Prediction of Drug-Gene Interactions: A *k*-NN Approach

Zonghao Feng[1]

Abstract

With the growing impact of genomics on drug discovery and development, it's critical to develop efficient algorithms for predicting new drug-gene interactions. In this report, we present a fast and accurate similarity-based algorithm following the k-NN scheme, which could make predictions from very limited known interactions. According to the results of evaluation using the AUC metric, our approach outperforms several methods in the given dataset.

1. Introduction

Among the numerous topics in genomics, the study of drug-gene interactions has become an increasingly important field. Predicting drug-gene interactions using computational methods could not only greatly reduce the time cost of drug discovery and development, but also help researchers better understanding the inner mechanism of various biological processes. Many advanced predictors have been invented to meet the demand, while nowadays machine learning-based methods stand out for their boosted efficiency respect to traditional approaches. [2]

The k-nearest neighbors algorithm (k-NN for short) is a widely-used supervised machine learning method. It follows a simple but effective idea: for each testing sample, find k nearest training samples in the feature space, then use them to vote for classification or average them for regression. Our method applied k-NN in predicting drug-gene interactions to see whether this classic algorithm works in the brand new field.

2. Methods

2.1. Notations

Assume that we have known the gene set $G = \{g_1, g_2, ..., g_m\}$ and the drug set $D = \{d_1, d_2, ..., d_n\}$. The similarity between genes could be represented as a matrix $S_g(g_i, g_j)$. Likewise, drug similarity matrix is $S_d(d_i, d_j)$. The interactions between drugs and genes could also be expressed as a matrix $I(d_i, g_j)$, where for each drug-gene pair, 1 means they interact with each other, and 0 means not. The goal of the algorithm is to predict whether a drug-gene pair will interact or not, that is given (d_x, g_y), return the possibility of interaction for them.

[1]Taishan College, Shandong University

2.2. Procedure

Consider a drug-gene pair (d_x, g_y), there are two ways to calculate the prediction score: drug-based method and gene-based method, which both follow a similar idea. The final prediction score could be set to the sum of these two scores. For the detailed description following, we only take the drug-based way for instance.

Firstly, find k nearest drugs of d_x. As the drug similarity matrix has been given, just extract vector $S_d[d_x]$ for processing. Two mainstream k-nearest-neighbor finding algorithms are sort-based method and heap-based method. They both share a $O(n \log n)$ time complexity, but the space usage of heap-based method could be reduced to $O(k)$ after taken appropriate optimization, while $O(n)$ for sort-based method. The key to save space is by using a min heap which size is limited in k. The heap is organized by drug similarity to d_x in increasing order. While the heap size is less than k, keep pushing drugs in heap; when the heap size is equal or more than k, we only push drugs with a larger similarity than the one on the heap's top, and then pop heap to keep heap size in k. In our implementation, we use a priority queue as a min heap.

After picking k nearest neighbors of d_x, we then calculate prediction score by the equation $score = \sum_k^{i=1} S_d[d_x][d_i] \cdot I[d_i][g_y]$. Obviously, the drugs with more similarity weights more in the prediction score. The pseudocode of the complete procedure is shown in K-NN Regression as below. As a matter of fact, in real implementation we could calculate scores of all pairs containing d_x in a row after obtained k nearest neighbors of d_x to prevent duplicate calculations.

K-NN Regression(k, d_x, g_y)

```
1  let pq be a new priority queue, increasing by S_d[d_x] from top
2  for i = 0 to S_d.length
3      if pq.size < k
4          Push(pq, d_i)
5      elseif S_d[d_x][Top(pq)] < S_d[d_x][i]
6          Pop(pq)
7          Push(pq, d_i)
8  while pq.size > 0
9      nearest = Top(pq)
10     Pop(pq)
11     score = score + S_d[d_x][nearest] · I[nearest][g_y]
12 return score
```

Note that when k is set to 1, our method will degenerate to the nearest profile method, which is usually being taken as a baseline method for comparison. On the other hand, when k is large enough to hold all drugs in the data set, our method will turn into the weighted profile method [3].

2.3. Evaluation

We use a 10-fold cross-validation to evaluate the performance of our algorithm. It is executed by divide the interaction list into 10 parts randomly first, then each fold will be taken as testing data in turn, while the remaining 9 folds are used as training data. The pairs outside the interaction list are used as testing data always. We take the average result of 5 trials of 10-fold cross-validation as the final result. For convenience, before each trial we use the GNU `shuf` tool to reorder the interaction list randomly, and then divide the list sequentially. It is equivalence to the original idea, but much more easier to implement.

As for performance measurement, we use the area under receiver operating characteristic (AUC) curve and the area under the precision-recall (AUPR) curve as metric. To draw the curves, we must sort all predictions by their score, then adjust threshold from highest to lowest and record the changes of true positive (TP), false positive (FP), true negative (TN) and false negative (FN) instances.

The receiver operating characteristic (ROC) curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR), where $TPR = TP/(TP+FN)$, and $FPR = FP/(FP+TN)$. We can calculate AUC by the following equation [4], where x represents for FPR and y represents for TPR:

$$\mathrm{AUC} = \frac{1}{2}\sum_{i=1}^{m-1}(x_{i+1} - x_i) \cdot (y_i + y_{i+1})$$

The equation above needs to iterate m times. In fact, for each adjustment of threshold, only one of TPR and FPR will change, so we can just sum up areas of small rectangles every time TPR changes.

The precision-recall (PR) curve is created by plotting precision to recall, where precision is also called positive predictive value (PPV), defined as $PPV = TP/(TP + FP)$, and recall is equal to TPR. The AUPR is harder to estimate than the AUC, as the PR curve may fluctuate because precision is unstable. We calculate AUPR by sum up areas of small trapezoids:

$$\mathrm{AUPR} = \frac{1}{2}\sum_{i=1}^{m-1}(x_{i+1} - x_i) \cdot (y_i + y_{i+1})$$

3. Results

We test our algorithm under various settings of parameter k first, and the result is shown in Figure 1. From the curve we can know that k-NN predictor reaches its best performance when k is in the range of $[5, 10]$ approximately.

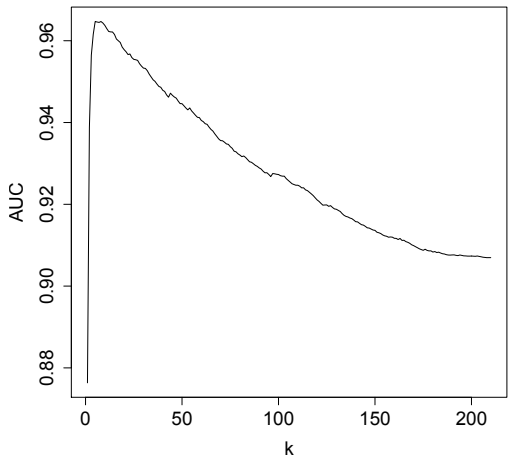

Figure 1. AUC under various settings of parameter k

So we compare the performance of best k-NN ($k = 5$) to nearest profile method ($k = 1$) and weighted profile method ($k =$ all). The ROC curves are drawn in Figure 2, and the PR curves are drawn in Figure 3. The average AUC and AUPR of 5 trials of 10-fold cross validation is shown in Table 1.

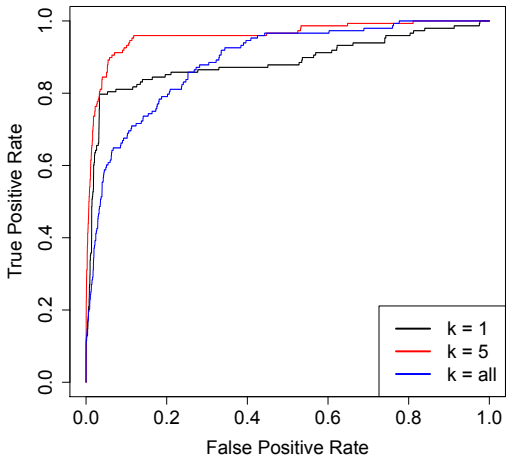

Figure 2. ROC curve

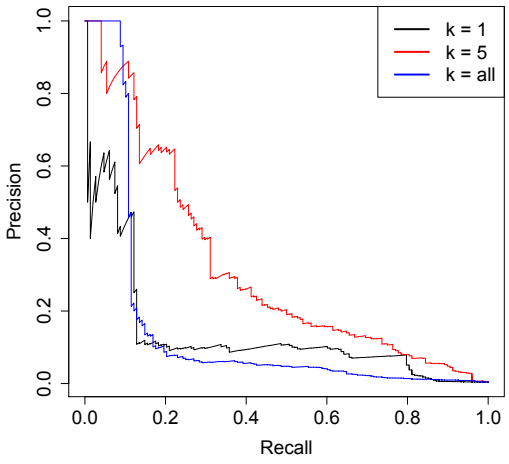

Figure 3. PR curve

	AUC	AUPR
$k = 1$	0.880023	0.175606
$k = 5$	0.962999	0.377858
$k = all$	0.906536	0.157509

Table 1. AUC and AUPR

The figure conveys that best k-NN outperforms other two methods. On the one hand, its ROC curve shows excellent performance; On the other hand, its PR curve shows poor outcome. We'll discuss this phenomenon in detail later.

4. Discussion

Our k-NN based algorithm shows great efficiency and optimal evaluation result using the AUC metric. The bottleneck of common k-NN algorithms is the cost of calculating distances between training samples. But as in our case, the similarity matrix of genes and drugs are given, which could be directly used as distance metrics, so that the time cost is sharply reduced. Meanwhile, the AUC is very close to optimal value 1, which reflects that our method performs well.

One major drawback of our method is the unsatisfactory performance under the judgement of AUPR. An algorithm that optimizes the area under the ROC curve is not guaranteed to optimize the area under the PR curve, as the AUPR punishes much more on false positive examples. [1] The poor result on AUPR is mainly due to that there are too few true drug–target interactions in the testing set, which makes it harder to keep a high precision.

References

1. Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

2. Hao Ding, Ichigaku Takigawa, Hiroshi Mamitsuka, and Shanfeng Zhu. Similarity-based machine learning methods for predicting drug–target interactions: a brief review. *Briefings in Bioinformatics*, page bbt056, 2013.

3. Yoshihiro Yamanishi, Michihiro Araki, Alex Gutteridge, Wataru Honda, and Minoru Kanehisa. Prediction of drug–target interaction networks from the integration of chemical and genomic spaces. *Bioinformatics*, 24(13):i232–i240, 2008.

4. Zhihua Zhou. *Machine Learning*. Tsinghua University Press, 2016.