.3389/fphar.2019.00971

12. X. Wang, X. Zhu, M. Ye, Y. Wang, C. D. Li, Y. Xiong, et al., STS-NLSP: A network-based label space partition method for predicting the specificity of membrane transporter substrates using a hybrid feature of structural and semantic similarity, *Front. Bioeng. Biotech.*, **7** (2019), 306. https://doi.org/10.3389/fbioe.2019.00306
13. A. Ruepp, O. N. Doudieu, J. van den Oever, B. Brauner, I. Dunger-Kaltenbach, G. Fobo, et al., The mouse functional genome database (MfunGD): functional annotation of proteins in the light of their cellular context, *Nucleic Acids Res.*, **34** (2006), D568−D571. https://doi.org/10.1093/nar/gkj074
14. V. D. Blondel, J. L. Guillaume, R. Lambiotte, E. Lefebvre1, Fast unfolding of communities in large networks, *J. Stat. Mech-Theory E.*, **2008** (2008), P10008. https://doi.org/10.1088/1742-5468/2008/10/P10008
15. G. Tsoumakas, I. Vlahavas, Random k-Labelsets: An ensemble method for multilabel classification, in *European conference on machine learningmachine learning*, (2007), 406−417. https://doi.org/10.1007/978-3-540-74958-5_38
16. C. Cortes, V. Vapnik, Support-vector networks, *Mach. Learn.*, **20** (1995), 273−297. https://doi.org/10.1007/BF00994018
17. L, Breiman, Random forests, *Mach. Learn.*, **45** (2001), 5−32. https://doi.org/10.1023/A:1010933404324
18. M. Ashburner, S. Lewis, On ontologies for biologists: the Gene Ontology-untangling the web, in Novartis Foundation Symposia (eds. N. Foundation), *Wiley Online Library*, **247** (2002), 66−80. https://doi.org/10.1002/0470857897.ch6
19. E. Camon, M. Magrane, D. Barrell, D. Binns, W. Fleischmann, P. Kersey, et al., The gene ontology annotation (GOA) project: implementation of GO in SWISS-PROT, TrEMBL, and InterPro, *Genome Res.*, **13** (2003), 662−672. https://doi.org/ 10.1101/gr.461403
20. 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
21. K. C. Chou, Y. D. Cai, Predicting protein structural class by functional domain composition, Biochem, *Bioph. Res. Co.*, **321** (2004), 1007−1009. https://doi.org/10.1016/j.bbrc.2004.07.059
22. L. Lu, Z. Qian, Y. D. Cai, Y. Li, ECS: an automatic enzyme classifier based on functional domain composition, *Comput. Biol. Chem.*, **31** (2007), 226−232. https://doi.org/10.1016/j.compbiolchem.2007.03.008
23. H. Zhou, Y. Yang, H. B. Shen, Hum-mPLoc 3.0: prediction enhancement of human protein subcellular localization through modeling the hidden correlations of gene ontology and functional domain features, *Bioinformatics*, **33** (2017), 843−853. https://doi.org/10.1093/bioinformatics/btw723
24. L. Chen, K. Y. Feng, Y. D. Cai, K. C. Chou, H. P. Li, Predicting the network of substrate-enzyme-product triads by combining compound similarity and functional domain composition, BMC *Bioinformatics*, **11** (2010), 293. https://doi.org/10.1186/1471-2105-11-293
25. M. Blum, H. Y. Chang, S. Chuguransky, T. Grego, S. Kandasaamy, A. Mitchell, et al., The InterPro protein families and domains database: 20 years on, *Nucleic Acids Res.*, **49** (2021), D344−D354. https://doi.org/10.1093/nar/gkaa977
26. T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, Preprint, arXiv: 1301.3781v3.

27. K. W. Church, Word2Vec, *Nat. Lang. Eng.*, **23** (2017), 155−162. https://doi.org/10.1017/S1351324916000334
28. B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in *20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (2014), 701−710. https://doi.org/10.1145/2623330.2623732
29. A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (2016), 855−864. https://doi.org/10.1145/2939672.2939754
30. H. Cho, B. Berger, J. Peng, Compact integration of multi-network topology for functional analysis of genes, *Cell Syst.*, **3** (2016), 540−548. https://doi.org/10.1016/j.cels.2016.10.017
31. H. Liu, B. Hu, L. Chen, L. Lu, Identifying protein subcellular location with embedding features learned from networks, *Curr. Proteomics*, **18** (2021): 646−660. https://doi.org/10.2174/1570164617999201124142950
32. X. Zhang, L. Chen, Z. H. Guo, H. Liang, Identification of human membrane protein types by incorporating network embedding methods, *IEEE Access*, **7** (2019), 140794−140805. https://doi.org/10.1109/ACCESS.2019.2944177
33. X. Pan, L. Chen, M. Liu, Z. Niu, T. Huang, Y. D. Cai, Identifying protein subcellular locations with embeddings-based node2loc, *IEEE ACM Trans. Comput. Bi.*, **2021** (2021). https://doi.org/10.1109/TCBB.2021.3080386
34. 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
35. D. Szklarczyk, A. Franceschini, S. Wyder, K. Forslund, D. Heller, J. Huerta-Cepas, et al., STRING v10: protein–protein interaction networks, integrated over the tree of life, *Nucleic Acids Res.*, **43** (2015), D447−D452. https://doi.org/10.1093/nar/gku1003
36. H. Tong, C. Faloutsos, J. Pan, Fast random walk with restart and its applications, in *Sixth International Conference on Data Mining*, (2006), 613−622. https://doi.org/10.1109/ICDM.2006.70
37. S. Kohler, S. Bauer, D. Horn, P. N. Robinson, Walking the interactome for prioritization of candidate disease genes, *Am. J. Hum. Genet.*, **82** (2008), 949−958. https://doi.org/10.1016/j.ajhg.2008.02.013
38. G. Tsoumakas, I. Katakis, Multi-label classification: An overview. *Int. J. Data Warehous.*, **3** (2007), 1−13.
39. J. Read, P. Reutemann, B. Pfahringer, G. Holmes, MEKA: A multi-label/multi-target extension to WEKA, *J. Mach. Learn. Res.*, **17** (2016), 1−5.
40. 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
41. L. Chen, S. Wang, Y. H. Zhang, L. Li, Z. H. Xing, J. Yang, et al., Identify key sequence features to improve CRISPR sgRNA efficacy, *IEEE Access*, **5** (2017), 26582−26590. https://doi.org/10.1109/ACCESS.2017.2775703
42. 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

43. Y. H. Zhang, H. Li, T. Zeng, L. Chen, Z. Li, T. Huang, et al., Identifying transcriptomic signatures and rules for SARS-CoV-2 infection, *Front. Cell Dev. Biol.*, **8** (2021), 627302. https://doi.org/10.3389/fcell.2020.627302
44. Y. H. Zhang, Z. Li, T. Zeng, L. Chen, H. Li, T. Huang, et al., Detecting the multiomics signatures of factor-specific inflammatory effects on airway smooth muscles, *Front. Genet.*, **11** (2021), 599970. https://doi.org/10.3389/fgene.2020.599970
45. 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. Method M.*, **2021** (2021), 6683051. https://doi.org/10.1155/2021/6683051
46. Y. Wang, Y. Xu, Z. Yang, X. Liu, Q. Dai, Using recursive feature selection with random forest to improve protein structural class prediction for low-similarity sequences, *Comput. Math. Method M.*, **2021** (2021), 5529389. https://doi.org/10.1155/2021/5529389
47. J. Platt, *Fast training of support vector machines using sequential minimal optimization*, MIT Press, 1998.
48. Y. Yang, L. Chen, Identification of drug-disease associations by using multiple drug and disease networks, *Curr. Bioinform.*, **17** (2022), 48−59. https://doi.org/10.2174/1574893616666210825115406
49. Y. Jia, R. Zhao, L. Chen, Similarity-based machine learning model for predicting the metabolic pathways of compounds, *IEEE Access*, **8** (2020), 130687−130696. https://doi.org/10.1109/ACCESS.2020.3009439
50. X. Zhao, L. Chen, J. Lu, A similarity-based method for prediction of drug side effects with heterogeneous information, *Math. Biosci.*, **306** (2018), 136−144. https://doi.org/10.1016/j.mbs.2018.09.010
51. K. K. Kandaswamy, K. C. Chou, T. Martinetz, S. Möllera, P. N. Suganthand, S. Sridharan, et al., AFP-Pred: A random forest approach for predicting antifreeze proteins from sequence-derived properties, *J. Theor. Biol.*, **270** (2011), 56−62. https://doi.org/10.1016/j.jtbi.2010.10.037
52. Y. B. Marques, A. de Paiva Oliveira, A. T. Ribeiro Vasconcelos, F. R. Cerqueira, Mirnacle: machine learning with SMOTE and random forest for improving selectivity in pre-miRNA ab initio prediction, *BMC Bioinformatics*, **17** (2016), 474. http://dx.doi.org/10.1186/s12859-017-1508-0
53. G. Pugalenthi, K. Kandaswamy, K. C. Chou, S. Vivekanandan, P. Kolatkar, RSARF: Prediction of residue solvent accessibility from protein sequence using random forest method, *Protein Peptide Lett.*, **19** (2011), 50−56. https://doi.org/10.2174/092986612798472875
54. M. Onesime, Z. Yang, Q. Dai, Genomic island prediction via chi-square test and random forest algorithm, *Comput. Math. Method M.*, **2021** (2021), 9969751. https://doi.org/10.1155/2021/9969751
55. M. Fernandez-Delgado, E. Cernadas, S. Barro, D. Amorim, Do we need hundreds of classifiers to solve real world classification problems?, *J. Mach. Learn. Res.*, **15** (2014), 3133−3181.
56. R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, in *International Joint Conference on Artificial Intelligence*, (1995), 1137−1145.
57. W. Chen, L. Chen, Q. Dai, iMPT-FDNPL: identification of membrane protein types with functional domains and a natural language processing approach, *Comput. Math. Method M.*, **2021** (2021), 7681497. https://doi.org/10.1155/2021/7681497

58. J. Zhang, Q. Chen, B. Liu, iDRBP_MMC: Identifying DNA-binding proteins and RNA-binding proteins based on multi-label learning model and motif-based convolutional neural network, *J. Mol. Biol.*, **432** (2020), 5860–5875. https://doi.org/10.1016/j.jmb.2020.09.008