REVIEW

Performance measures in evaluating machine learning based bioinformatics predictors for classifications

Yasen Jiao and Pufeng Du*

School of Computer Science and Technology, Tianjin University, Tianjin 300350, China

Received June 8, 2016; Revised September 6, 2016; Accepted October 21, 2016

***Background*: Many existing bioinformatics predictors are based on machine learning technology. When applying these predictors in practical studies, their predictive performances should be well understood. Different performance measures are applied in various studies as well as different evaluation methods. Even for the same performance measure, different terms, nomenclatures or notations may appear in different context.**
***Results*: We carried out a review on the most commonly used performance measures and the evaluation methods for bioinformatics predictors.**
***Conclusions*: It is important in bioinformatics to correctly understand and interpret the performance, as it is the key to rigorously compare performances of different predictors and to choose the right predictor.**

Keywords: machine learning; performance measures; evaluation methods

INTRODUCTION

The high-throughput sequencing technology, which was developed in the most recent years, has enabled us to acquire high quality genomic sequences at different levels for every living cell on this planet [1]. The sequencing technology has been widely applied in clinical trials to find target genes of a drug or develop therapies for complex diseases precisely [2]. All these developments rely on the fact that the cost of sequencing is dropping exponentially (https://www.genome.gov/sequencing-costs/). On the contrary, the high-throughput technology to determine the function of a gene is still lacking. Even with comprehensive genomic sequences in hand, it is still costly and time consuming to find out the molecular functions of a particular gene in a cellular process. There is a huge information gap between biological sequences and molecular functions [3]. Moreover, this gap is becoming wider and wider every day.

To bridge this gap, one feasible solution is to use computational methods to predict the molecular functions from the primary sequences. The computational predictions of molecular functions may contain errors. However, it still provides informative hints to the experimental studies [4]. Since the cost of computational predictions is minimal, the computational approaches become more and more popular in life sciences over the last two decades.

Various computational predictors have been established to estimate functional information from the primary sequences of proteins and genes. Most of these predictors use machine learning algorithms, such as support vector machines [5], artificial neuron networks [6], K-nearest neighbor algorithms [7], Bayes analysis [8] and many more [9]. Because machine learning algorithms usually use existing data to establish predictive models, it is possible that a predictive model is over-fitted or over-optimized on the existing data. Here, the term "over-fitted" or "over-optimized" means that the predictive model works well when it is tested with existing data. However, the prediction performance drops drastically when it is applied in practical studies with novel data. Therefore, when applying these computational predictors, it is vitally important to understand their mechanisms and the conditions of their performances in the first place. Without such knowledge, applying predictive computational methods in studies is no better than mixing reagents in tubes without knowing what they are.

When a paper describes a predictive model, it always reports its predictive performances as well as how these performances are estimated and measured. We would like to emphasize that the way how these performances are estimated and measured is much more important than the values. For example, when a predictor reports that its prediction accuracy is 86% and the other one 84%, it is not sure that the former one performs better than the latter one. The actual qualities of their predictions are tightly related with how these numbers are obtained. This is always confusing to the users of predictors, who do not have a strong statistics or machine learning background.

In addition, the authors of bioinformatics papers usually have different expertise and professional backgrounds. They may use different terminologies on the same performance measures, or vice versa. For example, the "accuracy", "sensitivity" and "precision" appear commonly in describing predictive performances of a bioinformatics predictor. However, they may have the same interpretations in some studies, while different in others. The readers should pay more attention to the definitions of these measures than their values.

In this paper, we will discuss in details on the performance measures as well as how these measures should be estimated and reported. We will first introduce the general concept of machine learning, which followed by the discussions of basic methods to test a machine learning system. We will then introduce several statistics, which are commonly used as performance measures in many bioinformatics studies. We will also discuss how to interpret the values of these measures and how to compare the performances of different predictors.

BASIC CONCEPTS OF MACHINE LEARNING IN BIOINFORMATICS

Machine learning, which is a subfield of computer science, aims at developing algorithms that create models of naturally exist systems from examples of their inputs and outputs [10]. Machine learning algorithms are usually based on statistics. It is also called learning from data or learning from examples. This is different to other kind of systems that follow strictly static program instructions [11].

Figure 1 gives a diagram of a machine learning system. Let S be a naturally exist system. It can be described by a deterministic generalized function $f(\boldsymbol{x})$. Due to our limited knowledge of its mechanism, it is difficult to obtain the details of $f(\boldsymbol{x})$. However, when we give an $\boldsymbol{x}$ as the input of S, S will give an output y that completely relies on $\boldsymbol{x}$. When we have many different pairs of $\boldsymbol{x}$ and y, a learning machine, as the LM in Figure 1, can be established using machine learning algorithms. LM can be described as a generalized function $f_e(\boldsymbol{x})$, which is an estimation to $f(\boldsymbol{x})$.

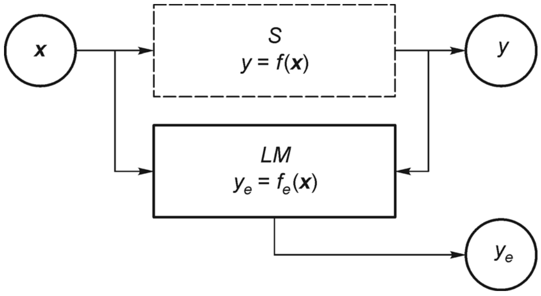


Figure 1. A diagram of a machine learning system. S represents a biological system. Its mechanism cannot be directly obtained. The details of $f(\boldsymbol{x})$ are unknown. We use a serial of $\boldsymbol{x}$ as the input of S. A serial of y can be obtained from the output of S. By using pairs of $\boldsymbol{x}$ and y, the machine learning system LM can be trained. A function $f_e(\boldsymbol{x})$ can be established. For the $\boldsymbol{x}$ that has never been seen by both S and LM, we expect that LM can produce a result y_e that is close to y as much as possible.

We expect that LM behaves like S as much as possible. When we give a serial of $\boldsymbol{x}$, which have never been seen by the LM, as the input of both LM and S, the serial of y_e, which are the output of LM, should be almost the same as the serial of y, which are the output of S. It seems like LM can predict the output of S, which is just why LM can be called a "predictor".

Due to our limited knowledge of $f(\boldsymbol{x})$, the following facts exist: (i) there is no way to guarantee that LM always produces exactly the same results as S. (ii) it is difficult to use analytical method to compare $f_e(\boldsymbol{x})$ and $f(\boldsymbol{x})$. (iii) $f_e(\boldsymbol{x})$ and $f(\boldsymbol{x})$ can be way different even if LM works well on many examples.

In bioinformatics, S is usually a biological system. For example, S can be the cellular mechanism that recognizes the cleavage sites of signaling peptide on protein sequences. Given $\boldsymbol{x}$, which represents a protein sequence, the output of S is y, which represents the position of the cleavage site on the sequence. LM is a predictor, like the LabCas [12] and Cascleave [13]. Although the exact mechanism of how S determines the position of cleavage sites is still unknown, LM can predict a large part of cleavage sites correctly. However, without thorough experimental validations, we cannot say that S works with a same or similar mechanism as LM.

A number of existing predictors in bioinformatics are classifiers, which is a type of learning machines. The input of a classifier is usually a vector $\boldsymbol{x}$, which is assumed to contain all necessary information of a sample. The vector $\boldsymbol{x}$ is usually defined in R^d ($\boldsymbol{x} \in R^d$). In machine learning, the vector $\boldsymbol{x}$ is termed as the features of a sample.

R^d, which contains all possible $\boldsymbol{x}$, is called the feature space. The output of a classifier is usually an integer y ($y \in Z$), which indicates the class that $\boldsymbol{x}$ belongs to. The integer y is called the class label of the vector $\boldsymbol{x}$. In bioinformatics, the vector $\boldsymbol{x}$ may represent different biological information, such as DNA sequences, RNA sequences, protein sequences, expression profiles and genetic variations. The output integer y can represent different attributes of samples, such as subcellular locations of proteins [14], structural classes of proteins [15,16], nucleosome positions in the genome [17], modification states of proteins [18,19], epistatic interactions [20,21] and many others [22–30].

The "multi-class classifier" [31] and the "multi-label classifier" [32] are two different terms in machine learning. Both of them have been introduced to bioinformatics [33]. Their meanings should not be confused. The term "multi-class classifier" describes that a classifier can assign one and only one label from more than two possible labels to every samples. This is to be distinguished from the "binary classifier", which can assign one of only two possible labels to every sample. The "multi-" in the "multi-class classifier" means that the number of all possible labels is more than two. However, the "multi-" in the "multi-label classifier" has nothing to do with the number of all possible labels. It means that the classifier can assign one or more than one label to every sample. This is to be distinguished from the classical "single-label classifiers", which assign only one label to every sample. For example, the Hum-PLoc [34], which can predict protein subcellular locations, is a "multi-class classifier". It can assign only a single subcellular location to every protein. On the contrary, the iLoc-Hum [35], which can also predict protein subcellular locations, is a "multi-label classifier". It can assign one or more than one subcellular locations to every protein.

Training and testing are two required procedures to establish a practically applicable machine learning based predictor. The training procedure establishes the learning machine with a dataset. This dataset is called the training dataset. The training procedure is usually an optimization procedure that aims at minimizing the error rate on the training dataset. The testing procedure is to validate whether the learning machine can actually work in predicting class labels from the features. It aims at simulating the practical application scenario. The dataset used in the testing procedure is called the testing dataset. The predictive performance values, which are reported in papers, are usually obtained from the testing procedures. The choice of training and testing dataset and the design of training and testing procedures are important in evaluating the predictors. In the next part of this review, we will discuss the commonly used protocols in evaluating machine learning based predictors in bioinformatics.

EVALUATION METHODS

While evaluating a bioinformatics predictor, it is important to understand that the predictor has a real predictive performance, which exists objectively and is independent to the evaluation methods. This real predictive performance can only be obtained by testing the predictor with infinite number of samples that can uniformly fill the entire feature space. Since this is always impossible and meaningless in practice, the purpose of all evaluation methods is to estimate this real predictive performance using finite number of samples with known features and labels. In bioinformatics, the samples, especially the labels of samples, are usually expensive. The number of samples to train a predictor is usually less than sufficient. For example, when we created the first protein submitochondrial location predictor, we have only 317 proteins to train a three-class predictor [36]. This is different to other common applications of machine learning, such as voice recognition, image recognition or natural language processing, where millions of samples can be acquired easily. Therefore, some concepts of testing and evaluating predictors in machine learning cannot be directly introduced to bioinformatics.

In machine learning, there are three different approaches to evaluate a predictor. As displayed in Figure 2, they are known as the independent dataset test, re-substituting test and cross validation. The cross validation method can be further divided into two different methods, the leave-one-out cross validation and the n-fold cross validation.

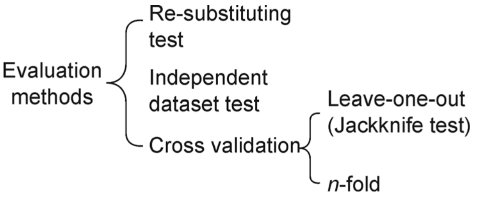


Figure 2. Taxonomy of evaluation methods. There are three different evaluation methods: Independent dataset test, re-substituting test and cross validation, which can be further divided into two different methods, the leave-one-out cross validation and the n-fold cross validation.

In the view of machine learning, the independent dataset test should be the most recommended. In an independent dataset test, the testing dataset must have sufficient size, which is usually much larger than the training dataset. There should be no overlap between the testing dataset and the training dataset. It is ideally that the testing dataset uniformly distributed in the entire feature space. However, in bioinformatics, since the available samples are usually limited, it is hardly possible to fulfill all these requirements.

As an accommodation in bioinformatics, the independent dataset tests are carried out in a different manner. The whole dataset is randomly partitioned into two parts: a larger part and a smaller part. The larger part usually contains over 70% of all samples. It is used as the training dataset. The remaining smaller part serves as the testing dataset. This method can be called a subsampling test. Although this method fulfills the requirement of independence, the size of testing dataset is usually small, which results in a large variance of the estimated performance. That is to say, if this test was carried out for several times, the predictive performance in every test would be different and distribute in a wide range. Therefore, in practical applications, the users are likely to find that the predictive performance is much higher or lower than the value that is estimated from a single run of this kind of independent dataset test. As an augment, in evaluating predictor SubMem [37], we reported the predictive performances as the average value of several times of subsampling test with different partitions of the dataset [37].

The re-substituting test should be the least recommended. This is also true in bioinformatics. In a re-substituting test, the testing dataset is identical to the training dataset. As the training process aims at minimizing the error rate on the training dataset, using the same dataset to test the predictor will always yield an over-estimated performance. The predictive performance in practical application can be much poorer than the re-substituting test. An amazing performance value, like 99% accuracy in a re-substituting test, usually indicates that the predictor is over-optimized and useless in practice.

In machine learning, cross-validation methods are considered as a compromise solution when the number of available samples is very limited. Due to the number of available samples, in bioinformatics, cross validation methods become the most recommended evaluation methods. In a cross-validation test, all the data are used as both training and testing dataset. However, it is not used in the same way as the re-substituting test. In an *n*-fold cross-validation test, the entire dataset was randomly partitioned into *n* parts of equal size. The training and testing process would be carried out for *n* rounds. In the *k*-th round, the *k*-th part was used as the testing dataset, while the remaining $n-1$ parts form the training dataset. After all *n* rounds of training and testing, every sample in the dataset was used as testing sample once and only once. The prediction performance can be estimated by averaging the prediction results over the whole dataset. Figure 3 illustrates the process of a 5-fold cross validation. When the number *n* is set to its maximum possible value, which is the number of all samples, an *n*-fold cross validation becomes a leave-one-out cross validation. A leave-one-out cross validation is also called a jackknife test.

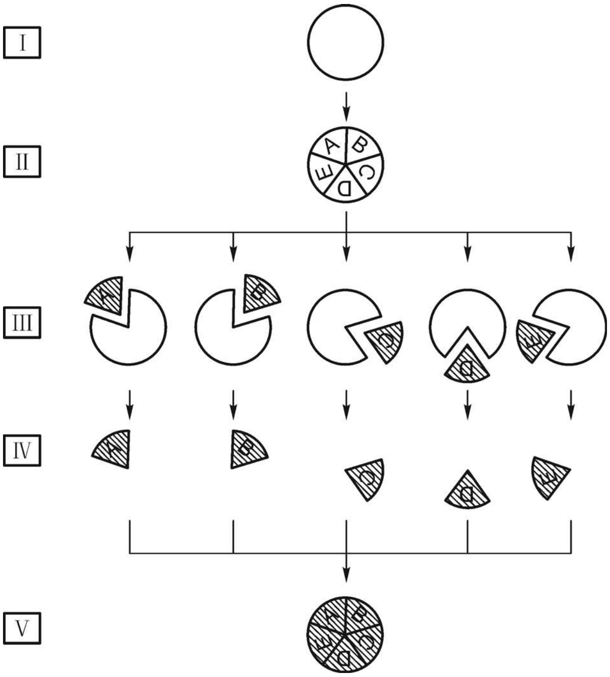


Figure 3. An illustration of the process of a 5-fold cross validation. The whole process contains five steps. Step I: the whole dataset was obtained. Step II: the whole dataset is randomly partitioned into five different parts (A, B, C, D and E). Step III: five rounds of training and testing are carried out. Every part is used as the testing dataset, while the remaining four parts are the training dataset. The shadowed part indicates that this part is used as testing dataset. Step IV: the testing results are collected from five rounds of training and testing. Step V: The testing results are pooled together to estimate the predictive performances.

The jackknife test has been widely applied in evaluating many bioinformatics predictors [4,17,38,39], as it can easily generate fixed values of performance measures, which makes it easy to compare different predictors [40,41]. However, there is a minor flaw in the jackknife test. In an *n*-fold cross-validation, the predictions and the real results have negative correlations even there is no relationship between them (http://www.russpoldrack.org/2012/12/the-perils-of-leave-one-out.html). This effect becomes more significant when *n* increases. As the jackknife test uses the maximal possible value of *n*, it is affected more than the 3-fold or 5-fold cross-validations. A more quantitative discussion has been provided by Hastie *et al.* [42]. Therefore, we suggest that, the *n*-fold cross-validation, where *n* is a small number, like 3 or 5, should be preferred in practice.

No matter which evaluation method is applied, the predictive performance has to be represented quantitatively with numbers. A set of statistics, which was called the performance measures, was introduced to represent the predictive performance in many different aspects.

PERFORMANCE MEASURES

Intuitively, users of a predictor would like to know how likely they may trust the prediction results. This can be roughly interpreted as the probability that the predictor makes the correct predictions. However, in practice, it is not enough to give only a single number that indicates the probability of getting correct predictions. For example, the users may only care about the correct predictions among the samples that they are interested in. The users may only need the correct predictions under some kind of conditions. Therefore, a set of statistical performance measures have been developed to quantitatively describe the predictive performances in different aspects under different conditions.

Different sets of performance measures are applied to the single-label predictors and multi-label predictors. Both types of predictors have been widely applied in bioinformatics. We will first focus on the performance measures of single-label predictors. After that, we will continue to introduce the performance measures of multi-label predictors.

In a binary predictor, which we have mentioned before, we call the two classes the positives and negatives. When we have m ($m > 2$) classes, we measure the performance on each class respectively. When the performance on the j-th class is measured, the j-th class is the positives. All remaining samples in $m-1$ classes serve as the negatives. When the overall performances are calculated, we use a different measure, which will be discussed later. Therefore, we now only focus on measuring prediction performances of binary predictors.

For every sample in a testing process, it always has a real label and a predicted label. The real label indicates the class that the testing sample really belongs to. The predicted label is the output of the predictor. Let s_k ($k = 1, 2, \ldots, n$) be one of n testing sample, $y(s_k)$ the real label of s_k and $y_e(s_k)$ the prediction result of s_k. Without loss of generality, in a binary predictor, we use +1 as the label of a positive sample and -1 as the label of a negative sample. We define the four most basic counts as the follows:

$$RP = |\{s_k | y(s_k) = +1\}|, \tag{1}$$

$$RN = |\{s_k | y(s_k) = -1\}|, \tag{2}$$

$$PP = |\{s_k | y_e(s_k) = +1\}|, \tag{3}$$

$$PN = |\{s_k | y_e(s_k) = -1\}|, \tag{4}$$

where RP, RN, PP and PN are real positives, real negatives, predictive positives and predictive negatives, $|.|$ the cardinal operator in the set theory. These four counts give the number of positives and negatives of all testing samples according to either their real labels or predictive labels. Although these four counts are not commonly seen in bioinformatics studies, they are components of commonly used performance measures.

The other four commonly used counts are the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). They can be defined as follows:

$$TP = |\{s_k | y(s_k) = +1, y_e(s_k) = +1\}|, \tag{5}$$

$$TN = |\{s_k | y(s_k) = -1, y_e(s_k) = -1\}|, \tag{6}$$

$$FP = |\{s_k | y(s_k) = -1, y_e(s_k) = +1\}|, \tag{7}$$

$$FN = |\{s_k | y(s_k) = +1, y_e(s_k) = -1\}|. \tag{8}$$

According to the above definitions, if both real label and predicted label of a sample are positive, this sample is a true positive sample. If both real label and predicted label of a sample are negative, this sample is called a true negative sample. If the real label of a sample is positive, while its predicted label is negative, this sample is called a false negative sample. If the real label of sample is negative, while its predicted label is positive, this sample is called a false positive sample. These definitions must be crystal clear, as almost all the performance measures rely on the correct number of these basic counts.

In addition, these four counts and the former four counts form a 2-by-2 contingency table, as displayed in Figure 4. The 2-by-2 matrix, which contains the TP, TN, FP and FN, can be called a confusion matrix. Intuitively, the following relationships exist:

$$TP + FP = PP, \tag{9}$$

$$TP + FN = RP, \tag{10}$$

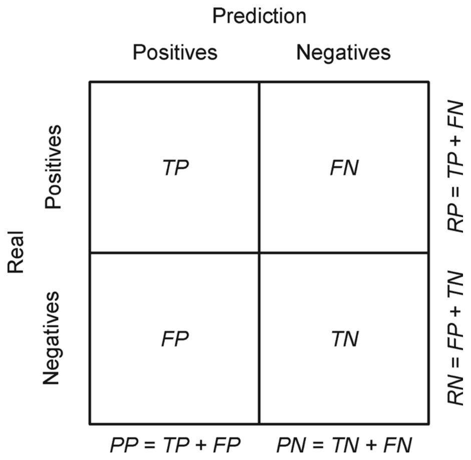
$$TN + FN = PN, \tag{11}$$

$$TN + FP = RN, \tag{12}$$

and

$$\begin{aligned} n &= TP + TN + FP + FN \\ &= RP + RN \\ &= PP + PN. \end{aligned} \tag{13}$$

Because of these relationships, the performance measures are not always represented by the commonly used four counts: TP, TN, FN and FP. A different set of notations may appear as well. For example, in literature



		Prediction		
		Positives	Negatives	
Real	Positives	*TP*	*FN*	*RP = TP + FN*
	Negatives	*FP*	*TN*	*RN = FP + TN*
		PP = TP + FP	*PN = TN + FN*	

Figure 4. The confusion matrix in testing a predictor. All the testing samples are divided into four categories, according to the real labels and the prediction results. There are altogether eight basic counts: *RP*, *RN*, *PP*, *PN*, *TP*, *TN*, *FP* and *FN*. The relationships between these counts are marked on the figure.

[43,44], four basic counts were actually the *RP*, *RN*, *FP* and *FN*. They were denoted by different notations. *FN* was denoted as N^{+}_{-}, while *FP* was denoted as N^{-}_{+}. *RP* was denoted as N^{+}, while *RN* was denoted as N^{-}. If the values of *TP* and *TN* were required, Equation (10) and Equation (12) were applied. *TP* was denoted as $N^{+}-N^{+}_{-}$, while *TN* was denoted as $N^{-}-N^{-}_{+}$. These notations are only different in appearance. Their interpretations are identical to our equivalents [5].

Based on the above definitions, three basic performance measures, which are called sensitivity (*Sen*), specificity (*Spe*) and accuracy (*Acc*), can be defined as follows:

$$Sen=\frac{TP}{TP+FN}=\frac{TP}{RP}, \tag{14}$$

$$Spe=\frac{TN}{TN+FP}=\frac{TN}{RN}, \tag{15}$$

$$Acc=\frac{TN+TP}{TN+FN+TP+FP}=\frac{1}{n}(TN+TP). \tag{16}$$

These three measures indicate the performance of a predictor in three different aspects. Sensitivity is the frequency of correctly predicted positive samples among all real positive samples. It measures the ability of a predictor in identifying positive samples. Similarly, specificity measures the ability of a predictor in identifying negative samples. Accuracy measures the ability of a predictor in correctly identifying all samples, no matter it is positive or negative. These performance measures have alias in different studies. Sensitivity may be termed as the true positive rate (*TPR*) or recall. Specificity may be termed as the true negative rate (*TNR*) or inverse recall.

In bioinformatics studies, there are two common challenges that most of the predictors must face to. The first challenge is that users of a predictor usually care only about positive outputs. The negative outputs are never important, no matter whether they are true negatives or false negatives. The second challenge is that the datasets are very imbalanced, no matter in training or testing. For example, when predicting protein phosphorylation sites, the negative samples are over 100 times more than the positives [23,24]. The users of the predictor always care only about the positive results, as only the phosphorylated sites worth further studies.

To face the first challenge, several other performance measures have to be introduced. The positive predictive value (*PPV*) is one of these kinds, which can be defined as follows:

$$PPV=\frac{TP}{TP+FP}=\frac{TP}{PP}. \tag{17}$$

The *PPV* indicates the frequency of true positives among all positive outputs. Usually, the positive outputs are subject to experimental validation in wet-lab. The wet-lab cost is always much higher than computational predictions. The average cost of getting every validated positive result depends on the *PPV*. Even if the sensitivity is very high, the predictor is still useless in practice if *PPV* is not ideal. The *PPV* is also termed as the precision [45]. Another performance measure that is tightly related to *PPV* is the false discovery rate (*FDR*), which can be defined as follows:

$$FDR=1-PPV=\frac{FP}{TP+FP}=\frac{FP}{PP}. \tag{18}$$

The *FDR* is widely used in analyzing expression profiles [46] and genetic association studies [47,48].

The *PPV* and *FDR* do not take the false negatives into consideration. However, false negatives are potential discoveries that are missed by the predictor. The number of false negatives may be considered in measuring the predictive performances. By simultaneously removing the true negatives from both the numerator and denominator of the accuracy, a performance measure called Jaccard index was introduced. It can be defined as follows:

$$J=\frac{TP}{TP+FP+FN}=\frac{TP}{n-TN}, \tag{19}$$

where *J* is the Jaccard index. The value of *J* indicates the frequency of true positives among all samples that are either real positives or positive predictions.

When the true positives are thought to be more important than the other samples, more weight can be put on the true positives when calculating the Jaccard index. If true positives are considered as twice important as the other samples, a performance measure called F_1-score can be defined as follows:

$$F_1 = \frac{2TP}{2TP + FP + FN}, \tag{20}$$

where F_1 is the F_1-score. The F_1-score is actually the harmonic mean of *PPV* and sensitivity. The following relationship exists:

$$F_1 = \frac{2PPV \cdot Sen}{PPV + Sen}. \tag{21}$$

The above performance measures remove the true negatives from the calculation. When the users care only about positive outputs, these performance measures work well.

In other cases, the training and testing datasets are very imbalanced. However, because of the practical requirements, the large number of true negatives cannot be ignored. Some other performance measures must be developed. Here, we introduce two of this kind, the balanced accuracy (*BAcc*) and the Matthew's Correlation Coefficients (*MCC*). The balanced accuracy can be defined as the average of sensitivity and specificity:

$$\begin{aligned} BAcc &= \frac{1}{2}(Sen + Spe) \\ &= \frac{1}{2}\left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right). \end{aligned} \tag{22}$$

The balanced accuracy solves most of the bias problems in an imbalanced dataset test [45,49]. The balanced accuracy has been applied in evaluating predictors for RNA editing [50] and posttranslational modification sites [51].

The Matthew's Correlation Coefficient was first proposed in 1975 [52]. It is well recognized and widely applied in evaluating predictors with imbalanced dataset. Especially, when the predictor was evaluated with a jackknife test, the *MCC* was usually reported along with the sensitivity, specificity and accuracy. Actually, if the classes are of very different sizes, there is no perfect way of describing the confusion matrix of true and false positives and negatives by a single number, the *MCC* is generally regarded as being one of the best balanced measures [45]. In most bioinformatics papers, the *MCC* is defined as the follows:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}. \tag{23}$$

Using Equations (9)–(12), *MCC* can also be defined as follows:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{PP \cdot PN \cdot RP \cdot RN}}. \tag{24}$$

This definition makes the calculation of *MCC* very easy. However, its statistical interpretation should be carefully carried out. *MCC* ranges from –1 to +1. A zero *MCC* value indicates that the predictor actually performs random guess. Its prediction results have no relationship to the real class labels of the samples. A positive or negative *MCC* value indicates that predictor is better than random guess. The +1 value of *MCC* indicates that the predictor is perfect, which reports the class labels correctly of every sample. The –1 value of *MCC* also indicates that predictor is perfect, while its outputs should be interpreted as the opposite meanings. In statistics, calculating the *MCC* value equals to the Chi-Square test on the confusion matrix. The following relationship exists:

$$MCC^2 = \frac{1}{n}\chi^2, \tag{25}$$

where χ^2 is the chi-square statistics on the confusion matrix. In statistics, the *MCC* value is also termed as the phi-coefficient or mean square contigency coefficient under the background of Pearson coefficients [45].

All the above performance measures rely on the values of *TP*, *TN*, *FP* and *FN*. However, most of the bioinformatics predictors give scores to samples before the class labels are assigned. The class labels are then assigned according to the scores. If the score is higher than a pre-defined cut-off value, the sample is predicted as a positive one. Otherwise, it is predicted as a negative one. Therefore, the choice of the cut-off value can affect the prediction performance largely. An improper choice of cut-off value makes some performance measures extremely high, while the others very low. To avoid this kind of bias, a comprehensive performance measure is necessary. Therefore, the receiver operating characteristic (ROC) curve methods are introduced.

An ROC curve describes the relationship between the sensitivity and false positive rate (*FPR*). The *FPR* can be defined as the follows:

$$FPR = 1 - Spe = \frac{FP}{FP + TN}. \tag{26}$$

Given a scoring scheme, the values of sensitivity and FPR will change along with the cut-off value. For every cut-off value, a dot can be plotted using the coordinate (*FPR*, *Sen*). A curve that connects all these dots is called an ROC curve. A demo of ROC curve can be found in Figure 5. The diagonal in Figure 5 separates the square between (0,0) and (1,1) into two parts. This diagonal is called the line of no-discrimination. An ROC curve

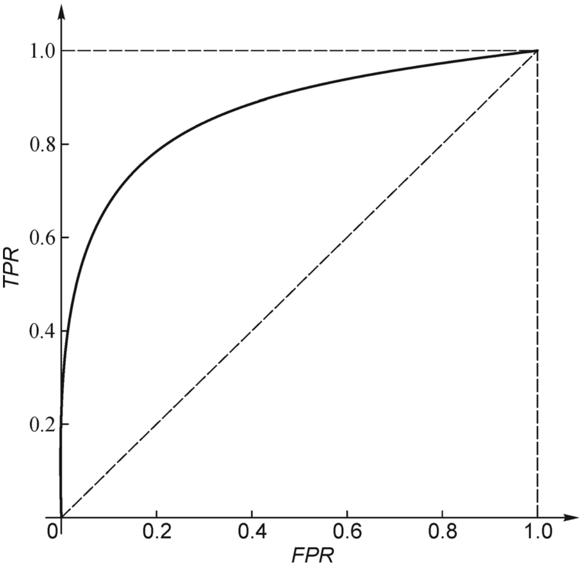


Figure 5. An ROC curve. The horizontal axis is the false positive rate (*FPR*). The vertical axis is the sensitivity, which can be termed as true positive rate (*TPR*). The solid curve is the ROC curve. The dashed diagonal is called the line of no-discrimination. An ROC curve, which is close to the top left corner, indicates the predictor has a good performance. The closer the curve to the top left corner, the better performance the predictor has. The area under curve (*AUC*) of ROC curve can be used as performance measures.

should appear in the top left part. An ROC curve, which is close to the diagonal, indicates that the predictions are close to random guesses. An ROC curve, which is close to the top left corner, indicates that predictor has a good performance. When an ROC curve is plotted, we can use the area under the curve (*AUC*) to measure the performance of the predictor. As ROC curve is not related to any particular cut-off value, the performance bias that is related to the choice of cut-off value does not exist. Actually, the *AUC* of an ROC curve equals to the probability that a randomly selected positive sample gets higher scores than a randomly selected negative sample. This can be formulated as:

$$AUC = P(X_{+} > X_{-}), \tag{27}$$

where AUC is the area under the curve, X_{+} the score of a randomly selected positive sample, and X_{-} the score of a randomly selected negative sample. Given scores of all testing samples, the probability on the right side of Equation (27) can be estimated.

It should be noticed that the shape of an ROC curve may be misleading when the dataset is highly imbalanced. An ROC curve, which is close to the upper-left corner, may not indicate a good predictor under imbalanced dataset condition [53]. To solve this problem, an alternative utility, which is known as the precision-recall (PR) curve, should be applied. The PR curve is plotted with a very similar method as the ROC curve. The only difference is that the coordinates are (recall, precision). The recall, as we mentioned before, equals to the sensitivity. The precision, as we mentioned before, equals to the PPV. As every dot in an ROC curve is related to a confusion matrix, which can be used to calculate a pair of recall and precision, every dot on an ROC curve can be mapped to a dot on a PR curve [54]. Although the AUC of a PR curve have no similar interpretations like Equation (27) in ROC, it can also be used to compare performances of different models. When comparing performances of different models or algorithms on the same dataset, the AUC of ROC curves and the AUC of PR curves always give the same results. A larger AUC of a PR curve indicates a better performance.

All the above discussions are based on binary classifiers. In a multi-class classifier, there is one more thing that we need to concern. That is the overall performance. The overall accuracy is the only measure that is commonly used in measuring the overall performance of a multi-class predictor. The overall accuracy of a multi-class predictor is usually defined as follows:

$$Acc_{Overall} = \frac{1}{n}\sum_{j=1}^{m} TP_j, \tag{28}$$

where n is the total number of testing samples, m the total number of classes and TP_j the number of true positives of the j-th classs. The other performance measures are usually reported separately for every class, as the definitions of basic counts under an overall condition will be difficult.

Recently, the multi-label classifiers are introduced to bioinformatics [55]. In a multi-label predictor, the label of a sample is not a single integer any more. Instead, we use a set of integers to label a sample. A new set of performance measures is also introduced. For the convenience of readers, in the multi-label context, we first revise the definitions of several notations. Let s_k be the k-th testing sample of n testing samples, $y(s_k)$ the set of real class labels of s_k, and $y_e(s_k)$ the set of predicted class labels of s_k. The labels in $y(s_k)$ but not in $y_e(s_k)$ are called the under-predicted labels. The labels in $y_e(s_k)$ but not in y (s_k) are called the over-predicted labels. We define an indicator function as follows:

$$\delta_k = \begin{cases} 1, & y_e(s_k) = y(s_k) \\ 0, & y_e(s_k) \neq y(s_k) \end{cases}, \tag{29}$$

where δ_k is the indicator function of s_k. This function indicates that whether the prediction result of s_k is completely correct, without any over-predicted or under-predicted label. If the prediction result of s_k is completely correct, its value is 1, otherwise 0.

The set of performance measures for multi-label predictors includes the aiming (*Aim*), coverage (*Cov*), accuracy (*Acc*), absolute-true-rate (*ATR*) and absolute-false-rate (*AFR*). They can be defined as follows:

$$Aim = \frac{1}{n}\sum_{k=1}^{n}\frac{|y_e(s_k) \cap y(s_k)|}{|y_e(s_k)|}, \tag{30}$$

$$Cov = \frac{1}{n}\sum_{k=1}^{n}\frac{|y_e(s_k) \cap y(s_k)|}{|y(s_k)|}, \tag{31}$$

$$Acc = \frac{1}{n}\sum_{k=1}^{n}\frac{|y_e(s_k) \cap y(s_k)|}{|y_e(s_k) \cup y(s_k)|}, \tag{32}$$

$$ATR = \frac{1}{n}\sum_{k=1}^{n}\delta_k, \tag{33}$$

$$AFR = \frac{1}{n}\sum_{k=1}^{n}\frac{1}{m}[|y_e(s_k) \cup y(s_k)| - |y_e(s_k) \cap y(s_k)|], \tag{34}$$

where n is the total number of testing samples and m the number of all possible labels.

In the above equations, *Aim* is aiming, which describes the average frequency of correctly predicted labels among all predicted labels [33]. Aiming is similar to the PPV in single label context. Therefore, it is also called the multi-label *PPV* [55] or multi-label precision [56]. *Cov* is coverage, which describes the average rate of correctly predicted labels among all true labels [33]. Coverage is similar to the sensitivity in single label context. Therefore, it is also called the multi-label sensitivity [55] or multi-label recall [56]. *Acc* is the multi-label accuracy, which reflects the average rate of correctly predicted labels among the labels that are either real labels or prediction results [33]. *Acc* is similar to the Jaccard index in the single-label context. *ATR* is the absolute-true-rate, which describes the frequency of absolutely correct predictions [33]. Neither over-predicted samples nor under-predicted samples are counted as correct predictions. This is the most strict performance measure in the multi-label context. *AFR* is the absolute-false-rate, which describes the average rate of wrongly predicted labels among all possible labels. The wrongly predicted labels include the over-predicted and under-predicted ones [33]. AFR is also called the Hamming-loss [55,56]. As the AFR describes the wrongly predicted label rate, the lower the AFR is, the better performance a predictor will have. This is different to other measures.

The readers may have already noticed that all these multi-label performance measures cannot be defined for every class separately. In fact, we agree to literatures [33,55–57] that it is meaningless and misleading to apply performance measures to every class separately in a multi-label context. The above multi-label performance measures should only be used for over-all performances.

CONCLUDING REMARKS

The readers of a bioinformatics paper tend to focus more on the performance values rather than the way how these values are obtained and what these values actually indicate. However, to correctly understand the performance of a predictor, the knowledge of the performance measures and the understanding of evaluation methods are necessary. Besides all the performance measures that we have defined and explained in the current review, we would like to take this opportunity to give the readers three tips in interpreting the performance values.

The first tip is on the performance comparison. Ideally, a fair and rigorous performance comparison must be carried out using identical testing dataset, identical training dataset and identical evaluation protocols [41,55]. These requirements are not easy to be satisfied practically. However, the readers should notice that, if the comparison is not carried out in a rigorous way, better values in performance measures may not guarantee a better performance in practical applications. The readers should try to use their own dataset to confirm the predictive performances in their own studies [58].

The second tip is on the feature selections. The feature selection should be regarded as part of the training procedures. If the feature selection uses whole dataset, and a cross validation is carried out after that on the same whole dataset with selected features, the predictive performance is likely to be over-estimated [59]. It is beyond the scope of this review too much to explain how this happened in a strict mathematical way. The readers may generally think that some information of the testing sample may slip into the training dataset by helping to decide that which features are selected. A safe evaluation protocol involving the feature selection is to leave the testing sample out before the feature selection process [60].

The last tip is on how to interpret the performance values that are obtained by a subsampling test or an n-fold test. As we have discussed, the performance values of subsampling test and n-fold test may vary due to different random partitioning of the dataset. Without details of the partitioning, which is usually not reported, it is not easy to reproduce the performance values exactly. Therefore, we recommend that the subsampling test and the n-fold test should be carried out for several times with different random partitioning. The average performance values should be reported as well as their standard deviations

[37].

With all the above discussions and explanations, we hope that the current work covers most evaluation methods and performance measures, which have been widely applied in bioinformatics. We expect that this paper can be a useful resource for the readers to check when they encounter glossaries of performance measures and evaluation methods in academic papers or reports.

ACKNOWLEDGEMENTS

This work was supported by the National Natural Science Foundation of China (NSFC 61005041), the Specialized Research Natural Fund for the Doctoral Program of Higher Education (SRFDP 20100032120039),Tianjin Natural Science Foundation (No. 12JCQNJC02300), and China Postdoctoral Science Foundation (Nos. 2012T50240 and 2013M530114).

COMPLIANCE WITH ETHICS GUIDELINES

The authors Yasen Jiao and Pufeng Du declare that they have no conflict of interests.

This article does not contain any studies with human or animal subjects performed by any of the authors.

REFERENCES

1. Eberwine, J., Sul, J.-Y., Bartfai, T. and Kim, J. (2014) The promise of single-cell sequencing. Nat. Methods, 11, 25–27
2. Ashley, E. A. (2015) The precision medicine initiative: a new national effort. JAMA, 313, 2119–2120
3. Chou, K.-C. (2009) Pseudo amino acid composition and its applications in bioinformatics, proteomics and system biology. Curr. Proteomics, 6, 262–274
4. Chou, K.-C. (2015) Impacts of bioinformatics to medicinal chemistry. Med. Chem., 11, 218–234
5. Jiao, Y.-S. and Du, P.-F. (2016) Predicting Golgi-resident protein types using pseudo amino acid compositions: approaches with positional specific physicochemical properties. J. Theor. Biol., 391, 35–42
6. Wang, Y. and Zeng, J. (2013) Predicting drug-target interactions using restricted Boltzmann machines. Bioinformatics, 29, i126–i134
7. Lee, K., Byun, K., Hong, W., Chuang, H. Y., Pack, C. G., Bayarsaikhan, E., Paek, S. H., Kim, H., Shin, H. Y., Ideker, T., *et al.* (2013) Proteome-wide discovery of mislocated proteins in cancer. Genome Res., 23, 1283–1294
8. Shao, J., Xu, D., Hu, L., Kwan, Y. W., Wang, Y., Kong, X. and Ngai, S. M. (2012) Systematic analysis of human lysine acetylation proteins and accurate prediction of human lysine acetylation through bi-relative adapted binomial score Bayes feature representation. Mol. Biosyst., 8, 2964–2973
9. Libbrecht, M. W. and Noble, W. S. (2015) Machine learning applications in genetics and genomics. Nat. Rev. Genet., 16, 321–332
10. Kohavi, R. and Provost, F. (1998) Glossary of terms. Mach. Learn., 30, 271–274
11. Simon P. (2013) Too Big to Ignore: The Business Case for Big Data. New Jersey: Wiley
12. Fan, Y.-X., Zhang, Y. and Shen, H.-B. (2013) LabCaS: labeling calpain substrate cleavage sites from amino acid sequence using conditional random fields. Proteins, 81, 622–634
13. Song, J., Tan, H., Shen, H., Mahmood, K., Boyd, S. E., Webb, G. I., Akutsu, T. and Whisstock, J. C. (2010) Cascleave: towards more accurate prediction of caspase substrate cleavage sites. Bioinformatics, 26, 752–760
14. Chou, K.-C. and Shen, H.-B. (2008) Cell-PLoc: a package of Web servers for predicting subcellular localization of proteins in various organisms. Nat. Protoc., 3, 153–162
15. Li X, Liu T, Tao P, Wang, C., Chen, L. (2015) A highly accurate protein structural class prediction approach using auto cross covariance transformation and recursive feature elimination. Comput. Biol. Chem., 59, 95–100
16. Kong, L., Zhang, L. and Lv, J. (2014) Accurate prediction of protein structural classes by incorporating predicted secondary structure information into the general form of Chou's pseudo amino acid composition. J. Theor. Biol., 344, 12–18
17. Guo, S.-H., Deng, E.-Z., Xu, L.-Q., Ding, H., Lin, H., Chen, W. and Chou, K. C. (2014) iNuc-PseKNC: a sequence-based predictor for predicting nucleosome positioning in genomes with pseudo *k*-tuple nucleotide composition. Bioinformatics, 30, 1522–1529
18. Xu, Y., Wen, X., Wen, L.-S., Wu, L. Y., Deng, N. Y. and Chou, K. C. (2014) iNitro-Tyr: prediction of nitrotyrosine sites in proteins with general pseudo amino acid composition. PLoS One, 9, e105018
19. Xu, Y. and Chou, K.-C. (2016) Recent progress in predicting posttranslational modification sites in proteins. Curr. Top. Med. Chem., 16, 591–603
20. Jiang, R., Tang, W., Wu, X. and Fu, W. (2009) A random forest approach to the detection of epistatic interactions in case-control studies. BMC Bioinformatics, 10, S65
21. Tang, W., Wu, X., Jiang, R. and Li, Y. (2009) Epistatic module detection for case-control studies: a Bayesian model with a Gibbs sampling strategy. PLoS Genet., 5, e1000464
22. Wu, X., Jiang, R., Zhang, M. Q. and Li, S. (2008) Network-based global inference of human disease genes. Mol. Syst. Biol., 4, 189
23. Li, T., Du, P. and Xu, N. (2010) Identifying human kinase-specific protein phosphorylation sites by integrating heterogeneous information from various sources. PLoS One, 5, e15411
24. Xue, Y., Liu, Z., Cao, J., Ma, Q., Gao, X., Wang, Q., Jin, C., Zhou, Y., Wen, L. and Ren, J. (2011) GPS 2.1: enhanced prediction of kinase-specific phosphorylation sites with an algorithm of motif length selection. Protein Eng. Des. Sel., 24, 255–260
25. Zhao, Q., Xie, Y., Zheng, Y., Jiang, S., Liu, W., Mu, W., Liu, Z., Zhao, Y., Xue, Y. and Ren, J. (2014) GPS-SUMO: a tool for the prediction of sumoylation sites and SUMO-interaction motifs. Nucleic Acids Res., 42, W325–W330
26. Nanni, L., Brahnam, S. and Lumini, A. (2012) Combining multiple approaches for gene microarray classification. Bioinformatics, 28, 1151–1157
27. Dong, X. and Weng, Z. (2013) The correlation between histone modifications and gene expression. Epigenomics, 5, 113–116
28. Dong, X., Greven, M. C., Kundaje, A., Djebali, S., Brown, J. B., Cheng, C., Gingeras, T. R., Gerstein, M., Guig, R., Birney, E., *et al.* (2012) Modeling gene expression using chromatin features in various cellular contexts. Genome Biol., 13, R53
29. Cheng, C., Shou, C., Yip, K. Y. and Gerstein, M. B. (2011) Genome-wide analysis of chromatin features identifies histone modification sensitive and insensitive yeast transcription factors. Genome Biol., 12, R111

30. Huang, J., Marco, E., Pinello, L. and Yuan, G. C. (2015) Predicting chromatin organization using histone marks. Genome Biol., 16, 162
31. Bishop CM. (2006) Pattern Recognition and Machine Learning. New York: Springer
32. Zhang, M.-L. and Zhou, Z.-H. (2007) ML-KNN: a lazy learning approach to multi-label learning. Pattern Recognit., 40, 2038–2048
33. Chou, K.-C. (2013) Some remarks on predicting multi-label attributes in molecular biosystems. Mol. Biosyst., 9, 1092–1100
34. Chou, K.-C. and Shen, H.-B. (2006) Hum-PLoc: a novel ensemble classifier for predicting human protein subcellular localization. Biochem. Biophys. Res. Commun., 347, 150–157
35. Chou, K.-C., Wu, Z.-C. and Xiao, X. (2012) iLoc-Hum: using the accumulation-label scale to predict subcellular locations of human proteins with both single and multiple sites. Mol. Biosyst., 8, 629–641
36. Du, P. and Li, Y. (2006) Prediction of protein submitochondria locations by hybridizing pseudo-amino acid composition with various physico-chemical features of segmented sequence. BMC Bioinformatics, 7, 518
37. Du, P., Tian, Y. and Yan, Y. (2012) Subcellular localization prediction for human internal and organelle membrane proteins with projected gene ontology scores. J. Theor. Biol., 313, 61–67
38. Lin, H., Deng, E.-Z., Ding, H., Chen, W. and Chou, K. C. (2014) iPro54-PseKNC: a sequence-based predictor for identifying sigma-54 promoters in prokaryote with pseudo *k*-tuple nucleotide composition. Nucleic Acids Res., 42, 12961–12972
39. Chou, K.-C. (2011) Some remarks on protein attribute prediction and pseudo amino acid composition. J. Theor. Biol., 273, 236–247
40. Chou, K. C. and Zhang, C. T. (1995) Prediction of protein structural classes. Crit. Rev. Biochem. Mol. Biol., 30, 275–349
41. Du, P., Li, T. and Wang, X. (2011) Recent progress in predicting protein sub-subcellular locations. Expert Rev. Proteomics, 8, 391–404
42. Hastie, T., Tibshirani, R. and Friedman, J. (2009) Model Assessment and Selection. In The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 219–260, New York: Springer-Verlag
43. Chou, K. C. (2001) Using subsite coupling to predict signal peptides. Protein Eng., 14, 75–79
44. Chen, W., Feng, P., Ding, H., Lin, H. and Chou, K. C. (2015) iRNA-Methyl: identifying N(6)-methyladenosine sites using pseudo nucleotide composition. Anal. Biochem., 490, 26–33
45. Powers, D. M. W. (2011) Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. Inter. J. Mach. Learn. Tech., 2, 37–63
46. Li, J., Witten, D. M., Johnstone, I. M. and Tibshirani, R. (2012) Normalization, testing, and false discovery rate estimation for RNA-sequencing data. Biostatistics, 13, 523–538
47. Andreassen, O. A., Thompson, W. K., Schork, A. J., Ripke, S., Mattingsdal, M., Kelsoe, J. R., Kendler, K. S., O'Donovan, M. C., Rujescu, D., Werge, T., *et al.* (2013) Improved detection of common variants associated with schizophrenia and bipolar disorder using pleiotropy-informed conditional false discovery rate. PLoS Genet., 9, e1003455
48. Chen, J. J., Roberson, P. K. and Schell, M. J. (2010) The false discovery rate: a key concept in large-scale genetic studies. Cancer Control, 17, 58–62
49. Brodersen, K. H., Ong, C. S., Stephan, K. E., Buhmann, J. M. (2010) The Balanced Accuracy and Its Posterior Distribution. In 2010 20th International Conference on Pattern Recognition (ICPR). 3121–3124
50. Mower, J. P. (2005) PREP-Mt: predictive RNA editor for plant mitochondrial genes. BMC Bioinformatics, 6, 96
51. Dayarian, A., Romero, R., Wang, Z., Biehl, M., Bilal, E., Hormoz, S., Meyer, P., Norel, R., Rhrissorrakrai, K., Bhanot, G., *et al.* (2015) Predicting protein phosphorylation from gene expression: top methods from the IMPROVER Species Translation Challenge. Bioinformatics, 31, 462–470
52. Matthews, B. W. (1975) Comparison of the predicted and observed secondary structure of T4 phage lysozyme. BBA – Protein Structure, 405, 442–451
53. Saito, T. and Rehmsmeier, M. (2015) The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. PLoS One, 10, e0118432
54. Davis, J. and Goadrich, M. (2006) The relationship between precision-recall and ROC curves. In Proceedings of the 23rd international conference on Machine learning. 233–240, New York: the Association for Computing Machinery
55. Du, P. and Xu, C. (2013) Predicting multisite protein subcellular locations: progress and challenges. Expert Rev. Proteomics, 10, 227–237
56. Tsoumakas, G., Katakis, I. and Vlahavas, I. (2010) Mining Multi-label Data. In Data Mining and Knowledge Discovery Handbook. 667–685, New York: Springer US
57. Tsoumakas, G. and Katakis, I. (2007) Multi-label classification: an overview. Int. J. Data Warehous. Min., 3, 1–13
58. Sprenger, J., Fink, J. L. and Teasdale, R. D. (2006) Evaluation and comparison of mammalian subcellular localization prediction methods. BMC Bioinformatics, 7, S3
59. Bermingham, M. L., Pong-Wong, R., Spiliopoulou, A., Hayward, C., Rudan, I., Campbell, H., Wright, A. F., Wilson, J. F., Agakov, F., Navarro, P., *et al.* (2015) Application of high-dimensional feature selection: evaluation for genomic prediction in man. Sci. Rep., 5, 10312
60. Varma, S. and Simon, R. (2006) Bias in error estimation when using cross-validation for model selection. BMC Bioinformatics, 7, 91