Inferring Gene Regulatory Networks by Combining Supervised and Unsupervised Methods

Turki Turki
Computer Science Department
King Abdulaziz University
Jeddah, Saudi Arabia
tturki@kau.edu.sa

Jason T. L. Wang
Department of Computer Science
New Jersey Institute of Technology
Newark, NJ 07102, USA
wangj@njit.edu

Ibrahim Rajikhan
Department of Biological Sciences
St. John's University
Queens, NY, 11439, USA
Ibrahim.rajikhan13@stjohns.edu

***Abstract*—Supervised methods for inferring gene regulatory networks (GRNs) perform well with good training data. However, when training data is absent, these methods are not applicable. Unsupervised methods do not need training data but their accuracy is low. In this paper, we combine supervised and unsupervised methods to infer GRNs using time-series gene expression data. Specifically, we use results obtained from unsupervised methods to train supervised methods. Since the results contain noise, we develop a data cleaning algorithm to remove noise, hence improving the quality of the training data. These refined training data are then used to guide classifiers including support vector machines and deep learning tools to infer GRNs through link prediction. Experimental results on several data sets demonstrate the good performance of the classifiers and the effectiveness of our data cleaning algorithm.**

Keywords—data mining; machine learning; network inference; systems biology

I. Introduction

Current biotechnology has allowed researchers in various fields to obtain immense amounts of experimental information, ranging from macromolecular sequences, gene expression data to proteomics and metabolomics. In addition to large-scale genomic information obtained through such methods as third generation DNA sequencing, newer technology, such as RNA-seq and ChIP-seq, has allowed researchers to fine tune the analysis of gene expression patterns [1]. More information on interactions between transcription factors and DNA, both qualitative and quantitative, is increasingly emerging from microarray data.

Although microarrays alone do not provide direct evidence of functional connections among genes, the attachment of transcription factors (TFs) and their binding sites (TFBSs), located at specific gene promoters, influences transcription and modulates RNA production from that particular gene, thus establishing a first level of functional interaction. Since the TFs are gene-encoded polypeptides and the target TFBSs belong to different genes, analyses of TFs-TFBSs interactions could uncover gene networks and may even contribute to elucidate unknown GRNs [2]. Besides contributing to infer and understand these interactions, determining GRNs also aims to provide explanatory models of such connections [3]. GRNs could be the basis to infer more complex networks, encompassing gene, protein, and metabolic spaces, as well as the entangled and often overlooked signaling pathways that interconnect them [4-6].

Maetschke *et al*. [7] categorized GRN inference methods into three groups: supervised, unsupervised and semi-supervised. While supervised algorithms are capable of achieving the highest accuracy among all the GRN inference methods, these algorithms require a large number of positive and negative training examples, which are difficult to obtain in many organisms [8-10]. Unsupervised algorithms infer GRNs based solely on gene expression profiles and do not need any training example; however, the accuracy of the unsupervised algorithms is low [7]. In this paper we explore ways of combining unsupervised and supervised methods for GRN inference using time-series gene expression data.

II. Related Work

Widely used unsupervised methods for time-series gene expression data include BANJO (Bayesian Network Inference with Java Objects) [11], TimeDelay-ARACNE (Algorithm for the Reconstruction of Accurate Cellular Networks) [12], tlCLR (Time-Lagged Context Likelihood of Relatedness) [13, 14], DFG (Dynamic Factor Graphs) [15], Jump3 [16], ScanBMA [17], and Inferelator [18].

BANJO models GRNs as a first-order Markov process; it searches through all possible GRNs, seeking the network with the best score. TimeDelay-ARACNE infers GRNs 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 GRNs 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.

On the other hand, supervised methods use training data along with a classification algorithm such as support vector machines (SVMs) [8, 9, 19, 20]. The training data includes known regulatory relationships between genes, also called links, which are used to guide the classification algorithm to reconstruct GRNs through link prediction. The performance

of the supervised methods depends on the quality and the amount of available training data.

III. Background and Overview

Central to our approach of combining supervised and unsupervised methods for GRN inference is a linear algebra-based data cleaning algorithm. The input of the data cleaning algorithm contains a portion of a weighted directed graph $G = (V, E)$ that represents the topological structure of a GRN. This GRN is inferred by an unsupervised method based on a time-series gene expression dataset. E is the set of directed 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 $0 < W(e) \leq 1$. As a case study, we use Inferelator [18] as the unsupervised method in this paper. Inferelator is one of the most widely used unsupervised methods in the field.

The main drawback of employing an unsupervised method such as Inferelator for GRN inference is that the method often creates missing and spurious links [7]. Let G be a network constructed by Inferelator. A missing link or edge e_m refers to a regulatory relationship that exists in the ground truth but is not created 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 created by Inferelator, and hence $e_s \in G$. These missing and spurious links will be used to train supervised methods. Our goal is to develop a data cleaning algorithm for removing the errors or noises in the links to get better quality training data. Since an inferred network is sizable, we select m links with the largest weights and m links with the smallest weights to form an original training set. Then, we construct feature vectors for the selected $2m$ links in the training set.

Our data cleaning algorithm consists of three steps. First, we calculate a distance matrix for the feature vectors using Laplacian kernel function. Second, we adopt a linear algebra technique to project the training set onto the eigenvectors of the distance matrix to obtain noise-removed features. Third, we select important features from the noise-removed features. The feature vectors containing the selected important features form a cleaned training set.

We build classifiers using the cleaned training set and apply these classifiers to predicting links in a regulatory network. The classification algorithms considered here include support vector machines and variants of deep neural networks. Support vector machines are commonly used in bioinformatics [21] while deep neural networks have recently received increasing attention for deep learning. We show in our case study how the proposed data cleaning algorithm improves the quality of training examples used to guide the classifiers for inferring GRNs through link prediction.

The rest of this paper is organized as follows. Section IV presents details of our data cleaning and link prediction algorithms for GRN inference. Section V reports experimental results, comparing the various classifiers studied in the paper. The results demonstrate the good performance of our approach and the effectiveness of the proposed data cleaning algorithm. Section VI concludes the paper.

IV. Methodology

A. Feature Vector Construction

We select a sample of links from the weighted network constructed by Inferelator. The sample contains m links (positive training examples) with the largest weights and m links (negative training examples) with the smallest weights in the constructed network. These $2m$ links form the original training set. (In the study presented here, $m = 100$.) For each ordered pair of genes u, v in a selected link, we create a feature vector x by concatenating the gene expression profiles of u and v as done in [9, 22]. That is,

$$\mathrm{x} = [u^1, u^2, ..., u^p, v^1, v^2, ..., v^p] \quad (1)$$

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

B. Data Cleaning

To facilitate the discussion of our data cleaning algorithm, we first summarize the mathematical symbols and notation used here. Matrices (vectors, respectively) are denoted by uppercase (lowercase, respectively) letters. The notation x_i denotes the ith vector in matrix X. Elements of a vector are denoted by italic lowercase with a superscript, e.g., x^i is the ith element of vector x; scalars are denoted by italic lowercase.

We construct a feature matrix $\mathrm{X} \in \mathbb{R}^{2m \times 2p}$ in which each row corresponds to a link (feature vector) in the original training set. Our data cleaning algorithm consists of three steps.

Step 1: Compute the distance matrix D in Equation (2):

$$\mathrm{D} = [d(\mathrm{x}_i, \mathrm{x}_j)]_{ij} \in \mathbb{R}^{2m \times 2m} \quad (2)$$

where

$$d(\mathrm{x}_i, \mathrm{x}_j) = \exp\left(-\sigma \| \mathrm{x}_i - \mathrm{x}_j \|\right), \forall i, j = 1, ..., 2m. \quad (3)$$

Here x_i (x_j, respectively) is the ith (jth, respectively) feature vector of the feature matrix X, $\| \mathrm{x}_i - \mathrm{x}_j \|$ is the Euclidean distance between x_i and x_j, and σ is a user-determined parameter. (In the study presented here, $\sigma = 1$.) The element of the distance matrix in the left hand side of Equation (3) corresponds to the element of the kernel K calculated using the Laplacian kernel function [23, 24] in Equation (4) below:

$$\mathrm{K} = [k(\mathrm{x}_i, \mathrm{x}_j)]_{ij} \in \mathbb{R}^{2m \times 2m} \quad (4)$$

where

$$k(\mathrm{x}_i, \mathrm{x}_j) = \exp\left(-\sigma \| \mathrm{x}_i - \mathrm{x}_j \|\right). \quad (5)$$

Step 2: Denote by $\lambda_1 < \lambda_2 < \ldots < \lambda_{2m}$ the eigenvalues of the distance matrix D, and $\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_{2m}$ the corresponding eigenvectors. According to Courant-Fischer Theorem [25], we have

$$\mathrm{v}_1 = \arg \min_{\mathrm{z}:\|\mathrm{z}\|_2=1} \mathrm{z}^{\mathrm{T}} \mathrm{D}\, \mathrm{z} \quad (6)$$

and

$$\mathrm{v}_l = \arg \min_{\mathrm{z}:\|\mathrm{z}\|_2=1, \mathrm{z} \perp \mathrm{span}\{\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_{l-1}\}} \mathrm{z}^{\mathrm{T}} \mathrm{D}\, \mathrm{z}. \quad (7)$$

Here $\| \mathrm{z} \|_2$ is the 2-norm of the eigenvector z, i.e.

$$\| \mathrm{z} \|_2 = \left(\sum_{i=1}^{2m} | z^i |^2 \right)^{\frac{1}{2}}. \quad (8)$$

The notation $\mathrm{span}\{\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_{l-1}\}$ is the span of the set of orthonormal eigenvectors, $\mathrm{z} \perp \mathrm{span}\{\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_{l-1}\}$ represents that the orthonormal eigenvector z is perpendicular to the span of the set of orthonormal eigenvectors, $\mathrm{span}\{\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_{l-1}\}$ [26, 27].

Step 3: Let $\mathrm{V}_t \in \mathbb{R}^{2m \times t}$, $1 \le t \le 2m$, be the matrix whose columns are the first t eigenvectors of the distance matrix D with the smallest eigenvalues. (In the study presented here, $t = 1$.) We project X onto V_t to obtain $\mathrm{C} \in \mathbb{R}^{2m \times 2p}$ with noise-removed features. That is,

$$\mathrm{C} = \mathrm{V}_t \mathrm{V}_t^{\mathrm{T}} \mathrm{X}. \quad (9)$$

Finally, we calculate $\mathrm{M} \in \mathbb{R}^{2p}$ using Equation (10):

$$\mathrm{M} = 1^{\mathrm{T}} \mathrm{C} \quad (10)$$

where 1 is a $2m$-dimensional column vector of all ones. We select the bottom k elements in M corresponding to the k minimum values in M and store their positions in $\mathrm{P} \in \mathbb{R}^k$. (In the study presented here, $k = 10$.) Construct feature vectors by selecting only k features, based on the positions in P, from the feature vectors in C, and store the feature vectors of k features in the transformed (cleaned) training set, $\overline{\mathrm{C}}$.

C. Link Prediction

We use the cleaned training set $\overline{\mathrm{C}}$ (with feature vectors of k features obtained from step 3 above) to train classification algorithms including support vector machines [23, 24], deep neural networks with weights initialized by deep belief networks, and deep belief networks with weights initialized by stacked AutoEncoder [28]. Given a testing set with n ordered gene pairs whose labels are unknown, the goal here is to predict the label of each gene pair (u, v) in the testing set using a trained classification model. That is, the classification model will predict whether there is a link from gene u to gene v. The predicted label is +1 if it is predicted that there is a link from u to v, and −1 otherwise.

To perform the link prediction, we construct a feature vector for each gene pair (u, v) in the testing set by concatenating the gene expression values of u and v as shown in Equation (1). We then create a feature matrix S for the testing set, selecting k features based on the positions in P, and store the feature vectors of k features of the testing set in $\overline{\mathrm{S}}$. The labels of the testing examples are then predicted by a trained classification model.

V. EXPERIMENTS AND RESULTS

A. Datasets

We carried out a series of experiments to evaluate the performance of our approach, using three time-series gene expression datasets available in the DREAM4 100-gene *in silico* network inference challenge [13, 14]. Each dataset contains 10 times series, where each time series has 21 time points, for 100 genes. Each gene has (10 × 21) = 210 gene expression values. Each link consists of two genes, and hence is represented by a 420-dimensional feature vector.

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 [18] outputs a list of ordered gene pairs where each gene pair is associated with a positive, non-zero weight. Gene pairs not shown in the output list are assumed to have a weight of −1. We selected m gene pairs (positive training examples) with the largest weights and m gene pairs (negative training examples) with the smallest weights in the output list of Inferelator. These 2m links formed the original training set. We then used our proposed data cleaning algorithm to clean the 2m links to obtain a cleaned training set.

Table I presents details of the data used in the experiments. The table shows the numbers of true present and true missing links in each gold standard network, as well as the numbers of gene pairs in the output list of Inferelator and the numbers of gene pairs not shown in the output list of Inferelator for each time-series dataset. Each network contains 100 nodes or genes, which form 9,900 ordered gene pairs totally.

TABLE I. DATA USED IN THE EXPERIMENTS

	Net1	Net2	Net3
Directed	Yes	Yes	Yes
Nodes	100	100	100
True present links	176	249	195
True missing links	9,724	9,651	9,705
Gene pairs in output	6,232	6,066	6,186
Gene pairs not in output	3,668	3,834	3,714

For each network, we generated three sets of testing data. Each testing set 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 sets, so the testing sets and training sets were disjoint. There were 9 testing sets totally.

B. Experimental Setup

We considered three classification algorithms, namely support vector machines (SVM), deep neural networks with weights initialized by deep belief networks (DNN_DBN), and deep neural networks with weights initialized by stacked AutoEncoder (DNN_Auto). Software used in this work included: the deepnet package in R [29], the SVM program with the linear kernel in the LIBSVM package [30] and other kernel functions in the kernlab package [31]. In addition, we used R to write some utility tools for performing the experiments.

The performance of each classification algorithm was evaluated as follows. As in [30], we trained each classification program where the option of probability estimation in each program was turned on. Given a testing link x, the program calculates the probability of x being in the positive class C_+, i.e., $P(C_+|x)$. Each gene pair in a testing test is predicted to have the label +1 (i.e., predicted as a present link) if its probability is greater than or equal to the median of the probability estimates produced by the program. The gene pair is predicted to have the label −1 (i.e., predicted as a missing link) if its probability is less than the median probability. In evaluating the classification algorithms, 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 consider a gene pair as an inferred present link if its weight is greater than or equal to the median of the weighs produced by Inferelator. The gene pair is an inferred missing link if its weight is less than the median weight. 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 set. The performance measure used in the study is the Area Under the ROC Curve (AUC) [21], defined as

$$\mathrm{AUC} = \frac{1}{2}\left(\frac{\mathrm{TP}}{\mathrm{TP+FN}} + \frac{\mathrm{TN}}{\mathrm{TN+FP}}\right). \qquad (11)$$

We applied each classification algorithm to each testing set and recorded the AUC the algorithm obtains for the testing set. The larger AUC a classification algorithm has, the better performance that algorithm achieves. We use MAUC to denote the mean of the AUC values averaged over the three testing sets generated from a network, and use AMAUC to denote the average of the MAUC values over the three networks used in the experiments.

C. Experimental Results

We first conducted experiments to evaluate the performance of SVM with different kernel functions, including the linear kernel (SVM_L), Gaussian kernel (SVM_G), sigmoid kernel (SVM_S), and polynomial kernel of degree 2 (SVM_P). Fig. 1 shows the AMAUC values of SVM with the different kernels. It can be seen from Fig. 1 that SVM with the linear kernel (SVM_L) performs the best. The nonlinear kernels including SVM_G, SVM_S and SVM_P yield smaller AMAUC values, and hence perform worse, than SVM_L.

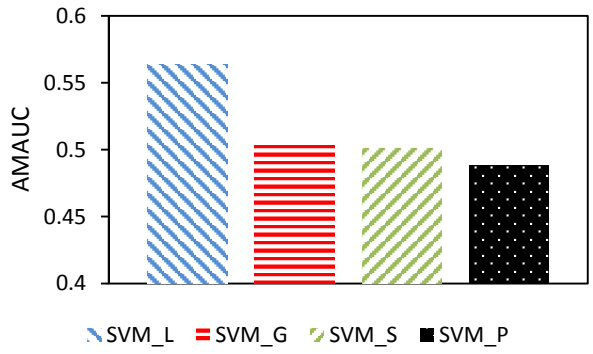

Fig. 1. Comparison of the AMAUC values of SVM with four different kernels including the linear kernel (SVM_L), Gaussian kernel (SVM_G), sigmoid kernel (SVM_S), and polynomial kernel of degree 2 (SVM_P).

In subsequent experiments, we fixed the SVM kernel at the linear kernel. Table II lists the MAUC values of SVM_L, deep neural networks with weights initialized by deep belief networks (DNN_DBN), deep neural networks with weights initialized by stacked AutoEncoder (DNN_Auto), and Inferelator (Inf). For each network, the classification algorithm with the best performance, i.e., the largest MAUC, is highlighted in boldface.

TABLE II. MAUC VALUES OF THREE CLASSIFIERS AND INFERELATOR

Dataset	SVM_L	DNN_DBN	DNN_Auto	Inf
Net1	**0.726**	0.626	0.546	0.380
Net2	0.460	0.440	**0.480**	0.353
Net3	0.506	**0.546**	0.506	0.380
Average	**0.564**	0.537	0.511	0.371

It can be seen from Table II that SVM_L is the best classifier for Net1 and yields the largest average MAUC (i.e., AMAUC) of 0.564. DNN_DBN is the best classifier for Net3 and yields the second largest average MAUC of 0.537. DNN_Auto is the best classifier for Net2 with the average MAUC of 0.511. All the three classifiers (i.e., su-

pervised methods) perform better than the unsupervised method, Inferelator, whose average MAUC is 0.371.

It is worth noting that the kernel-based program, SVM_L, performs better than the deep learning programs DNN_DBN and DNN_Auto. Deep learning is a powerful tool for image classification on big data with hundreds of classes. The deep learning programs model high-level abstractions in data through multiple non-linear transformations. In contrast, our work focuses on binary classification with relatively small datasets in which the learned linear relationship between feature vectors and labels was shown to be effective in testing data classification. As a consequence, the deep learning programs perform worse than the kernel-based program.

Fig. 2 shows the AMAUC values of the three classifiers, SVM_L, DNN_DBN and DNN_Auto, into which the proposed data cleaning algorithm was not incorporated. Thus, the classifiers were trained by uncleaned data. Comparing Fig. 2 with Table II where data was cleaned, we can see that the AMAUC values in Fig. 2 are smaller than those in Table II. The performance of the classifiers degrades when running on uncleaned data, showing the effectiveness of our proposed data cleaning algorithm. Notably, SVM_L suffers the most when data is not cleaned.

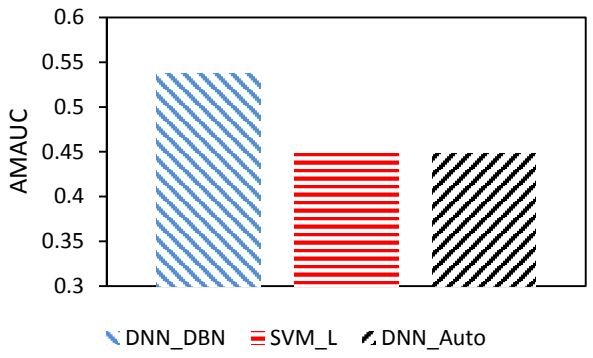

Fig. 2. Comparison of the AMAUC values of three classifiers SVM_L, DNN_DBN and DNN_Auto where the AMAUC values were obtained by running the classifiers on uncleaned data.

One component of our data cleaning algorithm is feature selection where we use Equation (10) to select k most important features to form feature vectors of k features. Fig. 3 shows the AMAUC values of the three classifiers, SVM_L, DNN_DBN and DNN_Auto, for varying k values. It can be seen from Fig. 3 that SVM_L continues to be the best classifier when k values change. Its behavior is stable with respect to k. Selecting k features makes our approach computationally efficient for large datasets with good predictive performance.

We also tested on different values for the parameters m in Equation (2), σ in Equation (3) and t in Equation (9) used in the proposed data cleaning algorithm. The results obtained were similar to those of using the default values for these parameters ($m = 100$, $\sigma = 1$, $t = 1$), and the qualitative conclusion remains the same.

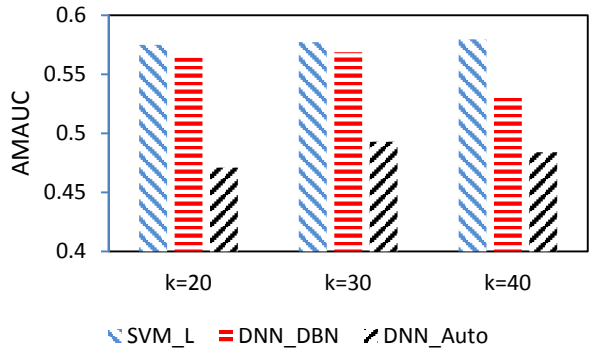

Fig. 3. Impact of the number of selected features, k, on the perfomance of three classifiers SVM_L, DNN_DBN and DNN_Auto.

VI. Discussion and Conclusion

Machine learning in biomedicine has received increasing attention recently [32-34]. In this paper we present a hybrid approach for learning gene regulatory networks (GRNs) by combining supervised and unsupervised methods. Central to our approach is a linear algebra-based algorithm for cleaning the results of unsupervised methods. The cleaned results are then used to train supervised methods to perform GRN inference through link prediction. In our case study, we adopt a widely used unsupervised method, Inferelator, as well as three popular classifiers including support vector machines (SVM), deep neural networks with weights initialized by deep belief networks (DNN_DBN) and deep neural networks with weights initialized by stacked AutoEncoder (DNN_Auto). Our experimental results show the superiority of the proposed hybrid approach over the unsupervised method. Among the three classifiers, SVM with the linear kernel outperforms the two variants of deep neural networks, DNN_DBN and DNN_Auto. This linear kernel is the best among all SVM kernel functions tested here. Our experimental results also show the effectiveness of the proposed data cleaning algorithm.

This data cleaning algorithm is related to the noise-filtering algorithm developed by Ouyang *et al*. [35]. While both algorithms aim to improve the quality of network data, they differ in two major ways. First, Ouyang *et al*.'s method is designed for undirected networks and employs the Laplacian matrix, which is symmetric for undirected networks. The eigenvalues of the symmetric matrix are real numbers, and the corresponding eigenvectors are orthonormal. In contrast, the networks we consider here are directed networks. When applying Ouyang *et al*.'s method to directed networks, one would get a non-symmetric Laplacian matrix, whose eigenvalues may contain complex numbers. In such a situation, an orthonormal set of eigenvectors cannot be found, nor even any pair of eigenvectors that are orthogonal (except perhaps by rare chance) [36]. Thus, instead of using the Laplacian matrix, we introduced a distance matrix D as shown in Equation (2), which can be calculated by using the Laplacian kernel function as shown in Equation (5). The eigenvalues of this symmetric positive semidefinite distance matrix D are real, non-negative numbers, and the corresponding eigenvectors are orthonormal. Second, Ouyang *et*

al.'s method does not include feature selection. In contrast, we use Equation (10) to select k most important features and use the selected features for link prediction. It should also be pointed out that the work of Ouyang *et al*. did not consider machine learning algorithms. In contrast, we use our data cleaning algorithm to get better quality training data, which are then used to guide machine learning tools to perform link prediction.

To the best of our knowledge, ours is the first work to combine supervised methods with an unsupervised method (Inferelator) for GRN inference. In future work, we plan to apply the proposed hybrid approach to some well-studied organisms such as *E. coli* and yeast. We will include other unsupervised methods and assess their feasibility for our hybrid approach. We will also explore new data cleaning and link prediction algorithms such as matrix completion [37, 38] and evaluate their performance on both the DREAM datasets and datasets from the well-studied organisms.

REFERENCES

[1] Angelini C, Costa V: Understanding gene regulatory mechanisms by integrating ChIP-seq and RNA-seq data: statistical solutions to biological problems. *Frontiers in cell and developmental biology* 2014, 2:51.

[2] Werner T, Dombrowski SM, Zgheib C, Zouein FA, Keen HL, Kurdi M, Booz GW: Elucidating functional context within microarray data by integrated transcription factor-focused gene-interaction and regulatory network analysis. *European cytokine network* 2013, 24(2):75-90.

[3] Brazhnik P, de la Fuente A, Mendes P: Gene networks: how to put the function in genomics. *Trends in biotechnology* 2002, 20(11):467-472.

[4] Ideker T, Krogan NJ: Differential network biology. *Molecular systems biology* 2012, 8:565.

[5] Margolin AA, Wang K, Lim WK, Kustagi M, Nemenman I, Califano A: Reverse engineering cellular networks. *Nature protocols* 2006, 1(2):662-671.

[6] Elloumi M, Iliopoulos C, Wang JTL, Zomaya AY (eds.): Pattern Recognition in Computational Molecular Biology: Techniques and Approaches: Wiley; 2015.

[7] Maetschke SR, Madhamshettiwar PB, Davis MJ, Ragan MA: Supervised, semi-supervised and unsupervised inference of gene regulatory networks. *Briefings in bioinformatics* 2014, 15(2):195-211.

[8] Ernst J, Beg QK, Kay KA, Balazsi G, Oltvai ZN, Bar-Joseph Z: A semi-supervised method for predicting transcription factor-gene interactions in Escherichia coli. *PLoS computational biology* 2008, 4(3):e1000044.

[9] Cerulo L, Elkan C, Ceccarelli M: Learning gene regulatory networks from only positive and unlabeled data. *BMC bioinformatics* 2010, 11:228.

[10] Patel N, Wang JTL: Semi-supervised prediction of gene regulatory networks using machine learning algorithms. *J Biosci* 2015, 40(4):731-740.

[11] Yu J, Smith VA, Wang PP, Hartemink AJ, Jarvis ED: Advances to Bayesian network inference for generating causal networks from observational biological data. *Bioinformatics* 2004, 20(18):3594-3603.

[12] Zoppoli P, Morganella S, Ceccarelli M: TimeDelay-ARACNE: reverse engineering of gene networks from time-course data by an information theoretic approach. *BMC bioinformatics* 2010, 11:154.

[13] Greenfield A, Madar A, Ostrer H, Bonneau R: DREAM4: combining genetic and dynamic information to identify biological networks and dynamical models. *PloS one* 2010, 5(10):e13397.

[14] Madar A, Greenfield A, Vanden-Eijnden E, Bonneau R: DREAM3: network inference using dynamic context likelihood of relatedness and the inferelator. *PloS one* 2010, 5(3):e9803.

[15] Krouk G, Mirowski P, LeCun Y, Shasha DE, Coruzzi GM: Predictive network modeling of the high-resolution dynamic plant transcriptome in response to nitrate. *Genome biology* 2010, 11(12):1-19.

[16] Huynh-Thu VA, Sanguinetti G: Combining tree-based and dynamical systems for the inference of gene regulatory networks. *Bioinformatics* 2015, 31(10):1614-1622.

[17] Young WC, Raftery AE, Yeung KY: Fast Bayesian inference for gene regulatory networks using ScanBMA. *BMC systems biology* 2014, 8(1):47.

[18] Bonneau R, Reiss DJ, Shannon P, Facciotti M, Hood L, Baliga NS, Thorsson V: The Inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo. *Genome biology* 2006, 7(5):R36.

[19] Mordelet F, Vert JP: SIRENE: supervised inference of regulatory networks. *Bioinformatics* 2008, 24(16):i76-82.

[20] Gillani Z, Akash M, Rahaman M, Chen M: CompareSVM: supervised, Support Vector Machine (SVM) inference of gene regularity networks. *BMC bioinformatics* 2014, 15(1):395.

[21] Wang JTL, Zaki MJ, Toivonen H, Shasha D (eds.): Data Mining in Bioinformatics: Springer-Verlag London; 2005.

[22] De Smet R, Marchal K: Advantages and limitations of current network inference methods. *Nature reviews Microbiology* 2010, 8(10):717-729.

[23] Schölkopf B, Smola AJ: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond: MIT Press; 2001.

[24] Mohri M, Rostamizadeh A, Talwalkar A: Foundations of Machine Learning: MIT Press; 2012.

[25] Horn RA, Johnson CR: Matrix Analysis: Cambridge University Press; 2012.

[26] Golub GH, Van Loan CF: Matrix Computations: Johns Hopkins University Press; 1996.

[27] Saad Y: Numerical Methods for Large Eigenvalue Problems, Revised Edition: SIAM; 2011.

[28] Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol PA: Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research* 2010, 11:3371-3408.

[29] Rong X: deepnet: deep learning toolkit in R. 2014.

[30] Chang CC, Lin CJ: LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2011, 2(3):27.

[31] Karatzoglou A, Smola A, Hornik K, Zeileis A: kernlab - an S4 package for kernel methods in R. *Journal of Statistical Software* 2004, 11(9):1-20.

[32] Gondara L: Random forest with random projection to impute missing gene expression data. In: *Proceedings of the IEEE 14th International Conference on Machine Learning and Applications*. 2015: 1251-1256.

[33] Chebil I, Elati M, Rouveirol C, Santini G: SetNet: ensemble method techniques for learning regulatory networks. In: *Proceedings of the IEEE 12th International Conference on Machine Learning and Applications*. 2013: 34-39.

[34] Turki T, Wang JTL: A new approach to link prediction in gene regulatory networks. In: *Proceedings of the 16th International Conference on Intelligent Data Engineering and Automated Learning*. 2015: 404-415.

[35] Ouyang B, Jiang L, Teng Z: A noise-filtering method for link prediction in complex networks. *PloS one* 2016, 11(1):e0146925.

[36] Press WH, Teukolsky SA, Vetterling WT, Flannery BP: Numerical Recipes: The Art of Scientific Computing, Third Edition: Cambridge University Press; 2007.

[37] Recht B: A simpler approach to matrix completion. *Journal of Machine Learning Research* 2011, 12:3413-3430.

[38] Keshavan RH, Montanari A, Oh S: Matrix completion from noisy entries. *Journal of Machine Learning Research* 2010, 11:2057-2078.