

RESEARCH

Open Access

# Predicting gene function using few positive examples and unlabeled ones

Yiming Chen<sup>1,3</sup>, Zhoujun Li<sup>2\*</sup>, Xiaofeng Wang<sup>4</sup>, Jiali Feng<sup>4</sup>, Xiaohua Hu<sup>5\*</sup>

From The 2009 International Conference on Bioinformatics & Computational Biology (BioComp 2009) Las Vegas, NV, USA. 13-16 July 2009

## Abstract

**Background:** A large amount of functional genomic data have provided enough knowledge in predicting gene function computationally, which uses known functional annotations and relationship between unknown genes and known ones to map unknown genes to GO functional terms. The prediction procedure is usually formulated as binary classification problem. Training binary classifier needs both positive examples and negative ones that have almost the same size. However, from various annotation database, we can only obtain few positive genes annotation for most offunctional terms, that is, there are only few positive examples for training classifier, which makes predicting directly gene function infeasible.

**Results:** We propose a novel approach SPE\_RNE to train classifier for each functional term. Firstly, positive examples set is enlarged by creating synthetic positive examples. Secondly, representative negative examples are selected by training SVM(support vector machine) iteratively to move classification hyperplane to a appropriate place. Lastly, an optimal SVM classifier are trained by using grid search technique. On combined kernel of Yeast protein sequence, microarray expression, protein-protein interaction and GO functional annotation data, we compare SPE\_RNE with other three typical methods in three classical performance measures recall  $R$ , precise  $P$  and their combination  $F$ : twoclass considers all unlabeled genes as negative examples, twoclassbal selects randomly same number negative examples from unlabeled gene, PSoL selects a negative examples set that are far from positive examples and far from each other.

**Conclusions:** In test data and unknown genes data, we compute average and variant of measure  $F$ . The experiments show that our approach has better generalized performance and practical prediction capacity. In addition, our method can also be used for other organisms such as human.

## Background

One of the important challenges in the post-genome era is determining the functional role of all genes in the cell although about one-third of the genes have been annotated and deposited in database such GO (gene ontology) [1]. With the recent invention of several large-scale experimental methods, a wealth of functional genomic data was accumulated, including sequence, micro-array expression profile and protein-protein interaction data.

These large data-sets have fueled an interest in computational approaches to gene function prediction, which promises to harness the information present in these large collections of data to automatically deduce accurate gene annotations [2,3]. Furthermore, many works have shown that integration of different kinds of data sources can considerably improve prediction results [4,5]. GO is a widely-used set of functional terms with which some genes are annotated, we also call functional terms as functional classes in related to classification problem from machine learning. GO functional annotation associates each gene or gene product to some functional terms. For an unknown gene, predicting its functions will assign some GO functional terms to it,

\* Correspondence: [lizj@buaa.edu.cn](mailto:lizj@buaa.edu.cn); [thu@cis.drexel.edu](mailto:thu@cis.drexel.edu)  
<sup>2</sup>School of Computer Science and Engineering BeiHang University, Beijing, China  
<sup>5</sup>College of Information Science and Technology, Drexel University, Philadelphia, PA, 19104, USA  
Full list of author information is available at the end of the article

which is called multi-label classification problem in machine learning community. The mainstream approach is to transform it into a binary classification task for each functional class, which focuses on training a classifier such as SVM (support vector machine) with some labeled positive and negative examples. However, the available information from the annotation databases, such as GO [1], is only about positive examples, i.e. for a functional class, we only know which gene is assigned to it, but we are not sure that a gene has no this function except for too few genes. As a result, when training classifier for a functional class, we can only obtain labeled positive examples and many unlabeled ones. In other words, for a functional class, we need to learn a classifier from positive and unlabeled examples. Thus, an important step is to select a suitable set of negative examples from unlabeled examples before training classifier.

Some approaches to select negative examples have been proposed. For example, Lanckriet et. al labeled the annotated genes as positive examples and the remaining ones as negative ones for each functional class [4]. Carter et al randomly selected the negative examples with the same size as the positive examples from the unlabeled examples [6]. We call these two methods *twoclass* and *twoclassbal* algorithm respectively. Chunlin Wang et.al selected a set of negative examples in two steps: firstly, identifying genes which are far from each other and the most dissimilar to positive examples as initial negative examples set. Then, using iteratively SVM to expand negative examples and stopping while the remaining unlabeled examples are less than given threshold. Their method is called PSoL(Positive Sample only learning) and its detail can be found in [7].

Above approaches can be divided into two categories and some problems can occur when only few positive examples and major unlabeled ones are given: 1. Regarding all unlabeled examples as negative examples [4]. On the one hand, it may lead to class imbalance problem because of few positive examples [8]. on the other hand, the false negative noise may seriously decrease the prediction accuracy. 2. Selecting negative examples with same size as positive examples [6,7]. These methods eliminate the impact of imbalanced problem, but, only few negative examples can be selected, as a result, the classifier is trained on a small training set and easily leads to over-fitting. When we use GO annotation, many functional classes have few annotated genes, which will lead to a lower prediction accuracy and need to be solved [9]. In this paper, aiming at both imbalance and over-fitting problem for genes function prediction with only few positive examples and unlabeled examples, we propose a novel strategy for predicting genes function using SVM. Firstly, we create some

synthetic positive examples with few negative noises to enlarge positive examples set  $P$ . Secondly, we extract a representative negative example set  $RN$  from unlabeled genes  $U$  using SVM iteratively. Finally, an optimal SVM classifier with RBF (Radial Basis Function) is trained by using Grid-search technique. This method is called SPE\_RNE(Learning classifier by Synthetic Positive Examples and Representative Negative Examples).

## Results and discussion

### Experiment setting

#### Data sets

**Gene annotation** We used gene ontology and corresponding gene function association of Yeast [10] released in April 2007. Gene association file contained 5,873 genes ,the number of known and unknown genes is 3,796 and 2,077 respectively. We up-propagate the gene annotation along GO hierarchical structure and obtained a reduced GO which has only 99 GO terms under guidance of biological experts. To compare the algorithm performance, we divide them into four groups according to number of annotated genes as shown in Table 1. There are 53 functional classes with annotated genes less than 60 among total 99 terms.

**Protein sequence** The protein sequence of all of the Yeast genes were downloaded from SGD [10].We applied the Smith-Waterman pairwise sequence alignment algorithm [11] to these sequences. Each protein is represented as a vector of Smith-Waterman log E-values, and computed with respect to all 5,873 Yeast genes. A 5873\*5873 similar matrix is obtained.

**Microarray expression profile** Microarray datasets are real-valued matrices measuring gene expression levels under different experimental conditions. We use gene expression microarray data from the Stanford Microarray Database(SMD) [12] containing results from several publications, providing a total of 294 real-valued features for all 5,873 genes. Microarray entries typically include missing values due to experimental imperfections. We estimate such entries using the widely accepted KNNimpute algorithm [13] with default  $k$  value. Then, we computed similarity between two genes using Gauss kernel with  $\gamma = 2$ . The second 5873\*5873 gene similar matrix is generated.

**Protein-protein interaction** We downloaded the protein-protein interaction data from BioGRID2.0.30 [14].

**Table 1 Four groups of function terms and number of term**

annotation interval	< 60	60-100	100-300	> 300
number of terms	53	17	18	11

99 functional terms are divided into four groups according to number of annotated genes. The first row are four groups with interval of genes number. The second row is number of terms.

Protein-protein interaction data is described as a graph in which nodes denote protein and edges denote interaction and diffusion kernel [15] with diffusion constant  $\beta = 2$  is used to measure the similarity between two proteins. Each gene is also represented as a vector of similarity with respect to all 5,873 genes. The third gene similar matrix is computed.

Several previous researches have shown that integrating various genomic data to predict gene function can improve prediction accuracy [4,5]. In this paper, we add three similar matrices and obtain a sum matrix. It is noticeable that each matrix should be centralized and normalized to eliminate the effect from major data before adding them [4]. While training SVM classifier, this pre-computed kernel matrix is used.

**Experiment setting and evaluation**

We used LIBSVM [16] to implement SPE-RNE and related algorithms two class SVM twoclass, two class balanced SVM (twoclassbal) and PSoL in matlab. First, we divided 3796 known genes into training set and validation set, after training SVM classifier on training set, the generalized performance of algorithms were compared on validation set. Widely-accepted measures, including precision rate  $P$ , recall rate  $R$  and their combination  $F1$ , are used. Their definitions are as follows:

$$P = \frac{TP}{TP + FP} \tag{1}$$

$$R = \frac{TP}{TP + FN} \tag{2}$$

$$F1 = \frac{2 \times P \times R}{P + R} \tag{3}$$

where  $TP, FP$  and  $FN$  denote the number of true positive, false positive and false negative respectively. Then, using 2,077 unknown genes released in April 2007 as test examples, we predict their functions and evaluate ROC (Receiver operating characteristics) score with gene function association released in December 2008 as annotation standard.

**Performance comparison on known genes**

For each functional class, 3796 genes are in two categories: genes assigned to this functional class and unlabeled genes. we randomly select 20 percent from these two categories as validation set, the others are training set. SVM classifiers are learned on training set and used to predict genes functions on validation set to evaluate generalized performance of algorithms.

When the number of negative examples is far more than positive examples, imbalanced problem occurs and

**Table 2 The number of functional class whose  $F1 = NaN$  for four algorithms and four functional groups**

algorithms	< 60	60-100	100-300	> 300
twoclass	31	8	3	1
twoclassbal	0	0	0	0
PSoL	22	6	0	1
SPE-RNE	0	0	0	0

When the number of negative examples is far more than positive examples, imbalanced problem occurs and the algorithm can not recall any true positive examples for some functional classes. As a result,  $P = 0$  and  $R = 0$  result in  $F1 = NaN$  (Not a Number) in matlab. Table 2 shows the number of  $NaN$  for each functional group.

the algorithm can not recall any true positive examples for some functional classes. As a result,  $P = 0$  and  $R = 0$  result in  $F1 = NaN$  (Not a Number) in matlab. Table 2 shows the number of  $NaN$  for each functional group.

As shown by table 2, twoclass has the most serious imbalance and PSoL has more serious imbalance, but our method SPE-RNE, like twoclassbal, doesn't suffer from imbalanced problem at all because we select reasonable quantity of negative examples after enlarging the positive examples set. In addition, functional classes with few annotated genes have more serious imbalance.

For twoclassbal algorithm, while serious imbalance does not occur, the over-fitting may arise to affect prediction performance due to few training examples. To evaluate the algorithm fairly, we set  $F1 = NaN$  to  $F1 = 0$ . For each algorithm and functional group, the means and variances of  $F1$  are listed in Table 3 and Table 4 respectively.

In table 3, although twoclassbal has not class imbalance problem, but it has worst performance because it has too few training examples, which easily causes over-fitting. For functional classes with few annotated genes, the existed algorithms, like twoclass, twoclassbal, PSoL have lower  $F1$  values, our algorithm significantly improves the  $F1$  values in this case. For functional classes with more annotated genes, our algorithm has better performance too. While only few training examples are used to learn SVM classifier, over-fitting problem may occur and make algorithms unstable. We compute the variances of  $F1$  for each functional group

**Table 3 The mean of  $F1$  for four algorithms and four functional groups**

algorithms	< 60	60-100	100-300	> 300
twoclass	0.1679	0.1881	0.3641	0.3675
twoclassbal	0.0912	0.1329	0.2877	0.3785
PSoL	0.2094	0.2223	0.4006	0.3999
SPE-RNE	0.5545	0.6589	0.5639	0.4848

Table 3 lists the average values of  $F1$  for all algorithms on four functional terms groups. These values can evaluate the comprehensive prediction performance of algorithms on different groups and compare their generalized capacity.

to evaluate the stability of algorithm. Table 4 shows that our algorithm has good stability.

### Predicting performance on unknown genes

Since April 2007, some of 2077 unknown genes have been annotated with some functions. We consider these 2077 genes as test examples and use trained SVM classifier to predict function for them. The GO function association released in December 2008 is regarded as complete annotation, that is, for each functional class, if a gene is assigned to it, the label is set to 1, otherwise -1. For each algorithm, the ROC score, which is area under ROC, is evaluated as comparison measures. In previous section, we use *F1* to evaluate algorithm performance because we think that GO function association released in April 2007 is incomplete. The ROC scores are listed in Table 5.

For group 1 and 2, our algorithm significantly improve the ROC score, which illustrates better prediction performance for unknown genes. For group 3, we only add synthetic examples as many as positive examples, and for group 4, we don't create any synthetic positive examples. But, our algorithm for extracting representative negative examples slightly improve the ROC score too. The average number of correctly predicted genes and true average number are displayed in Figure 1 for each group. In each group, our algorithm can recall more positive genes on average.

### Predicting result on unknown genes

We list predicted functional classes for ten genes with most predicted functional terms in Table 6, these genes were unknown in April 2007, but they were annotated with one or multiple functional classes in December 2008.

### Conclusions

In this paper, We propose a novel approach to predicting gene function for genes with few positive examples and unlabeled ones SPE-RNE: creating synthetic examples to enlarge the set of positive examples, extracting representative negative examples from unlabeled

**Table 4 The variance of F1 for four algorithms and four functional groups**

algorithms	< 60	60-100	100-300	> 300
twoclass	0.0636	0.0929	0.1034	0.0536
twoclassbal	0.0142	0.0083	0.0462	0.0215
PSoL	0.0594	0.0813	0.0629	0.0456
SPE-RNE	0.0523	0.0485	0.0359	0.0420

Table 4 lists the variances of *F1* for all algorithms on four functional terms groups. These values can evaluate the stability of algorithms on different groups. While only few training examples are used to learn classifier, overfitting may occur and makes algorithm unstable.

**Table 5 The average ROC score for four algorithms and four functional groups**

algorithms	< 60	60-100	100-300	> 300
twoclass	0.5081	0.7448	0.7743	0.7044
twoclassbal	0.5207	0.7529	0.8002	0.7118
PSoL	0.5313	0.7626	0.7726	0.7142
SPE-RNE	0.6827	0.7969	0.8084	0.7266

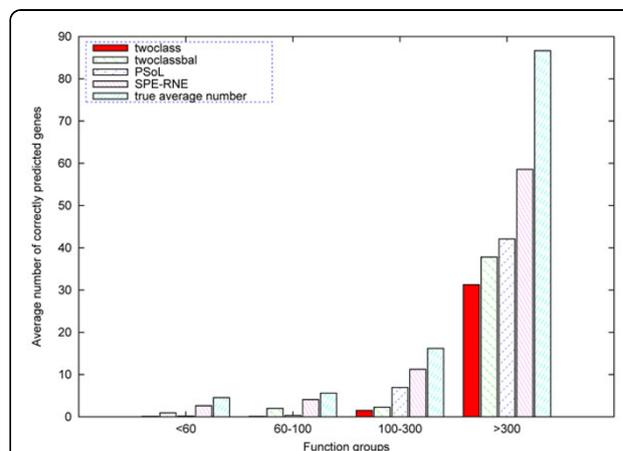
ROC score, that is area under ROC curve, is a widely-accepted performance measure for prediction classification problem. The larger ROC score is, the better the performance of classification algorithm is.

examples and training SVM classifier using Grid-search technique. For SPE-RNE, the validation on known gene data set shows its best *F1* value and good stability. Prediction on unknown genes set illustrates its higher ROC scores and better prediction performance than several classic algorithms. All the algorithms run in a sum matrix which is obtained by adding simply several similarity matrixes from heterogeneous data sources, which may loss some information. How to integrate effectively these heterogeneous data to predict gene function is our next research subject in future. In addition, our method can also be used for other organisms such as human.

### Methods

#### Creating synthetic examples to enlarge the positive examples set

The problem about learning classifier with few positive examples can be found in text classification domain. An intuitive idea is to enlarge the positive examples set. Li et al [17] assume that positive examples in *P* and likely positive examples from *U* have common underlying feature dimensions (or subspaces) as they belong to the



**Fig. 1 The average number of correctly predicted genes according to GO association released in December 2008 for unknown gene in April 2007** In figure 1, the height of bar denotes the average number of genes predicted correctly by four algorithms and average true number of genes on different groups.

**Table 6 The ten genes with the most predicted functions and their predicted functions**

protein	predicted functions			
YKR084c	GO:0000166	GO:0003676	GO:0003824	GO:0003924
	GO:0005488	GO:0016462	GO:0016787	GO:0017076
	GO:0017111	GO:0045182		
YEL030w	GO:0000166	GO:0003824	GO:0005488	GO:0005515
	GO:0016787	GO:0017076	GO:0017111	GO:0051082
YDR332w	GO:0003676	GO:0003824	GO:0004386	GO:0016462
	GO:0016787	GO:0016818	GO:0016887	GO:0017111
YBR025c	GO:0000166	GO:0003824	GO:0016462	GO:0016818
	GO:0016887	GO:0017076	GO:0017111	
YMR078c	GO:0003676	GO:0003824	GO:0016787	GO:0016818
	GO:0016887	GO:0017111		
YLR035c	GO:0000166	GO:0003676	GO:0003677	GO:0003824
	GO:0016787	GO:0017076		
YGL175c	GO:0003676	GO:0003677	GO:0003824	GO:0004519
	GO:0016787	GO:0016788		
YLR419w	GO:0003676	GO:0003824	GO:0004386	GO:0016818
	GO:0016887	GO:0017111		
YNL001w	GO:0003824	GO:0004518	GO:0004519	GO:0004540
	GO:0016788			
YJL057c	GO:0003824	GO:0004672	GO:0016301	GO:0016740
	GO:0016772			

Table 6 lists ten genes with top predicted functional classes and their predicted functional terms. The first column is gene name and other columns are functional classes predicted correctly by algorithm SPE-RNE.

same class. The representative words ( $RW$ ) extracted from  $P$  are used to identify more hidden positive examples from  $U$ . Fung et al [18] firstly identified feature words in  $P$  and select a set of reliable negative examples from  $U$ , then, all negative examples are divided into some clusters due to the diversity of negative examples, for an example from  $U$ , computing similarity  $d_P$  between it and centroid of positive examples and similarity  $d_U$  between it and cluster centroid of negative examples, if  $d_U - d_P$  is greater than a given threshold, it is added to  $P$ .

Due to sparse and discrete features of text vectors, Li and Fung's method can not be used for continuous feature vector of gene. We can not find so-called feature words in gene vectors to identify hidden positive examples. We can only use some distance, such as Euclidean distance, to measure the similarity between genes [7]. Fung's method enlarges the positive examples set after selecting the reliable negative examples, but, if we use Euclidean distance or cosine distance, a better positive example centroid can not be found due to irregularity of positive examples distribution, which has been validated by our many experiments. Our experiments also show that enlarging the  $P$  by identifying hidden positive examples from  $U$  generates easily false positive noise because of few hidden positive examples in  $U$ .

We create synthetic positive examples to enlarge the  $P$ . For each  $e \in P$ , we find its  $k$  nearest neighbors in  $P$  and then create synthetic examples along the line segments joining  $e$  and some/all of the  $k$  nearest neighbors. Depending on the required amount of enlarging,  $n$  neighbors from the  $k$  nearest neighbors are randomly chosen. For instance, if the needed amount of enlarging is 500%, only five neighbors from the  $k$  nearest neighbors are chosen and one synthetic example is generated in each direction. Given  $e' \in P$  is one of  $k$ -nearest neighbors of  $e$ ,  $\alpha \in (0, 1)$  is a random number, Synthetic examples are generated in the following way:

$$\tilde{e} = \alpha * e + (1 - \alpha) * e' \quad (4)$$

The following propositions convinces us of likely positive examples of  $\tilde{e}$ .

**Proposition 0.1**

*Let  $e$  and  $e'$  be two positive examples. Then,*

$$\tilde{e} = \alpha * e + (1 - \alpha) * e', \alpha \in (0, 1)$$

is a likely positive example.

**Proof**

Let the classification hyperplane be

$$f(x) = w^T x + b$$

According to Vapnik's theory minimizing empiric risk [19], following two inequalities are correct with probability close to 1:

$$w^T e + b > 0$$

$$w^T e' + b > 0$$

$$\text{Further } f(\alpha e + (1 - \alpha)e')$$

$$= w^T(\alpha e + (1 - \alpha)e') + b$$

$$= \alpha w^T e + (1 - \alpha)w^T e' + b$$

$$= \alpha(w^T e + b) + (1 - \alpha)(w^T e' + b)$$

Thus

$$f(\alpha e + (1 - \alpha)e') > 0$$

That is,  $\tilde{e} = \alpha e + (1 - \alpha)e'$  is a positive example with probability close to 1.

Therefore, the synthetic example  $\tilde{e}$  is probably positive example and the enlarged set of positive examples,  $\tilde{P}$  has few negative noise. In fact, from the point of view of algebra,  $\tilde{e}$  is convex combination of  $e$  and  $e'$ , and from the point of view of geometry,  $\tilde{e}$  is a random point in line segment from  $e$  to  $e'$ . We have following algorithm 1 for creating synthetic examples.

In our experiments, all the functional classes are divided into four groups according to number of annotated genes. We enlarge positive example set with different times for different groups and set  $k = 10$ , which is shown in Table 7. Particularly, we did not make

synthetic positive examples for functional classes with more than 300 annotated genes.

**Algorithm 1**

Algorithm creating synthetic examples

```

1: function MAKESYNEXPS( $P, n, k$ )
2:  $P$  is positive examples set;
3:  $n$  is amount of enlarging
4:  $k$  is number of nearest neighbors
5: pnum=size of  $P$ 
6: nnarray=array of nearest neighbor with size  $k$ 
7: synexps=array of synthetic examples
8: making synthetic positive examples
9: for all  $e \in P$  do
10: nnarray= $k$  nearest neighbors of  $e$  from  $P$ 
11: selnn= $n$  nearest neighbors randomly selected from
    nnarray
12: for all  $e' \in$  selnn do
13: generating a random number  $\alpha \in (0, 1)$ 
14:  $\tilde{e} = \alpha * e + (1 - \alpha) * e'$ 
15: adding  $e$  into synexps
16: end for
17: end for
18: Return synexps;
19: end function
    
```

**Extracting the representative negative examples**

After enlarging the positive examples set  $P$ , we need to train SVM on new positive examples set  $\tilde{P}$  and unlabeled examples set  $U$ . To learn a better classifier, we should extract a subset of the most probably negative examples from unlabeled data  $U$  so that it can best recover the positive examples hidden in  $U$ . This extracted negative examples subset can represent the whole negative set well and should have suitable size to avoid the class imbalance problem. To achieve this goal, our algorithm extracts representative negative examples and consists of three steps.

Step 1, identifying a reliable initial negative examples set. For gene vectors, only distance-based similarity can be used and the most dissimilar genes to positive examples are assumed as reliable negative examples. In PSoL, initial negative set  $N$  met two conditions: ① all elements in  $N$  are most dissimilar to positive example set ②. ② elements in  $N$  are far from each other. Since this problem is NP-hard, an approximate solution was used.

**Table 7 The enlarging times for different functional groups**

annotation interval	< 60	60-100	100-300	> 300
enlarging times	4	2	1	0

Algorithm SPE-RNE enlarges the positive examples set with different times according to the number of positive examples. The first row is functional groups and the second row is times. The functional classes with few positive examples enlarges more times.

In our algorithm, one-class SVM [20] is utilized to extract efficiently initial negative examples. Give a percentage of negative examples, such as 10 percent, it can draw an initial decision boundary to cover most of the positive and unlabeled examples. The data points not covered by the decision boundary can be regarded as negative example points because these data points are far from the positive set in the feature space.

Step 2, learning SVM iteratively to move classification hyperplane to an appropriate place. In this iteration process, a set of classifier,  $C_S$  and corresponding set of negative example set,  $N_S$  are obtained. In  $i^{th}$  iteration, we obtain not only SVM classifier  $C_i$  but also negative example set  $N_{i+1}$  to be used to train next SVM

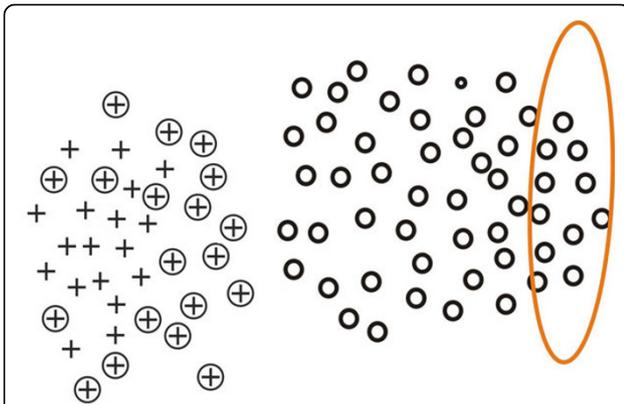
$$N_{i+1} = N_{pred} \cup N_{SVs} \quad (5)$$

where  $N_{pred}$  is the most reliable negative examples predicted in current iteration and  $N_{SVs}$  is negative support vector set of  $C_i$ . To avoid class imbalance, the size of  $N_{pred}$  is set as  $N_{pred} \leq m | \tilde{P} |$  and many experiments shown that  $m = 3$  is suitable. In addition, only the negative support vectors of  $C_i$  are selected as representatives of previous negative training examples. In this iteration process,  $|U|$  becomes smaller and smaller. When only few unlabeled examples are remained, the  $N_i$  may has more false positive examples and the classifier may become bad, therefore, we stop iterating. According to a large number of experiments, the stopping criteria is set as:

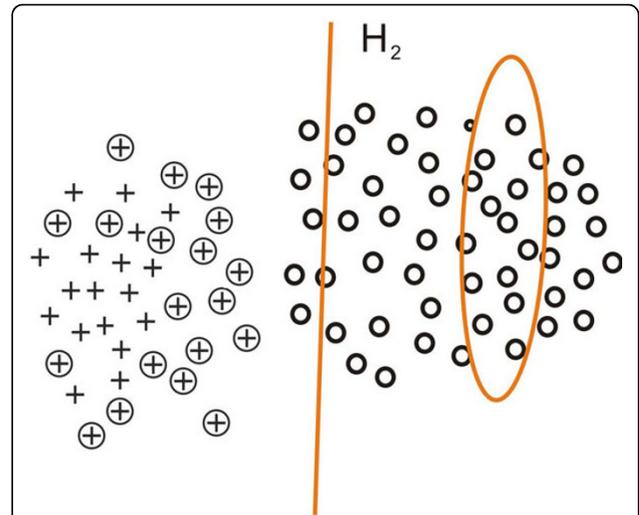
$$|U| \leq 4 * | \tilde{P} | \quad (6)$$

This process is inspired by [21] and can be intuitively demonstrated using figure 2, 3, 4, 5. In figure 2, 3, 4, 5, plus signs, plus signs with circle and circles denote positive examples, potential positive examples and unlabeled examples respectively. The points covered by ellipse are negative examples set  $N_i$  and the line is classification hyperplane. Figure 2 demonstrates how one-class SVM extracts initial negative examples. Figure 3, 4, 5 illustrate three iterations in which the classification hyperplane moves towards positive examples set.

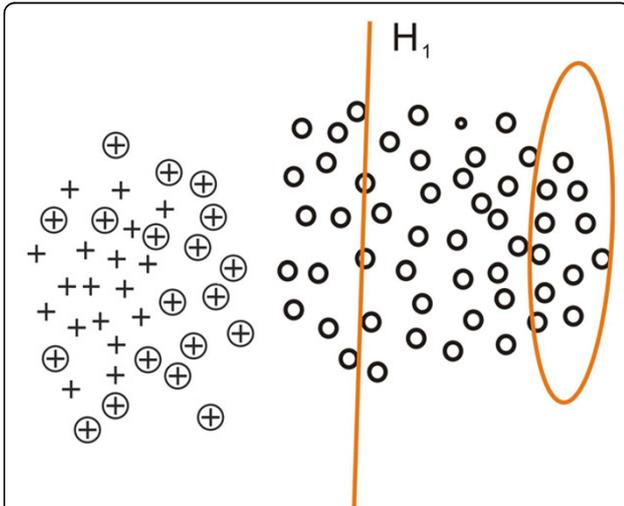
Step 3, selecting the best representative negative set. The representative negative set should have best classification performance, therefore, we use each SVM from  $C_s$  to classify a validation set  $V$  that is selected randomly form  $\tilde{P}$  and  $U$  with 10 percent of total respectively at the start of algorithm. The discrimination ability of the trained classifiers is evaluated with  $F1$ . Accordingly, the negative set corresponding to the best classifier is returned as the representative negative samples  $RN$ . In stead PSoL selects final classifier to classify remaining examples, our algorithm selects best classifier according to classification performance on validation set and corresponding negative set is regarded as representative



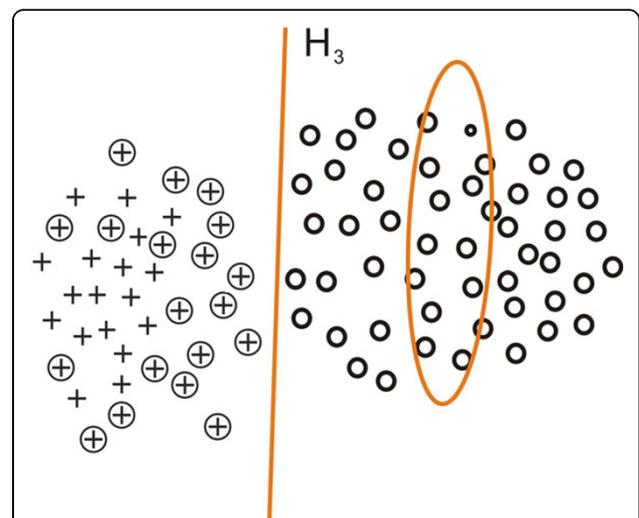
**Fig. 2 One-class SVM extracts initial negative example set** In figure 2, plus signs, plus signs with circle and circles denote positive examples, potential positive examples and unlabeled examples respectively. The points covered by ellipse are negative examples set  $N_0$  and the line is classification hyperplane. One-class SVM is utilized to extract the initial negative examples. Give a percentage of negative examples, such as 10 percent, it can draw an initial decision boundary to cover most of the positive and unlabeled data. The data points not covered by the decision boundary can be regarded as negative data points because these data points are far from the major positive set.



**Fig. 4 The negative example set  $N_2$  is obtained with the second iteration** In figure 4, plus signs, plus signs with circle and circles denote positive examples, potential positive examples and unlabeled examples respectively. The points covered by ellipse are negative example set  $N_2$  and the line is classification hyperplane. With the new training set  $\tilde{p} \cup N_1$ , the SVM classifier  $C_1$  is learned, the negative example set  $N_2$  consists of the support vectors of  $C_1$  and the unlabeled examples predicted as negative examples.



**Fig. 3 The first iteration in which the negative example set  $N_1$  is obtained by moving the classification hyperplane towards positive example set** In figure 3, plus signs, plus signs with circle and circles denote positive examples, potential positive examples and unlabeled examples respectively. The points covered by ellipse are negative examples set  $N_1$  and the line is classification hyperplane. With the positive examples  $\tilde{p}$  and initial negative example set  $N_0$ , the SVM classifier  $C_0$  is learned, the negative example set  $N_1$  consists of the support vectors of  $C_0$  and the unlabeled examples predicted as negative examples.



**Fig. 5 Obtaining the negative example set  $N_3$  with the more iteration** In figure 5, plus signs, plus signs with circle and circles denote positive examples, potential positive examples and unlabeled examples respectively. The points covered by ellipse are negative example set  $N_3$  and the line is classification hyperplane. With the new negative example set  $N_2$ , the SVM classifier  $C_2$  is learned, the negative example set  $N_3$  consists of the support vectors of  $C_2$  and the unlabeled examples predicted as negative examples. The process can proceed until the condition  $|U| \leq 4 * |\tilde{p}|$  is met.

negative set, final classifier is trained in the third stage. The algorithm for extracting representative negative examples is displayed in Algorithm 2.

#### Algorithm 2

Algorithm selecting representative negative examples

- 1: **function** SELNEGEXPS( $\tilde{p}$ ,  $U$ )
- 2: randomly selecting 10 percent of  $\tilde{p}$  and  $U$  respectively as validation set  $V$
- 3:  $\tilde{p}' = \tilde{p} - V$ ,  $U' = U - V$ ;
- 4: *identifying the initial reliable negative examples*
- 5: Training one-class SVM classifier  $C_0$  based on  $\tilde{p}'$  and  $U'$ ;
- 6: Classify  $U'$  using  $C_0$ . The predicted negative set  $N_1$  is used as the initial negative training set
- 7:  $U' = U' - N_1$
- 8: *Training iteratively SVMs.*
- 9: Classifier set  $C_s$ ;
- 10: negative set  $N_s$
- 11:  $i = 1$
- 12: **while**  $|U'| \geq 4 * |\tilde{p}'|$  **do**
- 13: Training two-class SVM classifier  $C_i$  based on  $\tilde{p}'$  and  $N_i$ ;
- 14:  $C_s(i) = C_i$ ,  $N_s(i) = N_1$ ;
- 15: Classify  $U'$  by  $C_i$ ,  $N_2$  is the predicted reliable negative set, where  $|N_2| \leq m|\tilde{p}'|$ ;
- 16:  $N_1 = N_2 + N_{SV}$ , where  $N_{SV}$  is the negative SVs of  $C_i$
- 17:  $U' = U' - N_2$ .
- 18:  $i = i + 1$ ;
- 19: **end while**
- 20: *selecting representative negative examples set*
- 21: **for all**  $C \in C_{sdo}$
- 22: computing the F1 on  $V$
- 23: **end for**
- 24: return  $RN$  from  $N_s$  with maximum F1
- 25: **end function**

#### Training the SVM for predicting genes function

After enlarging the positive examples set and extracting the representative negative examples, we merge these two kinds of examples into a training set  $\tilde{p} \cup RN$ . A SVM classifier with RBF kernel is trained on it. Grid-search technique [22] is used to search the optimal parameters  $c$  and  $g$  and an optimal SVM classifier can be successfully obtained.

#### Acknowledgements

This work is supported in part by the National Science Foundation NSF CCF 0905291, NSF IIS 0448023, NSF IIP 0934197, Natural Science Foundation in China with grant No 60573057 "Research on Key Technology of Data Mining" and grant No 90920005 "Chinese Language Semantic Knowledge Acquisition and Semantic Computational Model Study". We would like to thank Prof. Chih-Jen Lin from National Taiwan University for using SVM matlab toolkit. Publication of this supplement was made possible with

support from the International Society of Intelligent Biological Medicine (ISIBM).

This article has been published as part of *BMC Genomics* Volume 11 Supplement 2, 2010: Proceedings of the 2009 International Conference on Bioinformatics & Computational Biology (BioComp 2009). The full contents of the supplement are available online at <http://www.biomedcentral.com/1471-2164/11?issue=52>.

#### Author details

<sup>1</sup>Computer School of National University of Defense Technology, Changsha, Hunan, China. <sup>2</sup>School of Computer Science and Engineering BeiHang University, Beijing, China. <sup>3</sup>College of Information Science and Technology, Hunan Agricultural University, Changsha, China. <sup>4</sup>College of Information Engineering, Shanghai Maritime University, Shanghai, China. <sup>5</sup>College of Information Science and Technology, Drexel University, Philadelphia, PA, 19104, USA.

#### Authors' contributions

Zhoujun Li, Xiaofeng Wang and Jiali Feng conceived this research and applied project. Xiaohua Hu put forward some suggestion about two algorithms and revised the manuscript. Yiming Chen designed and coded algorithms, wrote the manuscript. All authors read and approved the final manuscript.

#### Competing interests

The authors declare that they have no competing interests.

Published: 2 November 2010

#### References

1. Ashburner M, Ball C, Blake J, Botstein D: **Gene Ontology: tool for the unification of biology.** The Gene Ontology Consortium. *Nat. Genet.* 2000, **25**:25-29.
2. Zhou X, Kao M, Wong W: **From the Cover: Transitive function annotation by shortest-path analysis of gene expression data.** *Proc. Natl. Acad. Sci.* 2002, **99**(20):12783-12788.
3. Chua H, Sung W, Wong I: **Exploiting indirect neighbours and topological weight to predict protein function from protein-protein interactions.** *Bioinformatics* 2006, **22**(13):1623-1630.
4. Lanckriet G, Deng M, Cristianini M, Jordan M, Noble W: **Kernel-based data fusion and its application to protein function prediction in yeast.** *In Bioinformatics, Pac Symp Biocomput* 2004, 300-11.
5. Barutcuoglu Z, Schapire RE, Troyanskaya OG: **Hierarchical multi-label prediction of gene function.** *Bioinformatics* 2006, **22**(7):830-836.
6. Carter RJ, Dubchak I, Holbrook SR: **A computational approach to identify genes for functional RNAs in genomic sequences.** *Nucleic Acids Research* 2001, **29**(19):3928-3938.
7. Wang C, Ding C, Meraz RF, Holbrook SR: **PSoL: a positive sample only learning algorithm for finding non-coding RNA genes.** *Bioinformatics* 2006, **22**(21):2590-2596.
8. Japkowicz N, Stephen S: **The class imbalance problem: A systematic study.** *Intelligent Data Analysis* 2002, **6**:429-449.
9. Guan Y, Myers CL, Hess DC, Barutcuoglu Z, Caudy AA, Troyanskaya OG: **Predicting gene function in a hierarchical context with an ensemble of classifiers.** *Genome Biology* 2008, **9**:S3.
10. SGD project: **"Saccharomyces Genome Database".** 2007.
11. Smith TF, Waterman MS: **Identification of common molecular subsequences.** *J Mol Biol* 1981, **147**:195-197.
12. Demeter J, Beauheim C, Gollub J, Hernandez-Boussard T, Jin H, Maier D, Matese J, Nitzberg M, Wymore F, Zachariah Z, Brown P, Sherlock G, Ball CA: **The Stanford Microarray Database: implementation of new analysis tools and open source release of software.** *Nucleic Acids Res. (Database Issue)* 2007, **1**(35):D766-770.
13. Troyanskaya Oa: **Missing value estimation methods for DNA microarrays.** *Bioinformatics* 2001, **17**:520-525.
14. Stark C, Breitkreutz B, Reguly T, Boucher L, Breitkreutz A, Tyers M: **Biogrid: A General Repository for Interaction Datasets.** *Nucleic Acids Res.* 2006, **34**(D):535-539.
15. Kondor RI, Lafferty J: **Diffusion Kernels on Graphs and Other Discrete Input Spaces.** *In In Proc Int Conf Machine Learning* 2002, 315-322.

16. Chang CC, Lin CJ: **LIBSVM : a library for support vector machines**. 2001 [<http://www.csie.ntu.edu.tw/~cjlin/libsvm/>], Software available at.
17. Li XL, Liu B, Ng SK: **Learning to Classify Documents with Only a Small Positive Training Set**. In *Proceedings of the 18th European conference on Machine learning* 2007.
18. Fung GPC, Yu JX, Lu H, Yu PS: **Text Classification without Negative Examples Revisited**. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*. 2006, **18**(1):6-20.
19. NVaplik V: **statistical learning theory**. JOHN WILEY & SONS INC. 1998.
20. Scholkopf B, Platt JC, Shawe-Taylor J, Smola AJ, Williamson RC: **Estimating the Support of a High-Dimensional Distribution**. *Neural Computation* 2001, **13**:1443-1471.
21. Liu B, Dai Y, Li X, Lee WS, Yu PS: **Building Text Classifiers Using Positive and Unlabeled Examples**. In *Neural Computation* Melbourne, Florida, USA: IEEE International Conference on Data Mining 2003.
22. Hsu C, Chang C, Lin C: **A Practical Guide to Support Vector Classification**. *Tech. rep.* Department of Computer Science National Taiwan University 2008.

doi:10.1186/1471-2164-11-S2-S11

**Cite this article as:** Chen *et al.*: Predicting gene function using few positive examples and unlabeled ones. *BMC Genomics* 2010 **11**(Suppl 2): S11.

**Submit your next manuscript to BioMed Central  
and take full advantage of:**

- Convenient online submission
- Thorough peer review
- No space constraints or color figure charges
- Immediate publication on acceptance
- Inclusion in PubMed, CAS, Scopus and Google Scholar
- Research which is freely available for redistribution

Submit your manuscript at  
[www.biomedcentral.com/submit](http://www.biomedcentral.com/submit)

