A Learning Framework to Improve Unsupervised Gene Network Inference

Turki Turki[1,2](✉), William Bassett[2], and Jason T.L. Wang[2](✉)

[1] Computer Science Department, King Abdulaziz University, P.O. Box 80221, Jeddah 21589, Saudi Arabia
tturki@kau.edu.sa

[2] New Jersey Institute of Technology Bioinformatics Program and Department of Computer Science, University Heights, Newark, NJ 07102, USA
{ttt2,wab2,wangj}@njit.edu

Abstract. Network inference through link prediction is an important data mining problem that finds many applications in computational social science and biomedicine. For example, by predicting links, i.e., regulatory relationships, between genes to infer gene regulatory networks (GRNs), computational biologists gain a better understanding of the functional elements and regulatory circuits in cells. Unsupervised methods have been widely used to infer GRNs; however, these methods often create missing and spurious links. In this paper, we propose a learning framework to improve the unsupervised methods. Given a network constructed by an unsupervised method, the proposed framework employs a graph sparsification technique for network sampling and principal component analysis for feature selection to obtain better quality training data, which guides three classifiers to predict and clean the links of the given network. The three classifiers include neural networks, random forests and support vector machines. Experimental results on several datasets demonstrate the good performance of the proposed learning framework and the classifiers used in the framework.

Keywords: Feature selection · Graph mining · Network analysis · Applications in biology and medicine

1 Introduction

Network inference through link prediction is a major research topic in computational social science [9,12,18] and biomedicine [1,6,8]. For example, computational biologists develop different methods for reconstructing gene regulatory networks (GRNs) using high throughput genomics data. Maetschke *et al.* [21] categorized the existing GRN reconstruction algorithms into three groups: unsupervised, supervised and semi-supervised. While supervised algorithms are capable of achieving the highest accuracy among all the network inference methods, these algorithms require a large number of positive and negative

P. Perner (Ed.): MLDM 2016, LNAI 9729, pp. 28–42, 2016.
DOI: 10.1007/978-3-319-41920-6_3

training examples, which are difficult to obtain in many organisms [21,24,29]. Unsupervised algorithms infer networks based solely on gene expression profiles and do not need any training example; however, the accuracy of the unsupervised algorithms is low [21]. In our previous work [24,29], we studied supervised and semi-supervised methods. Here we explore ways for improving the accuracy of unsupervised methods.

Specifically we propose a learning framework to clean the links of the GRNs inferred by unsupervised methods using time-series gene expression data. These methods include BANJO (Bayesian Network Inference with Java Objects) [36], TimeDelay-ARACNE (Algorithm for the Reconstruction of Accurate Cellular Networks) [37], tlCLR (Time-Lagged Context Likelihood of Relatedness) [10,20], DFG (Dynamic Factor Graphs) [16], BPDS (Boolean Polynomial Dynamical Systems) [30], MIDER [31], Jump3 [14], ScanBMA [35], and Inferelator [3]. BANJO models networks as a first-order Markov process; it searches through all possible networks, seeking the network with the best score. TimeDelay-ARACNE infers networks from time-series data using mutual information from information theory.

The tlCLR method also uses mutual information and depends on ordinary differential equations to model time-series data. DFG models experimental noise as a fitted Gaussian and then infers networks based on an assumed underlying, idealized gene expression pattern. Jump3 uses a non-parametric procedure based on decision trees to reconstruct GRNs. ScanBMA is a Bayesian inference method that incorporates external information to improve the accuracy of GRN inference. Inferelator uses ordinary differential equations that learn a dynamical model for each gene using time-series data. Recent extensions of Inferelator incorporate prior knowledge into the tool, and are resilient to noisy inputs.

The main drawback of the unsupervised methods is that they often create missing and spurious links [21]. The learning framework proposed here consists of several steps to clean the links. First, since an inferred network is sizable, we develop a graph sparsification technique to generate dual sample graphs, which represent significant portions of the network constructed by an unsupervised method. Graph sparsification is a technique for generating sample graphs from a large network, which speeds up the learning process from the network [2,17,23]. Next, we use principal component analysis (PCA) as a dimensionality reduction technique for feature learning. PCA is commonly used in the bioinformatics community to select important features from data [26,34]. Finally, we build three classifiers using the important features learned from the dual sample graphs and apply these classifiers to predicting and cleaning the links in the noisy network constructed by an unsupervised method. The three classifiers include neural networks, random forests and support vector machines. Like PCA, these three classifiers are commonly used in bioinformatics[32]. As a case study, we focus on Inferelator [3] in the paper and show how to use the proposed framework to predict and clean the links constructed by Inferelator, which is one of the most widely used unsupervised methods in the field.

The rest of this paper is organized as follows. Section 2 presents our learning framework, describing the techniques of graph sparsification, feature construction and principal component analysis, as well as the proposed link prediction and cleaning algorithm. Section 3 reports experimental results, comparing the three classifiers used in the framework. The results demonstrate the effectiveness of the framework, showing how it improves the accuracy of Inferelator. Section 4 concludes the paper.

2 The Learning Framework

2.1 Graph Sparsification

The input of the proposed learning framework is a weighted directed graph $G = (V, E)$ that represents the topological structure of (a subgraph of) the gene regulatory network (GRN) constructed by Inferelator based on a time series gene expression dataset. E is the set of edges or links, and V is the set of vertices or nodes in G, where each link represents a regulatory relationship and each node represents a gene. Each edge $e = (u, v) \in E$ is associated with a weight, denoted by $W(e)$, where $0 < W(e) \leq 1$.

Our graph sparsification method, named GeneProbe (reminiscent of LinkProbe [5] for social network analysis), takes as input the graph G and two genes of interest: an origin or regulator gene, and a destination or regulated gene. GeneProbe creates six sets of genes, described below, and produces as output an inference subgraph that contains all genes in the six sets and all edges in E that connect the genes in the six sets.

Two Sets of k-backbone Genes. These include one set of k-backbone hub genes and one set of k-backbone authority genes. The k-backbone hub genes include all genes whose weighted outgoing degree is greater than or equal to a user-specified positive real value $k_{hub} \in \mathbb{R}^+$. The weighted outgoing degree of a gene or node u is defined as the sum of edge weights for all outgoing edges of u. Likewise, the k-backbone authority genes include all genes whose weighted incoming degree is greater than or equal to a user-specified value $k_{authority} \in \mathbb{R}^+$. The weighted incoming degree of a node u is defined as the sum of edge weights for all incoming edges of u. (In the study presented here, $k_{hub} = 15$ and $k_{authority} = 10$.) Intuitively we select few "highly social" individuals who would represent "social hubs/authorities" for inference across geographical regions. The genes most likely to be regulators (with the largest weighted outgoing degrees) are selected as the "hubs" of the network G for inclusion in the inference subgraph. Furthermore, the genes most likely to be regulated genes (with the largest weighted incoming degrees) are selected as the "authorities" of the network G for inclusion in the inference subgraph.

Formally, let $W\text{-}out(u)$ ($W\text{-}in$(u), respectively) denote the weighted outgoing (incoming, respectively) degree of node u. Then

$$W\text{-}out(u) = \sum_{e \in E\text{-}out(u)} W(e) \tag{1}$$

$$W\text{-}in(u) = \sum_{e \in E\text{-}in(u)} W(e) \tag{2}$$

where $W(e)$ is the weight of edge e, and $E\text{-}out(u)$ ($E\text{-}in(u)$, respectively) denotes the set of edges leaving (entering, respectively) u. GeneProbe retrieves all genes $u \in G$ where $W\text{-}out(u) \geq k_{hub}$ and $W\text{-}in(u) \geq k_{authority}$.

Two Sets of d-local Genes. These include one set of d-local genes for the origin and one set of d-local genes for the destination. The d-local genes are the genes adjacent to each of the two genes of interest, i.e., the origin and destination, with incident edge weights greater than or equal to a user-specified positive real value $d \in \mathbb{R}^+$. (In the study presented here, $d = 0.95$.) Intuitively, the d-local genes represent the genes most likely to be regulated by and most likely to regulate the two genes of interest.

Two Sets of Random Walk Metropolis Genes. These include one set of random walk metropolis genes for the origin and one set of random walk metropolis genes for the destination. The random walk metropolis (RWM) genes provide a stochastic path from the genes of interest back to a k-backbone gene (if possible). The RWM does not differentiate between k-backbone hub and k-backbone authority genes. All of the genes encountered along the RWM path are added to the inference subgraph. For the origin or regulator gene, the random walk is a walk along outgoing edges towards the k-backbone, whereas the random walk for the destination or regulated gene is a backtrack to the k-backbone along incoming edges. Each step along the random walk metropolis is selected based on a randomized chance until a k-backbone gene is reached (or a maximum number of tries is exceeded).

The randomized chance at each step along the random walk for the regulator gene (i.e., origin) can be characterized as follows. Given a current gene u, we select a random edge from the list of outgoing edges of gene u. Let w represent the gene at the end of the randomly selected outgoing edge. The random walk will proceed from gene u to gene w if a randomly selected number between 0 and 1 is less than or equal to the minimum of 1 and the weighted outgoing degree of w divided by the weighted outgoing degree of u. That is, w is accepted as the next state with the probability of less than or equal to an acceptance rate α_{out}. Otherwise, another random outgoing edge of gene u is selected and similar calculations are performed. This move can be formalized in Equation (3). $P(u \rightarrow w)$ is the probability that a random walk proceeds from u to w where

$$\alpha_{out} = P(u \rightarrow w) = min\left\{1, \frac{W\text{-}out(w)}{W\text{-}out(u)}\right\} \tag{3}$$

This process is repeated until a maximum number of tries is reached (or a k-backbone gene is reached). Note that given enough chances in a connected gene regulatory network, the random walks will always reach a k-backbone gene. It logically follows that a setting that includes few k-backbone genes will likely generate many RWM genes and vice versa.

The randomized chance at each step along the random walk for the regulated gene (i.e., destination) can be characterized as follows. Given a current gene v, we select a random edge from the list of incoming edges of gene v. Let w represent the gene at the end of the randomly selected incoming edge. The random walk will backtrack from gene v to gene w if a randomly selected number between 0 and 1 is less than or equal to the minimum of 1 and the weighted incoming degree of w divided by the weighted incoming degree of v. That is, w is accepted as the next state with the probability of less than or equal to an acceptance rate α_{in}. Otherwise, another random incoming edge of gene v is selected and similar calculations are performed. This move is formalized in Equation (4).

$$\alpha_{in} = P(w \leftarrow v) = min\left\{1, \frac{W\text{-}in(w)}{W\text{-}in(v)}\right\} \tag{4}$$

where $P(w \leftarrow v)$ is the probability that a random walk moves backward from v to w. This process is repeated until a maximum number of tries is reached (or a k-backbone gene is reached). Figure 1 illustrates an inference subgraph.

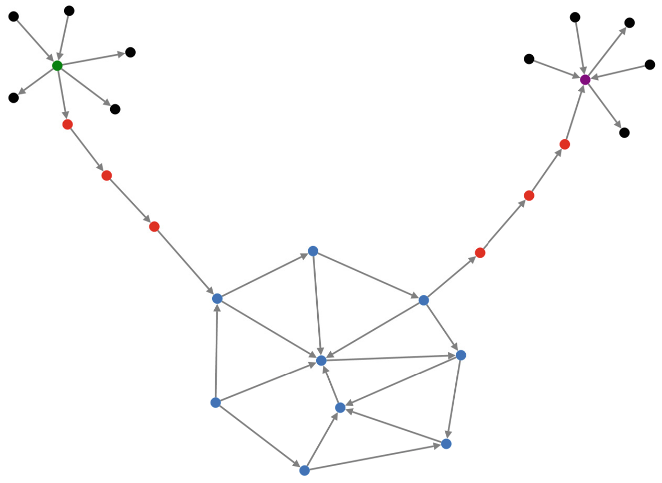

Fig. 1. Example of an inference subgraph containing an origin (green), a destination (purple), d-local genes (black), k-backbone genes (blue) and random walk metropolis genes (red).

2.2 Feature Selection

An inference subgraph may still have missing and spurious links. We select a more reliable sample from the inference subgraph where the weight of each edge in the sample is greater than or equal to 0.5. For each pair of genes u, v in the sample graph, we create a feature vector B by concatenating the gene expression profiles of u and v, as in [7,29]. That is,

$$B = [u^1, u^2, \ldots, u^p, v^1, v^2, \ldots, v^p] \tag{5}$$

where $u^1, u^2, \ldots, u^p$ are the gene expression values of u, and $v^1, v^2, \ldots, v^p$ are the gene expression values of v. Each gene expression value is a feature.

We employ principal component analysis (PCA) to reduce the dimensionality of the feature vectors of a sample graph [28]. Specifically, we combine the feature vectors into a $2p \times N$ matrix X where $2p$ is the total number of features and N is the number of links in the sample graph. Let the rank of the matrix X be r where the rank represents the maximum number of uncorrelated column vectors in X [33]. We represent X through singular value decomposition (SVD) as

$$X = U \cdot S \cdot V^T \tag{6}$$

Both U and V are orthogonal matrices. Each column of U is one of the eigenvectors of the covariance matrix $X \cdot X^T$ where X^T is the transpose of X. Each column of V is one of the eigenvectors of the matrix $X^T \cdot X$. The $r \times r$ matrix S contains eigenvalues of X on the diagonal line of S.

In our case, each column vector of the matrix X represents a (present or missing) link of the sample graph. That is, each link of the sample graph is a vector in the $2p$-dimensional Euclidean space. The dot product between two column vectors reflects the extent to which the two corresponding links share similar feature occurrences. Thus, we can use the dot product to get pairwise link distances. Let M contain pairwise link distances, i.e., M_{ij} is the dot product distance between link L_i and link L_j. Then M can be derived by:

$$M = X^T \cdot X \tag{7}$$

which can be generalized as:

$$\begin{aligned} M &= (U \cdot S \cdot V^T)^T \cdot (U \cdot S \cdot V^T) && (8)\\ &= (V \cdot S \cdot U^T) \cdot (U \cdot S \cdot V^T) && (9)\\ &= V \cdot S^2 \cdot V^T && (10)\\ &= (V \cdot S) \cdot (V \cdot S)^T && (11) \end{aligned}$$

The new representation of the matrix M shows that the pairwise link comparison matrix M can be obtained through the dot product of $(V{\cdot}S)$ and $(V{\cdot}S)^T$. That is, the ith row of the $N \times r$ matrix $(V \cdot S)$ is an r-dimensional vector representing the ith link in the sample graph. This result indicates that after performing projection transformation with respect to the matrix V, we can keep pairwise distances between link-vectors as in the original setting.

SVD reduces the dimensionality (from $2p$ to r where $r < min(2p, N)$) of a link-vector. However, the reduced r-dimensional link-vector might still contain redundant dimensions. In practice, the dimensionality of these link-vectors could be further reduced without losing characteristics of the link-vectors in the original $2p$-dimensional Euclidean space. Specifically, based on the optimal solution of the squared-error criterion of PCA [28], an r-dimensional vector could be projected onto a k-dimensional subspace, $k < r$, spanned by the eigenvectors corresponding to the k largest eigenvalues of the covariance matrix $X \cdot X^T$. As a result, we can obtain X_k, which is an approximation of the original matrix X, by keeping the k largest eigenvalues of the covariance matrix $X \cdot X^T$ and replacing the remaining eigenvalues with zeros. Then, Equation (6) can be rewritten as

$$X_k = U_k \cdot S_k \cdot V_k^T \tag{12}$$

Therefore, the $N \times k$ matrix $(V_k \cdot S_k)$ replaces the matrix $(V \cdot S)$ in Equation (11), where the ith row of the matrix $(V_k \cdot S_k)$ is a k-dimensional vector that represents the ith link in the sample graph. (In the study presented here, $k = 10$.)

2.3 The Link Prediction Algorithm

After explaining the concepts of graph sparsification and feature selection, we now describe how the proposed learning framework (i.e., link prediction algorithm) works. The main assumption here is that the network G constructed by Inferelator is not accurate, and there are many missing and spurious links in G. A missing link or edge e_m refers to a regulatory relationship that exists in the ground truth but is not inferred by Inferelator, and hence $e_m \notin G$. A spurious link e_s refers to a regulatory relationship that does not exist in the ground truth, but is inferred by Inferelator, and hence $e_s \in G$. The goal here is for our link prediction algorithm to detect these missing and spurious links, so as to clean them. To achieve this goal, the algorithm predicts whether there is a link between two nodes and uses the predicted outcome to replace the result obtained from Inferelator if the predicted outcome differs from Inferelator's result.

Let $G = (V, E)$ be the gene regulatory network (GRN) constructed by Inferelator based on a time series gene expression dataset. Our algorithm first creates two subgraphs $G_+ = (V_+, E_+)$ and $G_- = (V_-, E_-)$ where V_+ (V_-, respectively) contains incident nodes of the edges in E_+ (E_-, respectively), the weight of each edge in E_+ (E_-, respectively) is greater than or equal to (less than, respectively) the median of the weights of the edges in E, $E_+ \cap E_- = \emptyset$ and $E_+ \cup E_- = E$. Thus, the edges in G_+ are likely to be positive instances and the edges in G_- are likely to be negative instances. Note, however, that in practice these two subgraphs G_+ and G_- have low quality data, i.e., they contain many missing and spurious links.

Our link-prediction algorithm consists of five steps.

Step 1: Suppose the algorithm aims to predict whether there is a link from node/gene u to node/gene v. There are two cases to consider. In case 1, the gene pair (u, v) is in G_+. Then the algorithm creates an inference subgraph $I_+ \subseteq G_+$ by invoking GeneProbe and using G_+, the origin u and the destination v as input. In addition, the algorithm randomly selects a pair of genes x, y in G_-, and creates an inference subgraph $I_- \subseteq G_-$ by invoking GeneProbe and using G_-, the origin x and the destination y as input. In case 2, the gene pair (u, v) is in G_-. Then the algorithm creates an inference subgraph $I_- \subseteq G_-$ by invoking GeneProbe and using G_-, the origin u and the destination v as input. In addition, the algorithm randomly selects a pair of genes x, y in G_+, and creates an inference subgraph $I_+ \subseteq G_+$ by invoking GeneProbe and using G_+, the origin x and the destination y as input. Without loss of generality, we assume that case 1 holds and will use case 1 to describe the following steps. Thus, the algorithm creates dual graph sparsifications I_+ and I_- in step 1.

Step 2: Create a sample graph $I_+^{'} \subseteq I_+$ where $I_+^{'}$ does not contain the testing gene pair (u, v), and the weight of each edge in $I_+^{'}$ is greater than or equal to 0.5. We consider the edges in $I_+^{'}$ to have higher quality and are more likely to be positive instances. Suppose there are K edges in $I_+^{'}$. We then randomly select K edges from I_- and use the randomly selected edges to form a sample graph $I_-^{'} \subseteq I_-$. In training the three classifiers including neural networks, random forests and support vector machines, we will use the edges in $I_+^{'}$ as positive training examples, and use the edges in $I_-^{'}$ as negative training examples. The dual sample graphs $I_+^{'}$ and $I_-^{'}$ together form the training dataset.

Step 3: Construct a feature vector for the testing gene pair (u, v) by concatenating the gene expression values of u and v, as shown in Equation (5). Also, construct a feature vector for each gene pair (p, q) in the training dataset by concatenating the gene expression values of p and q.

Step 4: Reduce the dimensionality of the feature vectors constructed in step 3 using principal component analysis (PCA), as described in Section 2.2.

Step 5: Use the training examples (reduced feature vectors obtained from step 4) to train three classifiers including neural networks, random forests and support vector machines. Use the trained models to predict whether there is a link from gene u to gene v.

3 Experiments and Results

We conducted a series of experiments to evaluate the performance of the proposed learning framework and the three classifiers used in the framework. Below, we describe the datasets and experimental methodology used in our study, and then present the experimental results.

3.1 Datasets

We adopted the five time-series gene expression datasets available in the DREAM4 100-gene *in silico* network inference challenge [10,22,25,27]. Each dataset contains 10 times series, where each time series has 21 time points, for 100 genes. Each gene has $(10 \times 21) = 210$ gene expression values. Each link consists of two genes, and hence is represented by a 420-dimensional feature vector; cf. Equation (5). Through principal component analysis, each reduced feature vector has only 10 dimensions.

Each time-series dataset is associated with a gold standard file, where the gold standard represents the ground truth of the network structure for the time-series data. Each link in the gold standard represents a true regulatory relationship between two genes. For a given time-series dataset, Inferelator [3] constructs a directed network, in which each link has a weight and represents an inferred regulatory relationship between two genes.

Table 1 presents details of the five networks, true and inferred, used in the experiments. The table shows the numbers of true present and missing links in each gold standard, and the numbers of inferred present and missing links in each network constructed by Inferelator. Each network contains 100 nodes or genes, which form 9,900 ordered gene pairs totally.

Table 1. Networks used in the experiments

	Net1	Net2	Net3	Net4	Net5
Directed	Yes	Yes	Yes	Yes	Yes
Nodes	100	100	100	100	100
True present links	176	249	195	211	193
True missing links	9,724	9,651	9,705	9,689	9,707
Inferred present links	6,232	6,066	6,186	5,930	6,180
Inferred missing links	3,668	3,834	3,714	3,970	3,720

For each network, we created four sets of testing data. Each testing dataset contains 50 randomly selected links from the gold standard. Among the 50 links, 25 are true present links and 25 are true missing links in the gold standard. The label (+1 or -1) of each selected link is known, where +1 represents a true present link and -1 represents a true missing link. These testing data were excluded from the training datasets used to train the classifiers studied in the paper. There were 20 testing datasets totally.

3.2 Experimental Methodology

We considered three classification algorithms, namely neural networks (NN), random forests (RF) and support vector machines (SVM). Software used in this work included: the neuralnet package in R [11], the random forest package in R [19], and the SVM program with the polynomial kernel of degree 2 in the LIBSVM package [4]. The principal component analysis (PCA) program was based on the prcomp function in R [13]. The graph sparsification method (GeneProbe) was implemented in C++. In addition, we used R to write some utility tools for performing the experiments.

The performance of each classification algorithm was evaluated as follows. We trained each classification algorithm as described in Section 2.3. For each link in a testing dataset, we used the trained model to predict its label. In evaluating our link prediction algorithm, we define a true positive to be a true present link that is predicted as a present link. A false positive is a true missing link that is predicted as a present link. A true negative is a true missing link that is predicted as a missing link. A false negative is a true present link that is predicted as a missing link. In evaluating Inferelator, we define a true positive to be a true present link that is an inferred present link. A false positive is a true missing link that is an inferred present link. A true negative is a true missing link that is an inferred missing link. A false negative is a true present link that is an inferred missing link. Let TP (FP, TN, FN, respectively) denote the number of true positives (false positives, true negatives, false negatives, respectively) for a testing dataset. We adopted the balanced error rate (BER) [29], defined as

$$BER = \frac{1}{2} \times \left(\frac{FN}{TP + FN} + \frac{FP}{FP + TN} \right) \tag{13}$$

We applied each classification algorithm to each testing dataset and recorded the BER for that testing dataset. The lower BER a classification algorithm has, the better performance that algorithm achieves. We define the *improvement rate*, denoted IR, on a testing dataset to be $(P^* - P) \times 100\%$ where P^* is the BER of Inferelator and P is the BER of a classification algorithm (NN, RF or SVM) for that dataset. Statistically significant performance differences were calculated using Wilcoxon signed rank tests [15]. As in [28], we considered p-values below 0.05 to be statistically significant.

3.3 Experimental Results

Table 2 shows the improvement rate (IR) each classification algorithm achieves on each of the 20 testing datasets. A positive (negative, respectively) IR for a classification algorithm indicates that the algorithm performs better (worse, respectively) than Inferelator. The larger the positive IR a classification algorithm has, the more improvement over Inferelator that algorithm achieves. For each testing dataset, the classification algorithm with the best performance, i.e., the largest positive IR, is shown in boldface.

It can be seen from Table 2 that SVM (support vector machines) outperforms Inferelator, NN (neural networks) and RF (random forests). SVM improves Inferelator on 16 testing datasets, and the improvement is statistically significant according to Wilcoxon signed rank tests ($p < 0.05$). NN improves Inferelator on 12 testing datasets; however, the improvement is not statistically significant according to Wilcoxon signed rank tests ($p > 0.05$).

Table 2. Improvement rates of three classification algorithms on twenty datasets

Dataset	NN	RF	SVM
Net1.test1	+11.1%	+0.90%	**+17.9%**
Net1.test2	**+17.0%**	+1.80%	+12.3%
Net1.test3	+7.10%	+4.90%	**+13.6%**
Net1.test4	+5.20%	+2.10%	**+9.20%**
Net2.test1	**+5.80%**	+1.80%	−1.60%
Net2.test2	**+14.8%**	−6.20%	+0.90%
Net2.test3	**+4.40%**	−12.1%	+1.20%
Net2.test4	−0.60%	+1.20%	**+5.20%**
Net3.test1	**+0.20%**	−9.40%	−1.10%
Net3.test2	+0.00%	**+8.00%**	+4.00%
Net3.test3	−8.00%	−6.00%	**+2.00%**
Net3.test4	−2.00%	−10.0%	**+8.00%**
Net4.test1	**+10.4%**	−4.80%	+3.00%
Net4.test2	−5.70%	−6.00%	**+4.50%**
Net4.test3	**+10.8%**	+2.50%	+5.20%
Net4.test4	−3.00%	**+8.20%**	+2.60%
Net5.test1	+1.20%	−5.00%	**+7.10%**
Net5.test2	−1.70%	−13.1%	**+2.90%**
Net5.test3	−7.00%	−13.3%	−3.50%
Net5.test4	−3.00%	−7.20%	−1.10%

We also carried out experiments to evaluate the performance of different SVM kernel functions, including the linear kernel (SVM_L), polynomial kernel of degree 2 (SVM_P2), polynomial kernel of degree 15 (SVM_P15), Gaussian kernel (SVM_G), and sigmoid kernel (SVM_S). Figure 2 shows the BER values, averaged over the 20 testing datasets, for the different kernel functions. It can be seen that SVM with the polynomial kernel of degree 2 (SVM_P2) used in this study performs the best.

Finally we conducted experiments to evaluate the effectiveness of the components of the proposed learning framework. There are two core components: graph sparsification (GeneProbe) and feature selection (PCA). Figure 3 compares the approach with graph sparsification (GS) only, the approach with feature selection (FS) only, and our proposed approach, which combines both graph sparsification (GS) and feature selection (FS). Each bar represents the average BER over the 20 testing datasets. The classifier used to generate the results was the SVM program with the polynomial kernel of degree 2. It can be seen from Figure 3 that the proposed approach combining GS and FS performs the best.

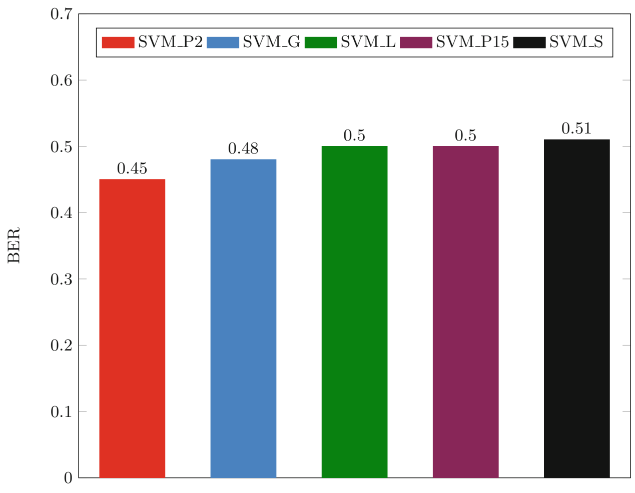

Fig. 2. Performance evaluation of different SVM kernel functions.

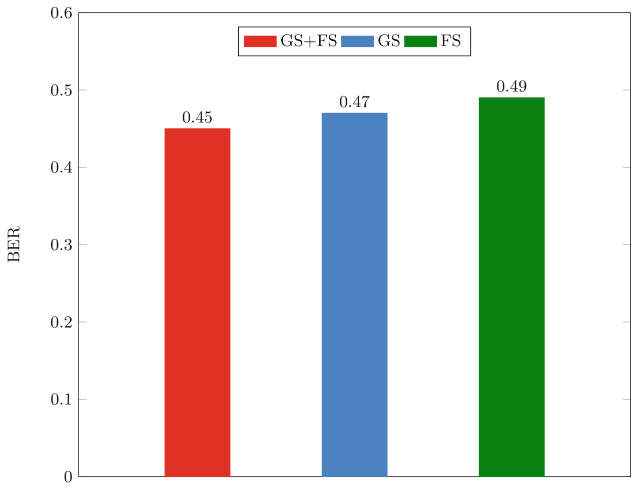

Fig. 3. Effectiveness of the components of the proposed learning framework.

4 Conclusion

Given gene regulatory networks constructed by unsupervised network inference methods, our goal is to predict and clean the links in the networks. To achieve this goal, we propose a learning framework, which employs (i) a graph sparsification technique (GeneProbe) for generating inference subgraphs from a given network, and (ii) principal component analysis (PCA) for selecting significant features from high-dimensional feature vectors. The selected feature values are then used to train three classifiers including neural networks (NN), random forests (RF) and support vector machines (SVM) for performing link prediction and link cleaning in the given network.

In our case study, the proposed framework is able to learn better quality training data from noisy networks constructed by a widely used network inference tool (Inferelator). Among the three classification algorithms studied in the paper, SVM with the polynomial kernel of degree 2 outperforms NN and RF in terms of improving the accuracy of Inferelator. This kernel is the best among all SVM kernel functions tested here. Our experimental results also show that combining both graph sparsification and PCA is better than using PCA or graph sparsification alone.

To the best of our knowledge, ours is the first study to predict and clean the links in gene regulatory networks constructed by unsupervised network inference methods. In future work, we plan to apply the proposed learning framework to other unsupervised network inference tools and evaluate its performance when used with those tools. We will also explore new feature learning and data classification methods including deep learning algorithms and assess their feasibility for our framework.

References

1. Barzel, B., Barabási, A.L.: Network link prediction by global silencing of indirect correlations. Nature Biotechnology **31**(8), 720–725 (2013)
2. Bogdanov, P., Singh, A.K.: Accurate and scalable nearest neighbors in large networks based on effective importance. In: Proceedings of the 22nd ACM International Conference on Information and Knowledge Management, pp. 1009–1018 (2013). http://doi.acm.org/10.1145/2505515.2505522
3. Bonneau, R., Reiss, D.J., Shannon, P., Facciotti, M., Hood, L., Baliga, N.S., Thorsson, V.: The Inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo. Genome Biology **7**(5), R36 (2006)
4. Chang, C., Lin, C.: LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2(3), 27 (2011). http://doi.acm.org/10.1145/1961189.1961199
5. Chen, H., Ku, W., Wang, H., Tang, L., Sun, M.: LinkProbe: probabilistic inference on large-scale social networks. In: Proceedings of the 29th IEEE International Conference on Data Engineering, pp. 290–301 (2013). http://dx.doi.org/10.1109/ICDE.2013.6544833
6. Clauset, A., Moore, C., Newman, M.E.: Hierarchical structure and the prediction of missing links in networks. Nature **453**(7191), 98–101 (2008)
7. De Smet, R., Marchal, K.: Advantages and limitations of current network inference methods. Nature **8**(10), 717–729 (2010)
8. Elloumi, M., Iliopoulos, C.S., Wang, J.T.L., Zomaya, A.Y.: Pattern Recognition in Computational Molecular Biology: Techniques and Approaches. Wiley (2015)
9. Getoor, L., Diehl, C.P.: Link mining: a survey. SIGKDD Explorations **7**(2), 3–12 (2005). http://doi.acm.org/10.1145/1117454.1117456
10. Greenfield, A., Madar, A., Ostrer, H., Bonneau, R.: DREAM4: combining genetic and dynamic information to identify biological networks and dynamical models. PLoS ONE **5**(10), e13397 (2010). http://dx.doi.org/10.1371%2Fjournal.pone.0013397

11. Günther, F., Fritsch, S.: Neuralnet: training of neural networks. Nature **2**(1), 30–38 (2010)
12. Hasan, M., Zaki, M.: A survey of link prediction in social networks. In: Aggarwal, C.C. (ed.) Social Network Data Analytics, pp. 243–275. Springer, US (2011). http://dx.doi.org/10.1007/978-1-4419-8462-3_9
13. Hothorn, T., Everitt, B.S.: A Handbook of Statistical Analyses Using R. CRC Press (2014)
14. Huynh-Thu, V.A., Sanguinetti, G.: Combining tree-based and dynamical systems for the inference of gene regulatory networks. Bioinformatics **31**(10), 1614–1622 (2015). http://dx.doi.org/10.1093/bioinformatics/btu863
15. Kanji, G.K.: 100 Statistical Tests. Sage (2006)
16. Krouk, G., Mirowski, P., LeCun, Y., Shasha, D., Coruzzi, G.: Predictive network modeling of the high-resolution dynamic plant transcriptome in response to nitrate. Genome Biology **11**(12), R123 (2010). http://dx.doi.org/10.1186/gb-2010-11-12-r123
17. Leskovec, J., Faloutsos, C.: Sampling from large graphs. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, NY, USA, pp. 631–636 (2006). http://doi.acm.org/10.1145/1150402.1150479
18. Leskovec, J., Huttenlocher, D., Kleinberg, J.: Predicting positive and negative links in online social networks. In: Proceedings of the 19th International Conference on World Wide Web, NY, USA, pp. 641–650 (2010). http://doi.acm.org/10.1145/1772690.1772756
19. Liaw, A., Wiener, M.: Classification and regression by randomForest. R News **2**(3), 18–22 (2002). http://CRAN.R-project.org/doc/Rnews/
20. Madar, A., Greenfield, A., Vanden-Eijnden, E., Bonneau, R.: DREAM3: network inference using dynamic context likelihood of relatedness and the Inferelator. PLoS ONE **5**(3), e9803 (2010). http://dx.doi.org/10.1371%2Fjournal.pone.0009803
21. Maetschke, S., Madhamshettiwar, P.B., Davis, M.J., Ragan, M.A.: Supervised, semi-supervised and unsupervised inference of gene regulatory networks. Briefings in Bioinformatics **15**(2), 195–211 (2014). http://dx.doi.org/10.1093/bib/bbt034
22. Marbach, D., Schaffter, T., Mattiussi, C., Floreano, D.: Generating realistic in silico gene networks for performance assessment of reverse engineering methods. Nature **16**(2), 229–239 (2009)
23. Mathioudakis, M., Bonchi, F., Castillo, C., Gionis, A., Ukkonen, A.: Sparsification of influence networks. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, NY, USA, pp. 529–537 (2011). http://doi.acm.org/10.1145/2020408.2020492
24. Patel, N., Wang, J.T.L.: Semi-supervised prediction of gene regulatory networks using machine learning algorithms. Journal of Biosciences 40(4), 731–740 (2015). http://dx.doi.org/10.1007/s12038-015-9558-9
25. Prill, R.J., Marbach, D., Saez-Rodriguez, J., Sorger, P.K., Alexopoulos, L.G., Xue, X., Clarke, N.D., Altan-Bonnet, G., Stolovitzky, G.: Towards a rigorous assessment of systems biology models: the DREAM3 challenges. PLoS ONE 5(2), e9202 (2010). http://dx.doi.org/10.1371%2Fjournal.pone.0009202
26. Ringnér, M.: What is principal component analysis? Nature **26**(3), 303–304 (2008)
27. Schaffter, T., Marbach, D., Floreano, D.: GeneNetWeaver: in silico benchmark generation and performance profiling of network inference methods. Bioinformatics 27(16), 2263–2270 (2011). http://dx.doi.org/10.1093/bioinformatics/btr373

28. Turki, T., Roshan, U.: Weighted maximum variance dimensionality reduction. In: Martínez-Trinidad, J.F., Carrasco-Ochoa, J.A., Olvera-Lopez, J.A., Salas-Rodríguez, J., Suen, C.Y. (eds.) MCPR 2014. LNCS, vol. 8495, pp. 11–20. Springer, Heidelberg (2014)
29. Turki, T., Wang, J.T.L.: A new approach to link prediction in gene regulatory networks. In: Jackowski, K., et al. (eds.) IDEAL 2015. LNCS, vol. 9375, pp. 404–415. Springer, Heidelberg (2015)
30. Vera-Licona, P., Jarrah, A.S., García-Puente, L.D., McGee, J., Laubenbacher, R.C.: An algebra-based method for inferring gene regulatory networks. BMC Systems Biology 8, 37 (2014). http://dx.doi.org/10.1186/1752-0509-8-37
31. Villaverde, A.F., Ross, J., Morn, F., Banga, J.R.: MIDER: network inference with mutual information distance and entropy reduction. PLoS ONE **9**(5), e96732 (2014). http://dx.doi.org/10.1371%2Fjournal.pone.0096732
32. Wang, J.T.L., Zaki, M.J., Toivonen, H.T.T., Shasha, D.: Data Mining in Bioinformatics. Springer (2005)
33. Wang, J.T.L., Liu, J., Wang, J.: XML clustering and retrieval through principal component analysis. International Journal on Artificial Intelligence Tools **14**(4), 683 (2005). http://dx.doi.org/10.1142/S0218213005002326
34. Yeung, K.Y., Ruzzo, W.L.: Principal component analysis for clustering gene expression data. Bioinformatics **17**(9), 763–774 (2001). http://dx.doi.org/10.1093/bioinformatics/17.9.763
35. Young, W., Raftery, A.E., Yeung, K.Y.: Fast Bayesian inference for gene regulatory networks using ScanBMA. BMC Systems Biology **8**, 47 (2014). http://dx.doi.org/10.1186/1752-0509-8-47
36. Yu, J., Smith, V.A., Wang, P.P., Hartemink, A.J., Jarvis, E.D.: Advances to Bayesian network inference for generating causal networks from observational biological data. Bioinformatics **20**(18), 3594–3603 (2004). http://bioinformatics.oxfordjournals.org/content/20/18/3594.abstract
37. Zoppoli, P., Morganella, S., Ceccarelli, M.: TimeDelay-ARACNE: reverse engineering of gene networks from time-course data by an information theoretic approach. BMC Bioinformatics **11**, 154 (2010). http://dx.doi.org/10.1186/1471-2105-11-154