ALGORITHMS FOR RECONSTRUCTION OF GENE REGULATORY NETWORKS FROM HIGH-THROUGHPUT GENE EXPRESSION DATA

Wenping Deng
Michigan Technological University, wenpingd@mtu.edu

Copyright 2018 Wenping Deng

Recommended Citation

Follow this and additional works at: https://digitalcommons.mtu.edu/etdr

Part of the Bioinformatics Commons, Multivariate Analysis Commons, and the Numerical Analysis and Scientific Computing Commons
ALGORITHMS FOR RECONSTRUCTION OF GENE REGULATORY NETWORKS
FROM HIGH-THROUGHPUT GENE EXPRESSION DATA

By

Wenping Deng

A DISSERTATION
Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

In Forest Molecular Genetics and Biotechnology

MICHIGAN TECHNOLOGICAL UNIVERSITY

2018

© 2018 Wenping Deng
This dissertation has been approved in partial fulfillment of the requirements for the Degree of DOCTOR OF PHILOSOPHY in Forest Molecular Genetics and Biotechnology.

School of Forest Resources and Environmental Science

Dissertation Advisor:  Hairong Wei
Committee Member:  Kui Zhang
Committee Member:  Qiuying Sha
Committee Member:  Victor Busov
School Dean:  Andrew Storer
# Table of Contents

List of figures .......................................................................................................................... vii
List of tables ........................................................................................................................... ix
Author contribution statement .............................................................................................. x
Acknowledgements ................................................................................................................ xii
List of abbreviations ............................................................................................................... xii
Abstract ....................................................................................................................................... xiii
1 Introduction ........................................................................................................................... 1

1.1 Acquisition of gene expression data .................................................................................. 1

1.1.1 Gene expression .............................................................................................................. 1

1.1.2 Microarray and RNA-Seq technology ............................................................................ 2

1.1.3 Microarray data normalization ....................................................................................... 3

1.1.4 RNA-Seq data analysis workflow .................................................................................. 3

1.2 Types of gene regulatory network .................................................................................... 4

1.2.1 Relevance network ......................................................................................................... 4

1.2.2 Partial correlation network ............................................................................................ 5

1.2.3 Causal network ............................................................................................................... 6

1.3 Computational tools for constructing gene regulatory networks .................................... 7

1.3.1 Statistical tools ............................................................................................................... 7

1.3.1.1 Correlation-based methods ....................................................................................... 8

1.3.1.2 Linear regression and shrinkage ............................................................................... 8

1.3.1.3 Gaussian graphical models ....................................................................................... 10

1.3.2 Machine Learning tools ................................................................................................ 12
1.3.2.1 Support vector machine .............................................12
1.3.2.2 Decision tree and random forest ..................................13
1.3.2.3 Bayesian networks ....................................................13
1.3.3 Convex optimization tools .............................................14
  1.3.3.1 Gradient, subgradient, and proximal gradient methods ..15
  1.3.3.2 Newton and barrier methods .....................................16
  1.3.3.3 Dual ascent method and ADMM algorithm ..................18
1.4 Organization of this dissertation .......................................19

2 HB-PLS: An algorithm for reconstruction of transcriptional regulatory networks by integrating Huber loss and Berhu penalty with Partial Least Square ............................................21
  2.1 Introduction ........................................................................21
  2.2 Variable selection via Huber-Berhu regression ......................24
    2.2.1 Huber and Berhu functions ...........................................24
    2.2.2 The algorithm to solve Huber Berhu regression ...............27
    2.2.3 Screening for non-zero variables ...................................28
    2.2.4 Simulation Results ......................................................30
      2.2.4.1 Running time .......................................................30
      2.2.4.2 Performance of Huber-Berhu regression ................31
  2.3 Huber-Berhu partial least square ......................................34
    2.3.1 Embedding Huber-Berhu regression into PLS .................34
    2.3.2 Tuning criteria and choice of the PLS dimension ..............36
    2.3.3 Analysis of Arabidopsis dataset using Huber-Berhu PLS ....37
  2.4 Conclusion ........................................................................39
3 BWERTF: a recursive random forest algorithm for constructing multilayered
hierarchical gene regulatory network that governs biological pathways..................41

3.1 Introduction .............................................................................................................42
3.2 Materials and methods..........................................................................................43

3.2.1 Materials ............................................................................................................43
3.2.1.1 Arabidopsis Microarray Data Sets .................................................................43
3.2.1.2 Mouse microarray data sets ........................................................................43
3.2.2 Methods ...............................................................................................................44
3.2.2.1 Construction of ML-hGRN with backward elimination random forest ..........44
3.2.2.2 Detail of random forest ................................................................................46
3.2.2.3 Details of Expectation Maximization (EM) algorithm ..................................47
3.2.2.4 Performance evaluation ..............................................................................47

3.3 Results ....................................................................................................................48

3.3.1 Comparison of variable recognizing capability between BWERTF and GENIE3 on a simulated “toys dataset” .................................................................48
3.3.2 Comparison of ML-hGRNs constructed from mouse gene expression data using BWERTF and GENIE3 .................................................................49
3.3.3 Comparison of ML-hGRNs constructed from Arabidopsis microarray data using BWERTF and GENIE3 .................................................................51

3.4 Discussion ..............................................................................................................52
3.5 Conclusion ..............................................................................................................54

4 JRmGRN: Joint reconstruction of multiple gene regulatory networks with common hub genes using data from multiple tissues or conditions ..........................56

4.1 Introduction ............................................................................................................56
4.2 Methods ..................................................................................................................59
4.2.1 Gaussian graphical model and regularization ...................................................59
### 4.2.2 The algorithm to estimate parameters .......................................................... 61

### 4.2.3 Selection of tuning parameters ............................................................... 64

### 4.3 Results ............................................................................................................. 68

#### 4.3.1 Results from simulated data ..................................................................... 68

##### 4.3.1.1 Results on ER-based network ........................................................... 68

##### 4.3.1.2 Results on the Barabási-Albert (BA)-based network .............. 70

#### 4.3.2 Results from real RNA-seq data of Arabidopsis thaliana ............ 71

### 4.4 Discussion ...................................................................................................... 73

### 4.5 Conclusion ...................................................................................................... 75

### 5 Reference List .................................................................................................... 76

### A Copyright documentation .................................................................................. 87
List of figures

Figure 1.1 central dogma of life........................................................................................................2
Figure 1.2 A true network and its corresponding relevance network........................................5
Figure 1.3 A true network and its corresponding partial correlation network......................5
Figure 1.4 A true network and its corresponding causal network .........................................6
Figure 1.5 Potential computational methods for reconstruction of GRNs...............................7
Figure 1.6 An example of a Bayesian network........................................................................14
Figure 1.7 The hierarchy of convex optimization problem ................................................15
Figure 2.1 Huber loss function (top left) and Berhu penalty function (top right) as well as their 2D contour (bottom row). .................................................................26
Figure 2.2 Estimation picture for Huber & Berhu regression (left). As a comparison, the estimation picture for LASSO (middle) and Ridge (right) are also included. ....27
Figure 2.3 A typical solution path for Huber & Berhu regression........................................29
Figure 2.4 The estimated coefficients (left) and their optimal clustering (right)...............29
Figure 2.5 An example of L-curve..........................................................................................30
Figure 2.6 Comparison of running time for Algorithm 2.1 and CVX.....................................31
Figure 2.7 Identified top 15 most important TFs for each lignin biosynthesis pathway gene. .................................................................................................................38
Figure 2.8 Comparison HB-PLS with PLS and SPLS on the identification of positive TFs for lignin pathway genes.................................................................39
Figure 3.1 Flowchart illustrating BWERF algorithm for constructing multilayered hierarchical gene regulatory network using expression data of pathway genes and regulatory genes .................................................................44
Figure 3.2 Comparison of the ordered important values of top 15 variables generated by GENIE3 (left) and BWERF (right) on a “toys dataset”.................................49
Figure 3.3 Comparison of precision-recall (PR) and receiver operating characteristic (ROC) curves of BWERF and GENIE3 for three mouse microarray datasets from Atlas of Pluripotency Evidence (ESCAPE), which were used in [101]..................50
Figure 3.4 Construction of ML-hGRN for lignocellulosic pathway with a compendium microarray dataset (128 chips) from Arabidopsis thaliana roots under salt stress condition. .................................................................52

Figure 4.1 A toy example of two gene regulatory networks from two tissues or environmental conditions.............................................................59

Figure 4.2 A simulated Erdős-Rényi gene network (A) and the estimated network (B)...69

Figure 4.3 Precision-Recall curve of JRmGRN and three existing methods on ER networks. ....................................................................................70

Figure 4.4 A simulated Barabási-Albert gene network (A) and the predicted network (B). .........................................................................................70

Figure 4.5 Precision-Recall curves of JRmGRN in comparison with three other existing methods, GLASSO, JGL and HGLASSO on BA networks. ........................................71

Figure 4.6 The gene regulatory networks built with JRmGRN using data sets generated from Arabidopsis thaliana leaves under low and high red:far-red (R:FR) regimes.................................................................73
List of tables

Table 2.1 Variable selection ability on model 1 based on 100 replications. ..................33
Table 2.2 Variable selection ability on model 2 based on 100 replications. ..................33
Table 3.1 AUPR and AUROC values for mouse datasets using BWERF and GENIE.....51
Table 4.1 The number of different types of edge identified by 4 methods....................74
Table 4.2 Comparison of JRmGRN and HGLASSO in identifying hub genes.................75
Author contribution statement

Chapter 3 was a collaborative work published in *PloS* one 12.2 (2017). W.D. and H.W. conceived of the presented idea. W.D. and H.W. developed the method and performed the analysis. K.Z. and V.B. performed data curation and validation. The initial manuscript was written by W.D. and reviewed by H.W. All authors discussed the results and contributed to the final manuscript.

Chapter 4 was a collaborative work published in *Bioinformatics* 1 (2018): 9. W.D. developed the theoretical formalism, performed the analytic calculations and performed the numerical simulations. K.Z. provided some statistical idea. S.L., P.X.Z. and S.X. performed data curation and validation. The initial manuscript was written by W.D. and H.W. All authors discussed the results and contributed to the final manuscript.
Acknowledgements

I would like to express my heartfelt thanks to my advisor Prof. Hairong Wei for many deliberative dialogues, opportunities to explore a diverse set of projects, and guidance for in doing interdisciplinary research. Words are not enough to express my thanks to his support and understanding that made my graduate study enjoyable. I benefited much from his sharing of knowledge, and encouraging me to be dedicated and stay motivated towards my career goals.

I would also like to express my gratitude to my other committee members who have been consistently supporting me. Professor Kui Zhang provided extra guidance, helped me on many technical issues in my research through inspiring discussions. Professor Qiuying Sha taught me statistical computing methods in her classes and directed me to opportunities my research. Professor Victor Busov gave me many positive feedbacks in my research.

I would also like to thank the faculty and staff in the School of Forest Resources and Environmental Science for their service and help, particularly the dean Dr. Andrew Storer. Thanks to all my colleagues and friends in Dr. Wei’s lab for their help in my life and research.

Finally, I would also like to express my deep gratitude to my parents, wife, and daughter for their long-lasting love, patience, and support.
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRN</td>
<td>Gene regulatory network</td>
</tr>
<tr>
<td>hGRN</td>
<td>Hierarchical GRN</td>
</tr>
<tr>
<td>TF</td>
<td>Transcription factor</td>
</tr>
<tr>
<td>DNA</td>
<td>Deoxyribonucleic acid</td>
</tr>
<tr>
<td>cDNA</td>
<td>Complementary DNA</td>
</tr>
<tr>
<td>RNA</td>
<td>ribonucleic acid</td>
</tr>
<tr>
<td>RNA-Seq</td>
<td>RNA sequencing</td>
</tr>
<tr>
<td>mRNA</td>
<td>Message RNA</td>
</tr>
<tr>
<td>FCI</td>
<td>Fast causal inference</td>
</tr>
<tr>
<td>GES</td>
<td>Greedy equivalence search</td>
</tr>
<tr>
<td>GGM</td>
<td>Gaussian graphical model</td>
</tr>
<tr>
<td>ADMM</td>
<td>Alternating direction method of multipliers</td>
</tr>
<tr>
<td>CI</td>
<td>Conditional independence</td>
</tr>
<tr>
<td>MI</td>
<td>Mutual information</td>
</tr>
<tr>
<td>KKT</td>
<td>Karush–Kuhn–Tucker</td>
</tr>
<tr>
<td>JGL</td>
<td>Joint graphical lasso</td>
</tr>
<tr>
<td>HGLASSO</td>
<td>Hub graphical lasso</td>
</tr>
<tr>
<td>SVM</td>
<td>Support vector machine</td>
</tr>
<tr>
<td>ADMM</td>
<td>Alternating direction method of multipliers</td>
</tr>
<tr>
<td>EM</td>
<td>Expectation maximization</td>
</tr>
<tr>
<td>DAG</td>
<td>Directed acyclic graph</td>
</tr>
<tr>
<td>RF</td>
<td>Random forest</td>
</tr>
<tr>
<td>BWERF</td>
<td>Backward elimination random forest</td>
</tr>
<tr>
<td>JRmGRN</td>
<td>Joint reconstruction of multiple GRNs</td>
</tr>
<tr>
<td>CV</td>
<td>Cross validation</td>
</tr>
<tr>
<td>PLS</td>
<td>Partial least square</td>
</tr>
<tr>
<td>SPLS</td>
<td>Sparse partial least square</td>
</tr>
</tbody>
</table>
Abstract
Understanding gene interactions in complex living systems is one of the central tasks in system biology. With the availability of microarray and RNA-Seq technologies, a multitude of gene expression datasets has been generated towards novel biological knowledge discovery through statistical analysis and reconstruction of gene regulatory networks (GRN). Reconstruction of GRNs can reveal the interrelationships among genes and identify the hierarchies of genes and hubs in networks. The new algorithms I developed in this dissertation are specifically focused on the reconstruction of GRNs with increased accuracy from microarray and RNA-Seq high-throughput gene expression data sets.

The first algorithm (Chapter 2) focuses on modeling the transcriptional regulatory relationships between transcription factors (TF) and pathway genes. Multiple linear regression and its regularized version, such as Ridge regression and LASSO, are common tools that are usually used to model the relationship between predictor variables and dependent variable. To deal with the outliers in gene expression data, the group effect of TFs in regulation and to improve the statistical efficiency, it is proposed to use Huber function as loss function and Berhu function as penalty function to model the relationships between a pathway gene and many or all TFs. A proximal gradient descent algorithm was developed to solve the corresponding optimization problem. This algorithm is much faster than the general convex optimization solver CVX. Then this Huber-Berhu regression was embedded into partial least square (PLS) framework to deal with the high dimension and multicollinearity property of gene expression data. The result showed this method can identify the true regulatory TFs for each pathway gene with high efficiency.

The second algorithm (Chapter 3) focuses on building multilayered hierarchical gene regulatory networks (ML-hGRNs). A backward elimination random forest (BWERF) algorithm was developed for constructing an ML-hGRN operating above a biological pathway or a biological process. The algorithm first divided construction of ML-hGRN into multiple regression tasks; each involves a regression between a pathway gene and all TFs. Random forest models with backward elimination were used to determine the importance of each TF to a pathway gene. Then the importance of a TF to the whole pathway was computed by aggregating all the importance values of the TF to the individual pathway gene. Next, an expectation maximization algorithm was used to cut the TFs to form the first layer of direct regulatory relationships. The upper layers of GRN were constructed in the same way only replacing the pathway genes by the newly cut TFs. Both simulated and real gene expression data were used to test the algorithms and demonstrated the accuracy and efficiency of the method.

The third algorithm (Chapter 4) focuses on Joint Reconstruction of Multiple Gene Regulatory Networks (JRmGRN) using gene expression data from multiple tissues or conditions. In the formulation, shared hub genes across different tissues or conditions were assumed. Under the framework of the Gaussian graphical model, JRmGRN method constructs the GRNs through maximizing a penalized log-likelihood function. It was formulated as a convex optimization problem, and then solved it with an alternating direction method of multipliers (ADMM) algorithm. Both simulated and real gene expression data manifested JRmGRN had better performance than existing methods.
1 Introduction

In a gene regulatory network, transcription factors (TFs), RNA and other small molecules act as regulators to activate or repress the expression levels of other genes. Gene regulation can occur in a form of direct physical binding of proteins to their targets’ gene promoter sequences, and result in the activation or repression of target genes. TFs also act as indirect translational regulators, binding to the mRNA of other regulatory proteins [1]. Although experiments can be used to identify direct regulatory relationships, it is still labor-intensive and time-consuming, the use of computational algorithms to identify regulatory relationships from terabits of gene expression data offers a much more time and cost efficient way to identify regulatory relationships. The major challenges for the computational approach lie in: (1) the difficulty in identifying the direct regulations from indirect ones; (2) high dimensionality data due to large number of variables (genes); (3) multicollinearity property engendered by large number of genes; (4) high noise data generated from high-throughput technologies; (5) small sample size. In the past decade, an enormous wealth of data has been generated with the advances in microarray and RNA-Seq technologies. Though various statistical and machine learning methods have been developed to analyze these data to infer qualitative and quantitative relationships between genes. Highly efficient methods are still demanded to infer the regulatory relationships with increased accuracy so that the results can facilitate biologists to carry out experimental validation.

In reality, the relationships between genes are dynamic in nature and they can change over time, differ in different tissues or in reaction to external stresses. Therefore, when modeling gene networks, the choice of modeling paradigm is largely dependent on the type and quality of data available, relevant biological questions to be addressed, and statistical and computational efficiency and feasibility. In this introductory chapter, I will provide some biological and computational background, and review some relevant literature of my research.

1.1 Acquisition of gene expression data

1.1.1 Gene expression

The famous double-helix structure of deoxyribonucleic acid (DNA) was discovered by James Watson and Francis Crick in 1953 [2]. DNA consists of two long polymers of nucleotides bases, with backbones made of sugars and phosphate groups joined by ester bonds. There are basically four nucleotide bases: Adenine (A), Guanine (G), Thymine (T) and Cytosine(C). In the double helical structure of DNA, each base has its complementary base. That is, A has T as its complementary base while G has C as its complementary base or vice versa. A complete DNA molecule, also called a chromosome, is very long. For example, there are 22 pairs chromosomes plus X and Y chromosomes with a total of 3.1 billion nucleotides in a human cell. These chromosomes harbor genes that occupy less than
10% regions of all chromosomes. The genes’ sequences are like command languages that instruct cells to manufacture particular proteins. The length of a gene usually varies from a few hundred to a few thousand nucleotides. When a protein is needed, the gene is transcribed into corresponding ribonucleic acid (RNA). The RNA is then processed so that non-coding parts are removed before it is transported from a nucleus into the cytoplasm, where the RNA is translated into a protein. During the translation process, every three nucleotides in an RNA constitute a genetic codon that is used to specify an amino acid. A DNA strand thus contains specifically ordered genetic codons that are translated into a specific protein with unique functions. This process is called the central dogma of life (Figure 1.1). Non-protein-coding genes are still transcribed to produce RNAs, but their RNAs are not translated into polypeptides. For either type of genes, the process of going from a DNA to a functional protein is known as gene expression. In any given cell, thousands of genes are expressed and work in concert to ensure the cell’s functions, fitness, and survival. Each gene, in turn, must be expressed at the proper time and in the proper amounts to ensure the appropriate functional outcome. Therefore, through measuring the levels of RNA transcripts or protein levels, we can get an understanding of the regulations and interactions among genes. However, due to the lack of efficient methods for measuring the abundance of a large number of proteins in a cell or tissues, the methods developed in this study are trained and tested with transcriptome, but they are applicable to proteomic data in the future.

![Central Dogma Diagram](https://via.placeholder.com/150)

**Figure 1.1 central dogma of life**

### 1.1.2 Microarray and RNA-Seq technology

DNA microarray and RNA-Seq are two technologies that can be used to obtain the expression data of all genes at one effort. DNA microarrays are made of nylon or glasses on which the gene-specific DNA fragments of 25 to 70 nt, also called probes, are synthesized into spots, with each spot represents a known DNA sequence or gene. These slides are also commonly known as gene chips or DNA chips. The gene-specific DNA fragments attached to each slide serve as probes to detect gene expression levels through hybridizing to complementary DNAs (cDNAs) that are synthesized from mRNAs. To perform a microarray analysis, mRNA molecules are typically collected from both a set of experimental samples and a set of reference samples. The mRNA in these samples are then converted into cDNAs, and cDNAs in each sample are labeled with a fluorescent dye of a specific color. For instance, the experimental cDNAs are labeled with a red fluorescent dye, whereas the reference cDNAs are labeled with a green fluorescent dye. The two samples are then mixed together to hybridize with the probes on microarray chips, a process in which the cDNA molecules bind to their respective DNA probes of complementarity.
Following hybridization, the microarray is scanned to quantify each gene’s the expression level in the form of the signal intensity of fluorescent dye. The data yielded from microarray need to undergo background correction, normalization, and summarization before the normalized data can be obtained and pooled to create gene expression profiles, which show simultaneous changes in the expression of many genes in response to a particular condition or treatment. RNA sequencing (RNA-Seq) is a newer technology that uses the high-throughput next-generation sequencers to obtain an estimate of each gene’s mRNA abundance in a sample. RNA-Seq technology eliminates many challenges posed by hybridization-based microarrays approaches and provides far higher coverage and greater resolution of the dynamic nature of the transcriptome. A typical RNA-Seq experiment consists of isolating RNAs, converting it to cDNA, preparing the sequencing library, and sequencing the sequence tags in each library on a next-generation sequencer.

### 1.1.3 Microarray data normalization

The performance of genome-wide gene regulatory network reconstruction algorithms depends on the sample size. It is generally considered that the larger the sample size is, the truer relationships the gene network constructed contain. However, the sample size in a typical Microarray and RNA-Seq experiment is usually small. We need to collect multiple experiments that are under similar experimental condition. The strategy of pooling data from independent investigations holds many benefits including increased statistical power, greater sample heterogeneity and the ability to estimate a variety of models [3]. Most microarray data are stored in the CEL file format. A CEL file is a data file created by Affymetrix DNA microarray image analysis software. The affy package [4] in Bioconductor provides many methods to process CEL files. The main steps are listed in following.

1. download the CEL files to a single folder. The CEL file describes the intensities determined for every feature on a chip, without providing information about which probes correspond to which probe sets (such information provided by the CDF file).
2. use the function “ReadAffy” to read the raw data with an extension of “.CEL” into an R object.
3. use function “expresso” to transform raw probe intensities to expression values. In this function, parameter “bgcorrect.method” can be set to “RMA (Robust Multi-Array)”, parameter “normalize.method” can be set to “quantile”, parameter “pmcorrect.method” is usually set to “pmonly”, parameter “summary.method” is usually set to “medianpolish” though you may choose another method for each parameter.

### 1.1.4 RNA-Seq data analysis workflow

For RNA-Seq data, some authors provide the read count data, we can download them directly from the GEO database. Some authors provide only the raw sequencing data files in fastq format, so we need to do an alignment to get read count data. Many workflows are
available to do this work, see [5-7]. I prefer to use Bioconductor’s Rsubread package [8] to do read alignment and count quantification. The main steps are listed in following.

1. use “fastq-dump” tools to download the FASTQ files. The FASTQ files containing the raw sequence reads were deposited to the Sequence Read Archive (SRA) repository.
2. use function “qualityScores” to check the quality scores of FASTQ files.
3. download a reference genome and use function “buildindex” to build an index for the reference genome.
4. use function “align” to align reads, we get many BAM files after this command.
5. to quantify read counts, we need to prepare an annotation file, then use function “featureCounts” to generate the count matrix. If the chromosome names used in BAM files are different from the chromosome names used in the annotation file, we must prepare a mapping file.

After we get the read count matrix, we can use the quasi-likelihood functionality of edgeR [9] package to do differential expression analysis. The main steps are listed in following.

1. use function “DGElist” to construct a DEGList object.
2. filter out low expressed genes. One simple method to do this is to choose a cutoff based on the median log-transformed counts per gene per million mapped reads (CPM).
3. use function “calcNormFactors” to do normalization for composition bias.
4. use function “estimateDisp” to model the variance of reading count per gene.
5. use function “exactTest” to get the differentially expressed genes.

1.2 Types of gene regulatory network

The enormous wealth of data generated by microarray and RNA-Seq technologies enable applying computational methods to infer gene regulatory relationships and then construct gene regulatory networks (GRNs), from which we can extract meaningful information like network motifs, modules, and subnetworks for inferring novel biological knowledge. Scientists apply various computational tools to learn different types of GRNs. Figure 1.2 shows three kinds of gene regulatory network.

1.2.1 Relevance network

Gene relevance network is also called a correlation network or co-expression network. It is an undirected graph, where each node corresponds to a gene, and a pair of nodes is connected with an edge if there is a significant correlation relationship between them, see Figure 1.2. Having gene expression profiles of a number of genes for a certain number of samples, a gene correlation network can be constructed by two steps: 1) the computation of all pairwise correlations for the genes, and 2) a thresholding or filtering procedure to identify significant correlations, each representing an edge, of the network. Pearson correlation [10, 11], Spearman rank correlation [12] and mutual information [13] are
commonly used to compute the relevance. The thresholding can be an assigned number of top relationships [14] or a $p$-value computed from a statistical test procedure. Gene correlation networks are of biological interest since co-expressed genes are considered to be the genes under the same transcriptional regulatory mechanism, and they are often functionally related ones or members of the same pathway/process or protein complex. However, relevance network may contain spurious relationships, which are resulted from coincident correlation owing to small sample size, or indirect regulatory relationship.

![Diagram of a true network and its corresponding relevance network](image1.png)

Figure 1.2 A true network and its corresponding relevance network

![Diagram of a true network and its corresponding partial correlation network](image2.png)

Figure 1.3 A true network and its corresponding partial correlation network

### 1.2.2 Partial correlation network

To deal with the drawback of indirect regulatory relationships in the correlation network, partial correlation networks were proposed [15]. In a partial correlation network, the indirect associations are removed. Partial correlation network is also represented by an undirected graph, where each node corresponds to a gene, and if there is no edge between two genes, then the two genes are conditionally independent given all other genes. Figure 1.3 shows a true network and its corresponding partial correlation network. Node B and node D are connected, because node B, node C and node D form a V structure. In a Gaussian multivariate distribution, the inverse of the covariance matrix, i.e. the precision matrix, corresponds to a partial correlation network. Because the number of genes is
usually much larger than the sample size, it is impossible to compute the partial correlation coefficient exactly. There are three approximate methods to construct partial correlation network: 1) use regularization [16-18]; 2) use one-order or two-order partial correlation networks as an alternative method [19]; 3) use regression coefficients as an approximation. This is because partial correlation coefficients are closely related to the multivariate regression coefficients [20]. Some regression techniques in statistical and machine learning can be used to reconstruct this kind of partial correlation network.

![Figure 1.4 A true network and its corresponding causal network](image)

**1.2.3 Causal network**

Causality is a fundamental notion in science and plays an important role in explanation, prediction and decision making. The causal network can not only remove the indirect associations but also distinguish response variables (effect) from covariates (cause). The causal network is represented by a directed graph, each directed edge points from cause to effect, see Figure 1.4. The causal network can offer deeper insight than undirected correlation relationships. Understanding causality is the most important and indispensable step towards the goal of effectively controlling and optimizing genetic engineering. A traditional way to discover causal relations is to use interventions or randomized experiments. As more and more data accumulate, revealing causal information by analyzing purely observational data has drawn more attention. The algorithms for causal discovery can be grouped into three broad categories: constraint-based, score-based algorithms and hybrid methods. For constraint-based algorithms that include PC and fast causal inference (FCI)[21], the conditional independence relationships in the data have been exploited to recover the underlying causal structure. Zhang et al. [22] use the PC algorithm and conditional mutual information to infer a causal network. Score-based algorithms assign a score to each candidate network for measuring how well the candidate network fits dataset [23, 24]. Greedy equivalence search (GES) algorithm [25] is one of the prominent examples of score-based algorithms. Hybrid methods learn the causal structure by combining constraint-based and score-based methods. Typically, hybrid methods estimate the skeleton of the causal structure using conditional independence tests before a search and score technique are applied while restricting the set of allowed edges to the estimated skeleton.
1.3 Computational tools for constructing gene regulatory networks

Gene expression dataset usually takes on the form of a tabular matrix with $p$ genes arranged in rows and their expression levels measured under $n$ experimental conditions in columns. A difficulty of inferring GRNs from gene expression data lies in their high dimensionality. That is, the number of genes ($p$) is often much larger than the number of samples ($n$), which poses many estimation and computation challenges. In this dissertation, we used the methods from three types of computational tools (statistics, machine learning and convex optimization) to reconstruct GRNs using high-throughput gene expression data (Figure 1.5). For each type of the three computational tools, there are multiple methods available for being potentially used to reconstruct GRNs from gene expression data. The six methods that are highlighted red in Figure 1.5 were the focuses of this dissertation.

![Diagram of computational tools]

Figure 1.5 Potential computational methods for reconstruction of GRNs.

### 1.3.1 Statistical tools

The fundamental of statistics is to use probability theory to infer population parameters using one or multiple sample data sets. Therefore, when we have gene expression sample data, we can use statistical methods to infer the regulatory relationships among genes.
Under certain conditions, we can even infer causal relationships. The challenge is that all gene expression data sets are high-dimensional, which requires the development of efficient and accurate algorithms.

1.3.1.1 Correlation-based methods

Using correlation analysis, we can infer undirected gene co-expression networks. Co-expression analysis is one of the earliest methods used to infer edges in a gene network and is based on the idea that genes that have similar expression profiles under different experimental conditions are likely to be co-regulated under the same genetics mechanism and hence are functionally relevant for being involved in the same pathway or biological process, or interacts with each other possible in a protein complex. Pearson correlation (eq. 1.1) is the most popular co-expression measure used in the literature.

$$r_p = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \quad (1.1)$$

When Pearson correlation coefficients are computed, either hard [14] or soft thresholding [26] can be applied to generate gene co-expression networks. Because noise and outliers are unavoidable in gene expression data, rank correlations such as Spearman rank correlation (eq. 1.2) can provide more robust and less sensitive solution compared to Pearson correlation. Though some information is lost in the process of converting numerical values to ranks, Spearman rank correlation can assess both the linear relationship and nonlinear monotonic relationship between two variables.

$$r_s = 1 - \frac{6 \sum_{i=1}^{n}(rank(x_i) - rank(y_i))^2}{n(n^2 - 1)} \quad (1.2)$$

Kumari et al. [12] compared eight methods for computation of the association, and find Spearman correlation is one of the best methods in the eight methods. In order to discover the nonlinear association, mutual information (eq. 1.3) has also been used to generate gene co-expression networks [27].

$$I(X, Y) = \sum_{x,y} P(x, y) \log \left( \frac{P(x, y)}{P(x)P(y)} \right) \quad (1.3)$$

1.3.1.2 Linear regression and shrinkage

The process of reconstruction of GRN is to select appropriate regulators for each pathway gene. Thus, linear regression methods fit this process well. By retaining a subset of the best predictors and discarding the rest, subset selection produces a model that is interpretable and has possibly lower prediction error than full model. However, because it is a discrete process (variables are either retained or discarded), it often exhibits high variance. Shrinkage methods are more continuous and don’t suffer much from high variability. Least
absolute shrinkage and selection operator (LASSO) [28] is most commonly used for gene regulatory network inference as it implements subset selection and regularization simultaneously. Many extensions of LASSO appear in the literature. Zhang et al. [29] propose to reconstruct GRN by solving the following robust LASSO problem (1.4).

$$\min_\beta \left( |y - X\beta|_1 + \lambda |\beta|_1 \right)$$  \hspace{1cm} (1.4)

where $y$ is the expression vector of the target gene, $X$ is the expression matrix of candidate transcription factors, $\lambda$ is a positive tuning parameter. Qin et al. [30] applied an iterative thresholding algorithms to solve the following $\ell_p(p = 1, \frac{1}{2}, 0)$ regularization models for gene regulatory network inference.

$$\ell_1: \min_\beta \left( |y - X\beta|_1^2 + \lambda |\beta|_1 \right)$$  \hspace{1cm} (1.5)

$$\ell_{\frac{1}{2}}: \min_\beta \left( |y - X\beta|_2^2 + \lambda |\beta|_{\frac{1}{2}} \right)$$  \hspace{1cm} (1.6)

$$\ell_0: \min_\beta \left( |y - X\beta|_2^2 + \lambda |\beta|_0 \right)$$  \hspace{1cm} (1.7)

They compared the performance of these three regularization models and showed $\ell_{1/2}$ and $\ell_0$ have better performance than $\ell_1$.

An assumption that we make in the reconstruction of gene regulatory networks is that each observation is drawn from the same distribution. However, when the gene expression data come from different tissues or under different treatments, this assumption is inappropriate. In this case, if one insists on modeling the gene expression data by one GRN, the results would be dubious and we cannot obtain the differential network we are interested in. A straightforward method to obtain the differential network is to reconstruct the network of each condition separately and then find the difference between them. However, this procedure ignores the similarity shared between GRNs across different tissues/treatments, which is critically important to reconstruct the GRNs, especially when the sample size is small. Omranian et.al [31] proposed a fused LASSO method for reconstructing gene regulatory networks by simultaneous consideration of data sets from different perturbation experiments and corresponding controls. Suppose there are $K$ datasets under different conditions, they tried to solve the following problem:

$$\min_\beta \sum_{k=1}^{K} \left| |y^{(k)} - X^{(k)}\beta^{(k)}|_2 \right|_2^2 + \lambda_1 \sum_{k=1}^{K} \left| W^{(k)}\beta^{(k)} \right|_1 + \lambda_2 \sum_{k' < k''} \left| \beta^{(k')} - \beta^{(k'')} \right|_1$$  \hspace{1cm} (1.8)

where $W$s are weight matrices, and the third item in (1.8) encourages the regulatory relationships to be similar for different conditions. Liu et al. proposed a group lasso-based method for robustly inferring gene regulatory networks from multiple time-course datasets [32]. Suppose there are $m$ time-course gene expression datasets for a gene network: $\tilde{X}(1), ..., \tilde{X}(m)$, each of which is measured at $n_k + 1$ time points, i.e., $\tilde{X}(k) \in \mathbb{R}^{p \times (n_k + 1)}$, they solved the following optimization problem:
\[
\min_A \sum_{i=1}^{p} \sum_{k=1}^{m} w_k \sum_{j=1}^{n_k} H_\delta \left( y_{ij}(k) - A_i(k)^T x_j(k) \right) + \\
\lambda \sum_{i=1}^{p} \sum_{l=1}^{p} \sqrt{a_{il}(1)^2 + \cdots + a_{il}(m)^2}
\]

(1.9)

where \( Y(k) = [\tilde{X}_2(k), ..., \tilde{X}_{n_k+1}(k)] \) i.e. the last \( n_k \) observations; \( X(k) = [\tilde{X}_1(k), ..., \tilde{X}_{n_k}(k)] \) i.e. the first \( n_k \) observations; \( A(k) \in R^{p \times p} \) the regulatory matrix for the \( k^{th} \) dataset; \( H_\delta \) is the Huber loss function. We can see the second item in (1.6) is a group LASSO type penalization, and it encourage all \( A(k) \)'s to have the same zero and nonzero pattern.

1.3.1.3 Gaussian graphical models

Gaussian graphical models (GGMs) are widely used to reconstruct gene networks using gene expression data [19]. The models assume that gene expression data on \( p \) genes from each sample follows a multivariate normal distribution with mean \( \mu \) and covariance matrix \( \Sigma \), where \( \mu \) is a vector with \( p \) elements and \( \Sigma \) is a \( p \times p \) positive definite matrix. The conditional independence of two genes given other genes corresponds to a zero entry in the inverse covariance matrix \( \Sigma^{-1} \) (also called the precision or concentration matrix) [33]. Therefore, Gaussian graphical models have the advantage of reconstructing direct dependencies between genes that represent edges in the reconstructed network: an edge corresponds to a non-zero entry in \( \Sigma^{-1} \). A natural way to estimate \( \Sigma^{-1} \)'s by maximizing the log-likelihood of the data. Under the Gaussian distribution, the log-likelihood function can be expressed as the following:

\[
\log \det(\Sigma^{-1}) - \text{tr}(S\Sigma^{-1})
\]

(1.10)

where \( S \) is the sample covariance matrix, \( \det \) is the determinant of a matrix, \( \text{tr} \) is the trace of a matrix. Define \( \Theta = \Sigma^{-1} \), then maximizing (1.10) with respect to \( \Theta \) leads to the maximum likelihood estimate of precision matrix \( \hat{\Theta} \). However, directly applying GGM to reconstruct GRN is not applicable due to two problems. First, since the number of samples (\( n \)) is generally much less than the number of genes (\( p \)) from gene expression data, the sample covariance matrix becomes singular and thus it is impossible to computing the likelihood function. Second, even if the sample covariance matrix is not singular, the elements (supposed to be zero) in the estimated precision matrix \( \Sigma^{-1} \) are in general not exactly equal to zero. For these reasons, Yuan and Lin [34] proposed to maximize the L1 regularized log-likelihood.

\[
\log \det(\Sigma^{-1}) - \text{tr}(S\Sigma^{-1}) - \lambda ||\Theta||_1
\]

(1.11)

where \( \lambda \) is a nonnegative tuning parameter, \( ||\Theta||_1 \) is the sum of the absolute value of each element in \( \Theta \). This log-likelihood (1.11) can be solved by the graphical LASSO algorithm.
(GLASSO) [35]. When $\lambda$ is sufficiently large, we can get a sparse and positive definite estimate of $\Theta$ from (1.11).

In order to incorporate multiple samples that are under different conditions, Danaher et al. [16] proposed a joint graphical LASSO model (JGL) to estimate multiple GRNs simultaneously based on the fused graphical LASSO or the group graphical lasso. In their models, the penalized log-likelihood function is:

\[
\sum_{k=1}^{K} n_k \left( \log \det \Theta^{(k)} - \text{tr} \left( S^{(k)} \Theta^{(k)} \right) \right) - P(\{\Theta\})
\] (1.12)

where $K$ is the number of conditions; $n_k$, $\Theta^{(k)}$ and $S^{(k)}$ are the sample size, the precision matrix and the sample covariance matrix for condition $k$ respectively; $P(\{\Theta\})$ is the penalty function. For the fused graphical lasso, $P(\{\Theta\})$ is:

\[
P(\{\Theta\}) = \lambda_1 \sum_{k=1}^{K} \left\| \Theta^{(k)} - \text{diag}(\Theta^{(k)}) \right\|_1 + \lambda_2 \sum_{k<k'} \left\| \Theta^{(k)} - \Theta^{(k')} \right\|_1
\] (1.13)

where $\lambda_1$ and $\lambda_2$ are nonnegative tuning parameters. The first item encourages the off-diagonal elements of $\{\Theta^{(k)}\}$ to be sparse, the second item encourages $\{\Theta^{(k)}\}$ to be the same. For the group graphical lasso, $P(\{\Theta\})$ is:

\[
P(\{\Theta\}) = \lambda_1 \sum_{k=1}^{K} \left\| \Theta^{(k)} - \text{diag}(\Theta^{(k)}) \right\|_1 + \lambda_2 \sum_{k=1}^{K} \sqrt{\sum_{i=1}^{K} (\Theta^{(k)}_{ij})^2}
\] (1.14)

Here, the second item encourages $\{\Theta^{(k)}\}$ to have similar sparsity pattern. Xie et al. decomposed the problem of reconstruction of multiple networks simultaneously into two graphical layers, the systemic layer which affects all outcomes and thereby induces cross-graph dependence, and the category-specific layer, which represents graph-specific variation [36]. They used a graphical expectation-maximization technique to estimate both layers jointly.

The above-mentioned methods do not impose any structural information of gene networks. That is, each gene has approximately the same number of interactions within the network, and each pair of nodes has equal probability to be an edge and all edges are independent. However, recent evidence points to scale-free properties in biological networks [37, 38], in which most genes interact with a few partners whereas a small proportion of genes, called hub genes, are densely-connected to many other genes (high connectivity). To incorporate hub genes in GRNs, Liu and Ihler [39] replaced the $l_1$ regularization in (1.11) with a power law regularization and optimized the objective function by solving a sequence of iteratively reweighted $l_1$ regularization problems, where the regularization coefficients of nodes with high degree were reduced, which encouraged the appearance of hub genes. Tan et al. [18] introduced a row-column overlap norm penalty to incorporate hub genes.
explicitly. In their model, called hub graphical LASSO (HGLASSO), the penalized log likelihood function is:

\[
\begin{aligned}
&\log \det \Theta - \text{tr}(S \Theta) + \lambda_1 |Z - \text{diag}(Z)|_1 + \lambda_2 |V - \text{diag}(V)|_1 \\
&+ \lambda_3 \sum_{j=1}^{p} \left( |V - \text{diag}(V)|_j \right)_q \text{ subject to } \Theta = V + V^T + Z
\end{aligned}
\]  

(1.15)

In this formulation, the precision matrix \( \Theta \) was decomposed into two parts, namely \( Z \) and \( V \), where \( Z \) is a symmetric matrix that is encouraged to be sparse, \( V \) is a matrix whose columns are encouraged to be either entirely zero or almost entirely non-zero through the \( l_1/l_q \) norm penalization. The non-zero columns of \( V \) correspond to hub genes.

### 1.3.2 Machine Learning tools

Many tools in machine learning are very related to statistics, but there are some differences. As stated by Breiman in [40]: statistics relies heavily on data modeling and emphasizes model and assumption checking, while machine learning relies on algorithmic modeling and put more emphasis on the predictive accuracy of models. A large number of machine learning methods have been used to reconstruct GRN from gene expression data.

#### 1.3.2.1 Support vector machine

Support vector machine (SVM) has attracted much interest within the bioinformatics community because of its good prediction performance for various tasks. The idea is to construct an optimal hyperplane between two classes +1 and -1 such that the distance from the closest point to the hyperplane, or the margin, is maximized. The primal form of SVM is

\[
\begin{aligned}
&\min_{w,p,s} \frac{1}{2} ||w||_2^2 + C \sum_{i=1}^{n} s_i \\
&s.t. \ y_i(wx_i + b) \geq 1 - s_i, \ s_i \geq 0, \ i = 1, ..., n
\end{aligned}
\]  

(1.16)

The dual form of (1.16) is

\[
\begin{aligned}
&\max_{\alpha} - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{n} \alpha_i \\
&s.t. \ y^T \alpha = 0, \ 0 \leq \alpha_i \leq C, \ i = 1, ..., n
\end{aligned}
\]  

(1.17)

To allow for nonlinear classification, the inner product \( x_i^T x_j \) in (1.17) can be replaced by kernel functions \( k(x_i, x_j) \), which can be thought of as special similarity measures. They implicitly map the original data into some high dimensional feature space, in which the
optimal hyperplane can be found. There are many kernel functions available; three mostly used kernel functions include:

- **Linear kernel**: \( k(x_i, x_j) = x_i^T x_j + c \)
- **Polynomial kernel**: \( k(x_i, x_j) = (x_i^T x_j + c)^d \)
- **Gaussian kernel**: \( k(x_i, x_j) = \exp\left(-\gamma ||x_i - x_j||^2\right) \)

Gillani et al. [41] used simulated gene expression data and corresponding regulatory relationships generated by GeneNetWeaver software to compare different kernel functions for inference of GRNs.

### 1.3.2.2 Decision tree and random forest

A decision tree is a method commonly used in machine learning. The goal is to create a model that predicts the value of a target variable based on several input variables. Algorithms for constructing decision trees usually work in a top-down manner, by choosing a variable at each step that best splits the set of items. Different algorithms use different metrics for measuring "best". They generally measure the homogeneity of the target variables within the subsets. Three commonly used metrics are given below. These metrics are applied to each candidate subset, and the resulting values are combined to provide a measure of the quality of the split. Assuming \( S = \{x_1, ..., x_n\} \) be a set of items with \( J \) classes, let \( p_i \) be the fraction of items labeled with class \( i \) in the set, then

- **Gini impurity**: \( Gini(p) = \sum_{j=1}^{J} p_j \sum_{k \neq j} p_k \)
- **Entropy**: \( H(p) = -\sum_{j=1}^{J} p_j \log p_j \)
- **Variance**: \( V(S) = \frac{1}{n^2} \sum_{i \in S} \sum_{j \in S} \frac{1}{2} (x_i - x_j)^2 \)

Decision tree model features human explainable and high efficiency, but it has high prediction variance and is prone to overfitting. Random forest is an effective method to decrease variance and avoid overfitting. Random forest uses bootstrap to generate a random subset of samples from original data, and then construct an individual decision tree using random feature (gene) selection. After the building of each decision tree, all the results from these trees are aggregated to produce the final result. GENIE3 [42] is a random forest algorithm for the inference of GRNs that was the best performer in the DREAM4 In Silico Multifactorial Challenge.

### 1.3.2.3 Bayesian networks

Bayesian network is a promising tool for reconstruction of GRN from expression data. First, it is particularly useful for describing processes composed of locally interacting components, that is, the value of each component directly depends on the values of a relatively small number of components. Second, statistical foundation for learning
Bayesian network from observations, and computational algorithms to do so are well understood and have been used successfully in many applications. Finally, Bayesian network provides models of causal influence: Although Bayesian network is mathematically defined strictly in terms of probabilities and conditional independence statements, a connection can be made between this characterization and the notion of direct causal influence [43].

Consider a finite set $\mathcal{X} = \{X_1, \ldots, X_p\}$ of random variables, a Bayesian network is a representation of a joint probability distribution of $\mathcal{X}$. There are two components in this presentation. First, a directed acyclic graph (DAG) $G$, each vertex of $G$ corresponds to a random variable, a directed edge corresponds to a direct causal influence; Secondly, a conditional distribution $\Theta$ for each variable, given its parents in $G$. The graph $G$ represents conditional independence assumptions that allow the joint distribution to be decomposed, that is

$$P(X_1, \ldots, X_p) = \prod_{i=1}^{p} P(X_i | Pa^G(X_i))$$

(1.18)

where $Pa^G(X_i)$ is the set of parents of $X_i$ in $G$. For instance, a Bayesian network is shown in Figure 1.6.

![Bayesian network example](image)

Figure 1.6 An example of a Bayesian network

The joint probability distribution of Figure 1.6 is

$$P(A, B, C, D, E) = P(A)P(B)P(C|A, B)P(D|B)P(E|C)$$

(1.19)

Using Bayesian network to reconstruct GRN means to learn a Bayesian network structure from gene expression data. Learning consists of two parts, one is a scoring function for a given network; the other is a method to explore the space of all networks. Friedman et al. [43] adopted a sparse candidate algorithm to facilitate efficient explore the network's space and implement reconstruction of GRN.

### 1.3.3 Convex optimization tools

Many models in statistics and machine learning can be formulated as a mathematical optimization problem,
\[
\begin{aligned}
\min_x f(x) \\
\text{s.t.} & \quad g_i(x) \leq 0, \quad i = 1, \ldots, m \\
& \quad h_j(x) = 0, \quad j = 1, \ldots, r
\end{aligned}
\]  

(1.20)

Generally, (1.20) is very difficult to solve, but a class of optimization problems, convex optimization problems, are the exception, which can be solved efficiently and reliably. The optimization problem (1.20) is a convex optimization problem if:

1. \( f \) and \( g_i, i = 1, \ldots, m \), are convex functions.
2. \( h_j, j = 1 \ldots r \), are affine functions.

The hierarchy of the convex optimization problem is shown in Figure 1.7. Although there have been many general purpose software packages available for a specific class of convex optimization problem, we still need to study how to solve it, because: (1) different algorithms can perform better or worse for different problems; (2) studying solving (1.20) can give us a deeper understanding of the original models.

![Figure 1.7 The hierarchy of convex optimization problem](image)

1.3.3.1 Gradient, subgradient, and proximal gradient methods

Gradient descent methods are used to solve unconstrained, smooth convex optimization problem: \( \min_x f(x) \). It works as follows:

1. Choose an initial point \( x^{(0)} \in \mathbb{R}^n \).
2. Repeat: \( x^{(k)} = x^{(k-1)} - t_k \nabla f(x^{(k-1)}), k = 1, 2, \ldots \)
3. Stop when \( ||\nabla f(x^{(k-1)})|| \) is small enough.

Where \( t_k \) is step size. It can be a fixed small number or determined by backtracking line search. When \( \nabla f \) is Lipschitz continuous, gradient descent has convergence rate \( O(\frac{1}{k}) \). If \( f \)
has strong convexity, gradient descent has convergence rate $O(\log \left( \frac{1}{\epsilon} \right))$ for some $0 < c < 1$. Das et al. [44] used a gradient descent method to solve a structural equation model and identified a set of optimal gene regulatory pathways.

When $f(x)$ is not differentiable, gradient descent method is not applicable. We can use the subgradient method instead of gradient descent method to solve an unconstrained optimization problem. Subgradient method is similar to the gradient descent method, but the gradient $\nabla f(x^{(k-1)})$ in step 2 is replaced by a subgradient $g(x^{(k-1)})$. The definition of subgradient is as follows:

**Definition** [45]: A subgradient of function $f$ at $x$ is any $g \in \mathbb{R}^*$, such that

$$f(y) \geq f(x) + g^T(y - x)$$

for all $y$.

The convergence rate of the subgradient method is $O\left( \frac{1}{\epsilon^2} \right)$, which is much slower than $O\left( \frac{1}{\epsilon} \right)$, the rate of the gradient descent method.

Suppose the objective function can be decomposed as $f(x) = g(x) + h(x)$, where $g(x)$ is a convex differentiable function and $h(x)$ is a convex non-differentiable function. Many LASSO based models have this form. Usually, $g(x)$ is a differentiable loss function and $h(x)$ is the sparsity induced penalization. We can use proximal gradient descent method to get a $O\left( \frac{1}{\epsilon} \right)$ convergence rate. The idea behind proximal gradient descent method is to make a quadratic approximation to $g(x)$, and leave $h(x)$ unchanged. That is,

$$f(z) = g(z) + h(z) \approx g(x) + \nabla g(x)^T(z - x) + \frac{1}{2t}||z - x||_2^2 + h(z) \quad (1.21)$$

At each step, we update $x$ by the minimum of the right side of (1.21).

$$x^+ = \arg\min_z g(x) + \nabla g(x)^T(z - x) + \frac{1}{2t}||z - x||_2^2 + h(z)$$

$$= \arg\min_z \frac{1}{2t}||z - (x - t\nabla g(x))||_2^2 + h(z)$$

The operator $\text{Prox}_{t,h}(x) = \arg\min_z \frac{1}{2t}||z - x||_2^2 + h(z)$ is called proximal mapping. For many $h(z)$, we can get the analytical solution of $\text{Prox}_{t,h}(x)$. Gui et al. [46] developed a proximal gradient descent method to identify genome-wide GRN structures, and their algorithm can scale to network size on the order of $10^4$.

1.3.3.2 Newton and barrier methods

Newton’s method is used to solve unconstrained, twice differentiable convex optimization problem: $\min_x f(x)$. It works as follows:

1. Choose an initial point $x^{(0)} \in \mathbb{R}^n$. 

   Choose an initial point $x^{(0)} \in \mathbb{R}^n$. 

1. Choose an initial point $x^{(0)} \in \mathbb{R}^n$.
2. Repeat: \( x^* = x + tv \), where

\[
v = \arg \min_z f(x) + \nabla f(x)^T z + \frac{1}{2} z^T \nabla^2 f(x) z
\]

i.e. \( x^{(k)} = x^{(k-1)} - t_k (\nabla^2 f(x^{(k-1)}))^{-1} \nabla f(x^{(k-1)}) \), \( k = 1, 2, \ldots \)

3. Stop when \( |\nabla f(x^{(k-1)})| \) is small enough.

Here, \( \nabla^2 f(x^{(k-1)}) \) is the Hessian matrix of \( f \) at \( x^{(k-1)} \). If \( f(x) \) satisfies the following additional conditions: (1) \( \nabla f \) is Lipschitz with parameter \( L \); (2) \( f \) is strongly convex with parameter \( m \); (3) \( \nabla^2 f \) is Lipschitz with parameter \( M \), then Newton’s method satisfies the following two-stage convergence bounds [47],

\[
f(x^{(k)}) - f(x^*) \leq \begin{cases} 
(f(x^{(0)}) - f(x^*)) - \gamma k & \text{if } k \leq k_0 \\
\frac{2m^3}{M^2} (\frac{1}{2})^{k-k_0+1} & \text{if } k > k_0
\end{cases}
\]

We can see the convergence of Newton’s method follows two stages. During the first stage, the value of objective function decreases at least \( \gamma \) for each iteration; During the second stage, Newton’s method has quadratic convergence rate, which is much faster than gradient descent method.

For a quality constraint optimization problem,

\[
\min_x f(x) \quad \text{s.t.} \quad Ax = b \tag{1.22}
\]

The updated direction of step (2) of former Newton’s method changes to

\[
v = \arg \min_{z:A(x+z)=b} f(x) + \nabla f(x)^T z + \frac{1}{2} z^T \nabla^2 f(x) z
\]

From the KKT condition, it follows \( v \) satisfies

\[
\begin{pmatrix} \nabla^2 f & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} v \\ w \end{pmatrix} = - \begin{pmatrix} \nabla f(x) \\ Ax - b \end{pmatrix}
\]

Barrier method is capable of solving inequalities constraint optimization problem,

\[
\begin{cases}
\min_x f(x) \\
s.t. \quad Ax = b \\
h_i(x) \leq 0 \\
i = 1, \ldots, m
\end{cases} \tag{1.23}
\]

the idea of Barrier method is to approximate the original problem with

\[
\begin{cases}
\min_x f(x) + \frac{1}{t} \phi(x) \\
s.t. \quad Ax = b
\end{cases} \tag{1.24}
\]

where \( \phi(x) = -\sum_{i=1}^m \log (-h_i(x)) \) is called barrier function. (1.24) can be solved by Newton’s method, then let \( t \to +\infty \), we can get the solution of (1.23). Yip et al. [48] proposed a method to reconstruct GRN that turned out to be a non-convex optimization
problem. They used Newton’s method to compute local minimums for different initial values.

1.3.3.3 Dual ascend method and ADMM algorithm

Newton’s method and barrier method need to compute the second differential matrix and solve a system of equations. For the large-scale problems, including reconstruction of GRNs, the first-order method is more applicable. Dual gradient ascend method and ADMM algorithm belong to this kind of first-order methods, they are widely used in large-scale optimization problems.

For the quality constraint optimization problem (1.22), dual gradient ascend method works as follows:

- starts with an initial dual guess \( u_0 \)
- repeats for \( k = 1, 2, 3 \ldots \), until converged
  1. \( x^{(k)} = \text{argmin}_z f(z) + (u^{(k-1)})^T A z \)
  2. \( u^{(k)} = u^{(k-1)} + t_k (A x^{(k)} - b) \)

The convergence rate of the dual gradient ascend method is the same as the gradient descend method. That is, if \( f \) is strongly convex, then dual gradient ascent method converges at the rate \( O\left(\frac{1}{k}\right) \); if \( f \) is strongly convex, and \( \nabla f \) is Lipschitz continuous, then dual gradient ascent method converges at the rate \( O(\log\left(\frac{1}{\epsilon}\right)) \).

A disadvantage of dual ascent method is that it requires strong convex condition to ensure convergence. Augmented Lagrangian method, also known as method of multipliers, transforms optimization problem (1.22) as

\[
\begin{align*}
\min_{x} &\quad f(x) + \frac{\rho}{2} ||Ax - b||^2 \\
\text{s.t.} &\quad Ax = b
\end{align*}
\]  

(1.25)

where \( \rho > 0 \) is a parameter. Clearly (1.25) is equivalent to the original problem (1.22), and the objective function is strongly convex when A has full column rank. Therefore, we can use dual gradient ascend method to solve (1.25).

The advantage of the augmented Lagrangian method is that it has much better convergence properties, but the disadvantage is that it loses decomposability. Alternating direction method of multipliers algorithm (ADMM algorithm) has the best of both worlds, i.e., it obtains strong convergence properties, along with decomposability. Consider

\[
\begin{align*}
\min_{x,z} &\quad f(x) + g(z) \\
\text{s.t.} &\quad Ax + Bz = c
\end{align*}
\]  

(1.26)

As the augmented Lagrangian method, the objective of (1.26) can be augmented as,
\[
\begin{aligned}
&\left\{ \min_{x,z} f(x) + g(z) + \frac{\rho}{2} \| Ax + Bz - c \|^2 \\
&\quad \text{s.t. } Ax + Bz = c
\end{aligned}
\]

(1.27)

The augmented Lagrangian of (1.27) is,

\[
L_p(x, z, u) = f(x) + g(z) + u^T (Ax + Bz - c) + \frac{\rho}{2} \| Ax + Bz - c \|^2
\]

ADMM repeats the steps, for \( k = 1, 2, 3, \ldots \), until converged

- \( x^{(k)} = \text{argmin}_x L_p(x, z^{(k-1)}, u^{(k-1)}) \)
- \( z^{(k)} = \text{argmin}_z L_p(x^{(k)}, z, u^{(k-1)}) \)
- \( u^{(k)} = u^{(k-1)} + \rho (Ax^{(k)} + Bz^{(k)} - c) \)

It is often easier to express the ADMM algorithm in a scaled form, where the dual variable \( u \) was replaced by a scaled variable \( w = u/\rho \). In this parametrization, the ADMM steps are:

- \( x^{(k)} = \text{argmin}_x f(x) + \frac{\rho}{2} \| Ax + Bz^{(k-1)} - c + w^{(k-1)} \|^2 \)
- \( z^{(k)} = \text{argmin}_z g(z) + \frac{\rho}{2} \| Ax^{(k)} + Bz - c + w^{(k-1)} \|^2 \)
- \( w^{(k)} = w^{(k-1)} + Ax^{(k)} + Bz^{(k)} - c \)

ADMM algorithm is widely used in the reconstruction of GRN from gene expression data. It was used to solve the Gaussian graphical model (1.9) [16], and the joint Gaussian graphical model (1.12) [18].

### 1.4 Organization of this dissertation

The dissertation is organized as follows: In this first chapter, I gave a brief introduction to the background needed for this study, which includes using microarray and RNA-Seq techniques to obtain the gene expression data, the three types of GRNs, and the computational tools that are used to reconstruct GRNs. Following that, I provided an analytical overview of the literature published on the reconstruction of GRNs. In chapter 2, I introduced a regression-based method to model the transcription factor (TF) and target interactions. This method uses Huber function as loss function and Berhu function as regularization item. Huber function gives a square loss for small error and gives a linear loss for large error, which can be more robust to deal with outliers and non-Gaussian error distribution. Berhu function penalizes small coefficients linearly and penalizes large coefficients quadratically, posing a better balance between ridge regression and LASSO regression. The group effect of Huber and Berhu regression makes it a good choice for modeling TF target interactions. In chapter 3, A backward elimination random forest (BWERF) algorithm was developed to model hierarchical GRNs. BWERF is based on random forest algorithm, and it uses recursive backward elimination to alleviate the impact
of a large number of noise variables, and designed an expectation maximization (EM) like algorithm to determine the number of TFs in each layer. In chapter 4, a joint reconstruction multiple GRN (JRmGRN) algorithm was developed to jointly reconstruct multiple GRNs using multiple gene expression datasets from different tissues or environmental conditions. JRmGRN attempts to jointly estimate the precision matrices based on GGM model. A feature of JRmGRN is that common hubs are explicitly modeled in the algorithm.
2 HB-PLS: An algorithm for reconstruction of transcriptional regulatory networks by integrating Huber loss and Berhu penalty with Partial Least Square

In this chapter, an algorithm for identifying pathway regulators was developed by reconstructing a regulatory network that comprises of both transcription factors and pathway genes. Such an algorithm is instrumental for identifying regulators governing various pathways and biological processes and can be used widely in many circumstances to discover novel biological knowledge from high-throughput expression data. Like Ridge and LASSO regression, a linear relationship between TFs and each pathway gene was assumed in this algorithm, but the square loss was replaced by Huber loss, and Berhu function was used as penalty function. An efficient proximal gradient descent algorithm was developed to solve this optimization problem. To deal with the high dimension and multicollinearity property of gene expression data, this Huber loss and Berhu penalty regression was embedded into partial least square (PLS) regression framework to acquire a regulatory network. The method was named HB-PLS, which was applied to both simulated and real gene expression data, and higher accurate results were obtained as compared to existing comparable methods.

2.1 Introduction

In a gene regulatory network (GRN), a node corresponds to a gene, and an edge represents a directional regulatory relationship between a transcription factor (TF) and a target gene. Understanding the regulatory relationships among genes in GRNs can help elucidate the various biological processes and underlying mechanisms in a variety of organisms. Although experiments can be conducted to acquire the evidence of gene regulatory interactions, it is labor-intensive and time-consuming. Computational tools utilizing gene expression data offer a much more time and cost efficient way to reconstruct GRNs. In the past one and half decade, advances in microarray and RNA-Seq technologies have generated an enormous wealth of data to which mathematical and statistical tools can be applied to infer qualitative and quantitative relationships between genes.

Many authors have tried different methods to reconstruct GRNs. Many works have been done under the information-theoretic framework. Relevance network (RN) [49] is one of the earliest methods, it infers a network in which a pair of genes are linked by an edge if the mutual information is larger than a given threshold. The context likelihood relatedness (CLR) algorithm [50] is an extension of RN, it derives a score from the empirical distribution of the mutual information for each pair of genes and eliminates the edges with scores that are not statistically significant. ARACNE (Algorithm for the Reconstruction of Accurate Cellular Networks) [51] is similar to RN. However, ARACNE makes use of the data processing inequality (DPI) to eliminate the least significant edge of a triplet of genes,
which decreases the false positive rate of the inferred network. MRNET (maximum relevance/minimum redundancy Network) [52] employs the maximum relevance and minimum redundancy feature selection method to infer GRNs. Although information theory-based methods are used extensively for reconstructing GRNs because these kinds of methods have a low computational complexity and are able to capture nonlinear dependencies among variables. However, the use of mutual information has some disadvantages, including that it cannot differentiate a positive regulatory interaction (activating) from a negative one (inhibiting).

Reconstruction of the transcriptional regulatory network can be implemented by the neighborhood selection method. Neighborhood selection [20] is a sub-problem of covariance selection. Assume Γ is a set containing all the variables (genes), the neighborhood ne_a of a variable a ∈ Γ is the smallest subset of Γ\{a} such that, given all variables in ne_a, variable a is conditionally independent of all remaining variables. Given n i.i.d. observations of Γ, neighborhood selection aims to estimate the neighborhood of each variable in Γ individually. The neighborhood selection problem can be casted as a multiple linear regression problem and solved by regularized methods.

Following the differential equation in [53], the expression levels of a target gene y and the expression levels of the TF genes x form a linear relationship.

\[ y_i = \alpha^* + x_i^T \beta^* + \epsilon_i \quad i = 1, 2, ..., n \]  

(2.1)

where n is the number of samples, \( x_i = (x_{i1}, ..., x_{ip})^T \) is the expression level of p TF genes and \( y_i \) is the expression level of the target gene in sample \( i \). \( \alpha^* \) is the intercept and \( \beta^* = (\beta_1^*, ..., \beta_p^*)^T \) are the associated regression coefficients, if \( \beta_j^* \neq 0 \), then TF gene \( j \) regulates target gene \( i \). \( \{\epsilon_i\} \) are independent and identically-distributed random errors with mean 0 and variance \( \sigma^2 \). The method to get an approximation \( \hat{\beta} \) for \( \beta^* \) is to transform this statistical problem to a convex optimization problem.

\[ \hat{\beta} = \arg\min_\beta f(\beta) = \arg\min_\beta \sum_{i=1}^n L(y_i - \alpha - x_i^T \beta) + \lambda P(\beta) \]  

(2.2)

where \( L(\cdot) \) is loss function, \( P(\cdot) \) is penalization function, and \( \lambda > 0 \) is a tuning parameter which determines the importance of penalization. Different loss functions, penalization functions and methods for determining \( \lambda \) have been proposed in the literature. Ordinary least square (OLS) is the simplest method with square loss function \( L(y_i - \alpha - x_i^T \beta) = (y_i - \alpha - x_i^T \beta)^2 \) and without a penalization function. OLS estimator is unbiased. However, as it is common for the number of genes \( p \) being much larger than the number of samples \( n \), i.e. \( p \gg n \), in any given gene expression data, there is no unique solution for an OLS, even when \( n > p \), OLS estimation features high variance. To conquer these problems, ridge regression [54] adds a \( \ell_2 \) penalty \( P(\beta) = \sum_{j=1}^p \beta_j^2 \) on the coefficients, which introduces a bias but reduces the variance of the estimated \( \hat{\beta} \). In ridge regression, there is
a unique solution even for \( p > n \) case. LASSO [28] is similar to ridge regression, but the \( \ell_2 \) penalty in ridge regression was replaced by \( \ell_1 \) penalty \( P(\beta) = \sum_{j=1}^{p} |\beta_j| \).

The main benefit of the LASSO is that it performs variables selection and regularization simultaneously, which generates a sparse solution, a desirable property for constructing GRNs. When using LASSO for selecting regulatory TFs for a target gene, there are two limitations. First, if several TF genes are correlated and have large effects on the target gene, the LASSO has the tendency to choose only one TF gene while zeroing out the other TF genes. Second, some studies [55] state that the LASSO does not have the oracle properties, that is, the capability to (1) identifies the right subset of true variables; (2) Has optimal estimation rate. It is claimed that there are cases where a given \( \lambda \) that leads to optimal estimation rate ends up with an inconsistent selection of variables. For the first limitation, Zou and Hastie [56] proposed elastic net, in which the penalty is a mixture of LASSO and ridge \( P(\beta) = \alpha \sum_{j=1}^{p} |\beta_j| + \frac{(1-\alpha)}{2} \sum_{j=1}^{p} \beta_j^2 \), where \( \alpha (0 < \alpha < 1) \) is called the elastic net mixing parameter. When \( \alpha = 1 \), the elastic net penalty becomes LASSO penalty; when \( \alpha = 0 \), the elastic net penalty becomes ridge penalty. For the second limitation, adaptive LASSO [55] was proposed as a regularization method, which enjoys the oracle properties. The penalty function adaptive LASSO is \( P(\beta) = \sum_{j=1}^{p} \tilde{w}_j |\beta_j| \), where adaptive weight \( \tilde{w}_j = \frac{1}{|\tilde{\beta}_{ini}|^\gamma} \), and \( \tilde{\beta}_{ini} \) is an initial estimate of the coefficients obtained through ridge regression or Lasso; \( \gamma \) is a positive constant, and is usually set to 1. We can see adaptive LASSO penalizes more those coefficients with lower initial estimates.

It is well known that the square loss function is sensitive to heavy-tailed errors or outliers. Therefore, the adaptive LASSO may fail to produce reliable estimates for datasets with heavy-tailed errors or outliers, which commonly appear in gene expression dataset. One possible remedy is to remove influential observations from the data before fitting a model, but it is difficult to differentiate the true outliers from the normal data. The other method is to use robust regression. Wang et al. [57] combined the least absolute deviation (LAD) and weighted LASSO penalty together to produce LAD- LASSO method. The objective function is

\[
\sum_{i=1}^{n} |y_i - \alpha - x_i^T \beta| + \lambda \sum_{j=1}^{p} \tilde{w}_j |\beta_j| \tag{2.3}
\]

With this LAD loss, LAD- LASSO is more robust than OLS to unusual \( y \) values, but it is sensitive to high leverage outliers. Moreover, LAD estimation degrades the efficiency of the resulting estimation if the error distribution is not heavy tailed [58]. To achieve both robustness and efficiency, Lambert-Lacroix et al. [59] proposed Huber Lasso, which combined the Huber loss function and weighted LASSO penalty. Huber function, see method section, is a hybrid of squared error for relatively small errors and absolute error for relatively large ones. Owen [60] proposed to use Huber function as a loss function and use a reversed version of Huber’s criterion, called Berhu, as a penalty function. The Berhu penalty, see method section, features that relatively small coefficients contribute their \( \ell_1 \)
norm to the penalty while larger ones cause it to grow quadratically. This Berhu penalty sets some coefficients to 0 as the LASSO does while shrinking the larger coefficients in the same way as ridge regression. In [61], the authors showed the combination of Huber loss function and adaptive Berhu penalty enjoys oracle properties, and they showed this procedure encourages a grouping effect. In [60, 61], the authors solved this Huber-Berhu optimization problem using software CVX [62], which is a Matlab-based modeling system for convex optimization. CVX turns Matlab into a modeling language, allowing constraints and objectives to be specified using standard Matlab expression syntax. However, CVX is very slow for large dataset. A proximal gradient descent algorithm was developed for this Huber-Berhu regression in this chapter, which runs much faster than CVX.

Reconstruction of gene regulatory networks often involves ill-posed problems due to high dimensionality and multicollinearity. Partial least squares (PLS) regression has been an alternative to ordinary regression for handling multicollinearity in several areas of scientific research. PLS couples a dimension reduction technique and a regression model. Although PLS has been shown to have good predictive performance in dealing with ill-posed problems, it is not particularly tailored for variable selection. Chun et al. [63] first proposed a sparse partial least squares regression for simultaneous dimension reduction and variable selection. Cao et al. [64] also proposed a sparse PLS method for variable selection when integrating omics data. They added sparsity into PLS with a LASSO penalization combined with SVD computation. In this chapter, the Huber-Berhu regression was embedded into this PLS framework and used simulated and real data to show this approach is applicable for reconstruction of GRNs.

2.2 Variable selection via Huber-Berhu regression

2.2.1 Huber and Berhu functions

In the process to estimate the regression coefficients, the square loss function is well suited if \( y_i \) follows a Gaussian distribution, but it gives a poor performance when \( y_i \) follows a heavy tailed distribution or there are outliers. On the other hand, the LAD loss function is more robust to outliers but the statistical efficiency is low when there are no outliers in the data. Huber function, introduced in [65], is a combination of the linear and quadratic loss functions. For any given positive real \( M \) (called shape parameter), Huber function is defined as

\[
H_M(z) = \begin{cases} 
  z^2 & |z| \leq M \\
  2M|z| - M^2 & |z| > M 
\end{cases}
\]  

(2.4)

This function is quadratic for small \( z \) but grows linearly for large value of \( z \). The parameter \( M \) determines where the transition from quadratic to linear take place, see Figure 2.1(top left). In this chapter, the default value of \( M \) was set to be one tenth of the interquartile range (IRQ), this idea came from [66]. Huber function is a smooth function, with a derivative function
\[ H'_M(z) = \begin{cases} \frac{2z}{2M \text{sign}(z)} & |z| \leq M \\ |z| > M \end{cases} \] (2.5)

The ridge regression uses the quadratic penalty on the regression coefficients, and it is equivalent to put a Gaussian prior on the coefficients. The LASSO uses linear penalty on the regression coefficients, and it is equivalent to put a Laplace prior on the coefficients. The advantage of LASSO over ridge regression is that it implements regularization and variable selection simultaneously. The disadvantage is that, if a group of predictors is highly correlated, LASSO picks only one of them and shrinks the others to zero. In this case, the prediction performance of ridge regression dominates the lasso. Berhu function, introduced in [60], is a hybrid of these two penalties. It gives a quadratic penalty to large coefficients while giving a linear penalty to small coefficients, shown in Figure 2.1(top right). The Berhu function is defined as

\[ B_M(z) = \begin{cases} \frac{|z|}{z^2 + M^2} & |z| \leq M \\ |z| > M \end{cases} \] (2.6)

the shape parameter \( M \) was set to be the same as that in Huber function. As shown in Figure 2.1, Berhu function is a convex function, but it is not differentiable at \( z = 0 \). Figure 2.1 also shows the 2D contour of Huber and Berhu functions. Put Huber loss function and Berhu penalty together, an objective function was obtained as below.

\[ f(\beta) = \sum_{i=1}^{n} H_M(y_i - x_i^T \beta) + \lambda \sum_{j=1}^{p} B_M(\beta_j) \] (2.7)
Figure 2.1 Huber loss function (top left) and Berhu penalty function (top right) as well as their 2D contour (bottom row).

Figure 2.2 provides some insight into the estimation of coefficients for Huber & Berhu (left), LASSO (middle) and Ridge (right) regression. The Huber loss corresponds to the rotated rounded rectangle contour at the top right corner, and the center of the contour is the solution of un-penalized Huber regression. The shaded area is a map of the Berhu constraint where a smaller $\lambda$ corresponds to a larger area. The estimated coefficient of Huber & Berhu regression is the first place the contours touch the shaded area. We can see, when $\lambda$ is small, the touch point is not on the axes, which means Huber & Berhu regression behaves more like Ridge regression, which doesn’t generate a sparse solution. When $\lambda$ increases, the correspondent shaded area changes to a diamond and the touch point more likely to locate on the axes. Therefore, for large $\lambda$, the Huber & Berhu regression behaves like Lasso, which can generate a sparse solution.
2.2.2 The algorithm to solve Huber Berhu regression

As Berhu function is not differentiable at $z = 0$, which poses difficulty to use gradient descent method to solve (2.4). Although we can use general convex optimization solver CVX [62] for a convex optimization problem, it is too slow for real biological application. Therefore, a proximal gradient descent algorithm was developed to solve (2.4). Proximal gradient descent is an effective algorithm to solve an optimization problem with decomposable objective function. Suppose the objective function can be decomposed as $f(x) = g(x) + h(x)$, where $g(x)$ is a convex differentiable function and $h(x)$ is a convex non-differentiable function. The idea behind proximal gradient descent [67] method is to make a quadratic approximation to $g(x)$, and leave $h(x)$ unchanged. That is,

$$f(z) = g(z) + h(z) \approx g(x) + \nabla g(x)^T(z - x) + \frac{1}{2t}||z - x||_2^2 + h(z)$$

At each step, $x$ is updated by the minimum of the right side of (2.5).

$$x^+ = \arg\min_z g(x) + \nabla g(x)^T(z - x) + \frac{1}{2t}||z - x||_2^2 + h(z)$$

$$= \arg\min_z \frac{1}{2t}||z - (x - t\nabla g(x))||_2^2 + h(z)$$

The operator $Prox_{t,h}(x) = \arg\min_z \frac{1}{2t}||z - x||_2^2 + h(z)$ is called proximal mapping for $h$. Therefore, to solve (2.4), the key is to compute the proximal mapping for Berhu function.

$$\lambda B_M(z) = \lambda |z| 1_{|z| \leq M} + \lambda \frac{z^2 + M^2}{2M} 1_{|z| > M} = \lambda |z| + \lambda \frac{(|z| - M)^2}{2M} 1_{|z| > M}$$

let $u(z) = \lambda \frac{(|z| - M)^2}{2M} 1_{|z| > M}$. As $u(z)$ satisfies the theorem 4 in [68],

$$Prox_{t,\lambda B}(x) = Prox_{t,\lambda u}(x) \circ Prox_{t,\lambda 1}(x) \quad (2.8)$$

It is not difficult to verify
\[ \text{Prox}_{t,\lambda}(x) = \text{sign}(x) \min \left\{ |x|, \frac{M}{M + t\lambda} (|x| + t\lambda) \right\} \]  
(2.9)

\[ \text{Prox}_{t,\lambda|x|}(x) = \text{sign}(x) \min \{|x| - t\lambda, 0\} \]  
(2.10)

Algorithm 2.1: accelerated proximal gradient descent method to solve (2.7)

Input: predictor matrix (X), dependent vector (y), and penalty constant (\lambda)
Output: regression coefficient (\beta)

1. Initiate \( \beta = 0, t=1, \beta_{prev} = 0 \)
2. For k in 1… MAX_ITER
3. \( v = \beta + (k/(k+3))(\beta - \beta_{prev}) \)
4. Compute the gradient of Huber loss at v using (2.5), denoted as \( G_v \)
5. While TRUE
6. Compute \( p_1 = \text{Prox}_{t,\lambda|x|}(v) \) using (2.9)
7. Compute \( p_2 = \text{Prox}_{t,\lambda}(p1) \) using (2.10)
8. If \( \sum_{i=1}^{n} H_M(y_i - x_i^T \cdot p_2) \leq \sum_{i=1}^{n} H_M(y_i - x_i^T \cdot v) + G_v(p_2 - v) + \frac{1}{2t} ||p_2 - v||_2^2 \)
   Break
9. Else \( t=t*0.5 \)
10. \( \beta_{prev} = \beta, \beta = p_2 \)
11. If converged
12. Break

Algorithm 2.1 uses accelerated proximal gradient descent method to solve (2.7). Line 3 implements the acceleration of [69]. Lines 6-7 compute the proximal mapping of Berhu function. Lines 5-10 use backtracking method to determine the step size.

### 2.2.3 Screening for non-zero variables

The left plot in Figure 2.3 shows a typical solution path for Huber & Berhu regression. The setting is model 1 in section 2.2.4.2. The dashed lines in Figure 2.3 represent the solution paths for zero variables, and the solid lines represent those for the nonzero ones. The absolute values of the solution are shown in the right plot of Figure 2.3. Huber & Berhu regression forces small coefficients to zero like LASSO but shrinks many variables to zero simultaneously like ridge regression.

We can see when \( \lambda \) is small, it is hard to differentiate nonzero variables from zero variables. However, with the increase of \( \lambda \), the absolute values of zero variables shrink more quickly and nonzero variables pop up. Figure 2.3 also shows that it is inappropriate to use AIC or BIC for selecting the tuning parameter \( \lambda \), because too many variables on the path are kept non-zero. An obvious feature shown in Figure 2.3 is that the orders of absolute values of the nonzero variable are rather stable when \( \lambda \) is large (\( \lambda/\lambda_{\text{max}} > 0.8 \)). In this research,
was fixed to be 0.9, and the absolute values of the estimated coefficients characterize the importance of each variable.

Figure 2.3 A typical solution path for Huber & Berhu regression

After the absolute values of the estimated coefficients were obtained, they were processed by an optimal one-dimensional k-mean clustering method. The k-mean cluster method is to partition data into K groups such that the sum of squared Euclidean distances to each group means is minimized. The standard k-mean algorithm is a heuristic algorithm, it iteratively calculates the within-cluster sum of squared distances, modifies group membership of each point to reduce the within-cluster sum of squared distances, and computes new cluster centers until local convergence is achieved. The disadvantage of this heuristic algorithm is that the solution may not be optimal and lack repeatability. In this chapter, I adopted an optimal one-dimensional k-mean clustering method, which is a dynamic programming algorithm implemented in an R package “Ckmeans.1d.dp” [70]. The number of clusters was determined by Bayesian information criteria. For example, the left plot in Figure 2.4 shows the absolute values of estimated coefficients when \( \frac{\lambda}{\lambda_{max}} \) is set to 0.9 in Figure 2.3. The right plot in Figure 2.4 shows their optimal clustering.

Figure 2.4 The estimated coefficients (left) and their optimal clustering (right).
Assume K clusters are obtained. In these clusters, larger index corresponds to larger absolute values. The variables in group K were used to fit the Huber loss without any penalization, and computed the mean error. Then variables in group K-1 and K were fitted to the Huber loss without any penalization, and computed the mean error. Repeat this process until variables in all groups were used. Then the mean errors versus the number of variables were plotted, which is an L-curve. The nonzero variables can be determined by the elbow point. Figure 2.5 shows the L-curve for the above example. From Figure 2.5, we can see the elbow point appears when the number of variables is 20. Thus the 20 variables with the largest absolute values of estimated coefficients can be identified as nonzero variables.

![L-curve](image.png)

**Figure 2.5 An example of L-curve.**

In summary, Huber & Berhu clustering algorithm for selecting regulatory TFs works as follows:

1. Use a binary search method to find the minimum $\lambda$ (denoted as $\lambda_{max}$) that makes the solution to (2.7) all zeros.
2. Set $\lambda = 0.9 * \lambda_{max}$, solve (2.7) to get $\hat{\beta}$
3. Do clustering analysis on $|\hat{\beta}|$
4. Using variables in top clusters to fit the Huber loss, then compute mean error.
5. Plot the L-curve and find the elbow point. Return the variables above the elbow point as regulatory TFs.

### 2.2.4 Simulation Results

#### 2.2.4.1 Running time

In this section, the running time of Algorithm 2.1 and general convex optimization solver CVX was compared. The underlying true model is $y = X\beta^* + \epsilon$, where $X$ is a $m \times p$ design
matrix and each element follows a standard normal distribution. $\beta^*$ is the true regression coefficient vector, which satisfies that 10% are non-zero and come from $N(0, 3^2)$, $\varepsilon$ is the normal noise and the signal-to-noise ratio was set to 5. The tuning parameter $\lambda$ in (2.7) was set to 10. All the computation is performed on my desktop computer with 2.2GHz Intel Core i7 processor and 16 GB 1600MHz DDR3 memory. Figure 2.6 shows the results of running time using Algorithm 2.1 and CVX for different setting of $m$ and $p$ based on 30 replications. For different $m$, the patterns are similar. Obviously, the proximal gradient descent algorithm runs much faster than CVX.

![Figure 2.6 Comparison of running time for Algorithm 2.1 and CVX.](image)

### 2.2.4.2 Performance of Huber-Berhu regression

In this section, the performance of variable selection for different loss functions, penalty functions and methods to determine tuning parameters were compared. The loss functions include square loss, LAD loss, and the Huber loss; The penalty functions include Lasso, elastic net, adaptive lasso, and Berhu; The methods to determine tuning parameters include: cross-validation, AIC, BIC, EBIC and the clustering method developed in this research. The combinations of these methods were named to be: square-lasso-cv, square-lasso-aic, square-lasso-bic, square-lasso-ebic, square-en-cv, square-en-aic, square-en-bic, square-en-ebic, square-adLasso-cv, square- adLasso-aic, square- adLasso-bic, square- adLasso-ebic, lad-lasso-cv, lad-lasso-aic, lad-lasso-bic, lad-lasso-ebic, lad-en-cv, lad-en-aic, lad-en-bic, lad-en-ebic, lad-adLasso-cv, lad-adLasso-aic, lad- adLasso-bic, lad- adLasso-ebic, Huberlasso-cv, Huber-lasso-aic, Huber-lasso-bic, Huber-lasso-ebic, Huber-Berhu-bic, Huber-Berhu-clustering. The fold of cross-validation was set to 5. The adaptive weights were obtained from the corresponding ridge estimator with a small $\lambda$ (0.001). The degree of freedom used in informatics criteria was set to the number of nonzero coefficients.
The models used to compare the performances of the algorithms are similar to those presented in [61]. They involve groups of highly correlated variables, which are commonly met GRNs. The underlying model is \( y = \mathbf{1}_n \beta^0 + X \beta^* + \sigma \epsilon \), where \( \mathbf{1}_n \) is a vector composed of ones and \( y = (y_1, ..., y_n) \) represents the expression levels of target genes. Each column of the design matrix \( X \) is the expression levels of a TF. The rows of \( X \) are \( n \) independent Gaussian vectors \( N_p(0, \Sigma) \). The covariance matrix \( \Sigma \) is a block diagonal matrix of size \( p \). The first block is a squared matrix of size 5 comprising of 1 outside the diagonal and taking values of 1.01 on the diagonal. The second and third blocks are the same as the first one. The last block is the identity matrix of size \( p - 15 \). The vector of true coefficients was set as follows: the first 5 coordinates are equal to 2, the second 5 coordinates are equal to 3, the third 5 coordinates are equal to 4 and the last \( p - 15 \) coordinates are equal to 0. Therefore, the first 15 variables are relevant variables and the others are pure noise. Among these 15 relevant variables, they are divided into three highly correlated variables. Variables in different groups are independent. The following two models according to the distribution of \( \epsilon \) were considered.

Model 1: Gaussian noise. In this case, \( \{\epsilon_i : i = 1, ..., n\} \) independently follows a standard normal distribution, \( \sigma \) was set to 10. There was no outlier in this model, which allows to quantify the deterioration of the performance of the robust algorithm in the absence of outliers.

Model 2: a mixture of Gaussians. In this case, \( \{\epsilon_i : i = 1, ..., n\} \) independently follows \( N(0,1) \) with probability 0.9 and follows \( N(0,15) \) with probability 0.1. \( \sigma \) was set to 4.1345. Thus the standard deviation of \( y_i \) is equal to \( 4.1345 \times \sqrt{0.9 \times 1 + 0.1 \times 15} = 20 \).

To compare the variable selection ability of various algorithms, the sample size \( n \) was fixed to 50 and set the number of TFs \( p \) to be 50, 100 and 200 respectively. For Lasso-type methods, a coefficient is considered to be zero if its absolute value is less than \( 10^{-5} \). The tuning parameter grid is composed of 100 points log-linearly spaced between \( 0.01 \times \lambda_{\text{max}} \) and \( \lambda_{\text{max}} \). The performance measures used are TPR (True Positive Rate), FPR (False Positive Rate), Precision and \( F_1 - \text{score} \), where

\[
\text{TPR} = \frac{TP}{TP + FN}, \quad \text{FPR} = \frac{FP}{FP + TN}, \quad \text{Precision} = \frac{TP}{TP + FP}, \quad F_1 - \text{score} = \frac{TP}{TP + FP} \times \frac{1}{TPR + \frac{1}{\text{Precision}}}
\]

Table 2.1 presents the results for Model 1, and Table 2.2 presents the results for Model 2. The values in each cell are the mean and the standard deviation of 100 replications. The performance of all methods for both models is similar. The Huber-berhu-cluster method has a high TPR while maintaining a rather low FPR. Based on the F1-score, it is obvious that Huber-berhu-cluster method has obvious better performance than other methods. For LAD loss, when \( p \) equals to 100 or 200, the results from aic, bic and ebic are the same. The definitions of aic, bic, and ebic were adopted from [59], which are similar to the definitions for square loss.
Table 2.1 Variable selection ability on model 1 based on 100 replications.

<table>
<thead>
<tr>
<th>method</th>
<th>p=50</th>
<th>F=100</th>
<th>F=200</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huber-Boost - Clustering</td>
<td>0.8647</td>
<td>0.5122</td>
<td>0.6728</td>
</tr>
<tr>
<td></td>
<td>0.0447</td>
<td>0.0236</td>
<td>0.0383</td>
</tr>
<tr>
<td>Huber-Boost-bic</td>
<td>0.1367</td>
<td>0.1959</td>
<td>0.1959</td>
</tr>
<tr>
<td></td>
<td>0.0385</td>
<td>0.0570</td>
<td>0.0702</td>
</tr>
<tr>
<td>Square-lasso-cv</td>
<td>0.3627</td>
<td>0.2703</td>
<td>0.1741</td>
</tr>
<tr>
<td></td>
<td>0.4233</td>
<td>0.3774</td>
<td>0.3187</td>
</tr>
<tr>
<td>Square-lasso-aic</td>
<td>0.1355</td>
<td>0.0649</td>
<td>0.0576</td>
</tr>
<tr>
<td></td>
<td>0.0597</td>
<td>0.0602</td>
<td>0.0602</td>
</tr>
<tr>
<td>Square-lasso-ebic</td>
<td>0.78</td>
<td>0.9411</td>
<td>0.8164</td>
</tr>
<tr>
<td></td>
<td>0.2642</td>
<td>0.3877</td>
<td>0.3548</td>
</tr>
<tr>
<td>Square-en-cv</td>
<td>0.8653</td>
<td>0.2866</td>
<td>0.1539</td>
</tr>
<tr>
<td></td>
<td>0.5921</td>
<td>0.5931</td>
<td>0.5931</td>
</tr>
<tr>
<td>Square-en-aic</td>
<td>0.38</td>
<td>0.3451</td>
<td>0.3451</td>
</tr>
<tr>
<td></td>
<td>0.4538</td>
<td>0.4538</td>
<td>0.4538</td>
</tr>
<tr>
<td>Square-en-ebic</td>
<td>0.8222</td>
<td>0.812</td>
<td>0.3188</td>
</tr>
<tr>
<td></td>
<td>0.8451</td>
<td>0.8451</td>
<td>0.8451</td>
</tr>
<tr>
<td>Square-lasso-ebc</td>
<td>0.0067</td>
<td>0.0069</td>
<td>0.0069</td>
</tr>
<tr>
<td></td>
<td>0.0576</td>
<td>0.0576</td>
<td>0.0576</td>
</tr>
<tr>
<td>LAD-lasso-cv</td>
<td>0.3458</td>
<td>0.2526</td>
<td>0.1924</td>
</tr>
<tr>
<td></td>
<td>0.4526</td>
<td>0.4526</td>
<td>0.4526</td>
</tr>
<tr>
<td>LAD-lasso-aic</td>
<td>0.1356</td>
<td>0.1373</td>
<td>0.1373</td>
</tr>
<tr>
<td></td>
<td>0.9263</td>
<td>0.9263</td>
<td>0.9263</td>
</tr>
<tr>
<td>LAD-lasso-ebc</td>
<td>0.756</td>
<td>0.7623</td>
<td>0.1608</td>
</tr>
<tr>
<td></td>
<td>0.3125</td>
<td>0.3125</td>
<td>0.3125</td>
</tr>
<tr>
<td>LAD-en-cv</td>
<td>0.45</td>
<td>0.3462</td>
<td>0.3823</td>
</tr>
<tr>
<td></td>
<td>0.6318</td>
<td>0.6318</td>
<td>0.6318</td>
</tr>
<tr>
<td>LAD-en-ebc</td>
<td>0.214</td>
<td>0.0289</td>
<td>0.0301</td>
</tr>
<tr>
<td></td>
<td>0.0907</td>
<td>0.0737</td>
<td>0.0737</td>
</tr>
<tr>
<td>LAD-lasso-cv</td>
<td>0.954</td>
<td>0.9389</td>
<td>0.3035</td>
</tr>
<tr>
<td></td>
<td>0.9389</td>
<td>0.9389</td>
<td>0.9389</td>
</tr>
<tr>
<td>LAD-lasso-ebc</td>
<td>0.8232</td>
<td>0.7648</td>
<td>0.3239</td>
</tr>
<tr>
<td></td>
<td>0.9269</td>
<td>0.9269</td>
<td>0.9269</td>
</tr>
<tr>
<td>LAD-en-cv</td>
<td>0.25</td>
<td>0.0359</td>
<td>0.0383</td>
</tr>
<tr>
<td></td>
<td>0.8101</td>
<td>0.8101</td>
<td>0.8101</td>
</tr>
<tr>
<td>LAD-en-ebc</td>
<td>0.68</td>
<td>0.0125</td>
<td>0.0142</td>
</tr>
<tr>
<td></td>
<td>0.6806</td>
<td>0.6806</td>
<td>0.6806</td>
</tr>
<tr>
<td>LAD-lasso-cv</td>
<td>0.954</td>
<td>0.9389</td>
<td>0.3035</td>
</tr>
<tr>
<td></td>
<td>0.9389</td>
<td>0.9389</td>
<td>0.9389</td>
</tr>
<tr>
<td>LAD-lasso-ebc</td>
<td>0.8232</td>
<td>0.7648</td>
<td>0.3239</td>
</tr>
<tr>
<td></td>
<td>0.9269</td>
<td>0.9269</td>
<td>0.9269</td>
</tr>
<tr>
<td>LAD-en-cv</td>
<td>0.25</td>
<td>0.0359</td>
<td>0.0383</td>
</tr>
<tr>
<td></td>
<td>0.8101</td>
<td>0.8101</td>
<td>0.8101</td>
</tr>
<tr>
<td>LAD-en-ebc</td>
<td>0.68</td>
<td>0.0125</td>
<td>0.0142</td>
</tr>
<tr>
<td></td>
<td>0.6806</td>
<td>0.6806</td>
<td>0.6806</td>
</tr>
</tbody>
</table>

Table 2.2 Variable selection ability on model 2 based on 100 replications.

<table>
<thead>
<tr>
<th>method</th>
<th>p=50</th>
<th>F=100</th>
<th>F=200</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huber-Boost - Clustering</td>
<td>0.8853</td>
<td>0.1958</td>
<td>0.1958</td>
</tr>
<tr>
<td></td>
<td>0.0075</td>
<td>0.0094</td>
<td>0.0094</td>
</tr>
<tr>
<td>Huber-Boost-bic</td>
<td>0.8113</td>
<td>0.1446</td>
<td>0.1446</td>
</tr>
<tr>
<td></td>
<td>0.0093</td>
<td>0.0093</td>
<td>0.0093</td>
</tr>
<tr>
<td>Square-lasso-cv</td>
<td>0.2807</td>
<td>0.2186</td>
<td>0.1489</td>
</tr>
<tr>
<td></td>
<td>0.3996</td>
<td>0.3996</td>
<td>0.3996</td>
</tr>
</tbody>
</table>

33
2.3 Huber-Berhu partial least square

2.3.1 Embedding Huber-Berhu regression into PLS

Let $X (n \times p)$ and $Y (n \times q)$ be the standardized predictor variables (TF genes) and dependent variables (pathway genes), respectively. PLS [71] looks for a linear combination of $X$ and a linear combination of $Y$ such that their covariance reaches maximum.

$$\text{max} ||u||_2^2 = 1, ||v||_2^2 = 1 \text{ cov}(Xu, Yv)$$  \hspace{1cm} (2.11)

Here, the linear combination $\xi = Xu$ and $\eta = Yv$ are called components scores, also called latent variables, and the $p$ and $q$ dimensional combinatorial coefficients $u$ and $v$ are called loadings. After getting this first component $\xi$, two regression equations, from $X$ to $\xi$ and from $Y$ to $\xi$, were set up.
\[ X = \xi c' + \varepsilon_1, Y = \xi d' + \varepsilon_2 = Xb + \varepsilon_3 \]  

(2.12)

Next, \( X \) was deflated as \( X = X - \xi c' \), and \( Y \) was deflated as \( Y = Y - \xi d' \), and this process was continued until enough components were extracted.

There exists a close relationship between PLS and SVD. Let \( M = X'Y \), then 
\[ \text{cov}(Xu, Yv) = \frac{1}{n} u'Mv. \]

Let the SVD of \( M \) be:
\[ M = U\Delta V' \]
where \( U(p\times r) \) and \( V(q\times r) \) are orthonormal and \( \Delta(r\times r) \) is a diagonal matrix whose diagonal elements \( \delta_k \ (k = 1 \ldots r) \) are called singular values. According to the property of SVD, the combinatorial coefficients \( u \) and \( v \) in (2.7) are exactly the first column of \( U \) and first column of \( V \). Therefore, the loadings of PLS can be computed by 
\[ \min_{u,v} ||M - uv'||_F^2 \]

where \( ||M - uv'||_F^2 = \sum_{i=1}^{p} \sum_{j=1}^{q} (m_{ij} - u_i v_j)^2. \)

Cao, et al [64] proposed a sparse PLS approach using SVD decomposition of \( M \) by adding \( \ell_1 \) penalty on the loadings. The optimization problem to solve is 
\[ \min_{u,v} ||M - uv'||_F^2 + \lambda_1 ||u||_1 + \lambda_2 ||v||_1 \]

As mentioned above, Huber function is more robust to outliers and has higher statistical efficiency than LAD loss, and Berhu penalty has a better balance between \( \ell_1 \) and \( \ell_2 \) penalty. In this chapter, the Huber loss and Berhu penalty were adopted to extract each component for PLS. The optimization problem becomes 
\[ \min_{u,v} \sum_{i=1}^{p} \sum_{j=1}^{q} H(m_{ij} - u_i v_j) + \lambda \sum_{i=1}^{p} B(u_i) + \lambda \sum_{i=1}^{q} B(v_i) \]  

(2.13)

The objective function in (2.13) is not convex on \( u \) and \( v \), but it is convex on \( u \) when \( v \) is fixed and it is convex on \( v \) when \( u \) is fixed. For example, when \( v \) is fixed, each \( u_i \) in parallel can be solved by
\[ \min_{u_i} \sum_{j=1}^{q} H(m_{ij} - u_i v_j) + \lambda B(u_i) \]  

(2.14)

similarly, when \( u \) is fixed, each \( v_j \) in parallel can be computed by
\[ \min_{v_j} \sum_{i=1}^{p} H(m_{ij} - u_i v_j) + \lambda B(v_j) \]  

(2.15)

(2.14) and (2.15) can be solved using Algorithm 2.1. So (2.9) can be solved iteratively through updating \( u \) and \( v \) alternately. Note, it is not cost-efficient to spend a lot of effort optimizing over \( u \) in line 6 before a good estimate for \( v \) is computed. As algorithm 2.1 is
an iterative algorithm, it may make sense to stop the optimization over $u$ early before going to update $v$. In the implementation, one step of proximal mapping was used to update $u$ and $v$. That is,

$$u = Prox_{\ell, \lambda B} \left( u - t \frac{\partial H(M - uu')}{\partial u} \right) \quad (2.16)$$

$$v = Prox_{\ell, \lambda B} \left( v - t \frac{\partial H(M - uu')}{\partial v} \right) \quad (2.17)$$

The Huber–Berhu PLS are detailed in Algorithm 2.2.

---

**Algorithm 2.2: Huber-Berhu PLS regression**

| Input: TF matrix (X), pathway matrix (Y), penalty constant ($\lambda$), and number of components (K) |
| Output: regression coefficient matrix (A) |

1. $X_0 = X, X_0 = Y, cF = I, A = 0$
2. For $k$ in 1...K
3. set $M_{k-1} = X'_{k-1}Y_{k-1}$
4. Initialize $u$ to be the first left singular vector and initialize $v$ to be the product of first right singular vectors and first singular value.
5. until convergence of $u$ and $v$
6. update $u$ using (2.16)
7. update $v$ using (2.17)
8. extract component $\xi = Xu$
9. compute regression coefficients in (2.8) $c = X'\xi/(\xi'\xi), d = Y'\xi/(\xi'\xi)$
10. update $A = A + cF \cdot u \cdot d'$
11. update $cF = cF \cdot (1 - u \cdot c')$
12. compute residuals for $X$ and $Y$, $X = X - \xi c', Y = Y - \xi d$

2.3.2 Tuning criteria and choice of the PLS dimension

Huber-Berhu PLS has two tuning parameters, namely, penalization parameter $\lambda$ and the number of hidden components $K$. For selecting the best penalization parameter $\lambda$, a common k-fold cross-validation (CV) procedure minimizing the overall prediction error is applied, using a grid of possible values. If the sample size is too small, CV can be replaced by leave-one-out validation, this procedure is also used in [63, 72] for tuning penalization parameters.
To choose the dimension of PLS, the $Q_h^2$ criteria were first proposed by Tenenhaus [73], it characterizes the predictive power of the PLS model by performing cross-validation computation. $Q_h^2$ is defined as

$$Q_h^2 = 1 - \frac{\sum_{k=1}^{q} PRESS_h^k}{\sum_{k=1}^{q} RSS_h^k}$$

where $PRESS_h^k = \sum_{l=1}^{n} (y_l^k - \hat{y}_{h(-i)}^k)^2$ is the Prediction Error Sum of Squares and $RSS_h^k = \sum_{l=1}^{n} (y_l^k - \hat{y}_h^k)^2$ is the Residual Sum of Squares for the variable $k$ and the PLS dimension $h$. The criterion for determining if $\xi_h$ contributes significantly to the prediction is:

$$Q_h^2 \geq (1 - 0.95^2) = 0.0975$$

This criterion is also used in the SIMCA-P software [74] and sparse PLS [64]. However, the choice of the PLS dimension still remains an open question. Empirically, there is little biological meaning when $h$ is too large, and good performance appears in 2-5 dimensions.

2.3.3 Analysis of Arabidopsis dataset using Huber-Berhu PLS

The HB-PLS algorithm was tested for its accuracy in identifying pathway regulators using an Arabidopsis gene expression data. The pathway used in this study is the lignin biosynthesis pathway. Lignin is the second most abundant plant biopolymer found in secondary cell walls and fibers of wood [75, 76]. Understanding how lignin is synthesized has long been a research focus of plant biologists and the wood industry because of the importance of lignin in plant structural integrity and stem stiffness [77]. The genes in this pathway were downloaded from the Arabidopsis Information Resource (https://www.arabidopsis.org/). The Arabidopsis gene expression data were downloaded from the public repository. The wood formation compendium dataset contains the 128 microarrays pooled from six experiments, which have the accession identifiers of GSE607, GSE6153, GSE18985, GSE2000, GSE24781, and GSE5633, in NCBI Gene Expression Omnibus (GEO) (http://www.ncbi.nlm.nih.gov/geo/). These datasets were obtained from hypocotyledonous stems under short-day that is known to induce secondary wood formation [78]. All data sets mentioned above were derived from hybridization of Affymetrix microarrays. The original CEL files were downloaded and processed by the robust multi-array analysis (RMA) algorithm using the Bioconductor package. The quality control methods followed [79].

The identified top 15 important TFs for each lignin biosynthesis pathway gene are depicted in Figure 2.7. The green oval shapes show pathway genes involved in the lignin biosynthesis. The identified top 15 important TFs are shown in the square shapes. The TFs, which have supporting literature evidence for regulating lignin biosynthesis pathway genes, are shown in red color. For example, it was reported in [80] that, SND1 is a high hierarchical regulator that controls SND2, SND3, MYB103, MYB85, MYB52, MYB54, MYB69, MYB42, MYB43, MYB86, MYB63, MYB46, MYB58, and KNAT7. HB-PLS algorithm identified 10 of these 15 TFs (SND1, SND2, SND3, MYB103, MYB85, MYB43,
MYB46, MYB86, MYB63, MYB58). NST1, NST2, VND4 and VND6, the functional NAC family homologs of SND1, regulate the same downstream targets in different cell types [81]. HB-PLS recognized out NST1 and VND4.

Figure 2.7 Identified top 15 most important TFs for each lignin biosynthesis pathway gene.

The performance of HB-PLS was compared with conventional partial least square method (PLS) [82] and sparse partial least square method [63]. In Figure 2.8, each facet shows the number of positive TFs in the top n most important TFs. For 4CL1, C3H and CCoAOMT1, the three methods have similar performance; For PAL1, C4H, CCR1, F5H, and COMT1, HB-PLS and SPLS have similar performance, and they behave better than PLS; For HCT and CAD8, HB-PLS behaves much better the other two methods.
2.4 Conclusion

In this chapter, a proximal gradient descend algorithm was developed to solve a regression optimization problem. In this regression, Huber function is used as the loss function and Berhu function is used as the penalty function. An optimal one-dimensional clustering algorithm was adopted to cluster the regression coefficients and then the elbow point is used to determine the non-zero variables. Huber function is more robust in dealing with outlier and non-Gaussian error. Berhu function integrates the advantages of both $\ell_1$ and $\ell_2$ penalty. The group effect of Huber-Berhu regression makes it very proper to model the transcriptional regulatory relationships. Simulation results showed that Huber-Berhu regression has better performance in identifying non-zero variables. When modeling the regulatory relationships from TFs to a pathway, the Huber-Berhu regression was embedded into the PLS framework, called HB-PLS, to deal with the high multicollinearity property.
of both TFs and pathway genes. Implementation of the HB-PLS to an Arabidopsis data showed it can identify more positive TFs known to regulate lignin biosynthesis pathway genes.
3 BWERF: a recursive random forest algorithm for constructing multilayered hierarchical gene regulatory network that governs biological pathways

Present knowledge indicates a multi-layered hierarchical gene regulatory network (ML-hGRN) often regulates a biological pathway or process. Although an ML-hGRN is very important for understanding how a pathway is regulated, there is almost no computational algorithm for directly building an ML-hGRN that operates above a biological pathway or process.

A backward elimination random forest (BWERF) algorithm was developed for constructing the ML-hGRN operating above a biological pathway or a biological process. The algorithm first divided construction of ML-hGRN into multiple regression tasks; each involves a regression between a pathway gene and all transcription factors (TFs). In order to infer the regulatory relationships between a pathway gene and TFs, a random forest model was built to acquire the importance value of each TF to the pathway gene. Then a portion (1/10 was used for illustration) of least important TFs was removed, known as backward elimination. A new random forest model was built on the left TFs, and their importance values are re-computed. This process was conducted repeatedly until all TFs are removed. The above procedure is termed BWERF, which was called for each of other pathway genes. The importance values of a TF to each individual pathway gene were aggregated to form an importance value of the TF to the group of pathway genes. An EM algorithm was used to fit a Gaussian mixture model to these importance values. The TFs that have the highest probability belonging to the component that had the largest mean in the Gaussian mixture model was identified to be the putative regulatory TFs.

BWERF improved the accuracy of inferring ML-hGRN from two aspects: first, it uses backward elimination to alleviate the inference of noise gene; second, it aggregated the individual regulatory importance (edge importance) to form the importance of TFs (node importance). The usefulness of BWERF was validated in constructing ML-hGRNs for a mouse pluripotency maintenance pathway and an Arabidopsis wood formation pathway. Compared to GENIE3, a champion algorithm of the DREAM4 challenge for building a non-hierarchical network, BWERF showed better performance in the evaluation of regulatory importance, and BWERF can recognize more positive TFs for a group of pathway genes. Compared to the Bottom-up GGM algorithm, BWERF can build ML-hGRNs with

1 The material contained in this chapter has been published in the PloS one 12.2 (2017): e0171532. https://doi.org/10.1371/journal.pone.0171532
significantly reduced edges, which enable biologists to choose the implicit edges for experimental validation.

### 3.1 Introduction

There are at least a few hundred metabolic pathways and a few thousand biological processes known to be present in plants and animals, but unfortunately, our knowledge on how these pathways or biological processes are regulated is very limited. For example, *Arabidopsis thaliana* has 549 metabolic pathways (http://www.arabidopsis.org/) and currently, the regulators for only a few pathways are partially known. The regulators for most pathways remain elusive. Present knowledge has shown that many pathways and biological processes are regulated by multi-layered hierarchical gene regulatory networks (ML-hGRN) [19, 83-90]. Therefore, it is imperative to develop the methods for reverse-engineering ML-hGRNs from high-throughput gene expression data. ML-hGRNs enable identification of high hierarchical regulators, middle-level and low-level regulators that indirectly or directly govern pathway or biological process genes at bottom-layer. Studies have shown high hierarchical regulators (HHRs) are global modulators that respond to various cellular signals [91, 92] and environmental cues [85, 93]. The middle-level regulatory genes play the manager-like roles or serve as a hub, through which the commands from high hierarchical regulators at upper layers are synthesized and then passed down to low-level regulators that exert regulation on bottom layer genes [85, 94]. It is important to identify the high hierarchical regulators at the top levels because they have more pleiotropic effects and are useful if we are intended to engineer multiple pathways. To understand how pathways are regulated, we should reconstruct ML-hGRNs that operate above biological pathways and processes. This kind of ML-hGRNs can provide not only the hierarchies of regulatory genes but also the connectivity among them, which can significantly increase our understanding of wired regulation exerted on a metabolic pathway and biological processes through multiple chains-of-command [95].

Although ML-hGRNs are important, there is a lack of methods for directly building ML-hGRNs from high-throughput gene expression data [19]. Currently, there are many methods that are available for building GRNs [27, 43, 50, 96-98], but these methods are not specifically tailored for constructing the ML-hGRNs that mimic the hierarchical regulation [19]. Although a Bottom-up GGM algorithm [19, 87] was developed for directly building ML-hGRNs from high-throughput gene expression data, it usually builds too many regulatory relationships (edges) among genes in the ML-hGRNs, making it hard to identify implicit “chains-of-commands” in particular between the bottom/first layer and second layer, and as well as the second and third layers. In this study, a novel algorithm was developed to construct ML-hGRNs based on random forest [99], with a purpose to reduce the number of edges among genes and also make the edges that represent the causal relationships to emerge.

GENIE3 is gene network inferring algorithm based on the random forest, and it is the champion of DREAM4 in silico network challenge.
There are two differences between BWERF and GENIE3: one difference is BWERF applies backward elimination to alleviate the interference of noise variables, and let the regulatory genes with medium-level regulatory strengths surface out from the noise; Another difference is that BWERF aggregates the individual importance values of a TF to each pathway gene to form the importance of the TF.

3.2 Materials and methods

3.2.1 Materials

3.2.1.1 Arabidopsis Microarray Data Sets

The *Arabidopsis* gene expression data used in this study were downloaded from the public repository. The wood formation compendium dataset contains the 128 microarrays pooled from six experiments, which have the accession identifiers of GSE607, GSE6153, GSE18985, GSE2000, GSE24781, and GSE5633, in NCBI Gene Expression Omnibus (GEO) ([http://www.ncbi.nlm.nih.gov/geo/](http://www.ncbi.nlm.nih.gov/geo/)). These datasets were obtained from hypocotyledonous stems under short-day that is known to induce secondary wood formation[78]. All data sets mentioned above were derived from hybridization of Affymetrix microarrays. The original CEL files were downloaded and processed by the robust multiarray analysis (RMA) algorithm using the Bioconductor package. The quality control methods were described [79].

3.2.1.2 Mouse microarray data sets

The mouse microarray datasets were downloaded from Embryonic Stem Cells Atlas of Pluripotency Evidence (ESCAPE) website ([http://www.maayanlab.net/ESCAPE/](http://www.maayanlab.net/ESCAPE/)). A time course gene expression data set was downloaded from R1 ESCs under undirected differentiation using Affymetrix MOE430A array. The time-course include 0 h, 6 h, 12 h, 18 h, 24 h, 36 h, 48 h, 4 d, 7d, 9d, and 14 d, with 3 replicates at each time point. The data file downloaded is named to be R1_ES_EB_MOE430A.txt.zip. The positive regulatory relationships were obtained from a file called chip_x.txt.zip, which contains the protein/DNA interactions table extracted from ChIP-X studies. ESCAPE includes 206,521 protein-DNA binding interactions in proximity to coding regions covering 48 ESC-relevant transcription factors and chromatin modifiers. The pathway selected are mouse pluripotency maintenance pathway with 24 genes. This pathway containing 24 genes, and there are 35 known TFs regulating them. 3 datasets are generated by adding 100, 200 and 300 randomly selected noise genes from the expression dataset into the TFs.
3.2.2 Methods

Figure 3.1 Flowchart illustrating BWERF algorithm for constructing multilayered hierarchical gene regulatory network using expression data of pathway genes and regulatory genes.

3.2.2.1 Construction of ML-hGRN with backward elimination random forest

In order to construct ML-hGRN that govern a biological pathway, the algorithm first placed pathway genes at the bottom (first) layer, and tried to identify the most significant regulatory genes (TFs) that are associated with the pathway genes with causal relationships, and then built the second layer of the network. After that, the regulatory genes shown at the second layer were removed from the pool of input genes and then used as the new bottom layer, and the remaining input genes were used as an input to construct the third layer. This process was repeated until the designated number of layers is achieved or no more layer can be built owing to lack of TFs that have causal relationships with the current bottom layer.

Assume there are $p$ TFs, $q$ pathway genes and $n$ samples, then the inputs are two expression matrices: TFs matrix with size $n \times p$ and pathway genes matrix with size $n \times q$ (Figure 3.1A). In order to build one layer of GRN, the inferring GRN problem was divided...
into $q$ different regression problems. For each regression problem, one pathway gene was taken out as the response variable, all the TFs were predictor variables. To identify the significant regulatory relationships from TFs to a pathway gene is to ascertain the most important predictor variables in the regression problem. For each regression, random forest method was used, which will be explained below, to identify the candidate regulatory relationships. The input data was fitted to a random forest model, and the importance value for each TF was recorded. Then, the importance of TFs was sorted in decreasing order and a certain portion (e.g. 1/10 or 1/5) of the least contributive TFs were eliminated. The remaining genes were re-fitted to random forest model. This process was performed recursively until all TFs are eliminated (Figure 3.1B). The purpose of using such a backward elimination strategy is to reduce the chance for mistakenly deleting important TF genes. After every pathway gene is evaluated using above random forest model with backward elimination procedure, all importance values of a TF were aggregated to form the importance value of TF to the group of pathway (Figure 3.1C), here, users can add weight to each pathway gene based on existing knowledgebase so that the regulatory relationships eventually identified will favor to some particular pathway genes or close to the reality of existing the rate-limited pathway genes. If no weights are available for pathway genes, users can enter 1 for each pathway gene. In order to identify one layer of gene regulatory network, users can assign the number of TFs empirically, or use expectation-maximization (EM) algorithm to fit a Gaussian mixture model to the importance values obtained, and extracted the most important TFs (Figure 3.1D). The TFs within the frame of interest as illustrated in Figure 3.1D have the highest probability belonging to the rightmost component, and were extracted. An edge will be added to each extracted TF and the corresponding pathway gene to form one layer of GRN (Figure 3.1E). By using the new TF layer as the bottom layer, all above procedure were repeated to obtain the next layer until the designated number of layers was achieved or the program terminated due to the lack of significant TFs as input for upper layers. BWERF algorithm was summarized in algorithm 3.1.

Algorithm 3.1 BWERF algorithm for reconstruction of ML-hGRN

Input: pathway matrix $Y_{n \times q}$, TF matrix $X_{n \times p}$, number of layers $K$. $n$ is sample size, $p, q$ are gene number.
Output: an ML-hGRN

1. Form $q$ regression problems using each column of $Y_{n \times q}$ as the dependent variable and using TF matrix $X$ as independent variables.
2. for each regression problem
3. while ncol(X)>1
4.     fit a random forest model on data $(X, y_j)$
5.     compute the importance value of each TF to $y_j$, denoted as $Im_{ij}$
6.     record the rank and importance value of a portion (1/10) of least important TFs
7.     eliminate this portion of TFs from X
for each $TF_i$
aggregate the importance of $TF_i$ to all pathway genes, denoted as $Imp_i$
fit a Gaussian mixture model on data $(Imp_1, ..., Imp_p)$
select the TFs belonging to the first component as direct regulators to the pathway genes
repeat above process to reconstruct another layer of GRN, only replacing Y by selected TFs and replacing X by non-selected TFs.

3.2.2.2 Detail of random forest

Random forest is a machine learning technique developed by Leo Breiman [99]. It can be used on both classification and regression problems. In general, random forest uses bootstrap to generate a random subset of samples (data sets) from the original data set and then constructs an individual decision tree. For the root node, a subset of randomly selected variables (genes) was used as the candidates for partition of the aforementioned random subset of samples. The best candidate gene being selected for the root node is the one by which we can split the random subset of samples into two children subsets with the minimal impurity. The impurity is a measure of whether the similar response values end up in the same set (child) in a partition. For classification problem, it is typically either Gini impurity or information entropy; for regression problem it is variance. After the root was set, the aforementioned random subset of samples was partitioned into two children nodes. The decision tree was then built recursively in the same fashion as described above for each child node until the impurity for each leaf node is zero. When completed, the decision tree is called a fully grown tree. In this study, bootstrap was used to generate n (where n = 1000) random subsets of samples, and 1000 decision trees were built.

To rank TFs associated with the genes in a specific pathway, it is essential to learn the importance value of each TF to the pathway genes. The random forest can return the importance value of each independent variable in a natural manner. For a node of the decision tree, the importance of the splitting variable is defined as the value of decreased variance using the formula: $IV = n \delta_p - n_1 \delta_{c1} - n_2 \delta_{c2}$ where $p$ stands for the parent node while $c$ stands for the children, $\delta_p$ represents the variance of all $n$ samples split at a parent node while $\delta_{c1}$ and $\delta_{c2}$ represents the variance of all $n_1$ samples split at children node C1 and variance of all $n_2$ samples split at child node C2 respectively. The importance of other variables is defined as zero for this node. For a decision tree, the importance of each independent variable is the sum of the importance of the variable appearing in each node of the tree. For a random forest, the importance of each independent variable is the average of importance in all trees.
3.2.2.3 Details of Expectation Maximization (EM) algorithm

Suppose $x_1, x_2, ..., x_N$ are the importance values of TFs that were obtained from BWERF. Assuming they are independent and identically distributed (i.i.d.) samples from a finite mixture of $K > 1$ Gaussian distributions. The density of each $x_i$ can be written as

$$g(x_i) = \sum_{k=1}^{K} \alpha_k \varphi_k(x_i; \mu_k, \sigma_k^2)$$

where $\alpha_k > 0$, $\sum_{k=1}^{K} \alpha_k = 1$, $\varphi_k$ is Gaussian density function. Let $\Theta = (\alpha_1, ..., \alpha_K, \mu_1, ..., \mu_K, \sigma_1^2, ..., \sigma_K^2)$ be the parameter vector want to be estimated. The EM algorithm is an iterative algorithm that starts from some initial estimate of $\Theta$ and then proceeds to iteratively update $\Theta$ until convergence is detected. Each iteration consists of an E-step and an M-step.

E-Step: Denote the current parameter values as $\Theta$. Compute the membership weight of $x_i$ belongs to component $k$.

$$w_{ik} = p(x_i \text{ belongs to component } k) = \frac{\alpha_k \varphi_k(x_i; \mu_k, \sigma_k^2)}{\sum_{k=1}^{K} \alpha_k \varphi_k(x_i; \mu_k, \sigma_k^2)}$$

For all data points $x_i$, $1 \leq i \leq N$ and all mixture components $1 \leq k \leq K$. This yields an $N \times K$ matrix of membership weights, where each of the rows sum to 1.

M-Step: Now use the membership weights and the data to calculate new parameter values. Let $N_k = \sum_{i=1}^{N} w_{ik}$, i.e., the sum of the membership weights for the $k^{th}$ component. Then,

$$\alpha_k^{new} = \frac{N_k}{N}, k = 1, ..., K$$

$$\mu_k^{new} = \frac{\sum_{i=1}^{N} w_{ik} x_i}{N_k}, k = 1, ..., K$$

$$\sigma_k^{new} = \frac{\sum_{i=1}^{N} w_{ik} (x_i - \mu_k^{new})^2}{N_k}$$

After the parameters were obtained, the TFs with the highest probability belonging to the first component in the Gaussian mixture model was identified as the putative regulatory TFs.

3.2.2.4 Performance evaluation

BWERF was compared with GENIE3, the latter is the best performer in DREAM4 In Silico Multifactorial challenge [42]. Huynh-Thu et al. used the random forest in GENIE3 to construct gene regulatory networks. However, GENIE3 is not tailored to construct GRNs that govern biological pathways or processes. A network constructed with GENIE3 is more
like a net with all directional connections but lack of hierarchy. In order to use GENIE3 as a comparison, some additional rules were enforced to use the top TFs ranked by GENIE3 as a layer over the bottom layer of pathway genes. After that, a new layer was derived from the same principle. The number of TFs kept at each layer was the same as BWERF for the purpose of comparison.

In order to evaluate and compare the efficiency of BWERF and GENIE3, the precision-recall (PR) curve and receiver operating characteristic (ROC) curves of two algorithms were plotted, and the statistics AUPR and AUROC were calculated. The PR curve is created by plotting the precision against the recall at various threshold settings, while the ROC curve is created by plotting the true positive rate (TPR) versus the false positive rate (FPR). The definitions of precision, recall, TPR, and FPR are as follows:

\[
\text{precision} = \frac{TP}{TP + FP} \\
\text{recall} = \text{TPR} = \frac{TP}{P_{\text{total}}} \\
\text{FPR} = \frac{FP}{F_{\text{total}}}
\]

Where TP and FP are the numbers of regulatory relationships above the threshold being true positive and false positive respectively. \(P_{\text{total}}\) and \(F_{\text{total}}\) are the total numbers of positive and negative regulatory relationships respectively. AUPR is calculated by Trapz [100] numerical integral of precision with respect to recall. AUROC is calculated by Trapz numerical integral of TPR with respect to FPR.

### 3.3 Results

#### 3.3.1 Comparison of variable recognizing capability between BWERF and GENIE3 on a simulated “toys dataset”

A simulated “toys dataset” was used to show the effect of backward elimination. This dataset simulates the expression values of 1 pathway gene (\(y\)) and 1006 TFs \(\{X_1, X_2, ..., X_{1006}\}\), where the first 6 TFs \(\{X_1, X_2, ..., X_6\}\) are true TFs that regulate the pathway gene and the other 1000 TFs are noise. The setting is as follows:

- \(X_i \sim N(1, 1)\) for \(i = 1, 2, 3\)
- \(X_i \sim N(3, 1)\) for \(i = 4, 5, 6\)
- \(X_i \sim N(0, 1)\) for \(i = 7, ..., 1006\)
- \(z \sim N(0, 0.1)\)
- \(y = X_1 + X_2 + X_3 + X_4 + X_5 + X_6 + z\)
Given this, the 6 true TFs can be divided into two groups \( \{X_1, X_2, X_3\} \) and \( \{X_4, X_5, X_6\} \). The TFs in the first group have weaker signals than the second group, and are more likely to be inundated by the noise TFs. The sample size is set to be 100.

As shown in Figure 3.2, the strong regulatory TFs \( X_5 \) and \( X_6 \) were correctly identified as the two most important top TFs by both methods. However, for GENIE3, the noise TF \( X_{536} \) appeared in the third place, its importance value surpassed the importance values of the other 4 true TFs. In addition, true TF \( X_3 \) was inundated by noise TFs. When BWERF was applied to this toys data set, the strong regulatory TFs \( \{X_4, X_5, X_6\} \) and weak regulatory TF \( X_1 \) had the larger importance values than any noise variables, and true TF \( X_3 \) was only surpassed by noise \( X_{536} \), which clearly manifested the roles of backward elimination in elevating the positions of true positive TFs. It was also noticed that the backward elimination implicitly increased the importance values and also their range rankings as backward elimination was advanced, leading to the true regulatory variables becoming more differentiable from noise variables.

![Figure 3.2 Comparison of the ordered important values of top 15 variables generated by GENIE3 (left) and BWERF (right) on a “toys dataset”](image)

### 3.3.2 Comparison of ML-hGNRs constructed from mouse gene expression data using BWERF and GENIE3

In order to evaluate the performance of BWERF in recognizing regulatory relationships, mouse time course gene expression data and ChIP-seq data were downloaded from embryonic stem cells Atlas of Pluripotency Evidence (ESCAPE). Mouse pluripotency maintenance pathway was selected for demonstrating the efficiency of BWERF. The 24 genes involved in pluripotency maintenance renewal were chosen for this \textit{in silico} validation. These 24 genes are regulated by 35 known TFs based on the ChIP-seq data provided by ESCAPE web portal. Three test data sets were subsequently generated by adding the profiles of 100, 200, and 300 simulated noise genes into the profiles of the 35
TFs. Since these 35 TFs are the direct regulators of the pathway genes based on ChIP-seq data, one-layered GRNs using BWERF and GENIE3 were constructed. With the known positive edges, the precision-recall (PR) curves and receiver operating characteristic (ROC) curves were generated for evaluating the efficiencies of BWERF and GENIE3 (Figure 3.3). A PR curve plots the proportion of true positives among all predictions versus the percentage of true positives that are retrieved (recall) for varying thresholds on the importance scores while a ROC curve plots the true positive rate versus the false positive rate. The results shown in Figure 3.3 suggest BWERF had a persistent advantage over GENIE3 in constructing gene regulatory network for a pathway. To summarize these curves, two statistics, the area under the PR curve (AUPR) and the area under the ROC curve (AUROC) were computed (Table 3.1). The improvement of BWERF over GENIE3 was obvious.

Figure 3.3 Comparison of precision-recall (PR) and receiver operating characteristic (ROC) curves of BWERF and GENIE3 for three mouse microarray datasets from Atlas of Pluripotency Evidence (ESCAPE), which were used in [101]
Table 3.1 AUPR and AUROC values for mouse datasets using BWERF and GENIE

<table>
<thead>
<tr>
<th></th>
<th>BWERF</th>
<th></th>
<th>GENIE3</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dataset 1</td>
<td>Dataset 2</td>
<td>Dataset 3</td>
<td>Dataset 1</td>
<td>Dataset 2</td>
<td>Dataset 3</td>
</tr>
<tr>
<td>PR</td>
<td>0.2405</td>
<td>0.1457</td>
<td>0.1958</td>
<td>0.1312</td>
<td>0.0705</td>
<td>0.1958</td>
</tr>
<tr>
<td>ROC</td>
<td>0.7778</td>
<td>0.8150</td>
<td>0.8134</td>
<td>0.6499</td>
<td>0.6868</td>
<td>0.6820</td>
</tr>
</tbody>
</table>

3.3.3 Comparison of ML-hGNRs constructed from *Arabidopsis* microarray data using BWERF and GENIE3

A four-layered hGRN was built, shown in Figure 3.4A, with 22 lignocellulosic pathway genes being used as bottom and 1602 TF being used as the input for the upper layers. A total of 14 positive TFs including MYB43, MYB46, MYB52, MYB58, MYB63 MYB83, MYB85, MYB103, SND1, 2, 3 and NST1, 2 and KNAT7 [102, 103] were identified as known positive regulators of lignocellulosic biosynthesis. In a previous work, 20 positive known TFs were identified by bottom-up GGM algorithm when 25 pathway genes and 1622 TFs were used [19]. However, these 20 positive TFs were identified by bottom-up GGM in a total of 1507 edges while the BWERF identified the 14 positive TFs in a total of 90 edges. Of these genes, MYB58, MYB63, SND1, 2, 3, and NST2 were located at the secondary layers in both networks built by Bottom-up GGM and BWERF. In the ML-hGRN constructed with BWERF, the regulatory relationships were sparser and more discernible, and the “top-down chains of commands” are much easier to be identified for experimental validation.
3.4 Discussion

At the time being, little is known about regulatory layers above the majority of over 500 metabolic and canonical pathways. With the availability of terabyte of gene expression data, it is imperative to develop novel methods to construct hGRNs via reverse-engineering approaches. This is because the new methods provide the means to recognize the novel TFs in multilayered hGRNs that can greatly facilitate our understanding and decipher how biological pathways are regulated. In a previous study, a bottom-up GGM algorithm was
developed to accomplish this goal through implementation of a hypothesis testing (Wald test-based approach) to each combined triple genes: one TF from TF pool and two current bottom-layered genes, leading to the identification of the significant triple regulatory blocks for building an ML-hGRN. The bottom-up GGM algorithm evaluates the triple genes by comparing the difference between the correlation coefficient of the two bottom-layered genes and the partial correlation coefficient of the two bottom-layered genes given a TF at the immediately upper layer. When the difference is statistically significant, the TF was defined to be the regulator of the two bottom-layered genes [19]. This is because the correlation coefficient of the two bottom-layered genes represents their coordination in the presence of the TF, whereas the partial correlation coefficient reflects their coordination after the effect of the TF on both bottom-layered genes is removed. The difference represents the interference strength of the TF on two bottom-layered genes. Though bottom-up GGM is very efficient in identifying regulatory genes (nodes) associated with a pathway through constructing multilayered hGRNs, a large number of edges are generated in the resulting hGRNs owing to exhaustive combinations of all remained TFs in the pool and all paired bottom-layered genes [19]. Too many edges benefit novel TF recognition, but make it hard to identify the regulatory edges for experimental validation. In addition, the regulatory strength of each edge in a network built with bottom-up GGM algorithm cannot be well characterized due to different regulation strengths among different triplet genes. For these reasons, a backward elimination random forest (BWERF) for constructing MH-hGRNs was developed, which can be used as an alternative approach especially when recognition of implicit regulatory edges is prioritized to be a primary goal. Compared to the bottom-up GGM algorithm, BWERF can significantly reduce the number of edges in the ML-hGRNs constructed. For example, the ML-hGRN operating above the lignocellulosic pathway constructed with BWERF had only 90 edges (Fig 3) in contrast to the 1507 edges in the ML-hGRN constructed with bottom-up GGM [19].

BWERF is based on the random forest model, and thus inherits the advantages of the random forest model in determining the important regulatory variables to each pathway gene. First, the random forest model is applicable to the data sets where the number of variables is much larger than the number of samples. The random forest model uses a randomly selected subset of variables at each splitting node, and thus important variables can be correctly separated from a large number of non-important regulatory variables. Secondly, the random forest model can detect non-linear regulatory relationships among variables. Thirdly, if only the parameter of $n_{tree}$ in random forest model is big enough, the random forest model is unlikely to be overfitted. Lastly, the criteria for choosing splitting variable in the decision tree for the random forest model is to maximally decrease the variance of dependent variables, and the decreased variances are used to characterize the importance of the variables. Therefore, the variable importance values returned by the random forest model are more intuitive. BWERF inherits all these features of the random forest model that are critically important for reverse-engineering of gene networks from the data where gene variables far exceed the number of samples.
GENIE3 was used as a comparison method because it also uses the random forest model to construct gene regulatory networks. However, GENIE3 is not tailored for constructing ML-hGRNs. GENIE3 creates the random forest model for each pathway gene only once while BWERF creates the random forest model for each pathway gene multiple times as it conducts backward elimination in a recursive manner. BWERF also considers the high correlation among a group of pathway genes and uses it by aggregating the importance values of TFs for ranking TFs. As we know that the gene regulatory relationships are intricate and the number of TFs is much larger than the number of samples. Non-regulatory TFs to the pathway of interest can severely twist the importance values of regulatory TFs if they are not appropriately modeled. The backward elimination step has its value in two aspects. First, it helps the regulatory genes with medium-level regulatory strength to emerge from noise variables, as shown with the “toys data”; secondly, the backward elimination can enlarge the range of importance values of the variables, which makes true regulatory variables more differentiable from noise variables. Although it takes more computational time to run the backward elimination, the running time is acceptable with moderate computational power. For example, when I used the Linux server with 22 cores to construct an ML-hGGRN from a compendium Arabidopsis thaliana microarray dataset with 128 samples, it took about one and a half hours to run.

Compared to GENIE3, the best performer in DREAM4 In Silico Multifactorial challenge [42], BWERF obviously has increased accuracy owing to the backward elimination to exclude non-regulatory TFs. The backward elimination was conducted recursively to the input regulatory genes towards obtaining the genuine regulators peculiar to a pathway gene, each time with the controlled deletion rate to avoid overdoing and to ensure only less important TFs were removed. In this research, an elimination rate of 10% was used, which showed a considerable improvement and also the acceptable computational time. In fact, there was a noticeable improvement in BWERF over GENIE3 when the elimination rate was as high as 50%. When elimination rate was decreased from 50%, the order of the importance of TFs to a pathway gene changed slightly, but the variance of the importance of each TF decreased accordingly, resulting in a more stable importance list. Though the computing time increased accordingly, the benefit is overweighed the time cost, especially considering the years may be taken for biologists to conduct experimental validation.

3.5 Conclusion

A random forest algorithm with backward elimination was developed for directly constructing an ML-hGRN that operates above a given metabolic or canonical pathway using microarray or RNA-Seq datasets. The algorithm was evaluated with both a synthetic “toys dataset” and two real gene expression data sets from Arabidopsis thaliana and Mus musculus (mouse), leading to the networks with significantly enriched known positive regulatory genes and much less number of regulatory edges. The efficiency and accuracy were corroborated by both PR and ROC curves and the over responded positive regulators in the final output. The BWERF method is especially useful for biologists who would like
to identify the hierarchical regulators associated with a metabolic pathway or a biological process of interest from exploded gene expression data in public repositories.
4 JRmGRN: Joint reconstruction of multiple gene regulatory networks with common hub genes using data from multiple tissues or conditions\(^2\)

Joint reconstruction of multiple gene regulatory networks (GRNs) using gene expression data from multiple tissues/conditions is very important for understanding common and tissue/condition-specific regulation. However, there are currently no computational methods available for directly constructing such multiple GRNs that not only share some common hub genes but also possess tissue/condition-specific regulatory edges.

In this chapter, a new Gaussian graphical model for joint reconstruction of multiple gene regulatory networks (JRmGRN) was proposed, which highlighted the identification of hub genes using gene expression data from several tissues/conditions. Under the framework of the Gaussian graphical model, JRmGRN method constructs the GRNs through maximizing a penalized log-likelihood function. It was formulated as a convex optimization problem, and then solved it with an alternating direction method of multipliers (ADMM) algorithm. The performance of JRmGRN was first evaluated with synthetic data and the results showed that JRmGRN outperformed several other methods for reconstruction of GRNs. This method was also applied to real *A. thaliana* RNA-seq data from two light regime conditions in comparison with other methods, and both common hub genes and some conditions-specific hub genes were identified with higher accuracy and precision.

4.1 Introduction

Though all cells in a multicellular organism carry out some common processes that are essential for survival, different tissues can exhibit some unique patterns in gene expression that help define their phenotypes. In addition, some organisms like sessile plants may experience various environmental conditions in particular stresses. These common and tissue/condition-specific processes are ultimately controlled by GRNs that contain both common and tissue/condition-specific hubs. These hubs play critical roles for organisms to complete their life cycle. For example, abiotic and biotic stresses-responsive genes in rice have 70% in common and these genes showed conserved expression status, and the majority of the rest were down-regulated in abiotic stresses and up-regulated in biotic stresses [104], indicating the presence of common hubs and network between two conditions. The local GRNs for different environmental conditions have been built in *Arabidopsis* [105, 106]. Tissue-specific genes for 38 tissues have been identified in humans and the GRNs for each of these 38 tissues in humans have been built [107] and analyzed.

---

\(^2\) The material contained in this chapter has been published in Bioinformatics, Volume 34, Issue 20, 15 October 2018, Pages 3470–3478, https://doi.org/10.1093/bioinformatics/bty354.
Comparison of global GRNs [108] has revealed that the GNRs are largely conserved and share remarkable commonalities though they can change in response to environmental stimuli or at different tissue types. The high similarity in GRNs is primarily caused by the relatively smaller number of tissue/condition-specific nodes. For example, 23.4% genes were indicated to be tissue-specific (with complicity equal to one) after studying multiple tissues of humans [107]. However, some local GRNs may be subject to some local topology changes [109, 110], making some regulatory interactions exist in all tissues or conditions while some others exist only in a specific tissue or specific treatment. Therefore, the identification of both common and tissue- or condition-specific gene regulation provides key insights into complex biological systems [111]. In the past two decades, advances in microarray and RNA-seq technology have led to the generation of an enormous wealth of gene expression data across various cell/tissue types and conditions. Although these data sets provide a valuable opportunity to more robustly reconstruct condition-specific GRNs, there are very limited methods for modeling the complicated GRNs with high accuracy. Advanced and highly efficient methods are still in great demand.

Gaussian graphical models (GGMs) are widely used to reconstruct gene networks using gene expression data [19]. The models assume that gene expression data on p genes from each sample follows a multivariate normal distribution with mean \( \mu \) and covariance matrix \( \Sigma \), where \( \mu \) is a vector with p elements and \( \Sigma \) is a p\times p positive definite matrix. The conditional independence of two genes given other genes corresponds to a zero entry in the inverse covariance matrix \( \Sigma^{-1} \) (also called the precision or concentration matrix) [33]. Usually, we set \( \Theta = \Sigma^{-1} \), called precision matrix or concentration matrix. Gaussian graphical models have the advantage of reconstructing direct dependencies between genes that represent edges in the reconstructed network: an edge corresponds a non-zero entry in \( \Sigma^{-1} \). A natural way to estimate \( \Sigma^{-1} \) is by maximizing the log-likelihood of the data, which result in an estimation of precision matrix \( \hat{\Theta} = S^{-1} \) where \( S \) is the sample covariance matrix.

However, directly applying GGM to reconstruct GRN is not applicable due to two problems. First, since the number of samples (n) is generally much less than the number of genes (p) from gene expression data, the sample covariance matrix \( S \) becomes singular and thus it is impossible to computing the inverse. Second, even if the sample covariance matrix is not singular, the elements in the estimated precision matrix \( \hat{\Theta} \) are in general not exactly equal to zero. For these reasons, Yuan and Lin [34] proposed to maximize a L1 regularized log-likelihood function. Similar to LASSO regression [28], they put a penalization on the sum of absolute value of each element in the precision matrix, which leads to a sparse and positive definite estimation of \( \Theta \). GLASSO [35] is a fast algorithm to solve this optimization problem.

When applying GGMs to reconstruct gene regulatory networks, the underlying assumption is that each observation is drawn from the same distribution. However, when the gene expression data come from different tissues or under different treatments, this assumption is inappropriate. In this case, if one insists on modeling the gene expression data by one
GRN, the result would be dubious and we cannot obtain the differential network. A straightforward method to obtain the differential network is to reconstruct the network of each condition separately and then find the difference between them. However, this procedure ignores the similarity shared between GRNs across different tissues/treatments, which is critically important to reconstruct the GRNs, especially when the sample size is small. To reconstruct these dependent GRNs, Guo et al. [112] proposed a joint penalized model using a hierarchical penalty and derived the convergence rate and sparsity properties of the resulting estimators. Danaher et al. [16] proposed a joint graphical LASSO model (JGL) to estimate multiple GRNs simultaneously. They proposed a fused graphical LASSO penalization and a group graphical LASSO penalization in addition to the sparsity penalization. In fused graphical lasso, the corresponding elements in the precision matrices are encouraged to have the same values. In group graphical lasso, the precision matrices in different conditions are encouraged to have similar sparsity pattern.

The above-mentioned methods do not impose any structural information of gene networks. That is, each gene has approximately the same number of interactions within the network, and each pair of nodes has equal probability to be an edge, and all edges are independent. However, recent evidence points to scale-free properties in biological networks [37, 38], in which most genes interact with a few partners whereas a small proportion of genes, called hub genes, are densely-connected to many other genes (high connectivity). To incorporate hub genes in GRNs, Liu and Ihler [39] replaced the l1 regularization in GLASSO with a power law regularization and optimized the objective function by solving a sequence of iteratively reweighted l1 regularization problems, where the regularization coefficients of nodes with a high degree were reduced, which encouraged the appearance of hub genes. Tan et al. [18] introduced a row-column overlap norm penalty to incorporate hub genes explicitly. In their model, called hub graphical LASSO (HGLASSO), the precision matrix $\Theta$ was decomposed into two parts, one is elementary matrix $Z$, the other is hub matrix $V$, where $Z$ is a symmetric matrix that is encouraged to be sparse, $V$ is a matrix whose columns are encouraged to be either entirely zero or almost entirely non-zero through the $l1/lq$ norm penalization. The non-zero columns of $V$ correspond to hub genes.

The aim of this research is to develop new and more accurate method for: (1) construction of GRNs containing the important common hubs that may play essential roles for survival and/or adaptation; (2) construction of GRNs containing tissue/condition-specific regulatory relationships that help us to understand phenotypes/traits of interest. In this manuscript, it was assumed that a network for a specific tissue/condition can be decomposed into an elementary network that is unique to the tissue/condition, and a common network centered on hub genes that is shared across multiple tissues/conditions. Based on this hypothesis, a new method was proposed to jointly reconstruct multiple GRNs for multiple tissues/conditions in just one effort. This method, JRmGRN, is different from the aforementioned methods. The methods from Yuan and Lin [34], Danaher et al. [16] cannot model hub genes. Although the methods from Liu and Ihler [39] and Tan et al. [18] can be used to model hub genes, their methods are dedicated to reconstruction of a gene network from each data set independently. With the availability of enormous amount of
gene expression data from multiple tissues/conditions in public repositories, it is important to use data sets from multiple tissues or conditions together to identify common hub genes across multiple tissues or conditions and some tissue- or condition-specific hub genes, which will advance our understanding on regulation of biological processes and pathways. The JRmGRN method hypothesizes that there are common hub genes in different tissues or under different environmental conditions. Figure 4.1 illustrates two example networks obtained from two tissues or conditions. There are many common edges (dash green) between two networks and some tissue- or condition-specific edges belonging to only one of the two networks (e.g. solid red and solid blue).

Figure 4.1 A toy example of two gene regulatory networks from two tissues or environmental conditions.

4.2 Methods

4.2.1 Gaussian graphical model and regularization

Suppose that there are K datasets, $Y^{(1)}, \ldots, Y^{(K)}$, where $K \geq 2$, to represent gene expression data from K tissues or conditions. $Y^{(k)}$ is a $n_k \times p$ matrix where $n_k$ is the number of samples and $p$ is the number of genes in the $k$th data set. Additionally, it was assumed that the rows of $Y^{(k)}$ are independent and each row of $Y^{(k)} \sim N(\mu_k, \Sigma_k)(k = 1, \ldots, K)$. Denote $S^{(k)} = \frac{1}{n_k} (Y^{(k)} - \mu_k)'(Y^{(k)} - \mu_k)$ as the sample covariance matrix of $Y^{(k)}$. The precision matrix, $\Theta^{(k)}$, is the inverse matrix of the covariance matrix $\Sigma_k$. For a gene regulatory network, the non-zero element $\theta_{ij}^{(k)}(i \neq j)$ in $\Theta^{(k)}$ indicate there is a conditional correlation between gene i and j for the kth tissue/condition. Since the number of genes, p, is large and only a small portion of genes are associated, most of the elements in $\Theta^{(k)}$ are expected to be zero. In addition, a few (hub) genes are expected to be associated with many other genes for different tissues or conditions, so the precision matrix $\Theta^{(k)}$ can be decomposed into two parts: one represents the elementary network for the kth tissue/condition and the other part represents the network for hub genes. Based on such sparsity and decomposition of $\Theta^{(k)}$, the joint reconstruction of multiple gene regulatory networks with common hubs (JRmGRN) was proposed by solving the following penalized log-likelihood function,
\[
\arg\min_{\{\Theta^{(k)} \in \mathcal{S}^+\}} \sum_{k=1}^{K} -n_k \left( \log \left( \det(\Theta^{(k)}) \right) - \text{trace}(S^{(k)} \Theta^{(k)}) \right) + P(\{\Theta\})
\]

where \(Z^{(k)} + V + V^T = \Theta^{(k)}\) for \(k = 1, ..., K\) and \(\mathcal{S}^+\) as the collection of symmetric positive semidefinite matrix. In the above penalized function, \(||V||_1 = \sum_{ij} |V_{ij}|\) denotes the sum of the absolute value of each element in \(V\). \(||V||_{1,2} = \sum_{j=1}^{p} ||V_j||_2\) where \(V_j\) is the \(j^{th}\) column of matrix \(V\). So \(||V||_{1,2}\) denotes the sum of the l2 norm of each column in \(V\). In (4.1), the first part is the log-likelihood function of the data based on the precision matrix and the second part is the penalized function with \(l_1\) norm to help us model the sparsity of \(\Theta^{(k)}\). The precision matrix \(\Theta^{(k)}\) under the \(k^{th}\) tissue/condition were decomposed into two parts: \(Z^{(k)}\) and \(V\). \(Z^{(k)}\) can be seen as the elementary network under the \(k^{th}\) tissue/condition. \(V\) represents common hub genes across all tissues/conditions. Four items in the penalized log-likelihood function \(P(\{\Theta\})\) were used to ensure that the reconstructed networks would satisfy the desired properties. The purpose and the prior assumption of the four penalties were summarized in following:

- \(\lambda_1 \sum_{k=1}^{K} ||Z^{(k)} - \text{diag}(Z^{(k)})||_1\). The prior assumption is that elementary network under each condition is sparse, meaning that most of the elements in \(Z^{(k)}\) is zero. Therefore, \(l_1\) penalty was used to encourage the off diagonal elements of \(Z^{(k)}\) to be zero.

- \(\lambda_2 \sum_{k < k'} ||Z^{(k)} - Z^{(k')}||_1\). The prior assumption is that each \(Z^{(k)}\) contains some unique edges, representing the specific network for the \(k^{th}\) tissue/condition; but \(\{Z^{(k)}\}\) have many common edges due to similarity among networks. Therefore, \(l_1\) penalty was used to encourage the elementary networks across different conditions to be the same.

- \(\lambda_4 ||V||_{1,2}\). It was assumed matrix \(V\) contains zero columns and dense non-zero columns, where the non-zero columns represent the common hub genes across all tissues/conditions. Therefore, group LASSO penalty was used to force some columns of \(V\) to be zero columns.

- \(\lambda_3 ||V||_1\). For the non-zero columns of \(V\), \(l_1\) penalty was also used to encourage some elements to be zero, so a hub gene would not connect to all other genes.
\( \lambda_1, \lambda_2, \lambda_3, \) and \( \lambda_4, \) are non-negative tuning parameters. Note that the model is different from methods that use a Gaussian graphical model and regularization to reconstruct gene associate networks [34, 35]. For example, GLASSO can only use data from single tissue or condition and cannot model hub genes [35]. JGL can use data from multiple tissues or conditions but cannot model hub genes [16]. HGLASSO incorporates hub genes in the reconstruction of gene networks but only handles data from a single tissue or condition [18]. Although we may reconstruct gene networks for each tissue or condition using GLASSO/JGL/HGLASSO then use reconstructed networks to identify common hub genes, such an approach is too subjective and less efficient. In contrast, the proposed method is to reconstruct gene networks with hub genes by jointly using data sets from multiple tissues/conditions, thus is more efficient, powerful and accurate.

### 4.2.2 The algorithm to estimate parameters

For fixed values of tuning parameters \( \lambda_1, \lambda_2, \lambda_3, \) and \( \lambda_4, \) the expression of (4.1) is a convex optimization problem, which can be solved by efficient algorithms available. The convexity of (4.1) can be proved by the following facts: the function of negative log determinant is a convex function, the norm functions are convex functions, and the nonnegative combination of convex functions is a convex function. The problem (4.1) were solved using the alternating directions method of multipliers (ADMM) algorithm, which allows us to decouple some of the terms in (4.1) that are difficult to optimize jointly. For more details on the ADMM algorithm and its convergence properties, please consult the previous publication [113].

The expression of (4.1) was written as a convex minimization problem with two blocks of variables and two separable functions as follows:

\[
\begin{align*}
\min & \ \phi(X) + \psi(X) \quad \text{s.t. } X - \bar{X} = 0 \\
\text{where } X &= (\Theta^{(k)}, Z^{(k)}, \mathcal{V}), \bar{X} = (\bar{\Theta}^{(k)}, \bar{Z}^{(k)}, \bar{\mathcal{V}}), \quad \text{and} \\
\phi(X) &= f(\Theta^{(k)}) + g(Z^{(k)}) + h(\mathcal{V}) \\
\psi(X) &= \sum_{k=1}^{K} I(\bar{\Theta}^{(k)} = \bar{Z}^{(k)} + \bar{\mathcal{V}} + \mathcal{V}')
\end{align*}
\]

where

\[
\begin{align*}
f(\Theta^{(k)}) &= \sum_{k=1}^{K} -n_k \left( \log \det \Theta^{(k)} - \text{trace} (S^{(k)} \Theta^{(k)}) \right) \\
g(Z^{(k)}) &= \lambda_1 \sum_{k=1}^{K} \left\| Z^{(k)} - \text{diag}(Z^{(k)}) \right\|_1 + \lambda_2 \sum_{k < k'} \left\| Z^{(k)} - Z^{(k')} \right\|_1
\end{align*}
\]

61
\[ h(V) = \lambda_3 ||V||_1 + \lambda_4 ||V||_{1,2} \quad (4.7) \]

\( I(P) \) is the indicator function on proposition \( P \)

\[ I(P) = \begin{cases} 
  0 & \text{if } P \text{ is TRUE} \\
  \infty & \text{if } P \text{ is FALSE} 
\end{cases} \]

The scaled augmented Lagrangian for (4.2) is given by

\[ L(\Theta, Z, V, W) = \phi(X) + \psi(\bar{X}) + \frac{\rho}{2} ||X - \bar{X} + W||_F^2 \quad (4.8) \]

where \( W = (\{W^{(k)}_\Theta\}, \{W^{(k)}_Z\}, W_V) \) is the dual variable and \( \rho \) is a parameter.

The iteration of ADMM applied to solve (4.8) can be described as follows:

\[ \begin{align*}
X^{(t+1)} &= \arg\min_X \left\{ \phi(X) + \frac{\rho}{2} ||X - \bar{X}^{(t)} + W^{(t)}||_F^2 \right\} \\
\bar{X}^{(t+1)} &= \arg\min_{\bar{X}} \left\{ \psi(\bar{X}) + \frac{\rho}{2} ||X^{(t+1)} - \bar{X} + W^{(t)}||_F^2 \right\} \\
W^{(t+1)} &= W^{(t)} + X^{(t+1)} - \bar{X}^{(t+1)} 
\end{align*} \quad (4.9) \]

Since the objective function is completely separable with respect to the variables \((\Theta^{(k)}, Z^{(k)}, V)\), updating \(X^{(t+1)}\) can be achieved by updating each variable and is given by:

\[ \Theta^{(k)} = \arg\min_{\Theta} \left\{ -n_k \left( \log \det \Theta - \text{trace} \left( S^{(k)} \Theta \right) \right) + \frac{\rho}{2} ||\Theta - \bar{\Theta}^{(k)} + W^{(k)}_\Theta||_F^2 \right\} \quad (4.10) \]

Let \( VDV^T \) denote the eigen decomposition of \( S^{(k)} - \frac{\rho}{n_k} (\bar{\Theta}^{(k)} - W^{(k)}_\Theta) \), the solution is given by \( V \tilde{D}V^T \), where \( \tilde{D} \) is the diagonal matrix with \( j^{th} \) diagonal element as follows: (see [114] for the derivation).

\[ \frac{n_k}{2\rho} (-D_{jj} + \sqrt{D_{jj}^2 + 4\rho/n_k}) \]

\[ Z^{(t+1)} = \arg\min_{Z \in \mathcal{S}^+} \left\{ \lambda_1 \sum_{k=1}^K ||Z^{(k)} - \text{diag}(Z^{(k)})||_1 + \lambda_2 \sum_{k<k'} ||Z^{(k)} - Z^{(k')}||_1 \\
+ \frac{\rho}{2} \sum_{k=1}^K ||Z^{(k)} - Z^{(k)} + W^{(k)}_Z||_F^2 \right\} \quad (4.11) \]

The objective function is completely separable with respect to each pair of matrix elements \((i, j)\). Therefore, it is equivalent to solve a \( p \times p \) optimization problem.
\[ Z_{ij(t+1)} = \arg \min_{Z_{ij}^{(k)}, Z_{ij}^{(k')}} \left\{ \lambda_1 1_{i \neq j} \sum_{k=1}^{K} |Z_{ij}^{(k)}| + \lambda_2 \sum_{k<k'} |Z_{ij}^{(k)} - Z_{ij}^{(k')}| + \frac{\rho}{2} \sum_{k=1}^{K} (Z_{ij}^{(k)} - Z_{ij}^{(k')} + W_{Z_{ij(t)}}^{(k)})^2 \right\} \] (4.12)

This form belongs to a class of fused LASSO problems. It can be solved in \(O(K \log K)\) operations, see [115] for more details. When \(K = 2\), there is a very simple closed form solution for the case \(\lambda_1 = 0\), see [16].

\[ (Z_{ij}^{(1)}, Z_{ij}^{(2)}) = \begin{cases} (A_{ij}^{(1)} - \frac{\lambda_2}{\rho} A_{ij}^{(2)} + \frac{\lambda_2}{\rho}), & \text{if } A_{ij}^{(1)} > A_{ij}^{(2)} + \frac{2\lambda_2}{\rho} \\ (A_{ij}^{(1)} + \frac{\lambda_2}{\rho} A_{ij}^{(2)} - \frac{\lambda_2}{\rho}), & \text{if } A_{ij}^{(2)} > A_{ij}^{(1)} + \frac{2\lambda_2}{\rho} \\ \left(\frac{A_{ij}^{(1)} + A_{ij}^{(2)}}{2}, \frac{A_{ij}^{(1)} + A_{ij}^{(2)}}{2}\right), & \text{if } |A_{ij}^{(2)} - A_{ij}^{(1)}| \leq \frac{2\lambda_2}{\rho} \end{cases} \] (4.13)

where \(A^{(k)} = Z_{ij}^{(k)} - W_{Z_{ij(t)}}^{(k-1)}\), when \(\lambda_1 > 0\), the solution to (4.12) can be obtained through soft-thresholding by \(\lambda_1 / \rho\).

\[ V_{(t+1)} = \arg \min_{V} \left\{ \lambda_3 ||V||_1 + \lambda_4 ||V||_{1,2} + \frac{\rho}{2} ||V - \breve{V} + W_V||_F^2 \right\} \] (4.14)

The objective function is column separable, Let \(V_j\) be the \(j\)th column of matrix \(V\), which can be solved by

\[ V_j = \arg \min_{V_j} \left\{ \lambda_3 ||V_j||_1 + \lambda_4 ||V_j||_2 + \frac{\rho}{2} ||V_j - \breve{V}_j + W_{V(j)}||_2^2 \right\} \] (4.15)

This problem looks like an elastic net problem [116], the difference between this problem and the elastic net is that the l2 norm is not squared. The solution is given in [16]:

Let \(C_j = \breve{V}_j - W_{V(j)}\)

\[ V_j = \max \left(1 - \frac{\lambda_4}{\rho} \frac{S(C_j, \frac{\lambda_3}{\rho})}{\|S(C_j, \frac{\lambda_3}{\rho})\|_2}, 0\right) \cdot S(C_j, \frac{\lambda_3}{\rho}) \] (4.16)

where \(S(C_j, \frac{\lambda_3}{\rho}) = \text{sign}(C_j) \cdot \max(\text{abs}(C_j) - \frac{\lambda_3}{\rho}, 0)\) is the soft-thresholding operation.

Although the objective function is not separable with respect to the variables \((\breve{S}^{(k)}, Z^{(k)}, \breve{V})\), updating \(X_{(t+1)}\), can be achieved as follows:
\[
\gamma^{(k)} = \frac{\rho}{2}\left((\Theta^{(k)} + W^{(k)_\Theta}) - (Z^{(k)} + W^{(k)_Z}) - (V + W_V - (V + W_V)^t) - 2\sum_{k=1}^{K} \gamma^{(k)} \right) (4.17) \\
\]

\[
\bar{\Theta}^{(k)} = \left(\Theta^{(k)} + W^{(k)_\Theta}\right) - \frac{1}{\rho} \gamma^{(k)} (4.18) \\
\bar{Z}^{(k)} = \left(Z^{(k)} + W^{(k)_Z}\right) + \frac{1}{\rho} \gamma^{(k)} (4.19) \\
\bar{V} = (V + W_V) + \frac{2}{\rho} \sum_{k=1}^{K} \gamma^{(k)} (4.20) \\
\]

### 4.2.3 Selection of tuning parameters

As pointed out in the previous study [117], a careful choice of the tuning parameters is much more important in this case than in the ordinary GLASSO since there are four tuning parameters. There are a wide variety of criteria to select appropriate tuning parameters. One criterion is validation set likelihood, a score that tries to assess how effective the estimator is at modeling new instances. However, three questions arise. First, if we partition the data as training set and validation set, it is inappropriate because the number of samples is very small. Secondly, if we use cross-validation score, we have to train multiple models and it is very slow. Thirdly, as discussed in [20], the optimal parameters under prediction-optimal value will, in general, have too many non-zero variables. In this research, a Bayesian information criterion (BIC)-type quantity was used to select tuning parameters. Recall that the precision matrix \(\Theta^{(k)}\) were factorized into \(\Theta^{(k)} = Z^{(k)} + V + V^T\), and placed a \(l_1\) penalty on \(Z^{(k)}\), a \(l_1\) penalty on the difference of \(\{Z^{(k)}\}\), and a \(l_1/l_2\) penalty on \(V\). \(\lambda_1, \lambda_2, \lambda_3, \text{and} \lambda_4\) were chosen to minimize the expression of 4.21, which is a tradeoff between model likelihood and model complexity.

\[
\left\{ \sum_{k=1}^{K} \left(-n_k \log(\det(\bar{\Theta}^{(k)})) + n_k \text{trace}(S^{(k)} \bar{\Theta}^{(k)})\right) + \right. \\
\left. \sum_{k=1}^{K} \log(n_k) |Z^{(k)}| - \log(n) |\cap Z^{(k)}| + \log(n) (v + c(|V| - v)) \right\} (4.21) \\
\]

where \(\{\bar{\Theta}^{(k)}, \bar{Z}^{(k)}, \bar{V}\}\) is the estimated parameters with a fixed set of tuning parameters \((\lambda_1, \lambda_2, \lambda_3, \lambda_4)\). \(|\bar{Z}^{(k)}|\) is the cardinality of \(\bar{Z}^{(k)}\), \(|\bar{V}|\) is the cardinality of \(\bar{V}\), \(v\) is the number of estimated hubs, \(c\) is a constant between zero and one. BIC in its standard form consists of a sum of model likelihood and \(\log(n) * d/2\), where \(n\) is the number of samples and \(d\) is the number of free parameters. In such case, as the elements in \(\bar{Z}^{(k)}\) and \(\bar{V}\) are inter-related, it is difficult to estimate the number of free parameters. Therefore, this BIC-type quantity
was proposed for selecting the set of tuning parameters, which is similar to the BIC quantity in [18].

BIC is just a guide for turning parameter selection. In reality, we may also consider other factors in addition to BIC. For example, network interpretability, stability, and the desire for an edge set with a low false discovery rate, as pointed out by some researchers [118].

The grid search was used to find the tuning parameters that maximized the expression of (4.21). The computational complexity for the network construction with a fixed set of tuning parameters mainly depends on the number of genes included in the analysis. The grid search is feasible when the number of genes falls into small to moderate ranges but quickly becomes impractical for a large number of genes. In this situation, we need to explore some theoretical properties of the problem that can be used to guide the search of tuning parameters.

Similar to Lemma 4.1 in [16], the following fact holds.

**Lemma 1.** Suppose that the solution to the expression of (4.1) is block diagonal with known blocks. That is, if the features are properly reordered and the estimated inverse covariance matrix takes the form

\[ \hat{\Theta}^{(k)} = \begin{pmatrix} \hat{\Theta}_1^{(k)} & 0 \\ 0 & \hat{\Theta}_2^{(k)} \end{pmatrix} \]

where each of \( \hat{\Theta}_1^{(1)}, ..., \hat{\Theta}_1^{(K)} \) has the same dimension, then \( \hat{\Theta}_1^{(1)}, ..., \hat{\Theta}_1^{(K)} \) and \( \hat{\Theta}_2^{(1)}, ..., \hat{\Theta}_2^{(K)} \) can be obtained by solving expression of (4.1) on just the corresponding set of features.

**Theorem 1.** A sufficient condition for the solution to (4.1) to be block diagonal with blocks given by \( C_1, C_2, ..., C_T \) is that

\[ \min \left\{ \frac{\lambda_1}{n_1}, ..., \frac{\lambda_1}{n_k}, \frac{\lambda_3}{2 \sum_{k=1}^{K} n_k} \right\} \geq \left| S_{ij}^{(k)} \right| \text{ for all } j \in C_{v}, j' \in C_{t'}, t \neq t' \]  

(4.22)

**Proof:**

Let \( \Omega = \bigcup_{k=1}^{K} \{ C_k \times C_k \} \), \( S_{max} = \max_{(i,j) \in \Omega} S_{ij}^{(k)} \), let \( A_{\Omega} \) denote the restriction of the matrix \( A \) to the set \( \Omega \), that is

\[ (A_{\Omega})_{ij} = \begin{cases} A_{ij}, & \text{if } (i,j) \in \Omega \\ 0, & \text{if } (i,j) \not\in \Omega \end{cases} \]

Assume \( (\Theta^{(k)}, Z^{(k)}, V) \) is a feasible solution of (4.1), then \( (\Theta_{\Omega}^{(k)}, Z_{\Omega}^{(k)}, V_{\Omega}) \) is also a feasible solution to (4.1). A fact is that if (4.22) holds, then the objective value of (4.1) evaluated at \( (\Theta_{\Omega}^{(k)}, Z_{\Omega}^{(k)}, V_{\Omega}) \) is smaller than the objective value of (4.1) evaluated at \( (\Theta^{(k)}, Z^{(k)}, V) \). By Fischer’s inequality,

\[ -\log \det(\Theta^{(k)}) > \log \det(\Theta_{\Omega}^{(k)}) \]
We need only to prove
\[
\sum_{k=1}^{K} n_k \text{trace} \left( S^{(k)} \Theta^{(k)} \right) + \lambda_1 \sum_{k=1}^{K} \left\| Z^{(k)} - \text{diag}(Z^{(k)}) \right\|_1 + \lambda_2 \sum_{k<k'} \left\| Z^{(k)} - Z^{(k')} \right\|_1 + \lambda_3 \left\| V \right\|_1 + \lambda_4 \left\| V \right\|_{1,2} \geq \sum_{k=1}^{K} n_k \text{trace} \left( S^{(k)} \Theta^{(k)} \right) + \lambda_1 \sum_{k=1}^{K} \left\| Z^{(k)} - \text{diag}(Z^{(k)}) \right\|_1 + \lambda_2 \sum_{k<k'} \left\| Z^{(k)} - Z^{(k')} \right\|_1 + \lambda_3 \left\| V \right\|_1 + \lambda_4 \left\| V \right\|_{1,2}
\]
As \( \lambda_2 \sum_{k<k'} \left\| Z^{(k)} - Z^{(k')} \right\|_1 \geq \lambda_2 \sum_{k<k'} \left\| Z^{(k)} - Z^{(k')} \right\|_1 \) and \( \lambda_4 \left\| V \right\|_{1,2} \geq \lambda_4 \left\| V \right\|_{1,2} \), we need only to prove
\[
\sum_{k=1}^{K} n_k \left\langle S^{(k)}, \Theta^{(k)} \right\rangle + \lambda_1 \sum_{k=1}^{K} \left\| Z^{(k)} \right\|_1 + \lambda_3 \left\| V \right\|_1 \geq 0
\]
\[
\sum_{k=1}^{K} n_k \left\langle S^{(k)}, \Theta^{(k)} \right\rangle = \sum_{k=1}^{K} n_k \left\langle S^{(k)}, Z^{(k)} + V + V^c \right\rangle
\]
\[
\leq \sum_{k=1}^{K} n_k \left\langle S^{(k)}, Z^{(k)} + V + V^c \right\rangle + \sum_{k=1}^{K} 2n_k \left\langle S^{(k)}, V + V^c \right\rangle
\]
\[
\leq \sum_{k=1}^{K} n_k S_{\max} \left\| Z^{(k)} \right\|_1 + \sum_{k=1}^{K} 2n_k S_{\max} \left\| V \right\|_1 \]
\[
\leq \lambda_1 \sum_{k=1}^{K} \left\| Z^{(k)} \right\|_1 + \lambda_3 \left\| V \right\|_1
\]
where the last inequality holds according to the sufficient condition.

\textbf{Theorem 2.} Let \( \{\Theta^{*\left(k\right)}\}, \{Z^{*\left(k\right)}\}, V^* \) be a solution to (4.1), a sufficient condition for \( V^* \) to be a zero matrix is that
\[
\lambda_1 \leq \frac{\lambda_4}{2K\sqrt{p}} + \frac{\lambda_3}{2K}
\]

\textbf{Proof:}
Let \( \bar{Z}^{(k)} = Z^{*\left(k\right)} + V^* + V^{*'} \), \( k = 1, \ldots, K \), let \( \bar{V} = 0 \), then \( \{\Theta^{*\left(k\right)}\}, \{\bar{Z}^{(k)}\}, \bar{V} \) is a feasible solution for (4.1), we want to show that if (4.23) holds, then the objective value of (4.1) evaluated at \( \{\Theta^{*\left(k\right)}\}, \{\bar{Z}^{(k)}\}, \bar{V} \) is smaller than the objective value of (4.1) evaluated at \( \{\Theta^{*\left(k\right)}\}, \{Z^{*\left(k\right)}\}, V^* \). As \( \lambda_2 \sum_{k<k'} \left\| Z^{*\left(k\right)} - Z^{*\left(k'\right)} \right\|_1 = \lambda_2 \sum_{k<k'} \left\| \bar{Z}^{(k)} - \bar{Z}^{(k')} \right\|_1 \)
We need only to prove,

\[
\lambda_1 \sum_{k=1}^{K} \|Z^{(k)} - diag(Z^{*(k)})\|_1 + \lambda_3 \|V^*\|_1 + \lambda_4 \|V^*\|_{1,2}
\]

\[
\geq \lambda_1 \sum_{k=1}^{K} \|Z^{(k)} - diag(Z^{(k)})\|_1
\]

\[
\lambda_1 \sum_{k=1}^{K} \|Z^{(k)} - diag(Z^{(k)})\|_1
\]

\[
\leq \lambda_1 \sum_{k=1}^{K} \|Z^{*(k)} - diag(Z^{*(k)})\|_1 + 2\lambda_1 \sum_{k=1}^{K} \|V^* - diag(V^*)\|_1
\]

\[
\leq \lambda_1 \sum_{k=1}^{K} \|Z^{*(k)} - diag(Z^{*(k)})\|_1 + 2K\lambda_1 \|V^*\|_1
\]

in order to prove above inequality, we need only to prove

\[
(2K\lambda_1 - \lambda_3)\|V^*\|_1 \leq \lambda_4 \|V^*\|_{1,2}
\]

That is

\[
\sum_{j=1}^{p} (2K\lambda_1 - \lambda_3)\|V_j^*\|_1 \leq \sum_{j=1}^{p} \lambda_4 \|V_j^*\|_2
\]

According to Cauchy inequality,

\[
\|V_j^*\|_1 = \langle sign(V_j^*), V_j^* \rangle \leq \sqrt{p} \|V_j^*\|_2
\]

we need only to prove \((2K\lambda_1 - \lambda_3)\sqrt{p} \leq \lambda_4\). This is true when (4.23) holds.

\[\blacksquare\]

If the conditions for Theorem 1 are satisfied, the reconstruction of a big network was decomposed into the reconstruction of two or more small networks separately, thus the computational time for (4.1) could be greatly reduced. With Theorem 2, we could reduce the search space of parameters \(\lambda_3\) and \(\lambda_4\) as these four tuning parameters are related. If \(\lambda_1\) is large and \(\lambda_3, \lambda_4\) are too small, then the elementary network matrices \(Z^{(k)}\) can become very sparse and the number of hub genes becomes too large. On the other hand, if \(\lambda_1\) is small and \(\lambda_3, \lambda_4\) are too large, the elementary network matrices \(Z^{(k)}\) can become dense, and the number of hub genes will become too small. In this research, a uniformed grid of log space from 0.01 to 20 (size=30) were used for parameter \(\lambda_1\), set \(\lambda_2\) to be 0.5, 1, and 2 folds of \(\lambda_1\), set \(\lambda_3\) to be 0.5, 1, 2, 2, 2K folds of \(\lambda_1\), and set \(\lambda_4\) to be 0.1, 0.5, 1, 2, 2, 2K folds of \(\lambda_1\).
4.3 Results

4.3.1 Results from simulated data

Two types of gene networks were simulated, Erdős-Rényi (ER)-based network [119] and Barabási-Albert(BA)-based network [120], and then generated corresponding gene expression data to assess and validate the method developed. JRmGRN was compared with three other GGM based methods, the graphical LASSO (GLASSO) [35], the joint graphical LASSO (JGL) [16], and the graphical LASSO with hubs (HGLASSO) [18]. The precision-recall curves were constructed based on the edges instead of hub genes in the network since GLASSO and JGL do not model hub genes explicitly.

4.3.1.1 Results on ER-based network

In an ER-based network, each pair of nodes was selected with equal probability and connected with a predefined probability. To simulate scale-free ER-based networks, a similar procedure used in [18] was adopted with some modifications. Specifically, for a given number of tissues or conditions (K), genes (p), samples (n_k, k = 1, ⋯, K), the following procedures were used to simulate the ER-based network and corresponding gene expression data. (1) A sparse p×p matrix A was generated by setting each element A_{ij} to be a random number in [−0.25, −0.75]∪[0.25,0.75] with probability α (elementary network sparsity 1 − α) and zero otherwise. This step is the same as the simulation procedure in [18]; (2) The matrix H was first set to be a p×p zero matrix, and then m hub genes were chosen randomly. For each element in the column of H that represents a hub gene, h_{ij}, it was set to be a random number in [−0.25, −0.75]∪[0.25,0.75] with probability β (hub sparsity 1 − β) and zero otherwise, then set H to (H + H^T)/2. (3) To construct the elementary network, Z^{(k)}, it was first set equal A, and then a fraction of δ (network difference) of elements were randomly chosen and reset to be a random number in [−0.25, −0.75]∪[0.25,0.75] with probability α (elementary network sparsity 1 − α) and zero otherwise. Z^{(k)}_{ij} was set to equal Z^{(k)}_{ji} for each i > j so that Z^{(k)} is symmetric. (4) The precision matrix, Θ^{(k)} was decomposed as Z^{(k)} + H. If Θ^{(k)} was not positive definite, the diagonal element of Θ^{(k)} was added by 0.1 − λ_{min}(Θ^{(k)}), where λ_{min}(Θ^{(k)}) is the minimum eigenvalue of Θ^{(k)}. (5) the gene expression of n_k samples were generated for the k-th tissue or condition with n_k independent multivariate normal distribution N(0, (Θ^{(k)})^{-1}).

For the sake of clearness in network display, the simulation was conducted based on 2 tissues or conditions, 40 samples for each tissue or condition. The elementary network sparsity, the hub sparsity, and the network difference were set as 0.98, 0.70, and 0.20, respectively. 3 networks with 80, 160 and 300 genes were simulated respectively. As described above, the BIC and the grid search were used to find the tuning parameters and best model.
Figure 4.2 shows the simulated ER gene network and the estimated gene network with 80 genes. There were 5 hubs genes that had an average of 30 edges. JRmGRN successfully identified 4 hub genes, and 95 out of 119 original edges of these 4 hub genes. Only one hub gene (76), which had only 26 edges, was not identified by JRmGRN. Two genes, 36 and 51, as shown in Figure 4.2B, were not hub genes but were identified as hub genes by JRmGRN. It was found that the numbers of edges of these two genes were 15 and 14, respectively. These numbers were slightly higher than other non-hub genes and deviated toward 30, the average number of edges from 5 hub genes. These results manifested the usefulness of JRmGRN in identifying true hub genes and their corresponding edges through network reconstruction.

JRmGRN was compared with several other methods. The precision-recall curve was constructed based on the edges instead of hub genes in the network since GLASSO and JGL do not model hub genes explicitly (Figure 4.3). The results clearly showed that JRmGRN method, JRmGRN, performed the best in all circumstances. JRmGRN jointly modeled multiple networks simultaneously so that the common network could be constructed more accurately by using data sets from multiple tissues/conditions, which resulted in more accurate tissue/condition-specific networks.
Figure 4.3 Precision-Recall curve of JRmGRN and three existing methods on ER networks.

4.3.1.2 Results on the Barabási-Albert (BA)-based network

The ba-based network is used in [16] to evaluate the performance of network inferring algorithm. A big network consisted of a number of disconnected BA subnetwork. There are no explicit hub genes in these subnetworks; genes that have more high connectivity were considered hub genes. For a given number of tissues or conditions (K), genes (p), samples (nk, k = 1, ⋯, K), the following procedures were used to generate a BA-based network and corresponding gene expression data. (1) p genes were divided into m groups evenly. (2) For each of first (1 − δ)m gene groups, the BA-based subnetwork was the same across all K tissues or conditions and were generated with the function “barabasi.game” from the R “igraph” package. (3) For each of the rest δm gene groups, the BA-based subnetworks were different for each tissue or condition and were generated separately. (4) For each edge in the network, the corresponding element in the precision matrix of the kth tissue/condition, Θ(k), was set to be a random number in [−0.25, −0.75] ∪ [0.25, 0.75]. (5) the gene expression data of nk samples for the kth tissue or condition were generated from independent multivariate normal distribution N(0, (Θ(k))−1).

Figure 4.4 A simulated Barabási-Albert gene network (A) and the predicted network (B).
The simulation was conducted based on two tissues or conditions, 40 samples for each tissue or condition, and each network consisted of eight subnetworks. The elementary network sparsity and hub sparsity were not explicitly implemented. The parameters were varied in the function “`barabasi.game`” from R “`igraph`” package to generate BA-based networks with desired elementary network sparsity ($1 - \alpha = 0.98$). Seven out of eight subnetworks were set to be the same across two tissues or conditions, and one subnetwork is different. Three networks with 80, 160 and 320 genes were simulated respectively. The simulated BA gene network (Figure 4A) and the estimated gene network (Figure 4B) with 80 genes. JRmGRN successfully identified 191 edges with a true positive rate of 0.702, and falsely identified 290 edges with a false positive rate of 0.048. JRmGRN identified 17 genes as hub genes. The average number of edges connected to these 17 genes in the true network was 5.76, and the average number of edges connected to the rest 63 genes in the true network are 3.05. Therefore, the hub genes identified by JRmGRN had a much higher degree of connectivity. As pointed out in [37, 38], the genes with higher degrees of connectivity may be more important in biological development, which validates and manifests the usefulness of JRmGRN.

![Figure 4.5 Precision-Recall curves of JRmGRN in comparison with three other existing methods, GLASSO, JGL and HGLASSO on BA networks.](image)

The comparison of PR curves of JRmGRN and other methods are shown in Figure 4.5. When the number of genes was 80 or less, JRmGRN and JGL had similar performance, and they were better than the other two methods. As the number of genes increased, the performance of JRmGRN surpassed that of JGL and became the best.

### 4.3.2 Results from real RNA-seq data of Arabidopsis thaliana

The *A. thaliana* gene expression data used in this study are RNA-seq data generated from cotyledon tissue of seedlings under two light regime conditions: low and high red:far-red (R:FR). There are 12 samples in each condition, with 2 replicates for each time point of 0.5, 1, 2, 3, 4 and 7 h. The SRA format data were downloaded through the accession identifier “GSE59722” in the NCBI Gene Expression Omnibus (GEO) (http://www.ncbi.nlm.nih.gov/geo/). The Rsusread software package [121] was first used to transform the raw sequence reads to a matrix of raw counts, and then the edgeR quasi-
likelihood pipeline [122] was used to obtain differentially expressed genes (DEGs) following the procedure provided earlier [123]. There were 321 light related DEGs. The Blom transform method [124] was employed to transform the read counts data. The Blom transformation is a rank-based transformation, which back-transforms the uniformly distributed ranks to a standard normal distribution, i.e.

$$x^{\text{blom}}_{ij} = \phi^{-1}\left(\frac{\text{rank}_{i=1,...,n}(x_{ij}) - c}{n - 2c + 1}\right)$$

with $c = 3/8$ and $\phi$ is the standardized cumulative distribution function.

The networks built based on the above method are shown in Figure 4.6. The common network of both low and high R and FR conditions is represented by the green edges with the 15 common hub genes being highlighted in yellow. All of these common hub genes had a connectivity $> 172$, which is at least five times larger than that of any the non-hub gene. Among these 15 hub genes, eight were up-regulated in overall trends (BZO2H3, CCL, TCP11, PLPC5, AT1G62310, AT3G15570, NAC102, and AT3G45260) in response to low and high R:FR treatments. PLPC encodes a blue light receptor protein while BLH10 were down-regulated in overall trends (BLH10, ELIP1, PD1, PEX11B, PLIM2a, WAV2, and POP12) upon low and high R and FR treatments. At least eight genes, including PLPC, BEL10, CCL, PD1, ELIP1, PEX11B, AT3G15570, POP1 and WAV2, were previously reported to encode a protein that interacts with PLPC (PAS/LOV PROTEIN). Their interaction diminishes by blue light [125]. PD1 encodes a plastid-localized arogenate dehydratase required for blue light-induced production of phenylalanine [126] while PEX11B is involved in light response [127]. ELIP1 is light-responsive [128] and plays an essential role in the assembly or stabilization of photosynthetic pigment-protein complexes [129]. CCL’s transcripts are differentially regulated at the level of mRNA stability at different times of day controlled by a circadian clock [130]. AT3G15570 encodes a phototropic-responsive NPH3 family protein. POP1 encodes a member of the NAP subfamily of ABC transporters whose expression pattern is regulated by light and sucrose [131]. WAV2 negatively regulates root bending when roots alter their growth direction in response to environmental stimuli such as light [132].
Figure 4.6 The gene regulatory networks built with JRmGRN using data sets generated from *Arabidopsis thaliana* leaves under low and high red:far-red (R:FR) regimes.

### 4.4 Discussion

The results from synthetic data, which were generated with the ER-based network or the BA-based network, clearly showed that JRmGRN outperformed several other methods, including GLASSO, HGLASSO, and JGL for the reconstructing GRNs. Since common hub genes were explicitly modeled in the ER-based network, it was not surprised to see that JRmGRN had a higher accuracy in identifying common hub genes and had the largest area under the PR curves when the synthetic data set from the ER-based network was used in the evaluation. In contrary, common hub genes were not clearly modeled although the genes that had high connectivity could be seen in the BA-based network. When the synthetic data set from the BA-based network was used in the evaluation, JRmGRN and
JGL had similar performance as a small number of genes were included, whereas JRmGRN had a much better performance than JGL as a large number of genes were used.

When JRmGRN was used to construct gene networks using real gene expression data set generated from *A. thaliana* cotyledons under low to high red:far-red light regime conditions, it successfully built three networks and identified 15 common hub genes, and at least 9 of them were explicitly documented in existing literature for their involvement in light-response or related biological processes. Some of them, like ELIP1, PLPC, and BLH10, play important roles in light perception and harvest in photosynthesis. In addition to these common hub genes, some other genes like bHLH071, a blue-light regulated gene [133], and RSM1, a light-responsive gene [134], are identified to be hubs in low and high R:FR-specific networks, respectively, indicating the usefulness of the method in building two or more condition-specific networks and a common network across all conditions. Other three methods were also implemented to the real data set from *A. thaliana* under high red:far-red light and low red:far-red condition. JRmGRN and JGL identified much more numbers of common edges and less number of condition-specific edges than GLASSO and HGLASSO. HGLASSO identified 37 hub genes, 14 high red:far-red-specific and 23 low red:far-red-specific hub genes. Of these 37 hub genes, TCP11, PD1 PLIM2A, and AT3G45260 are among the 15 hub genes identified by JRmGRN. Among the 15 hub genes identified by JRmGRN, nine are involved in light response and considered to be positive, whereas, among 37 hub genes identified by HGLASSO, eight genes are involved in light response and considered to be positive. Details are provided in Table 4.1 and Table 4.2. It appears that JRmGRN is more efficient in recognizing positive hubs.

Table 4.1 The number of different types of edge identified by 4 methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Common edges</th>
<th>high red:far-red-specific edges</th>
<th>low red:far-red-specific edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>JRmGRN</td>
<td>3702</td>
<td>150</td>
<td>52</td>
</tr>
<tr>
<td>GLASSO</td>
<td>192</td>
<td>1813</td>
<td>2304</td>
</tr>
<tr>
<td>JGL</td>
<td>3495</td>
<td>300</td>
<td>299</td>
</tr>
<tr>
<td>HGLASSO</td>
<td>35</td>
<td>902</td>
<td>2545</td>
</tr>
</tbody>
</table>

The execution of JRmGRN algorithm needs four tuning parameters and a grid search was employed to find the optimal turning parameter set, which was evaluated by BIC-like criteria. For a fixed set of tuning parameters, an efficient ADMM algorithm was derived and implemented to enable a fast estimation of precision matrices. Theoretical properties of the penalized likelihood function were also investigated and used to reduce the search space of tuning parameters. For a fixed set of tuning parameters, using a Mac desktop computer with 2.2 GHz Intel Core i7 processor and 16 GB 1600 MHz DDR3 memory, the average running times for estimating the precision matrices were about 30 seconds for 100 genes, 2.5 minutes for 200 genes, 6 minutes for 300 genes, 25 minutes for 500 genes, and 3.7 hours for 1000 genes, respectively. Therefore, implementation of JRmGRN allows us to reconstruct GRNs for 500 genes within a reasonable time frame by an ordinary desktop computer. In the future, two possible strategies may be explored to reduce the
computational burden so that JRmGRN can be used for a large number of genes. First, instead of using the grid search, the heuristic search algorithms, such as the genetic algorithm [135] and taboo [136, 137], can be investigated. Secondly, the domain knowledge on gene networks and differentially expressed genes can be used to reduce the search space of tuning parameters.

Table 4.2 Comparison of JRmGRN and HGLASSO in identifying hub genes

<table>
<thead>
<tr>
<th>JRmGRN Known gene</th>
<th>JRmGRN Currently unknown gene</th>
<th>HGLASSO Known gene</th>
<th>HGLASSO Currently unknown gene</th>
</tr>
</thead>
<tbody>
<tr>
<td>PD1</td>
<td>AT1G62310</td>
<td>SOB7</td>
<td>ERF11</td>
</tr>
<tr>
<td>ELIP1</td>
<td>PLIM2a</td>
<td>EIP6</td>
<td>ETC2</td>
</tr>
<tr>
<td>PLPC</td>
<td>AT3G45260</td>
<td>ELIP2</td>
<td>WRKY33</td>
</tr>
<tr>
<td>CCL</td>
<td>BZO2H3</td>
<td>TED_5</td>
<td>GATA8</td>
</tr>
<tr>
<td>POP12</td>
<td>NAC102</td>
<td>TCP11</td>
<td>PFG2</td>
</tr>
<tr>
<td>BLH10</td>
<td>AT3G15570</td>
<td></td>
<td>AT5G61590</td>
</tr>
<tr>
<td>WAV2</td>
<td></td>
<td></td>
<td>ERF104</td>
</tr>
<tr>
<td>PEX11B</td>
<td></td>
<td></td>
<td>MYBR1</td>
</tr>
<tr>
<td>TCP11</td>
<td></td>
<td></td>
<td>GA4H</td>
</tr>
</tbody>
</table>

JRmGRN assumed that all hub genes are shared across different tissues or conditions. In many situations, hub genes that are specific to an individual network also exist. One of the future works is to extend the current model to incorporate both common and unique hub genes. This can be done by adding an additional symmetric matrix to the decomposition of the precision matrix. The corresponding penalized log-likelihood function and an efficient algorithm will be developed accordingly.

4.5 Conclusion

JRmGRN was developed as a novel method for joint construction of GRNs using gene expression data from multiple tissues or environmental conditions. The model was based on a convex penalized log-likelihood function that not only took gene network sparsity and similarity into account but also explicitly modeled common hub genes across multiple GRNs, leading to multiple networks with common network moieties being emphasized. The resulting networks can significantly advance our understanding of the genetic regulation of various biological processes. Reconstruction of both common moieties and each individual network corresponding to a tissue or a condition was improved by sharing information of common hub genes and regulatory relationships from other individual networks. Therefore, JRmGRN can potentially generate more accurate gene networks, as manifested by the precision-recall curves.
5 Reference List


A Copyright documentation

- The material contained in Chapter three has been published in PLoS one journal, which is an open access freely available online journal. PLOS applies the Creative Commons Attribution (CC BY) license to articles and other works they publish.


- The material contained in Chapter four has been published in Bioinformatics journal, which applies the New Creative Commons license to articles.