Computational Biology and Chemistry 76 (2018) 23–31

Contents lists available at ScienceDirect

Computational Biology and Chemistry

journal homepage: www.elsevier.com/locate/cbac

C-PUGP: A cluster-based positive unlabeled learning method for disease gene prediction and prioritization

Akram Vasighizaker, Saeed Jalili*

Computer Engineering Department, Tarbiat Modares University, Tehran, Iran

ARTICLE INFO

Keyword:
Candidate disease genes
Identification
Classification
Clustering
Semi-supervised learning
Pul

ABSTRACT

Disease gene detection is an important stage in the understanding disease processes and treatment. Some candidate disease genes are identified using many machine learning methods Although there are some differences in these methods including feature vector of genes, the method used to selecting reliable negative data (non-disease genes), and the classification method, the lack of negative data is the most significant challenge of them. Recently, candidate disease genes are identified by semi-supervised learning methods based on positive and unlabeled data. These methods are reasonably accurate and achieved more desirable results versus preceding methods. In this article, we propose a novel Positive Unlabeled (PU) learning technique based upon clustering and One-Class classification algorithm. In this regard, unlike existing methods, we make a more Reliable Negative (RN) set in three steps: (1) Clustering positive data, (2) Learning One-Class classifier models using the clusters, and (3) Selecting intersection set of negative data as the Reliable Negative set. Next, we attempt to identify and rank the candidate disease genes using a binary classifier based on support vector machine (SVM) algorithm. Experimental results indicate that the proposed method yields to the best results, that is 92.8, 93.6, and 93.1 in terms of precision, recall, and F-measure respectively. Compared to the existing methods, the increase of performances of our proposed method is 11.7 percent better than the best method in terms of F-measure. Also, results show about 6% increase in the prioritization results.

1. Introduction

The occurrence of mutation in the function of one or more human genes causes some genetic disorders. For the sake of brevity, herein, we call the genes known to cause disease “disease genes” and name the rest “unknown genes”. Since the mutation of genes is always likely to happen, and also there is a likelihood of this mutation leading to disease, there is no available conception asserts that some proteins are not involved in disease (non-disease genes). In other words, because of the incomplete knowledge of humans up to now, “non-disease genes” are not introduced in biological contexts. According to certain conditions, a gene that biologists consider as provisionally “non-disease gene” today, may be registered as a “disease gene” in the future. In fact, the purpose of the biological experiments is to diagnose “disease genes”, not “non-disease gene”. Due to the time-consuming and expensive in-lab experiments, the discovery of new “disease genes” is still challenging. Computational approaches can be useful to speed up predicting disease genes among a large number of unknown genes. After that, the predicted disease genes are presented to biologists for lab experiments.

Most of the genes that have been experimentally confirmed as associated with different disorders are useful to develop machine learning methods to predict new disease genes. Based on the guilt-by-association principal, wherein similar phenotypes have similar functionality or physically causative genes (Oliver, 2000), some methods have been developed. These methods are based on different biological data, such as protein sequence, functional annotation, gene expression profile and PPI networks. A key step in these methods is to measure the similarity between “unknown genes” and “known disease genes”.

Improvement of the candidate disease gene identification methods is based on exploring different datasets and strategies (Van Driel and Brunner, 2006). Some proposed methods try to define new similarity measure to improve the identification of disease genes, considering both microarray and protein–protein interaction data, by maximizing the relevance and functional similarity of the selected genes (Maji et al., 2017). Wu et al. (2016) try to analyze strengthens of protein interaction network. They find that compared to other genes, disease genes are weakly connected with essential genes in protein interaction network and propose a novel global distance measurement for gene prioritization. Adie et al. (2005) tried to train a Decision Tree (DT) algorithm on a variety of genomic and evolutionary features, such as evolutionary

* Corresponding author.
E-mail address: sjalili@modares.ac.ir (S. Jalili).

https://doi.org/10.1016/j.compbiolchem.2018.05.022
Received 13 September 2017; Received in revised form 14 April 2018; Accepted 27 May 2018
Available online 01 June 2018
1476-9271/

conservation, presence, closeness of paralogs in the human genome, coding sequence length, etc. Xu and Li (2006) have employed the K-nearest neighbor (KNN) classifier for K = 1 and K = 5 to predict disease genes based on the topological features in PPI networks, such as the percentages of disease genes in proteins' neighborhood, proteins' degree, etc. Smalter, et al. (2007) used support vector machines (SVM) classifier using PPI topological, evolutionary age, and sequence-derived features. Radivojac et al. (2008) applied a combination of three SVM classifiers on three types of features, protein sequence, PPI network, and protein functional information to build a classifier for gene prediction. In all aforementioned techniques, binary classification technique has been used, and the confirmed “disease genes” are used as positive and other genes, “unknown genes”, as negative set. Since negative set, which has been achieved from unknown genes, suffers from noisy data aforementioned studies cannot yield very accurate results.

The major challenge for this problem is that there are no negative data available which makes conventional supervised learning techniques inapplicable. Several techniques have tried to learn from “positive” and “unlabeled” samples. These computational methods have used unknown genes as unlabeled set (the mixed set which comprises both positive and negative instead of just the negative set). Then, have applied Positive Unlabeled learning (PU) technique on positive and unlabeled set. Therefore, they have gained better results than binary classification with the noisy negative set. Mordelet and Vert (2011) have proposed a bagging method, ProDiGe, that randomly selects subsets from unlabeled set and trains multiple classifiers using bias SVM (BSVM) to discriminate positive instances from each subset. Aggregating the individual classifications cause to achieve final classification. ProDiGe method behaves equally all the instances in random subsets. Jowkar and Mansoori (2016) present a Perceptron ensemble of graph-based positive unlabeled learning (PEGPUL).They derived a reliable set of negative data using co-training schema in order to form a two-class classification problem. Yang et al. (2012) have proposed a multi-level PU learning algorithm for disease gene identification (PUDI). They have quantified similarity between ‘positive mean vector’ and instances in unlabeled set according to the Euclidean distance. They have estimated sample likelihood of being positive or negative in the unlabeled set using random walk with restart (RWR) algorithm. So, they have partitioned samples into four sets: Reliable Negative (RN) set, Likely Positive set (LP), Likely Negative set (LN) and Weak Negative set (WN). Then, they have built a multi-level Weighted SVM (WSVM) classifier using these sets together with positive set. So, they have gained a better result than other methods. Others proposed SVDD classification method to assign genes to disease class (yes, no) and achieved better results (Yousef and Charkari, 2015).

While these two latter methods appear promising in disease gene prediction, there are some limitations. Unlike Yang et al. (2012) and Yousef and Charkari (2015) there is not a specific technique to extract reliable negative data from unlabeled data, we introduce a technique to achieve more reliable results through aggregating classifiers. Also, unlike the ProDiGe method that deals with all the instances (positive and negative) equally, we can consider especially negative instances to correctly estimate their label as much as possible.

In this paper, we present a novel method for disease gene prediction and prioritization. First, we cluster positive data and then learn One Class classifiers on each cluster. By setting the False Negative (FN) of models equal to zero and applying models on unlabeled data, we can claim that models could classify negative data accurately. We can aggregate the votes of individual classifiers on negative data and introduce intersection of outliers as final reliable negative data. By doing this, we can guarantee reliable negative data and use of them to predict disease genes (positive genes). Finally, we learn different binary classifiers using positive data and the extracted reliable negative data. A comparison between previous methods and the proposed method helps us to show that our experimental results outperform in predicting general disease genes.

Rest parts of this article are organized as follow. First, the proposed method is presented in Section 2. The section 3 presents experimental results and comparison between our results and other methods. The final section is the conclusion.

2. Materials and methods

Given that we do not have any negative genes, the first step is to extract a set of reliable negative genes, *{RN}*, from *U* by computing the similarities between the unlabeled genes in *{U}* and the positive genes in *{P}* based on the idea that those genes in *{U}* that are very dissimilar to the genes in *{P}* are likely to be reliable negatives.

This section proposes a heuristic approach based on the EM clustering algorithm, One Class classification, and binary classification. Our proposed method takes a set composed of positive and unlabeled examples as input and returns a classifier to label unlabeled set. Also, it ranks the genes based on their belonging to the positive class. It is a three-phase method. In the first phase, C-PUGP tries to extract a set of Reliable Negative sample set, *{RN},* from the unlabeled set *{U}* with respect to positive sample set *{P}* using the clustering and One Class classification technique. In the second phase, we use *{P}* as the positive training set together with *{RN}* as the negative training set, which has no intersection, to create some binary classifiers. Then, learned binary classifier is applied to the remaining unlabeled data set, a refined unlabeled set *{U-RN},* to determine their labels. Finally, in the prioritization phase, the predicted positive set (that are candidate disease genes) are ranked based on the degree of similarity of the instances with their proper class (disease or non- disease). Fig. 1 illustrates the outline of the proposed method for the prediction and prioritization of disease genes.

Since the combination of multiple methods enables us to exploit them all, it is a powerful solution to solve difficult classification

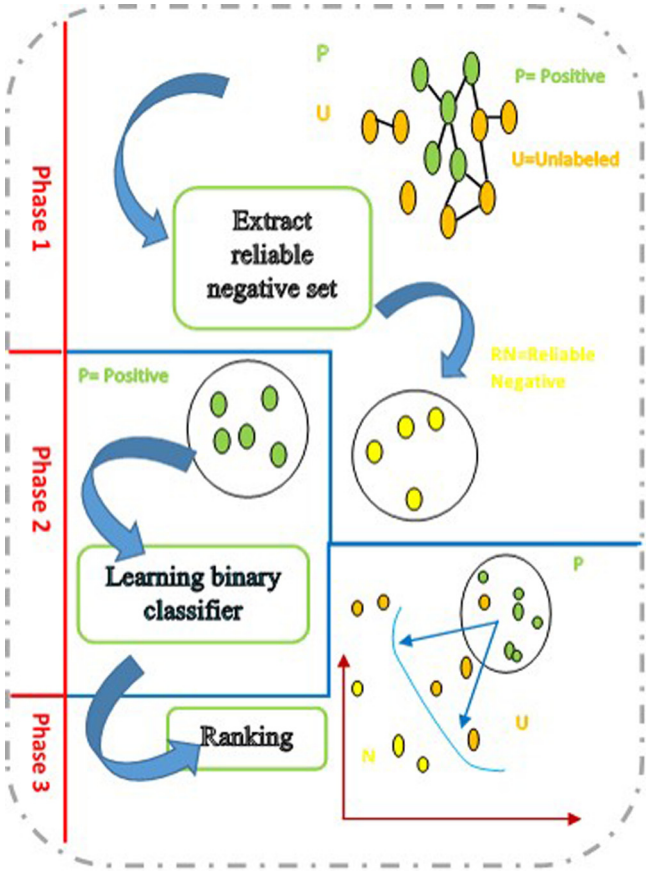

Fig. 1. Proposed C-PUGP method for ranking disease genes.

problems which contain noisy data. In order to minimize the overfitting of the identification model, 10-fold cross-validations have been carried out in all of the experiments. The rest of this section explains each of the phases in detail.

2.1. Phase 1: reliable negative data extraction

As there is no information about the negative data (non-disease gene), and for training and testing the method some negative data are required, clustering and One Class classification (OCC) method would help efficiently solve this issue by finding some more reliable negative data instead of using randomly unknown genes as negative data.

In this section, a stepwise algorithm is presented to extract reliable negative data. It is based on the dissimilarity between *{U}* and *{P}*. All the feature vectors of instances are normalized according to Min-Max formulation presented by Eq. (1):

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

Where $x' \in [0.1]$, and $x_{\min}$ and $x_{\max}$, minimum and maximum value of the features respectively, denote the normalized value of x. First, the feature set is reduced using PCA method. Then, we divide the positive set P into C clusters and learn a One-Class classifier with zero false negative rates on each cluster. Finally, we predict negative instance of $\{U\}$ using each learned One Class model and then intersect negative sets predicted by each One Class classifier. Because C-PUGP considers all positive patterns (clusters) to extract more reliable negative set, this approach is more reliable than PUDI algorithm. Fig. 2 depicts the pseudo-code of phase 1 and its overview is presented in Fig. 3. In the following paragraphs, we explain its major steps.

2.1.1. Principal component analysis (PCA)

To achieve the high level of performance of the proposed method, we try to obtain the most related and functional features from the represented feature vectors with 180 features generated by AC method. PCA is presented as dimensionality reduction methods and is employed to specify the most important features which remove the unnecessary components and keep the most of the information.

1. *Clustering positive data set, P into C clusters:*
   $$\boldsymbol{P = \{p_1, p_2, \ldots, p_C\}}$$
1. *Extracting likely negatives*
   2.1: for each cluster $p_i \in P$
   (1) Learn its correspondent OCSVM model with FN=0
   $$\boldsymbol{OCSVM\ Model, M = \{M_1, \ldots, M_c\}}$$
   2.2: for each OCSVM model Mi predict the label of members of U and save the negative members to LNi
   $$\boldsymbol{Likely\ Negative\ for\ Model_c\ (LN_c)}$$
   2.3: Make intersection of a negative labeled LNi
   $$\boldsymbol{LN = \bigcap_{i=1}^{C} \{LN_i\}}$$
1. *Return* **LN**

Fig. 2. Reliable negative data extraction. Inputs: P, Positive samples and U, Unlabeled samples.
Output: LN, Likely Negative samples\

2.1.2. Extracting clusters (patterns) of disease genes

Since disease genes have different patterns of sequence-based features (Adie et al., 2005), and there is a variety of disease genes in the positive set, we try to cluster positive gene set into some clusters such that the similarity among members of each cluster is high. Clustering helps us to separate the negative genes more accurately using separate disease gene patterns.

Finding the accurate number of clusters is an important part of the proposed method. Translate corresponding gene products (proteins) into numerical feature vector using the physicochemical descriptor, and Min-Max normalization method, which is used to normalize these properties, help using these feature vectors in algorithms and using similarity measures such as Correlation, Cosine, and Euclidean to find the distance between two genes.

In this study, we use the Expectation Maximization (EM) clustering algorithm which does not need to determine the number of clusters in advance. It is a popular iterative algorithm for maximum likelihood estimation in problems with incomplete data (Alpaydin, 2014). To find the most similar instances, we use similarity measure between their corresponding feature vectors. Since similarity measure has a high impact on finding the correct number of clusters, we investigate Correlation, Cosine, and Euclidean similarity measures. Table 1 summarizes the available distance measures. In the formulae, p and q are two points (feature vector), A_i and B_i are components of vector A and B, respectively.

Clustering is used just in order to assess the “closeness” of two points. We consider two nearest samples in P as one cluster according to the Correlation.

2.1.3. Building one class classifiers on each cluster

The OCC method makes a description of a disease gene set and has been used for outlier detection in each cluster separately. Setting the false negative of the models to zero will reduce the overall errors. One Class classifier tries to learn just positive class. One Class SVM (OCSVM) and Support Vector Data Description (SVDD) are two state-of-the-art One Class classifiers. OCSVM is a One Class learning method which was introduced by Schölkopf et al. (2001). Class of interest is called target class. The idea behind OCSVM is to describe target class using a function which maps most samples to a region where the function is non-zero. To this end, the origin is treated as the only available member of the non-target class (as an outlier). Then the problem is solved by finding a hyperplane (decision function) with the maximum distance from the origin that separates the surface region containing data from the region containing no data (as shown in Fig. 4, Left) (Muñoz-Marí et al., 2010).

The primal form of OCSVM is shown in Eq. (2):

$$\min_{w,\rho,\xi} \frac{1}{2} w^t w - \rho + \frac{1}{\nu l} \sum_{l}^{i=1} \xi i \tag{2}$$

Subject to:

$$w^t \phi(x_i) \geq (xi) \geq \rho - \xi i$$

$$\xi i \geq 0,\ i = 1,\ \ldots, l$$

Where $l \in \mathbb{N}$ is the number of observations, and ξi is slack variables that allow each example $x_i$to lie within the margin. Also, $\nu \in (0,1]$ controls the tradeoff between the maximizing the margin and l.

Also, there are other One Class SVM formulations, namely Support Vector Domain Description (SVDD). It is introduced by Tax and Duin (2004) and instead of finding a hyperplane (decision function), it finds a hypersphere with minimal radius containing only the target class

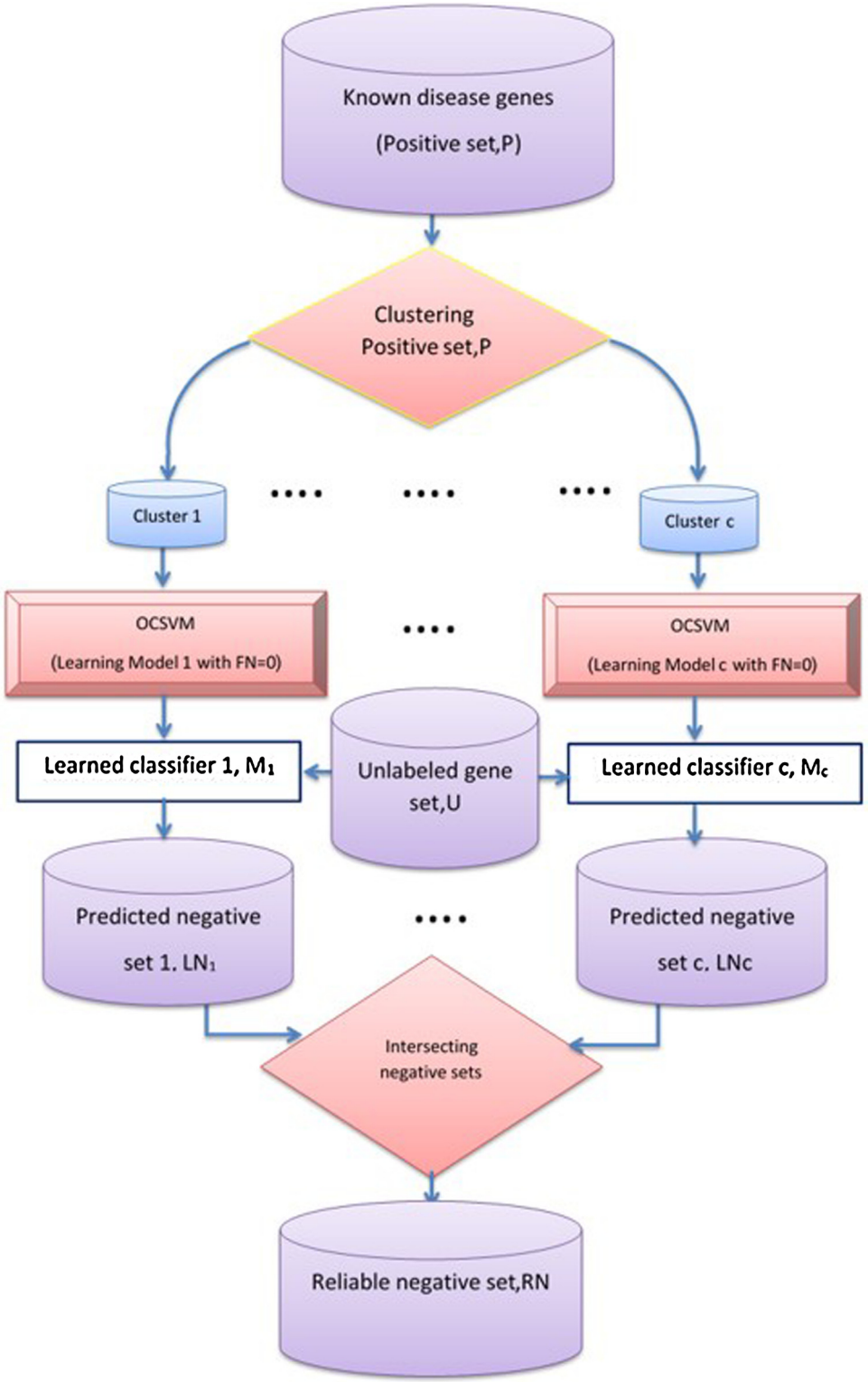

Fig. 3. Overview of the proposed method for extracting negative data.

samples. Samples lying outside are outliers (as shown in Fig. 4, Right).

It is shown that, when working with isotropic kernels, for example, Radial Basis Function (RBF) kernel and normalized data, both OCSVM and SVDD method yield the same solution usually (Schölkopf et al., 2001). Also, Tax found that the RBF kernel works better for most data sets. Indeed, because of the deficiency of SVDD technique which often needs a large data set, in particular in high dimensional feature spaces, it becomes very inefficient. On the other hand, OCSVM can extract patterns which are very hard to assign to their respective classes and some outliers were identified (Tax and Duin, 2004).

After finding M best clusters on positive set (Step 1 of Fig. 2), we learn a highly accurate set $\{M\}$ of m One Class models, $M = \{M_1 \ldots M_m\}$, where M_i is learned on its corresponding cluster i. For the reasons mentioned above, we use One Class support vector machine (OCSVM) to model disease genes. In order to train our One Class SVM, we use RBF kernel as shown in Eq. (3)

Table 1
Similarity measures formula.

Similarity measure	Formula
Euclidean	$\sqrt{\sum_{n}^{i=1}(q_i - p_i)^2}$
Cosine	$\cos(\theta) = \frac{A.B}{\|\|A\|\|.\|\|B\|\|} = \frac{\sum_{n}^{i=1} A_i.B_i}{\sqrt{\sum_{n}^{i=1} A_i^2}\sqrt{\sum_{n}^{i=1} B_i^2}}$
Correlation	$1 - \frac{(p-\bar{p})(q-\bar{q})'}{\sqrt{(p-\bar{p})(p-\bar{p})'}\sqrt{(q-\bar{q})(q-\bar{q})'}}$

$$K(X, X') = \exp\left(-\frac{X - X'^2}{2\sigma^2}\right) \quad (3)$$

Where σ^2,X and X' stand for similarity parameter, support vectors, and input feature vector, respectively.

Each model $M_l \in M$ is used to estimate the likelihood of a given instance in U being negative or positive. Given case $x \in U$, some models in M will classify it correctly, while the rest of them will misclassify. If false negative (FN) of each learned model is equal to zero, we could expect that most of the models in M will classify negative samples with a high likelihood. It takes the advantage to separate negative samples efficiently. An instance is negative if all learned models predict it as negative. Since unlabeled data set U are classified by all learned classifiers, there is a small overlap between predicted negative data of classifiers. So, the intersection of all negative data shows the majority voting of all models on negative data. In other words, each classifier makes a decision (vote) on each instance. The final decision is made based upon the great number of votes. As a result, the unlabeled set U is partitioned into two sets, RN, and $\{U - RN\}$ using all One Class models.

2.2. Phase 2- model construction and selection

PU learning method aims to build a binary classification model using positive and unlabeled data which can separate positive samples (i.e., disease gene) from unlabeled samples correctly and accurately. To this end, in the previous section, we have tried to extract some reliable negative data. In this section, the reliable negative data together with original positive data are employed to learn a robust binary classifier. We learn a set of binary classifiers using SVM, Decision Tree, Random Forest, Ridor, Naïve Bayes, and Regression techniques, then select the best classifier. Since the extracted reliable negative set size is a small fraction of the positive set size and binary classifiers usually are sensitive to imbalanced datasets, we use the SMOTE method (Chawla et al., 2002) to balance the size of two groups of samples.

2.3. Phase 3- prioritization

Given a large number of data items and a 'query', a scoring method ranks the items according to their similarity to the 'query'. Prioritizing candidate disease genes can be considered as a ranking problem. The score of candidate disease genes can be determined based on their similarity to known disease genes. Most of the existing methods define a scoring function S: $U \rightarrow \mathbb{R}$ using only positive set, to measures the similarity between samples in the unlabeled set U and positive set P (Mordelet and Vert, 2011). Various machine learning methods define this function using different scoring strategies. For example, SVM attempts to find an optimal separating hyperplane that increase the margin between the hyperplane and the nearest samples. The distance of a candidate disease gene from this hyperplane can be considered as the scoring function. We learn the scoring function using SVM classification method. To rank each predicted positive sample, we quantify its distance from the learned SVM hyperplane. The farthest point from hyperplane has the higher score, so has more likelihood of involvement in disease.

3. Results and discussion

3.1. Experimental data

In this paper, a set of known disease gene relevancy based upon a comparative study has been taken from research of Yang et al. (2012). Disease gene data was downloaded from the latest version of GENECARD (Safran et al., 2010) and OMIM (McKusick, 2007). We choose the genes with at least two-thirds non-zero features as our data. Totally 2136 disease gene products were chosen as a positive labeled set, P. Also, we use ~16 k genes downloaded from Ensembl (Flicek et al., 2011) as the unknown gene set from which we choose an unlabeled set, U, contains 3174 unknown gene products. Since not all the genes possess all features in the current data source.

The dataset consists of some features extracted from protein sequences. Information drowns from protein sequences are universal than some other information about the genes (Guo et al., 2008) and could be the useful feature of genes to predict potential disease genes. In this study, corresponding gene products have been characterized based on protein sequences.

Many representation methods have been introduced to extract the information from the amino acid in the protein sequence, including autocovariance (AC), Normalized Moreau-Broto autocorrelation (NA), Geary autocorrelation (GA), and Moran autocorrelation (MA). In this study, we follow the method outlined by Guo (Safran et al., 2010) as a representation method to extract the significant features of genes in which they are fully encoded using physicochemical properties of the

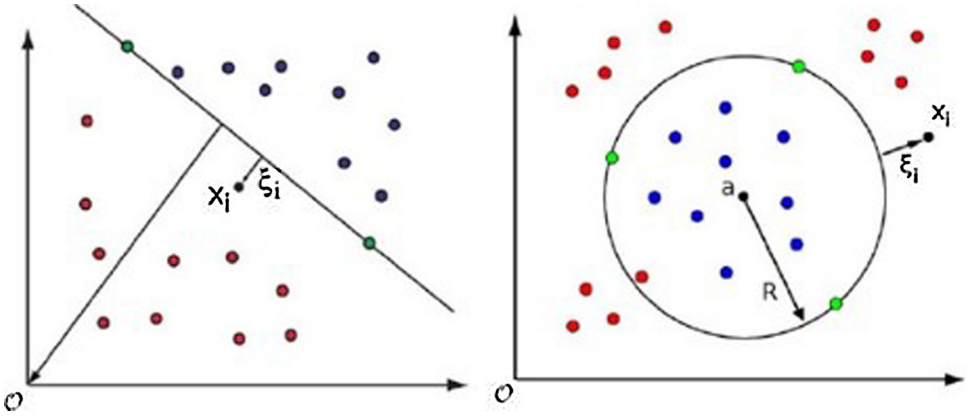

Fig. 4. (Left) OCSVM: The hyper-plane separates with maximum margin all target data from the origin by mapping all targets to the upper side of the hyper-plane and outliers to the lower side. (Right) SVDD: The hyper-sphere surrounds all target samples with the center, a, and radius of the hyper-sphere, R.

Table 2
Physicochemical properties of amino acid that used as feature.

	Feature Name	Abbreviation
1	Entropy Of Formation	EOF
2	Partition Coefficient	PC
3	POLarity	POL
4	Amino Acid Composition	AAC
5	Residue Accesible Surface area in tripeptide	RAS
6	Transfer Free Energy	TFE
7	CC in regression analysis	CC
8	Hydro PHILicity	HY-PHIL
9	POLarizability	POL2
10	Hydro PHOBicity	HY-PHOB
11	Solvation Free Energy	SFE
12	Graph Shape Index	GSI

amino acid.

Each protein sequence was translated into some vectors with each amino acid represented by the normalized values of some descriptors.

At first, 12 physicochemical properties of amino acids are employed to present the amino acid characteristics in this work. Table 2 lists these features.

Since by adding one more physicochemical property, 30 features will be added to the feature vector, we try to select which physicochemical property has more effect on the identification of disease gene. We select the more effective physicochemical properties with the minimum number of these properties to avoid complexity.

In this paper, 6 physicochemical properties of amino acid which were used in many applications have been selected. These properties are residue accessible surface area in tripeptide (RAS), polarizability (POL2), hydrophilicity (HY-PHIL), hydrophobicity (HY-PHOB), and polarity (POL) solvation free energy (SFE), respectively. Finally, we obtain $30 \times 6 = 180$ features for each feature vector. Then, to improve performance by reducing the noise of data it is reduced to 66 features with PCA method. The original values of these physicochemical properties for each amino acid were normalized using Min-Max normalization method.

3.2. Evaluation method

To investigate how the proposed method can improve the performance of disease gene identification compared with the state-of-the-art methods, we implement 10-fold cross-validation and leave-one-out cross-validation (LOOCV) test strategies.

A stationary benchmark dataset is composed of positive and reliable negative samples with equal size. In 10-fold cross validation, data is divided into 10 equal partitions. During each cross-validation fold, a different partition is put aside for test and the remaining partitions are used for training. It repeats 10 times and takes the mean of metrics. Then, predicted positive samples are ranked.

We use LOOCV strategy to report the prioritization result. In this test strategy, a training set and a test set is generated in every LOOCV run. First, a positive sample of disease genes is removed (defector gene) and inserted into the 99 non-disease genes (that are randomly selected from extracted reliable negative) as the test set. Remaining of genes (positive and reliable negative with equal size) will be used to construct the training set. Then, the score of defector gene is determined using learner. This process is repeated for each positive sample. Those cases that have been scored greater than a specific threshold are considered as positive, and others as negative.

3.2.1. Performance evaluation metrics

In binary classification algorithms, the confusion matrix is normally used to estimate a classifier performance criteria. Each element of this matrix is explained as follow:

Table 3
Representation of confusion matrix.

Predicted Label		
Real Label	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

Table 4
Performance criteria formulation.

Measure	Formula
Precision	$\boldsymbol{p} = \frac{TP}{TP+FP} \times 100$
Recall	$\boldsymbol{r} = \frac{TP}{TP+FN} \times 100$
F1 − measure	$[(2 \times \boldsymbol{p} \times \boldsymbol{r})/(\boldsymbol{p} + \boldsymbol{r})] \times 100$

- TP: The number of positive cases correctly classified.
- TN: The number of negative cases correctly classified.
- FP: The number of misclassified negative cases.
- FN: The number of misclassified positive cases.

The confusion matrix is defined in Table 3, and measures that are calculated from the confusion matrix are presented in Table 4. These measures are defined as follows:

- Precision: Percentage of positive predictions were performed correctly.
- Recall: Percentage of positive samples were correctly classified.
- F-measure: Harmonic mean of precision and recall. F-measure value is large when both precision and recall are high, and small when either of them is small. This is suitable for our objective of predicting disease genes if either precision or recall is small, F-measure shows that.

Also, we compute Matthew correlation coefficient (MCC) according to Eq. (4):

$$TP \times TN - FP \times FN / \sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)} \quad (4)$$

An alternative interpretation of AUC is that it gives the probability that a randomly selected positive is ranked above a randomly selected negative. Thus, AUC is 1 if and only if all positives are ranked above all negatives. However, it is still possible to obtain sensitivity or specificity under 1 by selecting a suboptimal cutoff.

As mentioned, LOOCV method is used to evaluate the outcome of prioritization phase. In LOOCV evaluation criteria such as Sensitivity and Specificity can be calculated by setting a specific threshold and plot ROC curve using these two measures. These two criteria are defined in Table 5:

- Sensitivity (Recall): From the total run of LOOCV, the rate of times which defector gene is located above a specific threshold in the prioritization list. This threshold can be either 1% or 10% of first 10.
- Specificity: Percentage of genes that are ranked below the threshold.

Table 5
Calculation of sensitivity and specificity.

Measure	Formula
Sensitivity	$\frac{TP}{TP+FN}$
Specificity	$\frac{TN}{TN+FP}$

For example, if threshold in a list of 100 genes would be equal to k = 10, this measure will be equal to 90.

By changing the threshold from 1% to 100% with the step increment of (scale) 1%, the ROC curve can be drawn. The area under this curve, AUC, is a standard criterion for computing the efficiency of the algorithm. For example, AUC = 100% indicates the defector gene is located at the first place in all prioritizations. While AUC = 50% means the defector genes are randomly ranked. Predicted genes at the top of prioritization list are more likely to be associated with the disease.

3.3. Environment and tools

In this study, MATLAB Version 8 (R2013a) and WEKA version 3.7 are used to develop and test the proposed method. We use LIBSVM library (Chang and Lin, 2011) and implemented OCSVM classifier included in the MATLAB. LIBSVM is an integrated tool for handling One Class SVM. We employ MATLAB to implement PCA. Also, WEKA is used to apply SMOTE method and EM algorithm in order to balance the training data and cluster positive data respectively. We use a computer with an Intel Core TM i5 processor, 2.4GH speed, and 4G internal memory. The Windows7 operating system was also used in this computer.

3.4. Evaluation of extracted reliable negative genes

3.4.1. Result of clustering

The version of EM clustering algorithm (called Simple EM in WEKA) that uses cross-validation method is employed to get the correct number of clusters in positive set. Experimental results show that among the three similarity measures, Correlation is the best measure in this study. The result of EM clustering is presented in Table 6.

According to Table 6 and after merging clusters which have the rate of 0% and 1%, we get 11 clusters finally.

3.4.2. Results of one class classification

We learn separate OCSVM using RBF kernel on each cluster in which false negative rate is set to zero. A drawback of OCSVM method is that its results are sensitive to free parameters which are difficult to tune (Schölkopf et al., 2001). Two parameters that should be tuned in OCSVM algorithm are the RBF kernel width, γ, and the fraction of rejection of OCSVM classifier, ν. To achieve the best performance, the selection of parameters was performed by grid search where we measure the cross-validation accuracy of the classifier over all combinations of input parameters. Accordingly, ν value was set to 0.004, and γ value was set to 0.02. We learn 11 One Class models according to obtained values for ν and γ corresponding to 11 clusters as shown in Table 6.

Table 6
The data rate of each cluster in the clustering of positive samples by EM algorithm.

Cluster Number	Rate of samples (%)	Number of samples
13	8	161
12	5	107
11	8	166
10	18	375
9	0	8
8	5	98
7	9	192
6	1	24
5	1	25
4	2	48
3	0	7
2	6	124
1	22	47
0	15	33

Table 7
Performance of different classifiers where10-fold cross validation is done, and SMOTE is applied (results of Weka and Matlab).

	Precision	Recall	F-measure	MCC
SVM (Weka)	99.8	93.6	96.6	95.1
Random Forest (Weka)	96.4	96.1	96.2	94.6
LR (Weka)	95.9	95.9	95.9	91.7
Naive Bayes (Weka)	94.4	93.6	93.6	88
DT (Weka)	92.5	92.5	92.5	85
Ridor (Weka)	91.3	91.2	91.2	82
SVM (Matlab)	92.8	93.6	93.1	80.7
Random Forest (Matlab)	91.3	90	90.6	79.8
LR (Matlab)	90.4	89.4	89.8	77.1
Naive Bayes (Matlab)	89.7	88.8	89.2	76.5
DT (Matlab)	87.2	87.9	87.5	74.2
Ridor (Matlab)	86.8	86.5	86.6	69.7

Then, we apply learned models on unlabeled set. Finally, we separate outliers (negatives) of models and considered negative samples of all models (intersection of all negative samples from all models) as the reliable negative set (Steps 2.2 and 3 of Fig. 2). As a result, we obtain 123 reliable negatives from unlabeled data.

3.5. Results of binary classification

We learn multiple binary classifiers using original positive (P) and reliable negative data (RN). We investigate the performance of binary SVM, Decision Tree (DT), Random Forest, Ridor, Naïve Bayes, and Linear Regression (LR) classifiers on the balanced and imbalanced dataset. SMOTE technique is applied to balance {P ∪ RN}.We investigate the effect of SMOTE on the performance of different classifiers.

Results indicate that SVM classifier has the highest performance in all balanced data sets. In SVM method, we can control the tradeoff between the complexity of decision function and frequency of errors by tuning the value of the parameter C. We employ grid search to achieve the best performance. Accordingly, C and γ are set to 0.1 and 0.2 respectively. The final results of classifiers are presented in Table 7.

3.6. Prioritization results

We investigate AUC value using some datasets with different sample size (shown in the first column of Table 8). To this end, we learn multiple scoring functions using binary SVM classification method. Also, we use extracted reliable negative data to evaluate the prioritization results. Table 8 presents the AUC values for different sized data, and Fig. 5 illustrates their corresponding ROC diagrams.

We evaluate the performance of our proposed method in term of TPR for the rank threshold K between $1 \leq k \leq 100$.

The results indicate that the average AUC score is 85.2% and the maximum AUC score is 90.5% for 400 samples.

Table 8
AUC value for prioritization of different introduced data sets.

Sample Size	AUC Value (%)
100	85.4
200	86.4
300	88.2
400	90.5
500	83.3
2000	86.9
Average	85.2

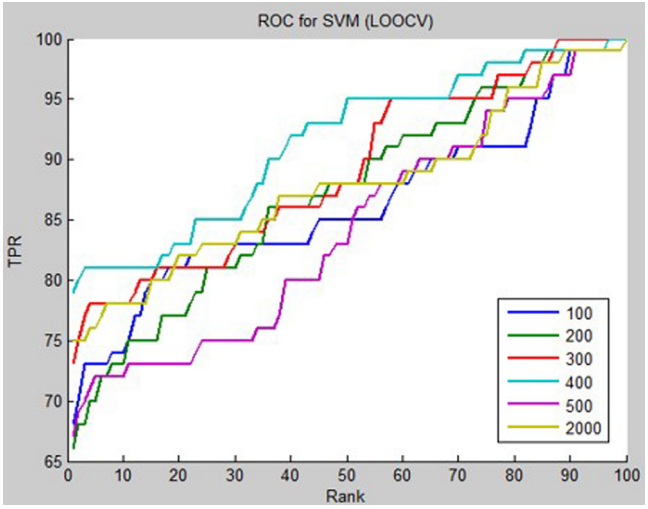

Fig. 5. ROC diagram for prioritization of different introduced data sets.

Table 9
Comparison between C-PUGP and five state-of-the-art techniques based on Precision, Recall and F-measures.

Method	Precision *p* (%)	Recall *r* (%)	F_measure *r* (%)
C-PUGP	92.8	93.6	93.1
Yousef's Method (Yousef and Charkari, 2015)	79.9	83.1	81.4
PUDI (Smalter et al., 2007)	73.2	82	77.3
ProDiGe (Xu and Li, 2006)	71.2	77	73.9
Smalter's method (Wu et al., 2016)	68.3	62.1	65
Xu's method (5NN) (Maji et al., 2017)	67.2	57.3	61.8
Xu's method (1NN) (Maji et al., 2017)	68.8	55	61.1

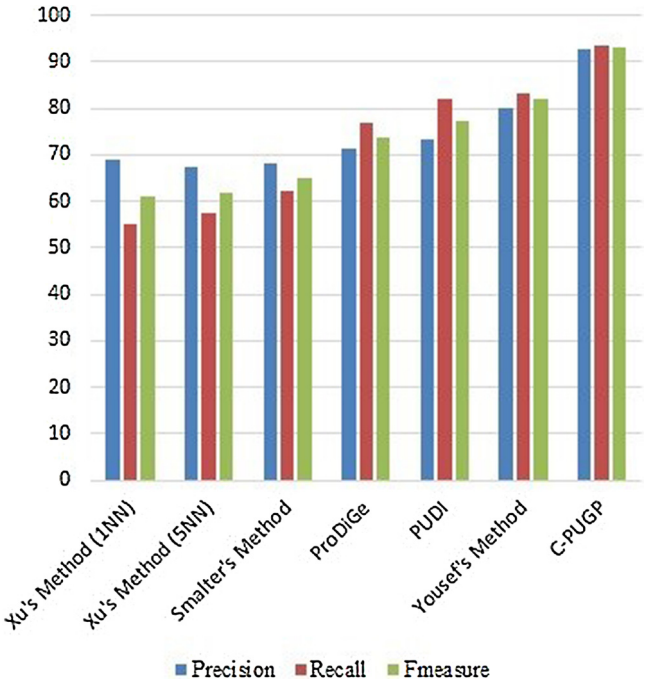

Fig. 6. Comparison between C-PUGP and five state-of-the-art techniques based on Precision, Recall and F-measures.

Table 10
Recall (TPR) of different methods at different rank levels.

	Top10	Top 10%
MKL1class (Zhu, 2005)	25.3	59.9
ProDiGe1 (Mordelet and Vert, 2011)	**27.8**	71.2
ProDiGe4 (Mordelet and Vert, 2011)	14.6	**78.4**
PRINCE (Vanunu et al., 2010)	6.8	65.4
C-PUGP	**34.1**	**85.2**

Table 11
Comparison between the results of proposed method with and without PCA extracted features.

Num. of features	Precision	Recall	F-measure
53	88.9	87.7	88.2
57	90.3	91.6	90.9
59	91.1	92.2	91.6
66	**92.8**	**93.6**	**93.1**
81	88.1	90.5	89.2
93	84.8	88.4	86.5
180 (without PCA)	79.8	85.1	82.3

Table 12
Performance of SVM classifiers with different positive sample size where10-fold cross validation is done, and SMOTE is applied (results of Matlab).

MCC	F-measure	Recall	Precision	Positive sample size
82.9	93.9	93.9	94.0	20%
83.1	93.9	93.9	94.1	40%
84.5	94.6	94.6	94.9	60%
84.2	94.4	94.5	94.8	80%
80.3	92.9	93.3	92.7	100%

3.7. Further testing: evaluation of the given method with simulated data

As mentioned earlier in Section 2, our proposed method takes a set composed of positive and unlabeled examples as input and returns a classifier to label unlabeled set. Also, it ranks the genes based on their relevance to the positive class. We use {*P*} as the positive training set together with {*RN*} as an expected negative training set, which has no intersection, to create some binary classifiers. Then, the learned binary classifier is applied to the remaining unlabeled data set, a refined unlabeled set {*U-RN*}, to predict their labels. The final results of classifiers are presented in Table 7.

Moreover, we investigate the effectiveness of SMOTE on the performance of different classifiers. Results indicate that SVM classifier has the highest performance in all balanced data sets. Also, in this experiment, SMOTE technique is applied to balance {P ∪ RN}.

Synthetic Minority Over-sampling TEchnique (SMOTE) interpolates existing samples to generate new instances. Instead of merely replicating existing observations, the technique generates artificial samples. We use the data mining tool Weka in order to run SMOTE technique to generate some artificial positive data (i.e., disease genes). Although SMOTE assumes a balanced class distribution and the original class distribution is preserved, these generated genes are simulated and differ from those of real data. Random oversampling (SMOTE) randomly duplicates minority class instances until the desired class distribution is reached. Indeed, we apply this technique to investigate well the given algorithm and to validate the proposed method.

To extensively investigate the effectiveness of the proposed method, we choose randomly some sets contains 20%, 40%, 60%, and 80% of gene products separately, as a positive set, among 2136 disease gene products. As a stationary benchmark dataset, it is composed of positive

and reliable negative samples with equal size (i.e., we also apply SMOTE method on these data sets to balance the number of positive and negative genes).We learn multiple classifiers using these data sets and binary SVM classification method. The results of the evaluation are presented in Table 12. As the results indicate, there were no significant changes in this experiment compared to the previous experiments.

As we apply our proposed method to unrealistic data (i.e., the simulated data) yields no meaningful results, therefore this type of simulation experiment is not recommended in such real-world problems.

3.8. Comparing the results with that of other works

We compare the findings of our proposed method with those of five state-of-the-art methods, namely Yousef's method, PUDI, ProDiGe, Smalter's method, and Xu's method for predicting general disease genes, i.e. classifying an unknown gene into a disease gene or a non-disease one. We employ 10-fold cross-validation and all the six methods above use the same groups of training and test set for fair evaluation. The comparison of the results is presented in Table 9 and depicted in Fig. 6.

According to Table 9, our proposed method yields to the best results in terms of precision, recall, and F-measure. The increase of performances is 11.7, 12.9, and 10.5 percent better than best method, Yousef's method, in terms of F-measure, precision, and recall, respectively. Besides, the difference between precision and recall, i.e. p-r, for C-PUGP and PUDI are 0.8 and 8.7 respectively. It means that C-PUGP is an even-break point support method (i.e., precision and recall have almost equal values). It shows C-PUGP do not sacrifice precision in favor of recall and conversely.

In order to perform a comparison between prioritization methods, we refer to the latest studies which use positive and unlabeled data. Results are presented in Table 10 in term of TPR measure.

As shown in Table 10, maximum values reported by the previous methods are 78.4% and 27.8%, at level 10% and top 10 respectively. While the proposed method reports average value 85.2% and 34.1% at level 10% and top 10, respectively. The results reveal significant increase about 6% in top 10, and about 7% at level 10% (Table 11).

The main difference between the proposed method and the related ones is the classification method used for disease gene identification problem. Some of the previous methods consider this problem as two-class classification problem which uses randomly unknown genes as negative instances. While in this work, the disease gene identification problem is treated as positive unlabeled learning problem which is trained and tested using reliable negative data extracted from the unknown set. Moreover, the difference between our reliable negative data and PUDI method (Yang et al., 2012) is that, there is no guarantee about the negative instances which are extracted from unknown genes in PUDI, but we try to find more reliable negative using combination advantages of multiple methods such as clustering (to separate different disease patterns) and One Class classification with zero false negative rates (to identify voting of all classifiers on outliers as negative set).

4. Conclusion

Based on the rule that genes which are associated with the similar disorders are likely to have similar features, some machine learning methods have been employed to predict novel disease gene. These methods focus on capturing different gene properties. Moreover, traditional methods typically have built a binary classification model using disease genes as positive training set and unknown genes as the negative training set. The negative set contains noisy data because the unknown gene set contains some positive ones. Therefore, such methods are not very effective. The absence of negative data is a challenging issue for disease gene prediction. Positive-Unlabeled (PU) learning techniques can remove with this inherent limitation.

In this paper, we use PU learning method to classify and rank candidate disease genes. We propose a novel method called C-PUGP based on protein sequence data. Extracting the reliable negative set with high probability from unlabeled data is the main idea and novelty of the proposed method. First, positive data are divided into some clusters. Then, One-Class classifier method is applied to each positive cluster. Therefore, each learned One-Class classifier can attribute a negative/positive label to unlabeled data. The intersection of negative data of all One-Class models is considered as reliable negative (RN), and together with original positive data are used to learn a binary classifier. The learned classifier is used to prioritize candidate disease genes between remained unlabeled genes, i.e. $\{U - RN\}$ according to the likelihood of membership in the positive class and their distance from the learned hyper-plain by SVM. Compared with other methods, C-PUGP achieves better performance (i.e., 11.7 percent in term of F1-measure).

References

Adie, E.A., Adams, R.R., Evans, K.L., Porteous, D.J., Pickard, B.S., 2005. BMC Bioinf. 6, 55.
Alpaydin, E., 2014. Introduction to Machine Learning. MIT press.
Chang, C.-C., Lin, C.-J., 2011. ACM Trans. Intell. Syst. Technol. (TIST) 2, 27.
Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. J. Artif. Iintell. Res. 16, 321–357.
Flicek, P., Amode, M.R., Barrell, D., Beal, K., Brent, S., Carvalho-Silva, D., Clapham, P., Coates, G., Fairley, S., Fitzgerald, S., 2011. Nucleic Acids Res. 40, D84–D90.
Guo, Y., Yu, L., Wen, Z., Li, M., 2008. Nucleic Acids Res. 36, 3025–3030.
Jowkar, G.-H., Mansoori, E.G., 2016. Comput. Biol. Chem. 64, 263–270.
Maji, P., Shah, E., Paul, S., 2017. Inf. Sci. 384, 110–125.
McKusick, V.A., 2007. Am. J. Hum. Genet. 80, 588.
Mordelet, F., Vert, J.-P., 2011. BMC Bioinf. 12, 389.
Muñoz-Marí, J., Bovolo, F., Gómez-Chova, L., Bruzzone, L., Camp-Valls, G., 2010. IEEE Trans. Geosci. Remote Sens. 48, 3188–3197.
Oliver, S., 2000. Nature 403, 601.
Radivojac, P., Peng, K., Clark, W.T., Peters, B.J., Mohan, A., Boyle, S.M., Mooney, S.D., 2008. Proteins: Struct. Funct. Bioinf. 72, 1030–1037.
Safran, M., Dalah, I., Alexander, J., Rosen, N., Iny Stein, T., Shmoish, M., Nativ, N., Bahir, I., Doniger, T., Krug, H., 2010. Database 2010.
Schölkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J., Williamson, R.C., 2001. Neural Comput. 13, 1443–1471.
Smalter, A., Lei, S.F., Chen, X.-W., 2007. Bioinformatics and biomedicine. BIBM 2007. IEEE International Conference On, IEEE, 2007 209–216.
Tax, D.M., Duin, R.P., 2004. Mach. Learn. 54, 45–66.
Van Driel, M.A., Brunner, H.G., 2006. Hum. Genomics 2, 429.
Vanunu, O., Magger, O., Ruppin, E., Shlomi, T., Sharan, R., 2010. PLoS Comput. Biol. 6.
Wu, S., Shao, F., Zhang, Q., Ji, J., Xu, S., Sun, R., Sun, G., Du, X., Sui, Y., 2016. Physica A 461, 262–269.
Xu, J., Li, Y., 2006. Bioinformatics 22, 2800–2805.
Yang, P., Li, X.-L., Mei, J.-P., Kwoh, C.-K., Ng, S.-K., 2012. Bioinformatics 28, 2640–2647.
Yousef, A., Charkari, N.M., 2015. J. Biomed. Inform. 56, 300–306.
X. Zhu, (2005).