Learning Gene Networks Underlying Clinical Phenotypes Under SNP Perturbations From Genome-Wide Data

Calvin McCarter

May 2019
CMU-ML-19-108

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee

Seyoung Kim, Chair
Pradeep Ravikumar
Kathryn Roeder
Dietrich Stephan (NeuBase Therapeutics, previously The University of Pittsburgh)

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

The author gratefully acknowledges support from NSF CAREER Award No. MCB-1149885, Sloan Research Fellowship, and Okawa Foundation Research Grant.

Keywords: probabilistic graphical models, sparse learning, structure learning, convex optimization, genomics, gene networks, systems biology

When thou shalt do wonderful things,
we shall not bear them:
thou didst come down,
and at thy presence the mountains melted away.

From the beginning of the world they have not heard,
nor perceived with the ears:
the eye hath not seen, O God, besides thee,
what things thou hast prepared for them that wait for thee.

Abstract

Recent technologies are generating an abundance of genome sequence data and molecular and clinical phenotype data, providing an opportunity to understand the genetic architecture and molecular mechanisms underlying diseases. Previous approaches have largely focused on the co-localization of single-nucleotide polymorphisms (SNPs) associated with clinical and expression traits, each identified from genome-wide association studies and expression quantitative trait locus (eQTL) mapping, and thus have provided only limited capabilities for uncovering the molecular mechanisms behind the SNPs influencing clinical phenotypes. Here we aim to extract rich information on the functional role of trait-perturbing SNPs that goes far beyond this simple co-localization. We introduce a computational framework called PerturbNet for learning the gene network that modulates the influence of SNPs on phenotypes, using SNPs as naturally occurring perturbation of a biological system. PerturbNet uses a probabilistic graphical model to directly model both the cascade of perturbation from SNPs to the gene network to the phenotype network and the network at each layer of molecular and clinical phenotypes. PerturbNet learns the entire model by solving a single optimization problem with an extremely fast algorithm that can analyze human genome-wide data within a few hours. In our analysis of asthma data, for a locus that was previously implicated in asthma susceptibility but for which little is known about the molecular mechanism underlying the association, PerturbNet revealed the gene network modules that mediate the influence of the SNP on asthma phenotypes. Many genes in this network module were well supported in the literature as asthma-related.

Acknowledgments

Firstly, credit goes to my advisor Seyoung Kim for her support and guidance. Over the last six years, I've learned a lot from her, not just about how to develop machine learning methods, but also how to make machine learning tools which are accessible and useful to biologists. Through high school and undergrad, I strongly preferred learning from books over learning from people. But it is not possible to learn from a book how to relate a machine learning model to a scientific question, or how to frame a machine learning tool for the scientific research community. So I'm fortunate to have had the irreplaceable opportunity to learn these things by working alongside someone who has been thinking seriously about these challenges.

I also appreciate the helpful feedback and insightful questions from my committee members, Pradeep Ravikumar, Kathryn Roeder, and Dietrich Stephan. There are also the many great professors of my classes at CMU who helped provide me with a better understanding of machine learning and molecular biology. Thanks also to Judie Howrylak for her help with the asthma data I studied.

The staff of MLD has made my time as a PhD student in the department a pleasant one. Diane Stidle has been so helpful; she and the other staff are so excellent that they are one of the reasons why I tell prospective PhD students to choose MLD.

On a personal level, I have no one to thank more than my parents for their care and support, and for their lifelong examples of a consistent work ethic, commitment to excellence, and dedication in the pursuit of knowledge. This inspiration was especially powerful and necessary during those times when I struggled to maintain momentum in research. I am also indebted to my brother for patiently listening to me during the video chats where I let out my frustrations when I felt stuck in life and research. If Simone Weil was correct in saying, "Attention is the rarest and purest form of generosity," then having him to talk to has been the ultimate earthly privilege over the last several years.

My time in Pittsburgh would not have been the same without all the people who made it feel like my home. I would like to particularly thank the following past or current Yinzers (ordered alphabetically by last names) without whom my time would have been inestimably worse: Jonathan, Meng, Ben, Kathryn, Molly, Derek, Julie, Wati, Chris, Jon, Kyle, Laura, Mathi, Dave and Sandy, Jeffery, Son (special thanks for housing me in the last months of my PhD!), Cindy, Alex, Father Paul, Eric and Hannah, and Hongyi. I am grateful also to my friends from MLD who made it an enjoyable experience, especially my academic "big sister" Jing and my officemates Ian, Juyong, Abu, Gus, and Hongyang.

Last but not least, I thank Jesus the First and the Last, though not simply the first or last person on my list of acknowledgments, nor separated from all the other persons whom I am indebted to. In the strange molecular networks of the body, in the statistical structures that makes these decipherable, in my own opportunities to study them, in the melting away of insurmountable obstacles, and in the people who have guided and encouraged me, I see revealed the hidden God who is Love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

One of the key questions in biology is how genetic variation perturbs gene regulatory systems to influence disease susceptibility or other phenotypes in a population. Recent advances in technologies have allowed researchers to obtain genome sequence data along with phenotype data at different levels of biological systems, such as gene expression [41], proteome [4], metabolome [85], and various clinical phenotype data. Combining genome sequence data with various types of molecular and clinical phenotype data in a computational analysis has the potential to reveal the complex molecular mechanisms controlled by different genetic loci that underlie diseases and other phenotypes.

To study gene regulatory systems, many previous works have considered the naturally-occurring perturbation of gene expression by genetic variants such as single nucleotide polymorphisms (SNPs), captured in expression and genotype data collected from a population. Compared to experimental perturbation methods such as gene knockdown [23] and genome editing techniques [83], SNP perturbation for functional genomics studies has an advantage of being more cost effective, being easily applicable to humans, and being potentially more meaningful subtle perturbations because they exist in nature [51]. However, it comes with the computational challenge of having to isolate the perturbation effect of each individual genetic variant, when a large number of genetic variants are perturbing the gene network simultaneously. Several computational methods have been proposed to address this challenge. Sparse conditional Gaussian graphical models (sCGGMs) have been introduced for simultaneously identifying the gene network and expression quantitative trait loci (eQTLs) from population SNP and expression data [31, 87, 104]. Many other works have relied on statistically less powerful approaches of identifying eQTLs first and then incorporating the eQTLs in the network learning procedure [27, 79, 80].

However, there have been relatively few works on modeling how a gene network perturbed by SNPs mediates the SNP perturbation of phenotypes. Most of the existing methods did not directly address this problem and thus, provided only limited capabilities for uncovering the molecular mechanisms behind the SNP perturbation of clinical phenotypes. Many of the previous approaches were concerned with identifying simply the co-localization of eQTLs and trait-associated SNPs [33, 36, 45], each of which were identified in a separate eQTL mapping

[27, 37, 40, 41] and a genome-wide association study [65, 93]. These methods did not provide a description of the regulatory roles of the trait-associated SNPs beyond their co-localization with eQTLs. The genome-transcriptome-phenome structured association method [22] focused only on identifying eQTLs and trait-associated SNPs, and was concerned with neither learning a gene network nor uncovering its role in modulating SNP effects on phenotypes. A predictive network model for diseases that involves Bayesian networks for gene regulatory networks have been proposed [73], but this approach relied on an elaborate pipeline of analysis to identify disease-related gene modules and genetic variants that could potentially lead to loss of statistical power.

Here, our goal is to extract rich information on the functional role of trait-perturbing SNPs that goes far beyond the simple co-localization with eQTLs, which was the focus of many of the previous studies [33, 36, 45]. Towards this goal, we introduce a computational framework called Perturb-Net for directly modeling and learning the gene network that modulates the influence of SNPs on phenotypes, using SNPs as naturally occurring perturbation of a biological system. Perturb-Net builds on the key idea in the previous work on sCGGMs [87, 104] for learning a gene network using SNP perturbations, and models the cascade of a gene network and a phenotype network under SNP perturbations as a cascade of sCGGMs, called a sparse Gaussian chain graph model.Our probabilistic graphical model framework naturally leads to a set of inference algorithms for inferring a detailed description of how different parts of the gene network mediate the influence of SNPs on phenotypes, given the model estimated from population genotype, expression, and phenotype data.The Perturb-Net model and inference procedures together provide a powerful tool for studying the gene regulatory mechanisms whose perturbations by SNPs lead to diseases.

We present a statistically powerful and extremely efficient algorithm for learning the Perturb-Net model. The Perturb-Net learning algorithm is statistically powerful, since it estimates the entire model by solving a single optimization problem with minimal loss of statistical power and with a guarantee in finding the optimal solution due to the convexity of the optimization problem. The Perturb-Net learning algorithm is also computationally efficient and can analyze human genome-wide data with 500,000 SNPs, 11,000 gene expression levels, and several dozens of phenotype data within a few hours. The performance of the Perturb-Net learning algorithm directly depends on that of sCGGM optimization, since it uses the sCGGM learning algorithm as a key module. The previous state-of-the-art method [100] had limited scalability due to expensive computation time and large memory requirement, requiring more than 4 hours for only 10,000 SNPs and running out of memory for 40,000 SNPs. We present a new learning algorithm Fast-sCGGM and its extension Mega-sCGGM with orders-of-magnitude speed-up in computation time that runs on a single machine without running out of memory and that is parallelizable. Our new sCGGM learning algorithms allow Perturb-Net to be applied to human genome-wide data.

We demonstrate Perturb-Net on the data collected for participants in the Childhood Asthma Management Program Continuation Study (CAMP-CS) [20, 21, 67]. Perturb-Net revealed the asthma gene network and how different parts of this gene network mediate the SNP perturbations of phenotypes. Furthermore, for a locus that was previously implicated in asthma susceptibility but for which little has been known about the molecular mechanism underlying the association, Perturb-Net revealed the gene network modules that mediate the influence of the SNP on asthma phenotypes. Many genes in this network module were well supported in the literature as asthma-

related, suggesting our framework can reveal the molecular mechanisms underlying the SNP perturbations of phenotypes.

1.2 Thesis objectives

We focus on four subgoals to provide statistical machine learning methods for genome-wide analysis of genotypic variation influencing clinical traits via gene regulatory networks.

1. **Develop scalable convex optimization methods for learning sCGGMs.** Our new learning algorithm has orders-of-magnitude speedup over previous methods and scales to genome-wide SNP data on machine with limited memory.
2. **Develop the sparse Gaussian chain graph model in order to model the cascaded influence from SNPs to gene expression levels to clinical phenotypes.** We address how to use the learned model to extract information about the role of both individual genes and also gene modules in mediating the effect of genetic variants on phenotypes. We also use our proposed model to enable utilization of samples where genotype and phenotype data are available but gene expressions are missing.
3. **We apply our method to asthma data to show how to extract from the estimated model rich information on the molecular interactions underpinning the association between gene variants and traits.** We show how to perform colocalization analysis on our learned model to uncover the gene modules with eQTLs that colocalize with the trait-associated SNPs. We also show how, using the learned model, one may focus in on the SNPs and gene modules which influence lung function traits and blood measurement traits.

The work described in the previous objectives has been completed. We plan to accomplish the following objectives.

4. **Compare the power and specificity of our model with other statistical models.** We will use both simulated and asthma data to examine whether our method provides better recovery of colocalized SNPs at the same false-positive error rate as other methods. To evaluate the power and specificity for recovering gene networks, we will compare our approach with sparse GGMs. Furthermore, we will examine how the power of our approach compares to univariate tests in detecting SNP-gene and SNP-trait associations.

1.3 Background on sparse modeling

In this section, we first describe approaches to learning the structure of sparse probabilistic graphical models that build on Lasso [90] and Graphical Lasso [30]. Next, we describe the sCGGM [86, 100], as well as previously developed optimization methods for this model.

1.3.1 The Lasso and sparse Gaussian graphical models

Our work on sparse probabilistic graphical model learning is closely related to problems in sparse regression. The Lasso [90] for sparse linear regression is a key building block for many recent

approaches to sparse regression and sparse graphical model learning, including our work. Rather than minimizing the least-squares error subject to a constraint on the number of non-zero parameters, the Lasso constrains the ℓ_1-norm of the parameters, leading to a convex problem with global optimum and efficient solution via a coordinate descent algorithm.

GGMs [30] have been extremely popular as a tool for learning a network structure over a large number of continuous variables in many different application domains including neuroscience [68] and biology [30]. A sparse GGM can be estimated as a sparse inverse covariance matrix by minimizing the convex function of ℓ_1-regularized negative log-likelihood. Non-zero elements in the estimated GGM parameter matrix indicate conditional independence relationships between the two corresponding variables given all the other variables.

Highly scalable learning algorithms such as Graphical Lasso [30], QUIC [47], and BigQUIC [48] have been proposed to learn GGMs. The QUIC algorithm finds in each iteration a generalized Newton descent direction by forming a second-order approximation of the smooth part of the objective and minimizing this along with the ℓ_1 penalty. Finding the Newton descent direction is itself solved iteratively, by solving a series of *Lasso* subproblems. Given this Newton direction, the parameter estimates are updated with a step size found by backtracking line search. BigQUIC extends QUIC to high-dimensional problems where the large dense matrices computed in QUIC do not fit in memory. This algorithm divides the parameters into blocks and passes through these blocks while computing the Newton direction, requiring only smaller portions of these large dense matrices to avoid exceeding computer system memory.

1.3.2 The sparse conditional Gaussian graphical model

Sparse CGGMs have been introduced as a discriminative extension of sparse GGMs to model a sparse network over outputs conditional on input variables [86, 100]. CGGMs can be viewed as a Gaussian analogue of conditional random field [61], while GGMs are a Gaussian analogue of Markov random field. Sparse CGGMs have recently been applied to various settings including energy forecasting [99], finance [103], and biology [104], where the goal is to predict structured outputs influenced by inputs. A sparse CGGM can be estimated by minimizing a convex function of l_1-regularized negative log-likelihood. This optimization problem is closely related to that for sparse GGMs because CGGMs also model the network over outputs. However, the presence of the additional parameters in CGGMs for the functional mapping from inputs to outputs makes the optimization significantly more complex than in sparse GGMs.

A CGGM [86, 100] models the conditional probability density of $\mathbf{x} \in \mathbb{R}^p$ given $\mathbf{y} \in \mathbb{R}^q$ as follows:

$$p(\mathbf{y}|\mathbf{x}; \mathbf{\Lambda}, \mathbf{\Theta}) = \exp\{-\mathbf{y}^T\mathbf{\Lambda}\mathbf{y} - 2\mathbf{x}^T\mathbf{\Theta}\mathbf{y}\}/Z(\mathbf{x}),$$

where $\mathbf{\Lambda}$ is a $q \times q$ matrix for modeling the network over $\mathbf{y}$ and $\mathbf{\Theta}$ is a $p \times q$ matrix for modeling the mapping between the input variables $\mathbf{x}$ and output variables $\mathbf{y}$. The normalization constant is given as $Z(\mathbf{x}) = (2\pi)^{q/2}|\mathbf{\Lambda}|^{-1}\exp(\mathbf{x}^T\mathbf{\Theta}\mathbf{\Lambda}^{-1}\mathbf{\Theta}^T\mathbf{x})$.

Non-zeros in $\mathbf{\Lambda}$ indicate conditional dependencies between the corresponding pairs of output variables, given all inputs and the other output variables. Similarly, non-zeros in $\mathbf{\Theta}$ encode conditional dependencies between the corresponding output variable and input variable, given

all the other input and output variables. Inference in a CGGM gives $p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{B}^T\mathbf{x}, \mathbf{\Lambda}^{-1})$, where $\mathbf{B} = -\mathbf{\Theta}\mathbf{\Lambda}^{-1}$, showing the connection to multivariate linear regression.

A sparse estimate of CGGM parameters can be obtained by minimizing the negative log-likelihood combined with ℓ_1-regularization. As observed in [86, 100, 103], this objective is convex. Several different approaches have been previously proposed to estimate sparse CGGMs, including OWL-QN [86], accelerated proximal gradient method [103], and Newton coordinate descent algorithm [100]. In particular, the Newton coordinate descent algorithm extends the QUIC algorithm [47] for sparse GGM estimation to the case of CGGMs, and has been shown to have superior computational speed and convergence. This approach finds in each iteration a descent direction by minimizing a quadratic approximation of the original negative log-likelihood function along with l_1 regularization. Then, the parameter estimate is updated with this descent direction and a step size found by line search.

The previous state-of-the-art Newton coordinate descent method [100] was not efficient enough to be applied to many real-world problems even with tens of thousands of variables. More importantly, it suffers from a large space requirement, because for very high-dimensional problems, several large dense matrices need to be precomputed and stored during optimization. For a CGGM with p inputs and q outputs, the algorithm requires storing several $p \times p$ and $q \times q$ dense matrices, which cannot fit in memory for large p and q. In this dissertation, we introduce a fast algorithm that substantially improves the scalability both in terms of time and space.

1.4 Background on modeling genomic data

In this section we review existing methods for inferring the genetic basis of clinical traits and expression traits, for learning gene networks from gene expression data, and for combining such information with integrative methods. We also discuss the shortcomings and difficulties with applying such methods in piecemeal fashion, rather than addressing these problems together with a single model.

1.4.1 GWAS and eQTL mapping

A genome-wide association study (GWAS) aims at finding the genome-wide set of genetic variants, typically SNPs, that are statistically associated with a trait in order to establish the genetic causes of phenotypic variation. The dominant paradigm in GWAS analysis is to perform separate univariate statistical tests for all SNP-trait pairs. While GWAS data have greatly facilitated discovery of the genetic basis of complex-trait diseases, such traits are very polygenic, with each causal SNP explaining on average a small proportion of population variation, so that GWAS analyses still often lack the statistical power to detect all the causal variants [92], especially in cases of linkage disequilibrium. To tackle this, combining all SNPs together into a single sparse regression model for each phenotype has been proposed [97], using either Lasso or ℓ_1-penalized logistic regression.

Expression quantitative trait loci (eQTL) analysis investigates the genetic basis of gene expression by treating gene transcript abundances as phenotypes, using computational methods similar to those for GWAS. The goal of such eQTL analyses is to learn the molecular basis of

associations between genetic variation and traits. Because gene expression levels are influenced not only by genotype, but also by tissue and disease state, eQTL mapping studies must be specific to a particular disease state or tissue type, as in the GTEx Project [41]. As in GWAS, most eQTL studies rely on simple univariate testing for each SNP-transcript pair, in part due to the computational demands of analyzing tens of thousands of gene transcripts. Various computational methods have been developed to improve the scalability of univariate testing in eQTL mapping [34, 82].

1.4.2 Statistical learning approaches to gene network learning

The graphical Lasso [30] for learning sparse GGMs was motivated by the problem of gene network estimation from gene expression level data gathered through microarray experiments. Since then, it has gained wide use as a gene regulatory network learning approach [59, 64]. Various extensions of the sparse GGM have also been proposed to utilize prior knowledge about regulatory pathways [39, 84], to jointly learn multiple related networks for different tissues or disease types [24, 66], and to learn time-varying gene networks [43, 95]. However, these approaches do not jointly learn the eQTLs, and are therefore unable to factor out the portion of the observed correlations among gene expressions that arise from shared causal SNPs rather than gene-gene regulatory effects [72].

Sparse multivariate linear regression with sparse inverse covariance estimation (MRCE) [78] has been applied to learn gene networks after adjusting for the effect of genotype covariates [15, 102]. However, because this approach led to a non-convex optimization problem that scales poorly with the number of SNPs, these studies limited their analysis to only a few hundreds of SNPs.

The sparse CGGM has also been proposed as an approach to learning gene networks under SNP perturbations [104]. This method, unlike MRCE, distinguished between marginal SNP-gene relationships, captured by sparse regression coefficients, and conditional SNP-gene dependencies after conditioning on the learned gene network, in order to identify the genes directly perturbed by a SNP versus those indirectly affected through the gene regulatory network. The gene network learned from the sparse CGGM model has been used to infer the correlation matrix among genes after accounting for the gene correlations induced by SNPs [3]. In that study, the sparse CGGM was shown to scale to a few thousand SNPs and a few thousand gene expression levels, but was not yet capable of performing genome-wide analysis of human data. Furthermore, this approach did not allow incorporation of phenotypic data into the learning algorithm.

1.4.3 Three-way integrative genomics analyses

Colocalization methods are a leading general approach to integrating genotype, gene expression, and phenotype data in order to uncover genes which lead to disease traits. In colocalization analyses, GWAS and eQTL mapping are first performed separately to find SNP-gene and SNP-trait associations. Next, these approaches link genes and traits with common patterns of underlying SNP associations. Sherlock [45], COLOC [36], and eCAVIAR [46] are Bayesian methods that compare p-values from GWAS and eQTL mapping hypothesis tests to find SNPs that show up as associated with both gene expression and clinical traits. However, these analyses are limited

by their reliance on univariate tests. Wen et al. [94] generalize COLOC and eCAVIAR with a model that incorporates the effect of multiple SNPs on each trait using a sparse regression model, but like the previous method it does not account for correlations among genes through the gene regulatory network or correlations between gene expression levels and clinical traits.

Another set of integrative techniques is built on imputation of gene expression levels using eQTL mapping study results, such as from the GTEx Project. These two-step procudures [33, 71] predict gene expression levels for individuals from their genotypes, then regress phenotypes on these imputed gene expression values. While these methods reduce the multiple-testing burden from examining genome-wide SNPs, they do not model the gene and trait networks, or the colocalization of SNPs regulating both genes and traits.

Another branch of three-way integrative analyses methods incorporate all variables into a Bayesian network [19, 91]. While Bayesian networks are able to model all the interactions among the genotype, gene expression, and phenotype variables, learning Bayesian networks poses serious statistical and computational difficulties. To address this, the authors of [19, 91] employed a complex analysis pipeline with rules to favor or exclude possible edges in the network model. The complexity of this pipeline makes it challenging to apply to data collected from other diseases. To overcome the complexities of the previous approaches, we will develop a method that models all interactions among genomic data types, automatically learns these interactions from data, and uses these learned interactions to discover the molecular basis of traits, including the role of the gene network and the colocalization between trait SNPs and eQTLs.

Chapter 2

Optimization for Sparse CGGMs

In this chapter, we address the problem of scaling up the optimization of the sCGGM to very large problem sizes without requiring excessive time or memory. We propose sparse CGGM learning algorithms called Fast-sCGGM for reducing computation time and Mega-sCGGM for further improving Fast-sCGGM to remove the memory constraint. Fast-sCGGM improves the computation time of the previous Newton coordinate descent method [100] by alternately optimizing the network parameters and the input-output perturbation parameters. While Fast-sCGGM improves the computation time of the previous method, it is limited by the memory size required to store large $q \times q$ or $p \times q$ matrices during the iterative optimization. Hence, we combine Fast-sCGGM with block coordinate descent and introduce the Mega-sCGGM algorithm to scale up the optimization to very large problems on a machine with limited memory. During the iterative optimization, we update blocks of the large matrices so that within each block, the computation of the large matrices can be cached and re-used.

Our Fast-sCGGM algorithm is based on the key observation that the computation simplifies drastically if we alternately optimize the two sets of parameters for output network and for mapping inputs to outputs, instead of updating all parameters at once as in the previous approach. The previous approach updated all parameters simultaneously by forming a second-order approximation of the objective on all parameters, which requires an expensive computation of the large Hessian matrix of size $(p + q) \times (p + q)$ in each iteration. Our approach of alternate optimization forms a second-order approximation only on the network parameters, which requires the Hessian of size $q \times q$, as the other set of parameters can be updated easily using a simple coordinate descent.

In order to overcome the constraint on the space requirement, we then introduce Mega-sCGGM, an alternating Newton block coordinate descent method that can be applied to problems of unbounded size on a machine with limited memory. A naive approach to reduce the memory footprint would be to recompute portions of these matrices on demand for each coordinate update, which would be very expensive. Hence, we divide the parameters into blocks for block-wise updates such that the results of computation can be reused within each block. Block-wise parameter updates were previously used in BigQUIC [48] for learning a sparse GGM, where the block sparsity pattern of the network parameters was leveraged to overcome the space limitations. We propose an approach for block-wise update of the output network parameters in CGGMs that extends their idea. We then propose a new block-wise update strategy for the parameters for

mapping inputs to outputs. These blocks are determined automatically in each iteration by exploiting the sparse structure within the network parameters and the input-output parameters. In our experiments, we show that we can solve problems with a million inputs and hundreds of thousands of outputs on a single machine.

2.1 Background on Newton Coordinate Descent Optimization

In this section we first describe the Newton coordinate descent algorithm [100], and then describe the difficulty with scaling this method to large problems due to its time and space complexity.

2.1.1 Optimization Method

Given a mean-centered dataset of $\mathbf{X} \in \mathbb{R}^{n \times p}$ and $\mathbf{Y} \in \mathbb{R}^{n \times q}$ for n samples, and their covariance matrices $\mathbf{S_{xx}} = \frac{1}{n}\mathbf{X}^T\mathbf{X}, \mathbf{S_{xy}} = \frac{1}{n}\mathbf{X}^T\mathbf{Y}, \mathbf{S_{yy}} = \frac{1}{n}\mathbf{Y}^T\mathbf{Y}$, a sparse estimate of CGGM parameters can be obtained by minimizing l_1-regularized negative log-likelihood. This can be written as the

$$\min_{\mathbf{\Lambda} \succ 0, \mathbf{\Theta}} f(\mathbf{\Lambda}, \mathbf{\Theta}) = g(\mathbf{\Lambda}, \mathbf{\Theta}) + h(\mathbf{\Lambda}, \mathbf{\Theta}), \tag{2.1}$$

where $g(\mathbf{\Lambda}, \mathbf{\Theta}) = -\log|\mathbf{\Lambda}| + \mathrm{tr}(\mathbf{S_{yy}}\mathbf{\Lambda} + 2\mathbf{S_{xy}}^T\mathbf{\Theta} + \mathbf{\Lambda}^{-1}\mathbf{\Theta}^T\mathbf{S_{xx}}\mathbf{\Theta})$ for the smooth negative log-likelihood and $h(\mathbf{\Lambda}, \mathbf{\Theta}) = \lambda_{\mathbf{\Lambda}}\|\mathbf{\Lambda}\|_1 + \lambda_{\mathbf{\Theta}}\|\mathbf{\Theta}\|_1$ for the non-smooth elementwise l_1 penalty. $\lambda_{\mathbf{\Lambda}}, \lambda_{\mathbf{\Theta}} > 0$ are regularization parameters. As observed in [86, 100, 103], this objective is convex.

The current state-of-the-art method for solving Eq. (2.1) for l_1-regularized CGGM is the Newton coordinate descent algorithm [100] that extends QUIC [47] for l_1-regularized GGM estimation. In each iteration, this algorithm found a generalized Newton descent direction by forming a second-order approximation of the smooth part of the objective and minimizing this along with the l_1 penalty. Given this Newton direction, the parameter estimates were updated with a step size found by line search using Armijo's rule [7].

In each iteration, the Newton coordinate descent algorithm found the Newton direction as follows:

$$\begin{aligned}\mathbf{D}_{\mathbf{\Lambda}}, \mathbf{D}_{\mathbf{\Theta}} = \operatorname*{argmin}_{\Delta_{\mathbf{\Lambda}}, \Delta_{\mathbf{\Theta}}} & \bar{g}_{\mathbf{\Lambda},\mathbf{\Theta}}(\Delta_{\mathbf{\Lambda}}, \Delta_{\mathbf{\Theta}}) \\ & + h(\mathbf{\Lambda} + \Delta_{\mathbf{\Lambda}}, \mathbf{\Theta} + \Delta_{\mathbf{\Theta}}),\end{aligned} \tag{2.2}$$

where $\bar{g}_{\mathbf{\Lambda},\mathbf{\Theta}}$ is the second-order approximation of g given by Taylor expansion:

$$\begin{aligned}\bar{g}_{\mathbf{\Lambda},\mathbf{\Theta}}(\Delta_{\mathbf{\Lambda}}, \Delta_{\mathbf{\Theta}}) = \mathrm{vec}(\nabla g(\mathbf{\Lambda}, \mathbf{\Theta}))^T \mathrm{vec}([\Delta_{\mathbf{\Lambda}}\ \Delta_{\mathbf{\Theta}}]) \\ + \frac{1}{2}\mathrm{vec}([\Delta_{\mathbf{\Lambda}}\ \Delta_{\mathbf{\Theta}}])^T \nabla^2 g(\mathbf{\Lambda}, \mathbf{\Theta})\, \mathrm{vec}([\Delta_{\mathbf{\Lambda}}\ \Delta_{\mathbf{\Theta}}]).\end{aligned}$$

The gradient and Hessian matrices above are given as:

$$\begin{aligned}
\nabla g(\mathbf{\Lambda}, \mathbf{\Theta}) &= [\nabla_{\mathbf{\Lambda}} g(\mathbf{\Lambda}, \mathbf{\Theta}) \qquad \nabla_{\mathbf{\Theta}} g(\mathbf{\Lambda}, \mathbf{\Theta})] \\
&= [\mathbf{S_{yy}} - \mathbf{\Sigma} - \mathbf{\Psi} \quad 2\mathbf{S_{xy}} + 2\mathbf{\Gamma}] \qquad (2.3) \\
\nabla^2 g(\mathbf{\Lambda}, \mathbf{\Theta}) &= \begin{bmatrix} \nabla^2_{\mathbf{\Lambda}} g(\mathbf{\Lambda}, \mathbf{\Theta}) & \nabla_{\mathbf{\Lambda}} \nabla_{\mathbf{\Theta}} g(\mathbf{\Lambda}, \mathbf{\Theta}) \\ \nabla_{\mathbf{\Lambda}} \nabla_{\mathbf{\Theta}} g(\mathbf{\Lambda}, \mathbf{\Theta})^T & \nabla^2_{\mathbf{\Theta}} g(\mathbf{\Lambda}, \mathbf{\Theta}) \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{\Sigma} \otimes (\mathbf{\Sigma} + 2\mathbf{\Psi}) & -2\mathbf{\Sigma} \otimes \mathbf{\Gamma}^T \\ -2\mathbf{\Sigma} \otimes \mathbf{\Gamma} & 2\mathbf{\Sigma} \otimes \mathbf{S_{xx}} \end{bmatrix}, \qquad (2.4)
\end{aligned}$$

where $\mathbf{\Sigma} = \mathbf{\Lambda}^{-1}$, $\mathbf{\Psi} = \mathbf{\Sigma}\mathbf{\Theta}^T\mathbf{S_{xx}}\mathbf{\Theta}\mathbf{\Sigma}$, and $\mathbf{\Gamma} = \mathbf{S_{xx}}\mathbf{\Theta}\mathbf{\Sigma}$. Given the Newton direction in Eq. (2.2), the parameters can be updated as $\mathbf{\Lambda} \leftarrow \mathbf{\Lambda} + \alpha \mathbf{D}_{\mathbf{\Lambda}}$ and $\mathbf{\Theta} \leftarrow \mathbf{\Theta} + \alpha \mathbf{D}_{\mathbf{\Theta}}$, where step size $0 < \alpha \leq 1$ ensures sufficient decrease in Eq. (2.1) and positive definiteness of $\mathbf{\Lambda}$.

The *Lasso* problem [89] in Eq. (2.2) was solved using coordinate descent. Despite the efficiency of coordinate descent for *Lasso*, applying coordinate updates repeatedly to all $q^2 + pq$ variables in $\mathbf{\Lambda}$ and $\mathbf{\Theta}$ is costly. So, the updates were restricted to an active set of variables given as:

$$\begin{aligned}
\mathcal{S}_{\mathbf{\Lambda}} &= \{(\Delta_{\mathbf{\Lambda}})_{ij} : |(\nabla_{\mathbf{\Lambda}} g(\mathbf{\Lambda}, \mathbf{\Theta}))_{ij}| > \lambda_{\mathbf{\Lambda}} \vee \mathbf{\Lambda}_{ij} \neq 0\} \\
\mathcal{S}_{\mathbf{\Theta}} &= \{(\Delta_{\mathbf{\Theta}})_{ij} : |(\nabla_{\mathbf{\Theta}} g(\mathbf{\Lambda}, \mathbf{\Theta}))_{ij}| > \lambda_{\mathbf{\Theta}} \vee \mathbf{\Theta}_{ij} \neq 0\}.
\end{aligned}$$

Because the active set sizes $m_{\mathbf{\Lambda}} = |\mathcal{S}_{\mathbf{\Lambda}}|, m_{\mathbf{\Theta}} = |\mathcal{S}_{\mathbf{\Theta}}|$ approach the number of non-zero entries in the sparse solutions for $\mathbf{\Lambda}^*$ and $\mathbf{\Theta}^*$ over iterations, this strategy yields a substantial speedup.

To further improve the efficiency of coordinate descent, intermediate results were stored for the large matrix products that need to be computed repeatedly. At the beginning of the optimization for Eq. (2.2), $\mathbf{U} := \Delta_{\mathbf{\Lambda}}\mathbf{\Sigma}$ and $\mathbf{V} := \Delta_{\mathbf{\Theta}}\mathbf{\Sigma}$ were computed and stored. Then, after a coordinate descent update to $(\Delta_{\mathbf{\Lambda}})_{ij}$, row i and j of $\mathbf{U}$ were updated. Similarly, after an update to $(\Delta_{\mathbf{\Theta}})_{ij}$, row i of $\mathbf{V}$ was updated.

2.1.2 Computational Complexity and Scalability Problems

Although the Newton coordinate descent method is computationally more efficient than other previous approaches, it does not scale to problems even with tens of thousands of variables. The main computational cost of the algorithm comes from computing the large $(p + q) \times (p + q)$ Hessian matrix in Eq. (2.4) in each application of Eq. (2.2) to find the Newton direction. At the beginning of the optimization in Eq. (2.2), large dense matrices $\mathbf{\Sigma}$, $\mathbf{\Psi}$, and $\mathbf{\Gamma}$, for computing the gradient and Hessian in Eqs. (2.3) and (2.4), are precomputed and reused throughout the coordinate descent iterations. Initializing $\mathbf{\Sigma} = \mathbf{\Lambda}^{-1}$ via Cholesky decomposition costs up to $O(q^3)$ time, although in practice, sparse Cholesky decomposition exploits sparsity to invert $\mathbf{\Lambda}$ in much less than $O(q^3)$. Computing $\mathbf{\Psi} = \frac{1}{n}\mathbf{R}^T\mathbf{R}$, where $\mathbf{R} = \mathbf{X}\mathbf{\Theta}\mathbf{\Sigma}$, requires $O(nm_{\mathbf{\Theta}} + nq^2)$ time, and computing $\mathbf{\Gamma}$ costs $O(npq + nq^2)$. After the initializations, the cost of coordinate descent update per each active variable $(\Delta_{\mathbf{\Lambda}})_{ij}$ and $(\Delta_{\mathbf{\Theta}})_{ij}$ is $O(p + q)$. During the coordinate descent for solving Eq. (2.2), the entire $(p + q) \times (p + q)$ Hessian matrix in Eq. (2.4) needs to be evaluated, whereas for the gradient in Eq. (2.3) only those entries corresponding to the parameters in active sets are evaluated.

A more serious obstacle to scaling up to problems with large p and q is the space required to store dense matrices $\mathbf{\Sigma}$ (size $q \times q$), $\mathbf{\Psi}$ (size $q \times q$), and $\mathbf{\Gamma}$ (size $p \times q$). In our experiments on a machine with 104 Gb RAM, the Newton coordinate descent method exhausted memory when $p + q$ exceeded $80{,}000$.

In the next section, we propose a modification of the Newton coordinate descent algorithm that significantly improves the computation time. Then, we introduce block-wise update strategies to our algorithm to remove the memory constraint.

2.2 Fast-sCGGM for Improving Computation Time

In this section, we introduce our Fast-sCGGM algorithm for learning an l_1-regularized CGGM that significantly reduces computation time compared to the previous method. Instead of performing Newton descent for all parameters $\mathbf{\Lambda}$ and $\mathbf{\Theta}$ simultaneously, our approach alternately updates $\mathbf{\Lambda}$ and $\mathbf{\Theta}$, optimizing Eq. (2.1) over $\mathbf{\Lambda}$ given $\mathbf{\Theta}$ and vice versa until convergence.

Our approach is based on the key observation that with $\mathbf{\Lambda}$ fixed, the problem of solving Eq. (2.1) over $\mathbf{\Theta}$ becomes simply minimizing a quadratic function with l_1 regularization. Thus, it can be solved efficiently using a coordinate descent method, without the need to form a second-order approximation or to perform line search. On the other hand, optimizing Eq. (2.1) for $\mathbf{\Lambda}$ given $\mathbf{\Theta}$ still requires forming a quadratic approximation to find a generalized Newton direction and performing line search to find the step size. However, this computation involves only $q \times q$ Hessian matrix and is significantly simpler than performing the same type of computation on both $\mathbf{\Lambda}$ and $\mathbf{\Theta}$ jointly as in the previous approach.

2.2.1 Coordinate Descent Optimization for $\mathbf{\Lambda}$

Given fixed $\mathbf{\Theta}$, the problem of minimizing the objective in Eq. (2.1) with respect to $\mathbf{\Lambda}$ becomes

$$\operatorname*{argmin}_{\mathbf{\Lambda} \succ 0} g_{\mathbf{\Theta}}(\mathbf{\Lambda}) + \lambda_{\mathbf{\Lambda}} \|\mathbf{\Lambda}\|_1,$$

where $g_{\mathbf{\Theta}}(\mathbf{\Lambda}) = -\log|\mathbf{\Lambda}| + \operatorname{tr}(\mathbf{S_{yy}}\mathbf{\Lambda} + \mathbf{\Lambda}^{-1}\mathbf{\Theta}^T\mathbf{S_{xx}}\mathbf{\Theta})$. In order to solve this, we first find a generalized Newton direction that minimizes the l_1-regularized quadratic approximation of $g_{\mathbf{\Theta}}(\mathbf{\Lambda})$:

$$\mathbf{D}_{\mathbf{\Lambda}} = \operatorname*{argmin}_{\Delta_{\mathbf{\Lambda}}} \bar{g}_{\mathbf{\Lambda},\mathbf{\Theta}}(\Delta_{\mathbf{\Lambda}}) + \lambda_{\mathbf{\Lambda}} \|\mathbf{\Lambda} + \Delta_{\mathbf{\Lambda}}\|_1, \tag{2.5}$$

where $\bar{g}_{\mathbf{\Lambda},\mathbf{\Theta}}(\Delta_{\mathbf{\Lambda}})$ is obtained from a second-order Taylor expansion and is given as

$$\begin{aligned}\bar{g}_{\mathbf{\Lambda},\mathbf{\Theta}}(\Delta_{\mathbf{\Lambda}}) = {} & \operatorname{vec}(\nabla_{\mathbf{\Lambda}} g(\mathbf{\Lambda},\mathbf{\Theta}))^T \operatorname{vec}(\Delta_{\mathbf{\Lambda}}) \\ & + \frac{1}{2} \operatorname{vec}(\Delta_{\mathbf{\Lambda}})^T \nabla^2_{\mathbf{\Lambda}} g(\mathbf{\Lambda},\mathbf{\Theta}) \operatorname{vec}(\Delta_{\mathbf{\Lambda}}).\end{aligned}$$

The $\nabla_{\mathbf{\Lambda}} g(\mathbf{\Lambda},\mathbf{\Theta})$ and $\nabla^2_{\mathbf{\Lambda}} g(\mathbf{\Lambda},\mathbf{\Theta})$ above are components of the gradient and Hessian matrices corresponding to $\mathbf{\Lambda}$ in Eqs. (2.3) and (2.4). We solve the *Lasso* problem in Eq. (2.5) via

input : Inputs $X \in \mathbb{R}^{n \times p}$ and $Y \in \mathbb{R}^{n \times q}$; regularization parameters $\lambda_{\mathbf{\Lambda}}, \lambda_{\mathbf{\Theta}}$
output: Parameters $\mathbf{\Lambda}, \mathbf{\Theta}$
Initialize $\mathbf{\Theta} \leftarrow 0, \mathbf{\Lambda} \leftarrow I_q$
for $t = 0, 1, \ldots$ **do**
  Determine active sets $\mathcal{S}_{\mathbf{\Lambda}}, \mathcal{S}_{\mathbf{\Theta}}$
  Solve via coordinate descent:
    $D_{\mathbf{\Lambda}} = \underset{\Delta_{\mathbf{\Lambda}}}{\operatorname{argmin}} \bar{g}_{\mathbf{\Lambda},\mathbf{\Theta}}(\mathbf{\Lambda} + \Delta_{\mathbf{\Lambda}}, \mathbf{\Theta}) + h(\mathbf{\Lambda} + \Delta_{\mathbf{\Lambda}}, \mathbf{\Theta})$
  Update $\mathbf{\Lambda}^+ = \mathbf{\Lambda} + \alpha D_{\mathbf{\Lambda}}$, where step size α is found
    with line search
  Solve via coordinate descent:
    $\mathbf{\Theta}^+ = \operatorname{argmin}_{\mathbf{\Theta}} \; g_{\mathbf{\Lambda}}(\mathbf{\Theta}) + \lambda_{\mathbf{\Theta}} \|\mathbf{\Theta}\|_1$
end

Algorithm 1: Fast-sCGGM

coordinate descent. Similar to the Newton coordinate descent method, we maintain $\mathbf{U} := \Delta_{\mathbf{\Lambda}} \mathbf{\Sigma}$ to reuse intermediate results of the large matrix-matrix product. Given the Newton direction for $\mathbf{\Lambda}$, we update $\mathbf{\Lambda} \leftarrow \mathbf{\Lambda} + \alpha \Delta_{\mathbf{\Lambda}}$, where α is obtained by line search.

During coordinate descent to solve the *Lasso* problem, each element of $\Delta_{\mathbf{\Lambda}}$ is updated as follows:

$$(\Delta_{\mathbf{\Lambda}})_{ij} \leftarrow (\Delta_{\mathbf{\Lambda}})_{ij} - c_{\mathbf{\Lambda}} + S_{\lambda_{\mathbf{\Lambda}}/a_{\mathbf{\Lambda}}}(c_{\mathbf{\Lambda}} - \frac{b_{\mathbf{\Lambda}}}{a_{\mathbf{\Lambda}}}),$$

where $S_r(w) = \text{sign}(w) \max(|w| - r, 0)$ is the soft-thresholding operator and

$$\begin{aligned} a_{\mathbf{\Lambda}} &= \mathbf{\Sigma}_{ij}^2 + \mathbf{\Sigma}_{ii}\mathbf{\Sigma}_{jj} + \mathbf{\Sigma}_{ii}\mathbf{\Psi}_{jj} + 2\mathbf{\Sigma}_{ij}\mathbf{\Psi}_{ij} + \mathbf{\Sigma}_{jj}\mathbf{\Psi}_{ii} \\ b_{\mathbf{\Lambda}} &= (\mathbf{S}_{\mathbf{yy}})_{ij} - \mathbf{\Sigma}_{ij} - \mathbf{\Psi}_{ij} + (\mathbf{\Sigma}\Delta_{\mathbf{\Lambda}}\mathbf{\Sigma})_{ij} + (\mathbf{\Psi}\Delta_{\mathbf{\Lambda}}\mathbf{\Sigma})_{ij} + (\mathbf{\Psi}\Delta_{\mathbf{\Lambda}}\mathbf{\Sigma})_{ji} \\ c_{\mathbf{\Lambda}} &= \mathbf{\Lambda}_{ij} + (\Delta_{\mathbf{\Lambda}})_{ij}. \end{aligned}$$

Restricting the generalized Newton descent to $\mathbf{\Lambda}$ simplifies the computation significantly for coordinate descent updates, compared to the previous approach [100] that applies it to both $\mathbf{\Lambda}$ and $\mathbf{\Theta}$ jointly. Our updates only involve $\nabla_{\mathbf{\Lambda}} g(\mathbf{\Lambda}, \mathbf{\Theta})$ and $\nabla^2_{\mathbf{\Lambda}} g(\mathbf{\Lambda}, \mathbf{\Theta})$, and no longer involve $\nabla_{\mathbf{\Theta}} g(\mathbf{\Lambda}, \mathbf{\Theta})$ and $\nabla_{\mathbf{\Lambda}} \nabla_{\mathbf{\Theta}} g(\mathbf{\Lambda}, \mathbf{\Theta})$, eliminating the need to compute the large $p \times q$ dense matrix $\mathbf{\Gamma}$ in $O(npq + nq^2)$ time. Our approach also reduces the computational cost for the coordinate descent update of each element of $\Delta_{\mathbf{\Lambda}}$ from $O(p + q)$ to $O(q)$.

2.2.2 Coordinate Descent Optimization for $\mathbf{\Theta}$

With $\mathbf{\Lambda}$ fixed, the optimization problem in Eq. (2.1) with respect to $\mathbf{\Theta}$ becomes

$$\underset{\mathbf{\Theta}}{\operatorname{argmin}} \; g_{\mathbf{\Lambda}}(\mathbf{\Theta}) + \lambda_{\mathbf{\Theta}} \|\mathbf{\Theta}\|_1, \tag{2.6}$$

where $g_{\mathbf{\Lambda}}(\mathbf{\Theta}) = \text{tr}(2\mathbf{S}_{\mathbf{xy}}{}^T\mathbf{\Theta} + \mathbf{\Lambda}^{-1}\mathbf{\Theta}^T\mathbf{S}_{\mathbf{xx}}\mathbf{\Theta})$. Since $g_{\mathbf{\Lambda}}(\mathbf{\Theta})$ is a quadratic function itself, there is no need to form its second-order Taylor expansion or to determine a step size via line search.

Instead, we solve Eq. (2.6) directly with coordinate descent method, storing and maintaining $\mathbf{V} := \mathbf{\Theta\Sigma}$. Our approach reduces the computation time for updating $\mathbf{\Theta}$ compared to the corresponding computation in the previous algorithm [100]. We avoid computing the large $p \times q$ matrix $\mathbf{\Gamma}$, which had dominated overall computation time with $O(npq)$. Our approach also eliminates the need for line search for updating $\mathbf{\Theta}$. Finally, it reduces the cost for each coordinate descent update in $\mathbf{\Theta}$ to $O(p)$, compared to $O(p+q)$ for the corresponding computation for $\Delta_{\mathbf{\Theta}}$ in the previous method.

The coordinate descent updates applied directly to $\mathbf{\Theta}$ take the following form:

$$\mathbf{\Theta}_{ij} \leftarrow \mathbf{\Theta}_{ij} - c_{\mathbf{\Theta}} + S_{\lambda_{\mathbf{\Theta}}/a_{\mathbf{\Theta}}}(c_{\mathbf{\Theta}} - \frac{b_{\mathbf{\Theta}}}{a_{\mathbf{\Theta}}}),$$

where

$$\begin{aligned} a_{\mathbf{\Theta}} &= 2\mathbf{\Sigma}_{jj}(\mathbf{S_{xx}})_{ii} \\ b_{\mathbf{\Theta}} &= 2(\mathbf{S_{xy}})_{ij} + 2(\mathbf{S_{xx}\Theta\Sigma})_{ij} \\ c_{\mathbf{\Theta}} &= \mathbf{\Theta}_{ij}. \end{aligned}$$

Our approach is summarized in Algorithm 1. In practice, we approximately solve Eqs. (2.5) and (2.6) by using a warm-start for $\mathbf{\Lambda}$ and $\mathbf{\Theta}$ with the results of the previous iteration and making a single pass over the active set. This ensures decrease in the objective in Eq. (2.1) and reduces the overall computation time in practice.

2.3 Mega-sCGGM for Removing Memory Requirement

The Fast-sCGGM algorithm in the previous section improves the computation time of the previous state-of-the-art method, but is still limited by the space required to store large matrices during coordinate descent computation. Solving Eq. (2.5) for updating $\mathbf{\Lambda}$ requires precomputing and storing $q \times q$ matrices, $\mathbf{\Sigma}$ and $\mathbf{\Psi}$, whereas solving Eq. (2.6) for updating $\mathbf{\Theta}$ requires $\mathbf{\Sigma}$ and a $p \times p$ matrix $\mathbf{S_{xx}}$. A naive approach to reduce the memory footprint would be to recompute portions of these matrices on demand for each coordinate update, which would be very expensive.

Here, we describe Mega-sCGGM that combines the alternating Newton coordinate descent algorithm in Fast-sCGGM with block coordinate descent to scale up the optimization to very large problems on a machine with limited memory. During coordinate descent optimization, we update blocks of $\mathbf{\Lambda}$ and $\mathbf{\Theta}$ so that within each block, the computation of the large matrices can be cached and re-used, where these blocks are determined automatically by exploiting the sparse stucture. For $\mathbf{\Lambda}$, we extend the block coordinate descent approach in BigQUIC [48] developed for GGMs to take into account the conditioning variables in CGGMs. For $\mathbf{\Theta}$, we describe a new approach for block coordinate descent update. Our algorithm can, in principle, be applied to problems of any size on a machine with limited memory.

2.3.1 Blockwise Optimization for $\mathbf{\Lambda}$

Block Coordinate Descent Method A coordinate-descent update of $(\Delta_{\mathbf{\Lambda}})_{ij}$ requires the ith and jth columns of $\mathbf{\Sigma}$ and $\mathbf{\Psi}$. If these columns are in memory, they can be reused. Otherwise,

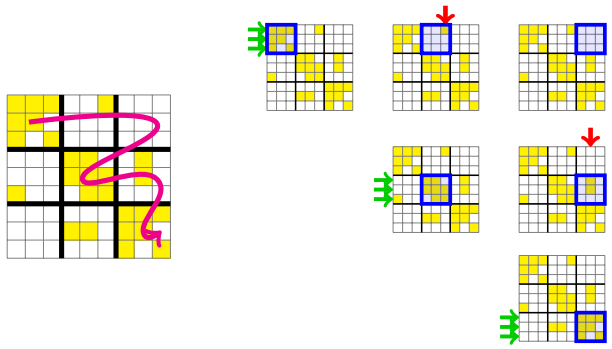


Figure 2.1: Schematic of block coordinate descent for $\mathbf{\Lambda}$ in Mega-sCGGM. The $\mathbf{\Lambda}$ of size $q = 9$ is updated for each of the $k_{\mathbf{\Lambda}}^2$ blocks in turn with $k_{\mathbf{\Lambda}} = 3$. Filled elements denote the parameters in the active set. The green arrows denote rows of $\mathbf{\Sigma}$ and $\mathbf{\Psi}$ that are computed once and reused while sweeping through a row of blocks. The red arrows denote cache misses and the corresponding columns of $\mathbf{\Sigma}$ and $\mathbf{\Psi}$ need to be recomputed.

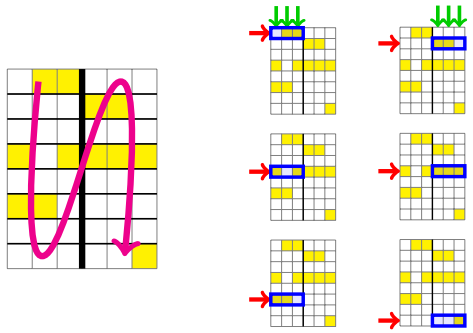


Figure 2.2: Schematic of block coordinate descent for $\mathbf{\Theta}$ in Mega-sCGGM. The $\mathbf{\Theta}$ of size $p = 8, q = 6$ is updated for each of the $p \times k_{\mathbf{\Theta}}$ blocks with $k_{\mathbf{\Theta}} = 2$. Filled elements denote the parameters in the active set. Green arrows denote columns of $\mathbf{\Sigma}$ that are computed once and reused while sweeping through the column of p blocks. The red arrows denote cache misses for $(\mathbf{S_{xx}})_i$.

it is a cache miss and we should compute them on demand. $\mathbf{\Sigma}_i$ for the ith column of $\mathbf{\Sigma}$ can be obtained by solving linear system $\mathbf{\Lambda}\mathbf{\Sigma}_i = \mathbf{e}_i$ with conjugate gradient method in $O(m_{\mathbf{\Lambda}} K)$ time, where K is the number of conjugate gradient iterations. Then, $\mathbf{\Psi}_i$ can be obtained from $\frac{1}{n}\mathbf{R}^T\mathbf{R}_i$ in $O(nq)$ time, where $\mathbf{R} = \mathbf{X}\mathbf{\Theta}\mathbf{\Sigma}$.

In order to reduce cache misses, we perform block coordinate descent, where within each block, the columns of $\mathbf{\Sigma}$ are cached and re-used. Suppose we partition $\mathcal{N} = \{1, \ldots, q\}$ into $k_{\mathbf{\Lambda}}$ blocks, $C_1, \ldots, C_{k_{\mathbf{\Lambda}}}$. We apply this partitioning to the rows and columns of $\Delta_{\mathbf{\Lambda}}$ to obtain $k_{\mathbf{\Lambda}} \times k_{\mathbf{\Lambda}}$ blocks. We perform coordinate-descent updates in each block, updating all elements in the active set within that block. Let $\mathbf{A}_{C_r}$ denote a q by $|C_r|$ matrix containing columns of $\mathbf{A}$ that corresponds to the subset C_r. In order to perform coordinate-descent updates on (C_z, C_r) block of $\Delta_{\mathbf{\Lambda}}$, we need $\mathbf{\Sigma}_{C_z}$, $\mathbf{\Sigma}_{C_r}$, $\mathbf{\Psi}_{C_z}$, and $\mathbf{\Psi}_{C_r}$. Thus, we pick the smallest possible $k_{\mathbf{\Lambda}}$ such that we can store $2q/k_{\mathbf{\Lambda}}$ columns of $\mathbf{\Sigma}$ and $2q/k_{\mathbf{\Lambda}}$ columns of $\mathbf{\Psi}$ in memory. When updating the variables within block (C_z, C_r) of $\Delta_{\mathbf{\Lambda}}$, there are no cache misses once $\mathbf{\Sigma}_{C_z}$, $\mathbf{\Sigma}_{C_r}$, $\mathbf{\Psi}_{C_z}$, and $\mathbf{\Psi}_{C_r}$

are computed and stored. After updating each $(\Delta_{\mathbf{\Lambda}})_{ij}$ to $(\Delta_{\mathbf{\Lambda}})_{ij} + \mu$, we maintain $\mathbf{U}_{C_z}$ and $\mathbf{U}_{C_r}$ by $\mathbf{U}_{it} \leftarrow \mathbf{U}_{it} + \mu\mathbf{\Sigma}_{jt}, \mathbf{U}_{jt} \leftarrow \mathbf{U}_{jt} + \mu\mathbf{\Sigma}_{it}, \forall t \in \{C_z \cup C_r\}$.

To go through all blocks, we update blocks $(C_z, C_1), \ldots, (C_z, C_{k_{\mathbf{\Lambda}}})$ for each $z \in \{1, \ldots, k_{\mathbf{\Lambda}}\}$. Since all of these blocks share $\mathbf{\Sigma}_{C_z}$ and $\mathbf{\Psi}_{C_z}$, we precompute and store them in memory. When updating an off-diagonal block $(C_z, C_r), z \neq r$, we compute $\mathbf{\Sigma}_{C_r}$ and $\mathbf{\Psi}_{C_r}$. In the worst case, overall $\mathbf{\Sigma}$ and $\mathbf{\Psi}$ will be computed $k_{\mathbf{\Lambda}}$ times.

Reducing Computational Cost Using Graph Clustering In typical real-world problems, the graph structure of $\mathbf{\Lambda}$ will exhibit clustering, with an approximately block diagonal structure. We exploit this structure by choosing a partition $\{C_1, \ldots, C_{k_{\mathbf{\Lambda}}}\}$ that reduces cache misses. Within diagonal blocks (C_z, C_z)'s, once $\mathbf{\Sigma}_{C_z}$ and $\mathbf{\Psi}_{C_z}$ are computed, there are no cache misses. For off-diagonal blocks (C_z, C_r)'s, $r \neq z$, we have a cache miss only if some variables in $\{(\Delta_{\mathbf{\Lambda}})_{ij} | i \in C_z, j \in C_r\}$ lie in the active set. We thus minimize the active set in off-diagonal blocks via clustering, following the strategy for sparse GGM estimation in [48]. In the best case, if all parameters in the active set appear in the diagonal blocks, $\mathbf{\Sigma}$ and $\mathbf{\Psi}$ are computed only once with no cache misses. We use the METIS [53] graph clustering library. Our method for updating $\mathbf{\Lambda}$ is illustrated in Figure 2.1.

2.3.2 Blockwise Optimization for $\mathbf{\Theta}$

Block Coordinate Descent Method The coordinate descent update of $\mathbf{\Theta}_{ij}$ requires $(\mathbf{S}_{\mathbf{xx}})_i$ and $\mathbf{\Sigma}_j$ to compute $(\mathbf{S}_{\mathbf{xx}})_i^T \mathbf{V}_j$, where $\mathbf{V}_j = \mathbf{\Theta}\mathbf{\Sigma}_j$. If $(\mathbf{S}_{\mathbf{xx}})_i$ and $\mathbf{\Sigma}_j$ are not already in the memory, it is a cache miss. Computing $(\mathbf{S}_{\mathbf{xx}})_i$ takes $O(np)$, which is expensive if we have many cache misses.

We propose a block coordinate descent approach for solving Eq. (2.6) that groups these computations to reduce cache misses. Given a partition of $\{1, \ldots, q\}$ into $k_{\mathbf{\Theta}}$ subsets, $C_1, \ldots, C_{k_{\mathbf{\Theta}}}$, we divide $\mathbf{\Theta}$ into $p \times k_{\mathbf{\Theta}}$ blocks, where each block comprises a portion of a row of $\mathbf{\Theta}$. We denote each block (i, C_r), where $i \in \{1, \ldots, p\}$. Since updating block (i, C_r) requires $(\mathbf{S}_{\mathbf{xx}})_i$ and $\mathbf{\Sigma}_{C_r}$, we pick the smallest possible $k_{\mathbf{\Theta}}$ such that we can store $q/k_{\mathbf{\Theta}}$ columns of $\mathbf{\Sigma}$ in memory. While performing coordinate descent updates within block (i, C_r) of $\mathbf{\Theta}$, there are no cache misses, once $(\mathbf{S}_{\mathbf{xx}})_i$ and $\mathbf{\Sigma}_{C_r}$ are in memory. After updating each $\mathbf{\Theta}_{ij}$ to $\mathbf{\Theta}_{ij} + \mu$, we update $\mathbf{V}_{C_r}$ by $\mathbf{V}_{it} \leftarrow \mathbf{V}_{it} + \mu\mathbf{\Sigma}_{jt}, \forall t \in C_r$.

In order to sweep through all blocks, each time we select a $q \in \{1, \ldots, k_{\mathbf{\Theta}}\}$ and update blocks $(1, C_r), \ldots, (p, C_r)$. Since all of these p blocks with the same C_r share the computation of $\mathbf{\Sigma}_{C_r}$, we compute and store $\mathbf{\Sigma}_{C_r}$ in memory. Within each block, the computation of $(\mathbf{S}_{\mathbf{xx}})_i$ is shared, so we pre-compute and store it in memory, before updating this block. The full matrix of $\mathbf{\Sigma}$ will be computed once while sweeping through the full $\mathbf{\Theta}$, whereas in the worst case $\mathbf{S}_{\mathbf{xx}}$ will be computed $k_{\mathbf{\Theta}}$ times.

Reducing Computational Cost Using Row-wise Sparsity We further reduce cache misses for $(\mathbf{S}_{\mathbf{xx}})_i$ by strategically selecting partition $C_1, \ldots, C_{k_{\mathbf{\Theta}}}$, based on the observation that if the active set is empty in block (i, C_r), we can skip this block and forgo computing $(\mathbf{S}_{\mathbf{xx}})_i$. We therefore choose a partition where the active set variables are clustered into as few blocks as possible.

Formally, we want to minimize $\sum_{i,r} I_\phi(\mathcal{S}_{(i,C_r)})$, where $I_\phi(\mathcal{S}_{(i,C_r)})$ is an indicator function that outputs 1 if the active set $\mathcal{S}_{(i,C_r)}$ within block (i, C_r) is not empty.We therefore perform graph clustering over the graph $G = (V, E)$ defined from the active set in $\mathbf{\Theta}$, where $V = \{1, \ldots, q\}$ with one node for each column of $\mathbf{\Theta}$, and $E = \{(j, k)|\exists i : \mathbf{\Theta}_{ij} \in \mathcal{S}_{\mathbf{\Theta}}, \mathbf{\Theta}_{ik} \in \mathcal{S}_{\mathbf{\Theta}}\}$, connecting two nodes j and k with an edge if both $\mathbf{\Theta}_{ij}$ and $\mathbf{\Theta}_{ik}$ are in the active set. This edge set corresponds to the non-zero elements of $\mathbf{\Theta}^T\mathbf{\Theta}$, so the graph can be computed quickly in $O(m_{\mathbf{\Theta}}q)$.

We also exploit row-wise sparsity in $\mathbf{\Theta}$ to reduce the cost of each cache miss. Every empty row in $\mathbf{\Theta}$ corresponds to an empty row in $\mathbf{V} = \mathbf{\Theta}\mathbf{\Sigma}$. Because we only need elements in $(\mathbf{S_{xx}})_i$ for the dot product $(\mathbf{S_{xx}})_i^T\mathbf{V}_j$, we skip computing the kth element of $(\mathbf{S_{xx}})_i$ if the kth row of $\mathbf{\Theta}$ is all zeros. Our strategy for updating $\mathbf{\Theta}$ is illustrated in Figure 2.2.

Our method is summarized in Algorithm 2. See Appendix for analysis of the computational cost.

2.3.3 Parallelization

The most expensive computations in our algorithm are embarrassingly parallelizable, allowing for further speedups on machines with multiple cores. Throughout the algorithm, we parallelize matrix and vector multiplications. In addition, for block-wise $\mathbf{\Lambda}$ updates, we compute multiple columns of $\mathbf{\Sigma}_{C_z}$ and $\mathbf{\Psi}_{C_z}$ as well as multiple columns of $\mathbf{\Sigma}_{C_r}$ and $\mathbf{\Psi}_{C_r}$ for multiple cache misses in parallel, running multiple conjugate gradient methods in parallel. For block-wise $\mathbf{\Theta}$ updates, we compute multiple columns of $\mathbf{\Sigma}$ in parallel before sweeping through blocks and perform a parallel computation within each cache miss, computing elements within each $(\mathbf{S_{xx}})_i$ in parallel.

2.3.4 Time Complexity Analysis

In this section we describe the time complexity of the alternating Newton block coordinate descent method. The active set sizes for $\mathbf{\Lambda}$ and $\mathbf{\Theta}$ are $m_{\mathbf{\Lambda}}$ and $m_{\mathbf{\Theta}}$, respectively. Also, K is the number of conjugate gradient iterations.

The time complexity of each $\mathbf{\Lambda}$ update is dominated by the cost of precomputing columns of $\mathbf{\Sigma}$ and $\mathbf{\Psi}$. The cost of these precomputations is $O\Big(\big[1 + \frac{B_{\mathbf{\Lambda}}}{q}\big][m_{\mathbf{\Lambda}}Kq + nq^2]\Big)$ where $B_{\mathbf{\Lambda}} = \sum_{z \neq r} |\{j|i \in C_z, j \in C_r, (i, j) \in \mathcal{S}_{\mathbf{\Lambda}}\}|$ is the number of cache misses. Although the worst-case of $B_{\mathbf{\Lambda}} = k_{\mathbf{\Lambda}}q$ requires computing $\mathbf{\Sigma}$ and $\mathbf{\Psi}$ a total of $k_{\mathbf{\Lambda}}$ times, in practice, graph clustering dramatically reduces this additional cost of block-wise optimization. In the best case, when graph clustering identifies perfect block-diagonal structure in the active set, the number of cache misses $B_{\mathbf{\Lambda}} = 0$ and we incur no runtime penalty from limited memory.

For $\mathbf{\Theta}$, the overall runtime is dominated by the cost of precomputing columns of $\mathbf{S_{xx}}$ and $\mathbf{\Sigma}$. The complexity of these operations is $O(m_{\mathbf{\Lambda}}Kq + m_{\mathbf{\Theta}}q + n\tilde{p}B_{\mathbf{\Theta}})$, where $\tilde{p}$ is the number of non-empty rows in $\mathbf{\Theta}$ and $B_{\mathbf{\Theta}} = \sum_{i,r} |\{i|j \in C_r, (i, j) \in \mathcal{S}_{\Theta}\}|$ is the number of cache misses. Without any row-wise sparsity we have $\tilde{p} = p$ and $B_{\mathbf{\Theta}} = k_{\mathbf{\Theta}}p$, so the worst-case is that $\mathbf{S_{xx}}$ is computed a total of $k_{\mathbf{\Theta}}$ times. This additional cost is substantially reduced in real datasets, where most input variables influence few or none of the outputs. In the best case, if the active set of $\mathbf{\Theta}$ has a block structure, where each group of inputs are influencing only one group of outputs, overall $\mathbf{S_{xx}}$ will be computed only once. We can further save the computation of $\mathbf{S_{xx}}$, if the entire

row of $\boldsymbol{\Theta}$ is not in the active set, by entirely skipping the computation for the corresponding column of $\mathbf{S}_{\mathbf{xx}}$.

input : Inputs $\mathbf{X} \in \mathbb{R}^{n \times p}$ and outputs $\mathbf{Y} \in \mathbb{R}^{n \times q}$; regularization parameters $\lambda_{\mathbf{\Lambda}}, \lambda_{\mathbf{\Theta}}$
output: Parameters $\mathbf{\Lambda}, \mathbf{\Theta}$
Initialize $\mathbf{\Theta} \leftarrow 0, \mathbf{\Lambda} \leftarrow I_q$
for $t = 0, 1, \ldots$ **do**
    Determine active sets $\mathcal{S}_{\mathbf{\Lambda}}, \mathcal{S}_{\mathbf{\Theta}}$
    Partition columns of $\mathbf{\Lambda}$ into $k_{\mathbf{\Lambda}}$ blocks
                                                ▷ Minimize over $\mathbf{\Lambda}$
    Initialize $\Delta_{\mathbf{\Lambda}} \leftarrow 0$
    for $z = 1$ **to** $k_{\mathbf{\Lambda}}$ **do**
        Compute $\mathbf{\Sigma}_{C_z}$, $\mathbf{U}_{C_z}$, and $\mathbf{\Psi}_{C_z}$
        for $r = 1$ **to** $k_{\mathbf{\Lambda}}$ **do**
            if $z \neq r$ **then**
                Identify columns $B_{zr} \subset C_r$ with active elements in $\mathbf{\Lambda}$
                Compute $\mathbf{\Sigma}_{B_{zr}}$, $\mathbf{U}_{B_{zr}}$, and $\mathbf{\Psi}_{B_{zr}}$
            end
            Update all active $(\Delta_{\mathbf{\Lambda}})_{ij}$ in (C_z, C_r)
        end
    end
    Update $\mathbf{\Lambda}^+ \leftarrow \mathbf{\Lambda} + \alpha \Delta_{\mathbf{\Lambda}}$, where step size α is found with line search
    Partition columns of $\mathbf{\Theta}$ into $k_{\mathbf{\Theta}}$ blocks
                                                ▷ Minimize over $\mathbf{\Theta}$
    for $r = 1$ **to** $k_{\mathbf{\Theta}}$ **do**
        Compute $\mathbf{\Sigma}_{C_r}$, and initialize $\mathbf{V} \leftarrow \mathbf{\Theta}\mathbf{\Sigma}_{C_r}$
        for row $i \in \{1, \ldots, p\}$ if $I_\phi(\mathcal{S}_{(i,C_r)})$ **do**
            Compute $(\mathbf{S}_{\mathbf{xx}})_{ij}$ for non-empty rows j in V_{C_r}
            Update all active $\mathbf{\Theta}_{ij}$ in (i, C_r)
        end
    end
end

Algorithm 2: Mega-sCGGM

2.4 Experiments

We compare the performance of our methods with the existing state-of-the-art Newton coordinate descent algorithm, using synthetic and real-world genomic datasets. All methods were implemented in C++ with parameters represented in sparse matrix format. All experiments were run on 2.6GHz Intel Xeon E5 machines with 8 cores and 104 Gb RAM, running Linux OS. We run the Newton coordinate descent and alternating Newton coordinate descent algorithms as a single thread job on a single core. For our alternating Newton block coordinate descent method, we run it on a single core and with parallelization on 8 cores.

2.4.1 Synthetic Data Experiments

We compare the different methods on two sets of synthetic datasets, one for chain graphs and another for random graphs with clustering for $\mathbf{\Lambda}$, generated as follows. For chain graphs, the true sparse parameters $\mathbf{\Lambda}$ is set with $\mathbf{\Lambda}_{i,i-1} = 1$ and $\mathbf{\Lambda}_{i,i} = 2.25$ and the true $\mathbf{\Theta}$ is set with $\mathbf{\Theta}_{i,i} = 1$. We perform one set of chain graph experiments with $p = q$, and another set with an additional q irrelevant features unconnected to any outputs, so that $p = 2q$. For random graphs with clustering, following the procedure in [48] for generating a GGM, we set the true $\mathbf{\Lambda}$ to a graph with clusters of nodes of size 250 and with 90% of edges connecting randomly-selected nodes within clusters. We set the number of edges so that the average degree of each node is 10, with edge weights set to 1. We then set the diagonal values so that $\mathbf{\Lambda}$ is positive definite. To set the sparse patterns for $\mathbf{\Theta}$, we randomly select $100\sqrt{p}$ input variables as having edges to at least one output and distribute total $10q$ edges among those selected inputs to influence randomly selected outputs. We set the edge weights of $\mathbf{\Theta}$ to 1.

Then, we draw samples from the CGGM defined by these true $\mathbf{\Lambda}$ and $\mathbf{\Theta}$. We generate datasets with $n = 100$ samples for the chain graphs and $n = 200$ samples for random graphs with clustering. We choose $\lambda_{\mathbf{\Lambda}}$ and $\lambda_{\mathbf{\Theta}}$ so that the number of edges in the estimated $\mathbf{\Lambda}$ and $\mathbf{\Theta}$ is close to ground truth. Following the strategy used in GGM estimation [48], we use the minimum-norm subgradient of the objective as our stopping criterion: $\|\text{grad}^S(\mathbf{\Lambda}_t, \mathbf{\Theta}_t)\|_1 < 0.01(\|\mathbf{\Lambda}\|_1 + \|\mathbf{\Theta}\|_1)$.

We compare the scalability of the different methods on chain graphs of different sizes. We show the computation times for datasets with $p = q$ in Figure 2.3A and for datasets with $p = 2q$ with q additional irrelevant features in Figure 2.3B. For large problems, computation times are not shown for Newton coordinate descent and alternating Newton coordinate descent methods because they could not complete the optimization with limited memory. In addition, for large problems, alternating block coordinate descent was terminated after 60 hours of computation. We provide results on varying the sample size n in Appendix. In Figure 2.3C, using the dataset with $p = 40{,}000$ and $q = 20{,}000$, we plot the suboptimality in the objective $f - f^*$ over time, where f^* is obtained by running alternating Newton coordinate descent algorithm to numerical precision. Our new methods converge substantially faster than the previous approach, regardless of desired accuracy level. We notice that as expected from the convexity of the optimization problem, all algorithms converge to the global optimum and find nearly identical parameter estimates.

In Figure 2.4, we compare scalability of different methods for random graphs with clustering. In Figure 2.4A, we vary p, while setting q to $10{,}000$. In Figure 2.4B, we vary q, fixing p to $40{,}000$. Similar to the results from chain graph, for larger problems, Newton coordinate descent

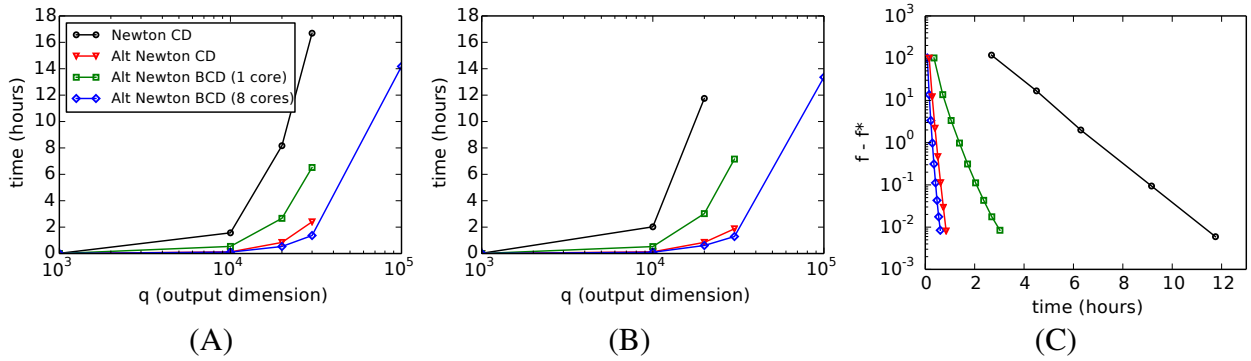


Figure 2.3: Comparison of scalability on chain graphs. We vary p and q, where (A) $p = q$ and (B) $p = 2q$. The Newton coordinate descent and alternating Newton coordinate descent methods could not be run beyond the problem sizes shown due to memory constraint. (C) Convergence when $q = 20{,}000$ and $p = 40{,}000$.

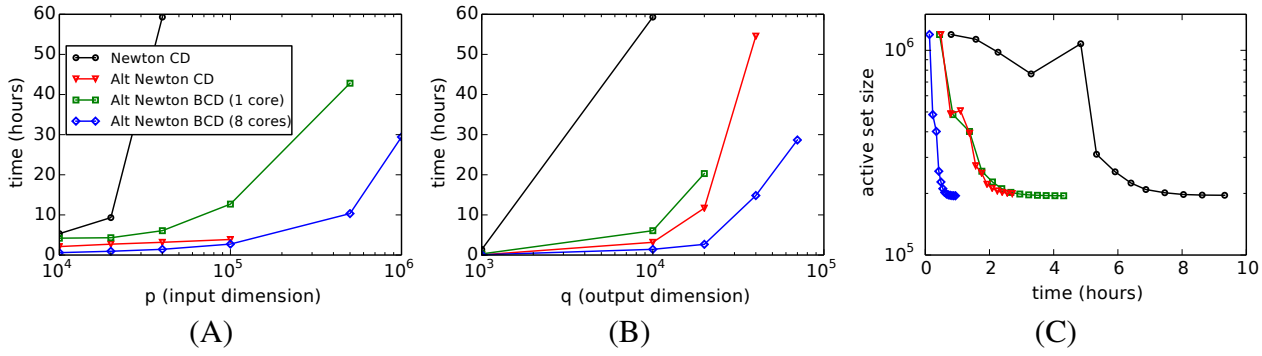


Figure 2.4: Comparison of scalability on random graphs with clustering. (A) Varying p with q fixed at $10{,}000$. (B) Varying q with p fixed at $40{,}000$. (C) Active set size versus time with $p = 20{,}000$ and $q = 10{,}000$.

Table 2.1: Computation time in hours on genomic dataset. '*' indicates running out of memory.

p	q	$\|\mathbf{\Lambda}^*\|_0$	$\|\mathbf{\Theta}^*\|_0$	Newton CD	Alternating Newton CD	Alternating Newton BCD
34,249	3,268	34,914	28,848	22.0	0.51	**0.24**
34,249	10,256	86,090	103,767	> 50	2.4	**2.3**
442,440	3,268	26,232	30,482	*	*	**11**

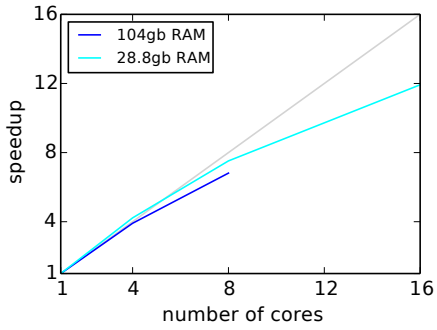


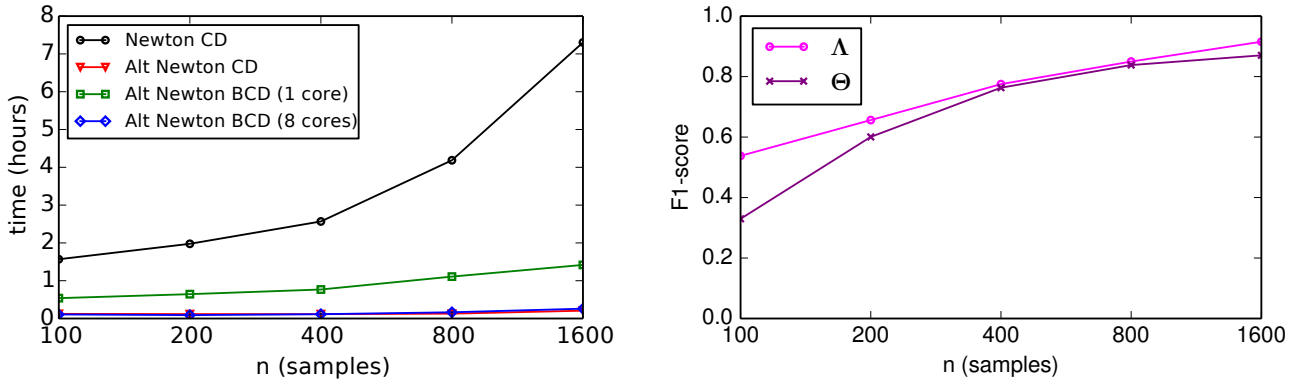
Figure 2.5: Speedup with parallelization for alternating Newton block coordinate descent.

and alternating Newton coordinate descent methods ran out of memory and alternating block coordinate descent was terminated after 60 hours. For all problem sizes, our alternating Newton coordinate descent algorithm significantly reduces the computation time of the previous method, the Newton coordinate descent algorithm. This gap in the computation time increases for larger problems. In Figure 2.4C, we compare the convergence in sparsity pattern for the different methods as measured by the active set size, for $p = 20{,}000$ and $q = 10{,}000$. All our methods recover the optimal sparsity pattern much more rapidly than the previous approach.

Figures 2.3 and 2.4 show that our alternating Newton block coordinate descent can run on much larger problems than any other methods, while those methods without block coordinate descent run out of memory. For example, in Figure 2.4A alternating Newton block coordinate descent could handle problems with one million inputs, while without block-wise optimization it ran out of memory when $p > 100{,}000$. We also notice that on a single core, the alternating Newton block coordinate descent is slighly slower than the same method without block-wise optimization because of the need to recompute $\mathbf{\Sigma}$ and $\mathbf{S_{xx}}$. However, it is still substantially faster than the previous method.

Finally, we evaluate the parallelization scheme for our alternating Newton block coordinate descent method on multi-core machines. Given a dataset generated from cluster graph with $p = 40{,}000$ and $q = 20{,}000$, in Figure 2.5, we show the folds of speedup for different numbers of cores with respect to a single core. We obtained about 7 times speedup on a 8-core machine with 104Gb RAM, and about 12 times speedup on a 16-core machine with 28Gb RAM. In general, we observe greater speedup on larger problems and for random graphs, because such problems tend to have more cache misses that can be computed in parallel.

We additionally analyzed the performance of our approach in scaling with the number of samples. We compare the performance of the different algorithms on synthetic datasets with different sample sizes n, using a chain graph structure with $p = q = 10{,}000$. Figure 2.6A shows that our methods run significantly faster than the previous method across all sample sizes. In



(A) (B)

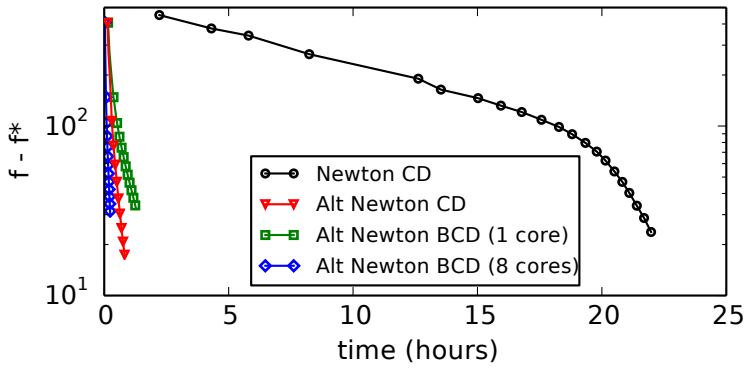
Figure 2.6: Results from varying sample size n on chain graph with $p = q = 10{,}000$. (A) Comparison of computation time of different methods. (B) Comparison of edge recovery accuracy as measured by F_1-score.

Figure 2.6B we measure the accuracy in recovering the true chain graph structure in terms of F_1-score for different sample sizes n. At convergence, F_1-score was the same for all methods to three significant digits. As expected, the accuracy improves as the sample size increases.

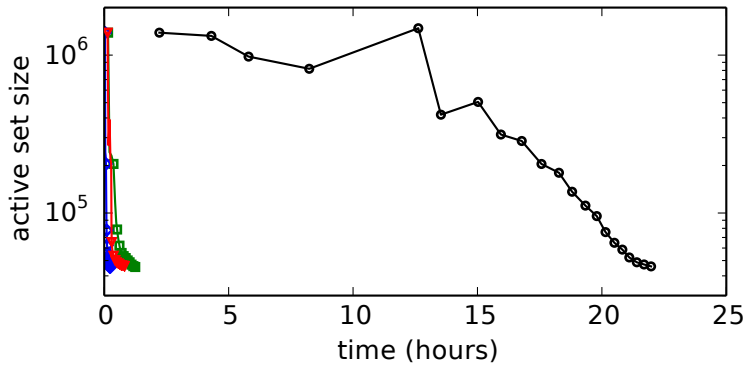
2.4.2 Genomic Data Analysis

We compare the different methods on a genomic dataset. The dataset consists of genotypes for $442{,}440$ single nucleotide polymorphisms (SNPs) and $10{,}256$ gene expression levels for 171 individuals with asthma, after removing genes with variance < 0.01. We fit a sparse CGGM using SNPs as inputs and expressions as outputs to model a gene network influenced by SNPs. We also compared the methods on a smaller dataset of $34{,}249$ SNPs from chromosome 1 and expression levels for $3{,}268$ genes with variance > 0.1. As typically sparse model structures are of interests in this type of analysis, we chose regularization parameters so that the number of non-zero entries in each of $\mathbf{\Theta}$ and $\mathbf{\Lambda}$ at convergence was approximately 10 times the number of genes. The alternating Newton block coordinate descent was run on a 8-core machine with parallelization.

The computation time of different methods are provided in Table 2.1. On the largest problem, the previous approach could not run due to memory constraint, whereas our block coordinate descent converged in around 11 hours. We also compare the convergence of the different methods on the dataset with $34{,}249$ SNPs and $3{,}268$ gene expressions in Figure 2.7, and find that our methods provide vastly superior convergence than the previous method.



(A)



(B)

Figure 2.7: Convergence results on genomic dataset. (A) Suboptimality and (B) active set size over time.

2.5 Conclusions for Optimization for Sparse CGGMs

In this chapter, we addressed the problem of large-scale optimization for sparse CGGMs. We proposed a new optimization procedure, called alternating Newton coordinate descent, that reduces computation time by alternately optimizing for the two sets of parameters $\mathbf{\Lambda}$ and $\mathbf{\Theta}$. Further, we extended this with block-wise optimization so that it can run on any machine with limited memory.

Chapter 3

Sparse Gaussian Chain Graph Models

Probabilistic graphical models have been extensively studied as a powerful tool for modeling a set of conditional independencies in a probability distribution [57]. In this chapter, we are concerned with a class of graphical models, called chain graph models, that has been proposed as a generalization of undirected graphical models and directed acyclic graphical models [13, 32, 62]. Chain graph models are defined over chain graphs that contain a mixed set of directed and undirected edges but no partially directed cycles.

In particular, we study the problem of learning the structure of Gaussian chain graph models in a high-dimensional setting. While the problem of learning sparse structures from high-dimensional data has been studied extensively for other related models such as Gaussian graphical models (GGMs) [29] and more recently conditional Gaussian graphical models (CGGMs) [86, 100], to our knowledge, there is little previous work that addresses this problem for Gaussian chain graph models. Even with a known chain graph structure, current methods for parameter estimation are hindered by the presence of multiple locally optimal solutions [2, 26, 101].

Since the seminal work on conditional random fields (CRFs) [60], a general recipe for constructing chain graph models [57] has been given as using CRFs as building blocks for the model. We employ this construction for Gaussian chain graph models and propose to use the recently-introduced sparse CGGMs [86, 100] as a Gaussian equivalent of general CRFs. When the goal is to learn the model structure, we show that this construction is superior to the popular alternative approach of using linear regression as component models. Some of the key advantages of our approach are due to the fact that the sparse Gaussian chain graph models inherit the desirable properties of sparse CGGM such as convexity of the optimization problem and structured output prediction. In fact, our work is the first to introduce a joint estimation procedure for both the graph structure and parameters as a convex optimization problem, given the groups of variables for chain components. Another advantage of our approach is the ability to model a functional mapping from multiple related variables to other multiple related variables in a more natural way via moralization in chain graphs than other approaches that rely on complex penalty functions for inducing structured sparsity [50, 70].

Our work on sparse Gaussian chain graphs is motivated by problems in integrative genomic data analyses [19, 91]. While sparse GGMs have been extremely popular for learning networks from datasets of single modality such as gene-expression levels [29], we propose that sparse Gaussian chain graph models with CGGM components can be used to learn a cascade of net-

works by integrating multiple types of genomic data in a single statistical analysis. We show that our approach can reveal the module structures as well as the functional mapping between modules in different types of genomic data effectively. Furthermore, as the cost of collecting each data type differs, we show that semi-supervised learning can be used to make effective use of both fully-observed and partially-observed data.

3.1 Sparse Gaussian Chain Graph Models

We consider a chain graph model for a probability distribution over J random variables $\mathbf{x} = \{x_1, \ldots, x_J\}$. The chain graph model assumes that the random variables are partitioned into C chain components $\{\mathbf{x}_1, \ldots, \mathbf{x}_C\}$, the τth component having size $|\tau|$. In addition, it assumes a partially directed graph structure, where edges between variables within each chain component are undirected and edges across two chain components are directed. Given this chain graph structure, the joint probability distribution factorizes as follows:

$$p(\mathbf{x}) = \prod_{\tau=1}^{C} p(\mathbf{x}_\tau | \mathbf{x}_{\text{pa}(\tau)}),$$

where $\mathbf{x}_{\text{pa}(\tau)}$ is the set of variables that are parents of one or more variables in $\mathbf{x}_\tau$. Each factor $p(\mathbf{x}_\tau | \mathbf{x}_{\text{pa}(\tau)})$ models the conditional distribution of the chain component variables $\mathbf{x}_\tau$ given $\mathbf{x}_{\text{pa}(\tau)}$. This model can also be viewed as being constructed with CRFs for $p(\mathbf{x}_\tau | \mathbf{x}_{\text{pa}(\tau)})$'s [60].

The conditional independence properties of undirected and directed graphical models have been extended to chain graph models [32, 62]. This can be easily seen by first constructing a moralized graph, where undirected edges are added between any pairs of nodes in $\mathbf{x}_{\text{pa}(\tau)}$ for each chain component τ and all the directed edges are converted into undirected edges (Figure 3.1). Then, subsets of variables $\mathbf{x}_a$ and $\mathbf{x}_b$ are conditionally independent given $\mathbf{x}_c$, if $\mathbf{x}_a$ and $\mathbf{x}_b$ are separated by $\mathbf{x}_c$ in the moralized graph. This conditional independence criterion for a chain graph is called c-separation and generalizes d-separation for Bayesian networks [57].

In this section, we focus on Gaussian chain graph models, where both $p(\mathbf{x})$ and $p(\mathbf{x}_\tau | \mathbf{x}_{\text{pa}(\tau)})$'s are Gaussian distributed. Below, we review linear regression models and CGGMs as chain component models, and introduce our approach for learning chain graph model structures.

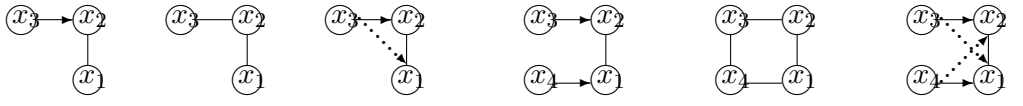


Figure 3.1: Illustration of chain graph models. (A) A chain graph with two components, $\{x_1, x_2\}$ and $\{x_3\}$. (B) The moralized graph of the chain graph in (A). (C) After inference in the chain graph in (A), inferred indirect dependencies are shown as the dotted line. (D) A chain graph with three components, $\{x_1, x_2\}$, $\{x_3\}$, and $\{x_4\}$. (E) The moralized graph of the chain graph in (D). (F) After inference in the chain graph in (D), inferred indirect dependencies are shown as the dotted lines.

3.1.1 Sparse Linear Regression as Chain Component Model

As the specific functional form of $p(\mathbf{x}_\tau|\mathbf{x}_{\text{pa}(\tau)})$ in Gaussian chain graphs models, a linear regression model with multivariate responses has been widely considered [5, 6, 26]:

$$p(\mathbf{x}_\tau|\mathbf{x}_{\text{pa}(\tau)}) = N(\mathbf{B}_\tau\mathbf{x}_{\text{pa}(\tau)}, \mathbf{\Omega}_\tau^{-1}), \tag{3.1}$$

where $\mathbf{B}_\tau \in \mathbb{R}^{|\tau|\times|\text{pa}(\tau)|}$ is the matrix of regression coefficients and $\mathbf{\Omega}_\tau$ is the $|\tau| \times |\tau|$ inverse covariance matrix that models correlated noise. Then, the non-zero elements in $\mathbf{B}_\tau$ indicate the presence of directed edges from $\mathbf{x}_{\text{pa}(\tau)}$ to $\mathbf{x}_\tau$, and the non-zero elements in $\mathbf{\Omega}_\tau$ correspond to the undirected edges among the variables in $\mathbf{x}_\tau$. When the graph structure is known, an iterative procedure has been proposed to estimate the model parameters, but it converges only to one of many locally-optimal solutions [26].

When the chain component model has the form of Eq. (3.1), in order to jointly estimate the sparse graph structure and the parameters, we adopt sparse multivariate regression with covariance estimation (MRCE) [77] for each chain component and solve the following optimization problem:

$$\min \sum_{\tau=1}^{C} \text{tr}((\mathbf{X}_\tau - \mathbf{X}_{\text{pa}(\tau)}\mathbf{B}_\tau^T)\mathbf{\Omega}_\tau(\mathbf{X}_\tau - \mathbf{X}_{\text{pa}(\tau)}\mathbf{B}_\tau^T)^T) - N\log|\mathbf{\Omega}_\tau| + \lambda\sum_{\tau=1}^{C}||\mathbf{B}_\tau||_1 + \gamma\sum_{\tau=1}^{C}||\mathbf{\Omega}_\tau||_1,$$

where $\mathbf{X}_\alpha \in \mathbb{R}^{N\times|\alpha|}$ is a dataset for N samples, $||\cdot||_1$ is the sparsity-inducing L_1 penalty, and λ and γ are the regularization parameters that control the amount of sparsity in the parameters. As in MRCE [77], the problem above is not convex, but only bi-convex.

3.1.2 Sparse CGGM as Chain Component Model

As an alternative model for $p(\mathbf{x}_\tau|\mathbf{x}_{\text{pa}(\tau)})$ in Gaussian chain graph models, a re-parameterization of the linear regression model in Eq. (3.1) with natural parameters has been considered [62]. This model also has been called a CGGM [86] or Gaussian CRF [100] due to its equivalence to a CRF. A CGGM for $p(\mathbf{x}_\tau|\mathbf{x}_{\text{pa}(\tau)})$ takes the standard form of undirected graphical models as a log-linear model:

$$p(\mathbf{x}_\tau|\mathbf{x}_{\text{pa}(\tau)}) = \exp\Big(-\frac{1}{2}\mathbf{x}_\tau^T\mathbf{\Lambda}_\tau\mathbf{x}_\tau - \mathbf{x}_\tau^T\mathbf{\Theta}_{\tau,\text{pa}(\tau)}\mathbf{x}_{\text{pa}(\tau)}\Big)/A(\mathbf{x}_{\text{pa}(\tau)}), \tag{3.2}$$

where $\mathbf{\Lambda}_\tau \in \mathbb{R}^{|\tau|\times|\tau|}$ and $\mathbf{\Theta}_{\tau,\text{pa}(\tau)} \in \mathbf{R}^{|\tau|\times|\text{pa}(\tau)|}$ are the parameters for the feature weights between pairs of variables within $\mathbf{x}_\tau$ and between pairs of variables across $\mathbf{x}_\tau$ and $\mathbf{x}_{\text{pa}(\tau)}$, respectively, and $A(\mathbf{x}_{\text{pa}(\tau)})$ is the normalization constant. The non-zero elements of $\mathbf{\Lambda}_\tau$ and $\mathbf{\Theta}_{\tau,\text{pa}(\tau)}$ indicate edges among the variables in $\mathbf{x}_\tau$ and between $\mathbf{x}_\tau$ and $\mathbf{x}_{\text{pa}(\tau)}$, respectively.

The linear regression model in Eq. (3.1) can be viewed as the result of performing inference in the probabilistic graphical model given by the CGGM in Eq. (3.2). This relationship between the two models can be seen by re-writing Eq. (3.2) in the form of a Gaussian distribution:

$$p(\mathbf{x}_\tau|\mathbf{x}_{\text{pa}(\tau)}) = N(-\mathbf{\Lambda}_\tau^{-1}\mathbf{\Theta}_{\tau,\text{pa}(\tau)}\mathbf{x}_{\text{pa}(\tau)}, \mathbf{\Lambda}_\tau^{-1}), \tag{3.3}$$

where marginalization in a CGGM involves computing $\mathbf{B}_\tau \mathbf{x}_{\text{pa}(\tau)} = -\mathbf{\Lambda}_\tau^{-1}\mathbf{\Theta}_{\tau,\text{pa}(\tau)}\mathbf{x}_{\text{pa}(\tau)}$ to obtain a linear regression model parameterized by $\mathbf{B}_\tau$.

In order to estimate the graph structure and parameters for Gaussian chain graph models with CGGMs as chain component models, we adopt the procedure for learning a sparse CGGM [86, 100] and minimize the negative log-likelihood of data along with sparsity-inducing ℓ_1 penalty:

$$\min -\mathcal{L}(\mathbf{X};\mathbf{\Theta}) + \lambda \sum_{\tau=1}^{C} ||\mathbf{\Theta}_{\tau,\text{pa}(\tau)}||_1 + \gamma \sum_{\tau=1}^{C} ||\mathbf{\Lambda}_\tau||_1,$$

where $\mathbf{\Lambda} = \{\mathbf{\Lambda}_\tau, \tau = 1,\ldots,C\}$, $\mathbf{\Theta} = \{\mathbf{\Theta}_{\tau,\text{pa}(\tau)}, \tau = 1,\ldots,C\}$, and $\mathcal{L}(\mathbf{X};\mathbf{\Lambda},\mathbf{\Theta})$ is the data log-likelihood for dataset $\mathbf{X} \in \mathbb{R}^{N\times J}$ for N samples. Unlike MRCE, the optimization problem for a sparse CGGM is convex, and efficient algorithms have been developed to find the globally-optimal solution with substantially lower computation time than that for MRCE [86, 100].

While maximum likelihood estimation leads to the equivalent parameter estimates for CGGMs and linear regression models via the transformation $\mathbf{B}_\tau = -\mathbf{\Lambda}_\tau^{-1}\mathbf{\Theta}_{\tau,\text{pa}(\tau)}$, imposing a sparsity constraint on each model leads to different estimates for the sparsity pattern of the parameters and the model structure [86]. The graph structure of a sparse CGGM directly encodes the probabilistic dependencies among the variables, whereas the sparsity pattern of $\mathbf{B}_\tau = -\mathbf{\Lambda}_\tau^{-1}\mathbf{\Theta}_{\tau,\text{pa}(\tau)}$ obtained after marginalization can be interpreted as indirect influence of covariates $\mathbf{x}_{\text{pa}(\tau)}$ on responses $\mathbf{x}_\tau$. As illustrated in Figures 3.1C and 3.1F, the CGGM parameters $\mathbf{\Theta}_{\tau,\text{pa}(\tau)}$ (directed edges with solid line) can be interpreted as direct dependencies between pairs of variables across $\mathbf{x}_\tau$ and $\mathbf{x}_{\text{pa}(\tau)}$, whereas $\mathbf{B}_\tau = -\mathbf{\Lambda}_\tau^{-1}\mathbf{\Theta}_{\tau,\text{pa}(\tau)}$ obtained from inference can be viewed as indirect and inferred dependencies (directed edges with dotted line).

We argue that when the goal is to learn the model structure, performing the estimation with CGGMs for chain component models can lead to a more meaningful representation of the underlying structure in data than imposing a sparsity constraint on linear regresssion models. Then the corresponding linear regression model can be inferred via marginalization. This approach also inherits many of the advantages of sparse CGGMs such as convexity of optimization problem.

3.1.3 Markov Properties and Chain Component Models

When a CGGM is used as the component model, the overall chain graph model is known to have Lauritzen-Wermuth-Frydenberg (LWF) Markov properties [32]. The LWF Markov properties also correspond to the standard probabilistic independencies in more general chain graphs constructed by using CRFs as building blocks [57].

Many previous works have noted that LWF Markov properties do not hold for the chain graph models with linear regression models [5, 6]. The alternative Markov properties (AMP) were therefore introduced as the set of probabilistic independencies associated with chain graph models with linear regression component models [5, 6]. It has been shown that the LWF and AMP Markov properties are equivalent only for chain graph structures that do not contain the graph in Figure 3.1A as a subgraph [5, 6]. For example, according to the LWF Markov property, in the chain graph model in Figure 3.1A, $x_1 \perp x_3 | x_2$ as x_1 and x_3 are separated by x_2 in the moralized graph in Figure 3.1B. However, the corresponding AMP Markov property implies a

different probabilistic independence relationship, $x_1 \perp x_3$. In the model in Figure 3.1D, according to the LWF Markov property, we have $x_1 \perp x_3 | \{x_2, x_4\}$, whereas the AMP Markov property gives $x_1 \perp x_3 | x_4$.

We observe that when using sparse CGGMs as chain component models, we estimate a model with the LWF Markov properties and perform marginalization in this model to obtain a model with linear-regression chain components that can be interpreted with the AMP Markov properties.

3.1.4 Sparse Multi-layer Gaussian Chain Graph Models

In this section, we extend the two-layer Gaussian chain graph model from the previous section into a multi-layer model to model data that are naturally organized into multiple layers. First, we describe how sparse Gaussian chain graph models with linearly chained components can be applied to problems in integrative genomics. Next, we compare the modeling assumptions of our approach to those of colocalization approaches in genomics.

Our approach is motivated by problems in integrative genomic data analysis. In order to study the genetic architecture of complex diseases, data are often collected for multiple data types, such as genotypes, gene expressions, and phenotypes for a population of individuals [19, 91]. The primary goal of such studies is to identify the genotype features that influence gene expressions, which in turn influence phenotypes. In such problems, data can be naturally organized into multiple layers, where the influence of features in each layer propagates to the next layer in sequence. In addition, it is well-known that the expressions of genes within the same functional module are correlated and influenced by the common genotype features and that the coordinated expressions of gene modules affect multiple related phenotypes jointly. These underlying structures in the genomic data can be potentially revealed by inference and moralization in sparse Gaussian chain graph models with CGGM components.

Models with linearly chained components

Given variables, $\mathbf{x} = \{x_1, \ldots, x_J\}$, $\mathbf{y} = \{y_1, \ldots, y_K\}$, and $\mathbf{z} = \{z_1, \ldots, z_L\}$, at each of the three layers, we set up a three-layer Gaussian chain graph model as follows:

$$\begin{aligned} p(\mathbf{z}, \mathbf{y}|\mathbf{x}) &= p(\mathbf{z}|\mathbf{y})p(\mathbf{y}|\mathbf{x}) \\ &= \left(\exp(-\frac{1}{2}\mathbf{z}^T\boldsymbol{\Lambda}_{\mathbf{zz}}\mathbf{z} - \mathbf{y}^T\boldsymbol{\Theta}_{\mathbf{yz}}\mathbf{z})/C_2(\mathbf{y})\right)\left(\exp(-\frac{1}{2}\mathbf{y}^T\boldsymbol{\Lambda}_{\mathbf{yy}}\mathbf{y} - \mathbf{x}^T\boldsymbol{\Theta}_{\mathbf{xy}}\mathbf{y})/C_1(\mathbf{x})\right), \end{aligned} \tag{3.4}$$

where $C_1(\mathbf{x})$ and $C_2(\mathbf{y})$ are the normalization constants. In our application, $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ correspond to genotypes, gene-expression levels, and phenotypes, respectively. As the focus of such studies lies on discovering how the genotypic variability influences gene expressions and phenotypes rather than the structure in genotype features, we do not model $p(\mathbf{x})$ directly.

Given the estimated sparse model for Eq. (3.4), structured sparsity pattern can be recovered via inference and moralization. Computing $\mathbf{B}_{\mathbf{xy}} = -\boldsymbol{\Lambda}_{\mathbf{yy}}^{-1}\boldsymbol{\Theta}_{\mathbf{xy}}^T$ and $\mathbf{B}_{\mathbf{yz}} = -\boldsymbol{\Lambda}_{\mathbf{zz}}^{-1}\boldsymbol{\Theta}_{\mathbf{yz}}^T$ corresponds to performing inference to reveal how multiple related y_k's in $\boldsymbol{\Lambda}_{\mathbf{yy}}$ (or z_l's in $\boldsymbol{\Lambda}_{\mathbf{zz}}$) are

jointly influenced by a common set of relevant x_j's (or y_k's). On the other hand, the effects of moralization can be seen from the joint distribution $p(\mathbf{z}, \mathbf{y}|\mathbf{x})$ derived from Eq. (3.4):

$$p(\mathbf{z}, \mathbf{y}|\mathbf{x}) = N(-\mathbf{\Lambda}^{-1}_{(\mathbf{zz},\mathbf{yy})}\mathbf{\Theta}^{T}_{(\mathbf{yz},\mathbf{xy})}\mathbf{x}, \mathbf{\Lambda}^{-1}_{(\mathbf{zz},\mathbf{yy})}),$$

where $\mathbf{\Theta}_{(\mathbf{yz},\mathbf{xy})} = (\mathbf{0}_{J\times L}, \mathbf{\Theta}_{\mathbf{xy}})$ and $\mathbf{\Lambda}_{(\mathbf{zz},\mathbf{yy})} = \begin{pmatrix} \mathbf{\Lambda}_{\mathbf{zz}} & \mathbf{\Theta}^{T}_{\mathbf{yz}} \\ \mathbf{\Theta}_{\mathbf{yz}} & \mathbf{\Lambda}_{\mathbf{yy}} + \mathbf{\Theta}_{\mathbf{yz}}\mathbf{\Lambda}^{-1}_{\mathbf{zz}}\mathbf{\Theta}^{T}_{\mathbf{yz}} \end{pmatrix}$. $\mathbf{\Lambda}_{(\mathbf{zz},\mathbf{yy})}$ corresponds to the undirected graphical model over $\mathbf{z}$ and $\mathbf{y}$ conditional on $\mathbf{x}$ after moralization.

Comparison of our modeling assumptions with those of colocalization methods

Here we compare the independence assumptions made by our model with those of previous colocalization-based approaches for integrating eQTL and GWAS analyses [36, 46, 94]. Both these previous colocalization techniques and our model focus on settings with three types of observed variables: genotypes $\mathbf{x}$, gene expression levels $\mathbf{y}$, and traits $\mathbf{z}$.

Without any independence assumptions, any model conditioned on genotypes $\mathbf{x}$ can be written as follows:

$$p(\mathbf{y}, \mathbf{z}|\mathbf{x}) = p(\mathbf{z}|\mathbf{y}, \mathbf{x})p(\mathbf{y}|\mathbf{x}). \tag{3.5}$$

The Naive Bayes assumption for the first term, $p(\mathbf{z}|\mathbf{y}, \mathbf{x}) = p(\mathbf{z}|\mathbf{y})p(\mathbf{z}|\mathbf{x})$, results in the following 3-layer model:

$$p(\mathbf{y}, \mathbf{z}|\mathbf{x}) = p(\mathbf{z}|\mathbf{y})p(\mathbf{y}|\mathbf{x})p(\mathbf{z}|\mathbf{x}) \tag{3.6}$$

Both previous colocalization approaches and our linear chain models can be written as special cases of of the model in Eq 3.6, with further independence assumptions. The conditional independence assumption $\mathbf{z} \perp \mathbf{y}|\mathbf{x}$, that phenotypes and gene expressions are independent given genotype, combined with Eq. 3.6, provides the model used by colocalization methods:

$$p(\mathbf{y}, \mathbf{z}|\mathbf{x}) = p(\mathbf{z}|\mathbf{x})p(\mathbf{y}|\mathbf{x}). \tag{3.7}$$

Colocalization methods learn $p(\mathbf{z}|\mathbf{x})$ from GWAS-type analyses and $p(\mathbf{y}|\mathbf{x})$ from eQTL-type analyses, then look for SNPs in $\mathbf{x}$ that show up in both $p(\mathbf{z}|\mathbf{x})$ and $p(\mathbf{y}|\mathbf{x})$.

In our approach, we instead assume $\mathbf{z} \perp \mathbf{x}|\mathbf{y}$, that phenotypes and genotype are independent given gene expressions. Combined with Eq. 3.6, this gives

$$p(\mathbf{y}, \mathbf{z}|\mathbf{x}) = p(\mathbf{z}|\mathbf{y})p(\mathbf{y}|\mathbf{x}), \tag{3.8}$$

with each factor modeled as a sparse CGGM. To find colocalized SNPs, we then look for SNPs in $\mathbf{x}$ that show up in $p(\mathbf{y}, \mathbf{z}|\mathbf{x})$.

A three-layer model for allele-specific expression

We present a three-layer Gaussian chain graph model for allele-specific expression. In this setting, we model allele-specific gene expression levels $\mathbf{a}$ given genotypes $\mathbf{g}$, and overall gene

expression levels $\mathbf{e}$ given $\mathbf{a}$. The distribution of allele-specific expressions $\mathbf{a}$ given genotypes $\mathbf{g}$ is given by an sCGGM with additional constraints on the parameters, to model known symmetries among allele-specific expressions and to distinguish between the effects of *cis* and *trans* eQTLs. The distribution of overall expressions given allele-specific expressions is given by an sCGGM at the limit of zero-noise, because the overall expression of a particular gene is simply given deterministically by the sum of its two allele's expression levels. Let us assume we have J SNPs and K genes, for which we observe expression levels for both alleles. We represent the allele-specific expressions so that all paternal allele expressions are together in $\mathbf{a}^{(P)} \in \mathbb{R}^K$, and all maternal allele data are together in $\mathbf{a}^{(M)} \in \mathbb{R}^K$. Furthermore, the genotype data consists of paternal haplotypes $\mathbf{g}^{(P)} \in \mathbb{R}^J$, maternal haplotypes $\mathbf{g}^{(M)} \in \mathbb{R}^J$, and overall genotypes $\mathbf{g}^* = \mathbf{g}^{(P)} + \mathbf{g}^{(M)} \in \mathbb{R}^J$. Combining these allele specific expression and genotype information, we have

$$\mathbf{a} = \begin{bmatrix} \mathbf{a}^{(P)} \\ \mathbf{a}^{(M)} \end{bmatrix} \in \mathbb{R}^{2K}, \qquad \mathbf{g} = \begin{bmatrix} \mathbf{g}^{(P)} \\ \mathbf{g}^* \\ \mathbf{g}^{(M)} \end{bmatrix} \in \mathbb{R}^{3p}. \tag{3.9}$$

We model these three data types as a sparse Gaussian chain graph model, with factorization $p(\mathbf{e}, \mathbf{a}|\mathbf{g}) = p(\mathbf{a}|\mathbf{g})p(\mathbf{e}|\mathbf{a})$. The distribution for $p(\mathbf{a}|\mathbf{g})$ is given as an sCGGM:

$$p(\mathbf{a}|\mathbf{g}) = p(\begin{bmatrix} \mathbf{a}^{(P)} \\ \mathbf{a}^{(M)} \end{bmatrix} |\mathbf{g}) \propto \exp\left(-\begin{bmatrix} \mathbf{a}^{(P)} \\ \mathbf{a}^{(M)} \end{bmatrix}^T \mathbf{\Lambda} \begin{bmatrix} \mathbf{a}^{(P)} \\ \mathbf{a}^{(M)} \end{bmatrix} - 2\mathbf{g}^T \mathbf{\Theta} \begin{bmatrix} \mathbf{a}^{(P)} \\ \mathbf{a}^{(M)} \end{bmatrix} \right) / Z(\mathbf{g}), \tag{3.10}$$

with parameters

$$\mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}^{(P)} & \mathbf{\Lambda}^{(P,M)} \\ \mathbf{\Lambda}^{(M,P)} & \mathbf{\Lambda}^{(M)} \end{bmatrix} \in \mathbb{R}^{2q \times 2q}, \qquad \mathbf{\Theta} = \begin{bmatrix} \mathbf{\Theta}^{(P,P)} & \mathbf{\Theta}^{(P,M)} \\ \mathbf{\Theta}^{(*,P)} & \mathbf{\Theta}^{(*,M)} \\ \mathbf{\Theta}^{(M,P)} & \mathbf{\Theta}^{(M,M)} \end{bmatrix} \in \mathbb{R}^{3p \times 2q}.$$

Because $\mathbf{\Lambda}$ is a positive definite matrix, we know that $\forall 1 \leq i, j \leq K$:

$$\mathbf{\Lambda}^{(P)} = \mathbf{\Lambda}^{(P)^T} \implies \mathbf{\Lambda}^{(P)}{}_{ij} = \mathbf{\Lambda}^{(P)^T}_{ji}, \tag{3.11}$$

$$\mathbf{\Lambda}^{(M)} = \mathbf{\Lambda}^{(M)^T} \implies \mathbf{\Lambda}^{(M)}{}_{ij} = \mathbf{\Lambda}^{(M)^T}_{ji}, \text{ and} \tag{3.12}$$

$$\mathbf{\Lambda}^{(P,M)} = \mathbf{\Lambda}^{(M,P)^T} \implies \mathbf{\Lambda}^{(P,M)}{}_{ij} = \mathbf{\Lambda}^{(M,P)^T}_{ji}. \tag{3.13}$$

We place additional constraints on $\mathbf{\Lambda}$ and $\mathbf{\Theta}$ to take advantage of known structure in this particular setting. We constrain the four edges among allelic expressions for all genes $i \neq j$ to be of equal weight:

$$\mathbf{\Lambda}^{(P)}{}_{ij} = \mathbf{\Lambda}^{(P,M)}{}_{ij} = \mathbf{\Lambda}^{(M,P)}{}_{ji} = \mathbf{\Lambda}^{(M)}{}_{ij}, \quad \forall i \neq j. \tag{3.14}$$

This means that $\mathbf{\Lambda}^{(P)} = \mathbf{\Lambda}^{(M)}$ and $\mathbf{\Lambda}^{(P,M)} = \mathbf{\Lambda}^{(M,P)}$. Furthermore, we assume that the gene node potentials for $\mathbf{a}^{(P)}{}_i$ and $\mathbf{a}^{(M)}{}_i$ are equal, so that $\mathbf{\Lambda}^{(P)}{}_{ii} = \mathbf{\Lambda}^{(M)}{}_{ii}$, with coefficients given in

$\delta \in \mathbb{R}^q$. We also have edges between ${\mathbf{a}^{(P)}}_i$ and ${\mathbf{a}^{(M)}}_i$ for all i, given in $\eta \in \mathbb{R}^q$. This implies the following structure for $\mathbf{\Lambda}$:

$$\mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}^{(P)} & \mathbf{\Lambda}^{(P,M)} \\ \mathbf{\Lambda}^{(M,P)} & \mathbf{\Lambda}^{(M)} \end{bmatrix} = \begin{bmatrix} \mathbf{\Omega} + \operatorname{diag}(\boldsymbol{\delta})) & \mathbf{\Omega} + \operatorname{diag}(\boldsymbol{\eta}) \\ \mathbf{\Omega} + \operatorname{diag}(\boldsymbol{\eta}) & \mathbf{\Omega} + \operatorname{diag}(\boldsymbol{\delta}) \end{bmatrix}, \tag{3.15}$$

where $\mathbf{\Omega}$ has zeros on the diagonal.

For $\mathbf{\Theta}$, we enforce symmetry between ${\mathbf{a}^{(P)}}_i$ and ${\mathbf{a}^{(M)}}_i$ for allele-specific edges from SNP k,

$${\mathbf{\Theta}^{(P)}}_{ki} = {\mathbf{\Theta}^{(M)}}_{ki}, \tag{3.16}$$

and remove all edges from $\mathbf{g}^{(P)}$ to $\mathbf{a}^{(M)}$ and from $\mathbf{g}^{(M)}$ to $\mathbf{a}^{(P)}$. Furthermore, we constrain equal weights for the edges from $\mathbf{g}^*$ to $\mathbf{a}^{(P)}$ and $\mathbf{a}^{(M)}$, respectively, corresponding to non-allele-specific effects. This implies the following structure for $\mathbf{\Theta}$:

$$\mathbf{\Theta} = \begin{bmatrix} \mathbf{\Theta}^{(P,P)} & \mathbf{\Theta}^{(P,M)} \\ \mathbf{\Theta}^{(*,P)} & \mathbf{\Theta}^{(*,M)} \\ \mathbf{\Theta}^{(M,P)} & \mathbf{\Theta}^{(M,M)} \end{bmatrix} = \begin{bmatrix} \mathbf{\Pi}^{(P)} & 0 \\ \mathbf{\Xi} & \mathbf{\Xi} \\ 0 & \mathbf{\Pi}^{(M)}. \end{bmatrix} \tag{3.17}$$

The generative model for overall expressions is given by the sum of both allelic expressions:

$$\mathbf{e} = \mathbf{a}^{(P)} + \mathbf{a}^{(M)} \in \mathbb{R}^K. \tag{3.18}$$

This can be written as an sCGGM with zero-noise:

$$p\Big(\mathbf{e}|\begin{bmatrix}\mathbf{a}^{(P)} \\ \mathbf{a}^{(M)}\end{bmatrix}\Big) \sim \mathcal{N}\Big(\mathbf{C}^T\mathbf{a}, \sigma^2 I_q\Big) = \mathcal{N}\Big(\begin{bmatrix} I_q & I_q \end{bmatrix}\begin{bmatrix}\mathbf{a}^{(P)} \\ \mathbf{a}^{(M)}\end{bmatrix}, \sigma^2 I_q\Big), \text{ where}$$
$$\mathbf{C} = \begin{bmatrix} I_q \\ I_q \end{bmatrix} \in \mathbb{R}^{2q \times q}, \text{ and } \sigma = 0.$$

3.2 Learning Sparse Gaussian Chain Graph Models

3.2.1 Learning Structured Sparsity

Another advantage of using CGGMs as chain component models instead of linear regression is that the moralized graph, which is used to define the LWF Markov properties, can be leveraged to discover the underlying structure in a correlated functional mapping from multiple inputs to multiple outputs. In this section, we show that a sparse two-layer Gaussian chain graph model with CGGM components can be used to learn structured sparsity. The key idea behind our approach is that while inference in CGGMs within the chain graph model can reveal the shared sparsity patterns for multiple related outputs, a moralization of the chain graph can reveal those for multiple inputs.

Statistical methods for learning models with structured sparsity were extensively studied in the literature of multi-task learning, where the goal is to find input features that influence multiple related outputs simultaneously [18, 50, 70]. Most of the previous works assumed the output

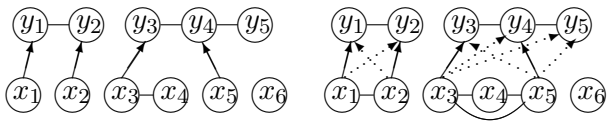


Figure 3.2: Illustration of sparse two-layer Gaussian chain graphs with CGGMs. (A) A two-layer Gaussian chain graph. (B) The results of performing inference and moralization in (A). The dotted edges correspond to indirect dependencies inferred by inference. The edges among x_j's represent the dependencies introduced by moralization.

structure to be known *a priori*. Then, they constructed complex penalty functions that leverage this known output structure, in order to induce structured sparsity pattern in the estimated parameters in linear regression models. In contrast, a sparse CGGM was proposed as an approach for performing a joint estimation of the output structure and structured sparsity for multi-task learning. As was discussed in Section 3.1.2, once the CGGM structure is estimated, the inputs relevant for multiple related outputs could be revealed via probabilistic inference in the graphical model.

While sparse CGGMs focused on leveraging the output structure for improved predictions, another aspect of learning structured sparsity is to consider the input structure to discover multiple related inputs jointly influencing an output. As CGGM is a discriminative model that does not model the input distribution, it is unable to capture input relatedness directly, although discriminative models in general are known to improve prediction accuracy. We address this limitation of CGGMs by embedding CGGMs within a chain graph and examining the moralized graph.

We set up a two-layer Gaussian chain graph model for inputs $\mathbf{x}$ and outputs $\mathbf{y}$ as follows:

$$p(\mathbf{y},\mathbf{x}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) = \left(\exp(-\frac{1}{2}\mathbf{y}^T\mathbf{\Lambda_{yy}}\mathbf{y} - \mathbf{x}^T\mathbf{\Theta_{xy}}\mathbf{y})/A_1(\mathbf{x})\right)\left(\exp(-\frac{1}{2}\mathbf{x}^T\mathbf{\Lambda_{xx}}\mathbf{x})/A_2\right),$$

where a CGGM is used for $p(\mathbf{y}|\mathbf{x})$ and a GGM for $p(\mathbf{x})$, and $A_1(\mathbf{x})$ and A_2 are normalization constants. As the full model factorizes into two factors $p(\mathbf{y}|\mathbf{x})$ and $p(\mathbf{x})$ with distinct sets of parameters, a sparse graph structure and parameters can be learned by using the optimization methods for sparse CGGM [100] and sparse GGM [29, 49].

The estimated Gaussian chain graph model leads to a GGM over both the inputs and outputs, which reveals the structure of the moralized graph:

$$p(\mathbf{y},\mathbf{x}) = N\left(\mathbf{0}, \begin{pmatrix} \mathbf{\Lambda_{yy}} & \mathbf{\Theta_{xy}}^T \\ \mathbf{\Theta_{xy}} & \mathbf{\Lambda_{xx}} + \mathbf{\Theta_{xy}}\mathbf{\Lambda_{yy}}^{-1}\mathbf{\Theta_{xy}}^T \end{pmatrix}^{-1}\right).$$

In the above GGM, we notice that the graph structure over inputs $\mathbf{x}$ consists of two components, one for $\mathbf{\Lambda_{xx}}$ describing the conditional dependencies within the input variables and another for $\mathbf{\Theta_{xy}}\mathbf{\Lambda_{yy}}^{-1}\mathbf{\Theta_{xy}}^T$ that reflects the results of moralization in the chain graph. If the graph $\mathbf{\Lambda_{yy}}$ contains connected components, the operation $\mathbf{\Theta_{xy}}\mathbf{\Lambda_{yy}}^{-1}\mathbf{\Theta_{xy}}^T$ for moralization induces edges among those inputs influencing the outputs in each connected component.

Our approach is illustrated in Figure 3.2. Given the model in Figure 3.2A, Figure 3.2B illustrates the inferred structured sparsity for a functional mapping from multiple inputs to multiple outputs. In Figure 3.2B, the dotted edges correspond to inferred indirect dependencies introduced

via marginalization in the CGGM $p(\mathbf{y}|\mathbf{x})$, which reveals how each input is influencing multiple related outputs. On the other hand, the additional edges among x_j's have been introduced by moralization $\mathbf{\Theta_{xy}\Lambda_{yy}^{-1}\Theta_{xy}^T}$ for multiple inputs jointly influencing each output. Combining the results of marginalization and moralization, the two connected components in Figure 3.2B represent the functional mapping from $\{x_1, x_2\}$ to $\{y_1, y_2\}$ and from $\{x_3, x_4, x_5\}$ to $\{y_3, y_4, y_5\}$, respectively.

3.2.2 Semi-supervised Learning

Here we explore the use of semi-supervised learning, where the top and bottom layer data are fully observed but the middle-layer data are collected only for a subset of samples. In our application, genotype data and phenotype data are relatively easy to collect from patients' blood samples and from observations. However, gene-expression data collection is more challenging, as invasive procedure such as surgery or biopsy is required to obtain tissue samples. We introduce a modification of our learning algorithm for semi-supervised learning, to handle the situation where expression data are available only for a subset of individuals because of the difficulty of obtaining tissue samples. This modification corresponds to an expectation maximization (EM) algorithm [25] that imputes the missing expression levels in the E-step and performs our Fast-sCGGM or Mega-sCGGM optimization in the M-step.

EM algorithm for semi-supervised learning

Given a dataset $\mathcal{D} = \{\mathcal{D}_o, \mathcal{D}_h\}$, where $\mathcal{D}_o = \{\mathbf{X}_o, \mathbf{Y}_o, \mathbf{Z}_o\}$ for the fully-observed data and $\mathcal{D}_h = \{\mathbf{X}_h, \mathbf{Z}_h\}$ for the samples with missing gene-expression levels, the data log-likelihood based on the three-layer model can be written as:

$$\mathcal{L}(\mathcal{D}; \mathbf{\Theta}) = \mathcal{L}_o(\mathcal{D}_o; \mathbf{\Theta}) + \mathcal{L}_h(\mathcal{D}_h; \mathbf{\Theta}),$$

where $\mathcal{L}_o(\mathcal{D}_o; \mathbf{\Theta})$ is the data log-likelihood with respect to the model in Eq. (3.4) and $\mathcal{L}_h(\mathcal{D}_h; \mathbf{\Theta})$ is the data log-likelihood based on the probability distribution after marginalizing out $\mathbf{y}$ from Eq. (3.4), given as:

$$p(\mathbf{z}|\mathbf{x}) = N(\mu_{\mathbf{z}|\mathbf{x}}, \mathbf{\Sigma_{z|x}}), \text{ where}$$
$$\mu_{\mathbf{z}|\mathbf{x}} = \mathbf{\Lambda_{zz}^{-1}\Theta_{yz}^T\Lambda_{yy}^{-1}\Theta_{xy}^T x} \quad \text{and} \quad \mathbf{\Sigma_{z|x}} = \mathbf{\Lambda_{zz}^{-1}} + \mathbf{\Lambda_{zz}^{-1}\Theta_{yz}^T\Lambda_{yy}^{-1}\Theta_{yz}\Lambda_{zz}^{-1}}.$$

Since the marginalization over $\mathbf{y}$ within $\mathcal{L}_h(\mathcal{D}_h; \mathbf{\Theta})$ leads to the coupling of many parameters, making the optimization non-trivial, we adopt an EM algorithm that iteratively maximizes the expected log-likelihood of complete data:

$$\mathcal{L}_o(\mathcal{D}_o; \mathbf{\Theta}) + \mathbb{E}\big[\mathcal{L}_o(\mathcal{D}_h, \mathbf{Y}_h; \mathbf{\Theta})\big], \tag{3.19}$$

combined with L_1-regularization.

Given the parameter estimate $\mathbf{\Theta}^{(t-1)}$ from the previous iteration $(t-1)$, the expectation in the above equation is taken with respect to $p(\mathbf{Y}_h|\mathbf{X}_h, \mathbf{Z}_h, \mathbf{\Theta}^{(t-1)})$, where

$$p(\mathbf{y}|\mathbf{z}, \mathbf{x}) = N(\mu_{\mathbf{y}|\mathbf{x},\mathbf{z}}, \mathbf{\Sigma_{y|x,z}}),$$
$$\mu_{\mathbf{y}|\mathbf{x},\mathbf{z}} = -\mathbf{\Sigma_{y|x,z}}(\mathbf{\Theta_{yz}z} + \mathbf{\Theta_{xy}^T x}) \quad \text{and} \quad \mathbf{\Sigma_{y|x,z}} = (\mathbf{\Lambda_{yy}} + \mathbf{\Theta_{yz}\Lambda_{zz}^{-1}\Theta_{yz}^T})^{-1}.$$

The expectation step computes the expected sufficient statistics to form Eq. (3.19) and the maximization step finds the parameter estimates that maximize Eq. (3.19).

Thus, during the M-step of the tth iteration, we maximize

$$\log|\mathbf{\Lambda_{yy}}| + \log|\mathbf{\Lambda_{zz}}| - \Big[\operatorname{tr}\widetilde{\mathbf{S_{yy}}}\mathbf{\Lambda_{yy}} + 2\widetilde{\mathbf{S_{xy}}}\mathbf{\Theta_{xy}} + \mathbf{\Lambda}_{\mathbf{yy}}^{-1}\mathbf{\Theta}_{\mathbf{xy}}^{T}\widehat{\mathbf{S_{xx}}}\mathbf{\Theta_{xy}} + \operatorname{tr}\widetilde{\mathbf{S_{zz}}}\mathbf{\Lambda_{zz}} + 2\widetilde{\mathbf{S_{yz}}}\mathbf{\Theta_{yz}} + \mathbf{\Lambda}_{\mathbf{zz}}^{-1}\mathbf{\Theta}_{\mathbf{yz}}^{T}\widehat{\mathbf{S_{yy}}}\mathbf{\Theta_{yz}}\Big],$$

using expected sufficient statistics computed from the parameters at step $t-1$:

$$\begin{aligned}\widetilde{\mathbf{S_{xy}}} &= \frac{1}{N_o+N_h}\Big(\mathbf{X}_o^T\mathbf{Y}_o + \mathbf{X}_p^T\boldsymbol{E}_1\Big),\\ \widetilde{\mathbf{S_{yy}}} &= \frac{1}{N_o+N_h}\Big(\mathbf{Y}_o^T\mathbf{Y}_o + \boldsymbol{E}_2\Big),\\ \widetilde{\mathbf{S_{yz}}} &= \frac{1}{N_o+N_h}\Big(\mathbf{Y}_o^T\mathbf{Z}_o + \boldsymbol{E}_1^T\mathbf{Z}\Big),\end{aligned}$$

where

$$\boldsymbol{E}_1 = \boldsymbol{E}\Big[\mathbf{Y}_h\Big] = -(\mathbf{X}_h\mathbf{\Theta_{xy}} + \mathbf{Z}_h\mathbf{\Theta}_{\mathbf{yz}}^T)\mathbf{\Sigma_{y|x,z}}, \quad \text{and} \quad \boldsymbol{E}_2 = \boldsymbol{E}\Big[\mathbf{Y}_h^T\mathbf{Y}_h\Big] = \boldsymbol{E}_1^T\boldsymbol{E}_1 + N_h\mathbf{\Sigma_{y|x,z}}.$$

The time complexity of each E-step is

$$O\Big(K^3 + L^3 + JK^2 + KL^2 + N_o(JK + K^2 + KL)\Big).$$

The first two terms come from matrix inversion, while the third and fourth terms come from matrix-matrix multiplications. The term with coefficient N_o comes from summing over the expectation for each of the samples. Of course this final cost would be incurred once, even without missing data and EM, though not repeated at every EM iteration.

Memory-efficient implementation of E-step

For problem sizes where we are constrained by limited memory, the M-step is carried out by using our Mega-sCGGM algorithm. However, in the E-step, a naive inversion of $\mathbf{\Lambda_{yy}} + \mathbf{\Theta_{yz}}\mathbf{\Lambda_{zz}}^{-1}\mathbf{\Theta}_{\mathbf{yz}}^T$ to obtain $\mathbf{\Sigma_{y|x,z}}$ is expensive and storage of this dense matrix may exceed computer memory for large gene expression datasets. We reduce the time cost and avoid memory limit in the E-step, assuming that the number of phenotypes r is relatively small compared to the number of genes (i.e., $r << q$), which is typical for most studies. Instead of explicitly performing the E-step, we embed the E-step within the M-step, such that the E-step results are represented implicitly to fit in memory and computed explicitly on-demand as needed in the M-step. Specifically, instead of performing the full E-step, we implicitly represent $\mathbf{\Lambda_{yy}} + \mathbf{\Theta_{yz}}\mathbf{\Lambda_{zz}}^{-1}\mathbf{\Theta}_{\mathbf{yz}}^T$ as $\mathbf{\Lambda_{yy}} + \mathbf{K}\mathbf{K}^T$, using low-rank component $\mathbf{K} = \mathbf{\Theta_{yz}}\mathbf{L}_{\mathbf{z}}^T$ and the sparse Cholesky factorization of trait network $\mathbf{L_z}\mathbf{L}_{\mathbf{z}}^T = \mathbf{\Lambda_{zz}}$. Then, during M-step, we invert $\mathbf{\Lambda_{yy}} + \mathbf{K}\mathbf{K}^T$, one column at a time as needed, using the conjugate gradient method. This modified EM algorithm is equivalent to the original EM algorithm that iterates between an M-step and an E-step, producing the same estimate.

EM algorithm with a deterministic chain component for allele-specific expression

In circumstances where, for certain samples, only overall gene expressions and not allele-specific expressions are available, we propose using semi-supervised with EM. Combining Eqs 3.10 and 3.18, gives the following joint distribution:

$$\begin{bmatrix}\mathbf{e}\\ \mathbf{a}\end{bmatrix}|\mathbf{g} \sim \mathcal{N}\Big(\begin{bmatrix}\mathbf{C}^T\boldsymbol{\mu}_{\mathbf{a}|\mathbf{g}}\\ \boldsymbol{\mu}_{\mathbf{a}|\mathbf{g}}\end{bmatrix}, \begin{bmatrix}\sigma^2 I_q + \mathbf{C}^T\boldsymbol{\Lambda}^{-1}\mathbf{C} & \mathbf{C}^T\boldsymbol{\Lambda}^{-1}\\ \boldsymbol{\Lambda}^{-1}\mathbf{C} & \boldsymbol{\Lambda}^{-1}\end{bmatrix}\Big) \text{ where } \boldsymbol{\mu}_{\mathbf{a}|\mathbf{g}} = -\boldsymbol{\Lambda}^{-1}\boldsymbol{\Theta}\mathbf{g}.$$

Then, the conditional distribution $p(\mathbf{a}|\mathbf{e},\mathbf{g})$ is as follows:

$$\mathbf{a}|\mathbf{e},\mathbf{g} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{a}|\mathbf{e},\mathbf{g}}, \boldsymbol{\Sigma}_{\mathbf{a}|\mathbf{e},\mathbf{g}}) \tag{3.20}$$

$$\text{where } \boldsymbol{\Sigma}_{\mathbf{a}|\mathbf{e},\mathbf{g}} = \boldsymbol{\Lambda}^{-1} - \boldsymbol{\Lambda}^{-1}\mathbf{C}\big(\sigma^2 I_q + \mathbf{C}^T\boldsymbol{\Lambda}^{-1}\mathbf{C}\big)^{-1}\mathbf{C}^T\boldsymbol{\Lambda}^{-1}\Big), \tag{3.21}$$

$$\text{and } \boldsymbol{\mu}_{\mathbf{a}|\mathbf{e},\mathbf{g}} = \boldsymbol{\mu}_{\mathbf{a}|\mathbf{g}} + \boldsymbol{\Lambda}^{-1}\mathbf{C}\big(\sigma^2 I_q + \mathbf{C}^T\boldsymbol{\Lambda}^{-1}\mathbf{C}\big)^{-1}(\mathbf{e} - \mathbf{C}^T\boldsymbol{\mu}_{\mathbf{a}|\mathbf{g}}).$$

Because $\sigma = 0$, the mean simplifies as follows:

$$\boldsymbol{\mu}_{\mathbf{a}|\mathbf{e},\mathbf{g}} = \frac{1}{2}\begin{bmatrix}\boldsymbol{\mu}_{\mathbf{a}^{(P)}|\mathbf{g}} + \mathbf{e} - \boldsymbol{\mu}_{\mathbf{a}^{(M)}|\mathbf{g}}\\ \boldsymbol{\mu}_{\mathbf{a}^{(M)}|\mathbf{g}} + \mathbf{e} - \boldsymbol{\mu}_{\mathbf{a}^{(P)}|\mathbf{g}}\end{bmatrix}.$$

Because $\sigma = 0$, the covariance simplifies as follows:

$$\boldsymbol{\Sigma}_{\mathbf{a}|\mathbf{e},\mathbf{g}} = \frac{1}{2}\begin{bmatrix}\boldsymbol{\Sigma}^{(P)} - \boldsymbol{\Sigma}^{(P,M)} & \boldsymbol{\Sigma}^{(P,M)} - \boldsymbol{\Sigma}^{(P)}\\ \boldsymbol{\Sigma}^{(P,M)} - \boldsymbol{\Sigma}^{(P)} & \boldsymbol{\Sigma}^{(P)} - \boldsymbol{\Sigma}^{(P,M)}\end{bmatrix}$$

3.3 Inference in Sparse Gaussian Chain Graph Models

While the sparse Gaussian chain graph model explicitly represents pair-wise dependencies among variables as edges in the graph, there are other dependencies that are only implicitly represented in the model but can be revealed by performing an inference on the estimated probabilistic graphical model. Here, we provide an overview of the inferred dependencies, all of which involve simple matrix operations.

3.3.1 Dependencies from inference procedures for sparse CGGMs

The following two inference methods directly follow from the inference method for a sparse CGGM (Figure 3.3A) [87, 104], which infers the indirect perturbation effects that arise from the direct perturbation effects propagating to other parts of the network.

- **Indirect SNP perturbation effects on gene expression levels:** $\mathbf{B}_{\mathbf{xy}} = -\boldsymbol{\Theta}_{\mathbf{xy}}{\boldsymbol{\Lambda}_{\mathbf{yy}}}^{-1}$, where $[\mathbf{B}_{\mathbf{xy}}]_{i,j}$ represents the indirect perturbation effect of SNP i on the expression level of gene j (blue dashed arrow in Figure 3.3A). This can be seen by deriving the marginal distribution from the sparse CGGM component model $p(\mathbf{y}|\mathbf{x})$ as follows:

$$p(\mathbf{y}|\mathbf{x}) = N(\mathbf{B}_{\mathbf{xy}}^T\mathbf{x}, {\boldsymbol{\Lambda}_{\mathbf{yy}}}^{-1}). \tag{3.22}$$

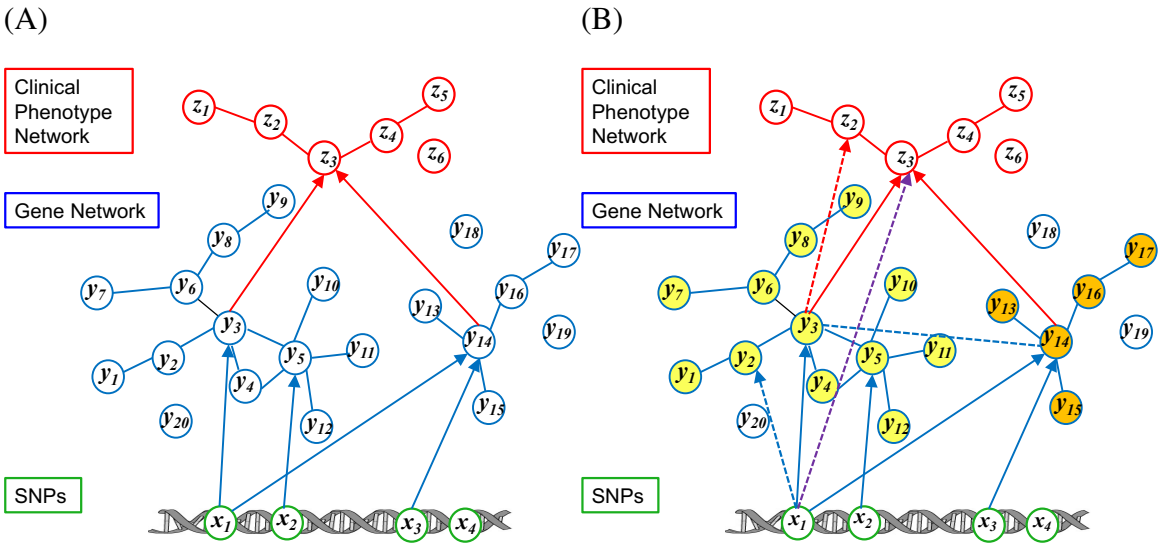


Figure 3.3: Illustration of the PerturbNet approach. (A) PerturbNet uses a sparse Gaussian chain graph model with a cascade of two sparse CGGMs, one for a gene network influenced by SNPs (blue solid edges and nodes) and the other for a clinical trait network influenced by gene expression levels (red solid edges and nodes). The sparse CGGM inference procedures can be used to infer hidden interactions in each of the two component sparse CGGMs, such as the indirect effect of SNP x_1 on expression level y_2 through expression level y_3 (blue dashed arrow) and the indirect effect of expression level y_3 on phenotype z_2 through phenotype z_3 (red dashed arrow). (B) The inference procedures of sparse Gaussian chain graph models are used to infer the information on how the gene network mediates SNP effects on phenotypes. Examples of such inferred interactions are shown for the perturbation effect of SNP x_1 on phenotype z_3 (purple dashed arrow), which can be decomposed into two components mediated by each of the two gene modules (yellow and orange nodes), and the inferred dependencies between expression level y_3 and expression level y_{14} (blue dashed line) induced by phenotype z_3 in the posterior gene network, after seeing the clinical phenotypes.

From Eq. (3.22), the marginal distribution for the expression level $[\mathbf{y}]_i$ of gene i can be obtained as $p([\mathbf{y}]_i|\mathbf{x}) = N\big(\big[[\mathbf{B_{xy}}]_{:,i}\big]^T\mathbf{x}, [{\mathbf{\Lambda_{yy}}}^{-1}]_{i,i}\big)$. While $[\mathbf{\Theta_{xy}}]_{i,j}$ represents the direct perturbation effect of SNP i on the expression of gene j, $[\mathbf{B_{xy}}]_{i,j}$ represents the overall perturbation effect that aggregates all indirect influence of this SNP on gene j through other genes. When SNP i does not influence the expression of gene j directly but exerts influence on gene j through other genes connected to gene j in the network $\mathbf{\Lambda_{yy}}$, we have $[\mathbf{\Theta_{xy}}]_{i,j} = 0$ but $[\mathbf{B_{xy}}]_{i,j} \neq 0$.

- **Indirect effects of gene expression levels on clinical phenotypes:** $\mathbf{B_{yz}} = -\mathbf{\Theta_{yz}}{\mathbf{\Lambda_{zz}}}^{-1}$, where $[\mathbf{B_{yz}}]_{i,j}$ represents the indirect influence of the expression level of gene i on phenotype j (red dashed arrow in Figure 3.3A). Similarly as above, this can be seen by deriving the marginal distribution from the sparse CGGM component model $p(\mathbf{z}|\mathbf{y})$, as follows:

$$p(\mathbf{z}|\mathbf{y}) = N(\mathbf{B}_{\mathbf{yz}}^T\mathbf{y}, {\mathbf{\Lambda_{zz}}}^{-1}).$$

  Then, the marginal distribution for $[\mathbf{z}]_i$ of phenotype i can be obtained as $p([\mathbf{z}]_i|\mathbf{y}) = N\big(\big[[\mathbf{B_{yz}}]_{:,i}\big]^T\mathbf{y}, [{\mathbf{\Lambda_{zz}}}^{-1}]_{i,i}\big)$. While $[\mathbf{\Theta_{yz}}]_{i,j}$ represents the direct influence of gene i on phenotype j, $[\mathbf{B_{yz}}]_{i,j}$ represents the overall influence that aggregates all indirect influence of this expression level on phenotype j through other phenotypes.

3.3.2 Dependencies from inference procedures for sparse Gaussian chain graph models

The sparse Gaussian chain graph model provides the following additional inference procedures for extracting the information on whether SNP perturbation effects on the gene network reach the phenotypes and how different genes or subnetworks of the gene network mediate SNP effects on phenotypes (Figure 3.3B).

- **SNP effects on clinical phenotypes:** $\mathbf{B_{xz}} = \mathbf{B_{xy}}\mathbf{B_{yz}}$, where $[\mathbf{B_{xz}}]_{i,j}$ represents the overall influence of SNP i on phenotype j mediated by gene expression levels in gene network $\mathbf{\Lambda_{yy}}$ (purple dashed arrow in Figure 3.3B). The effects of SNPs on phenotypes are not directly modeled in our model but can be inferred by deriving the marginal distribution $p(\mathbf{z}|\mathbf{x})$ as follows:

$$p(\mathbf{z}|\mathbf{x}) = N(\mathbf{B}_{\mathbf{xz}}^T\mathbf{x}, {\mathbf{\Lambda_{zz}}}^{-1} + {\mathbf{\Lambda_{zz}}}^{-1}\mathbf{\Theta}_{\mathbf{yz}}^T{\mathbf{\Lambda_{yy}}}^{-1}\mathbf{\Theta_{yz}}{\mathbf{\Lambda_{zz}}}^{-1}).$$

  The marginal distribution for the phenotype $[\mathbf{z}]_i$ of phenotype i given $\mathbf{x}$ can be obtained as $p([\mathbf{z}]_i|\mathbf{x}) = N\big(\big[[\mathbf{B_{xz}}]_{:,i}\big]^T\mathbf{x}, [{\mathbf{\Lambda_{zz}}}^{-1} + {\mathbf{\Lambda_{zz}}}^{-1}\mathbf{\Theta}_{\mathbf{yz}}^T{\mathbf{\Lambda_{yy}}}^{-1}\mathbf{\Theta_{yz}}{\mathbf{\Lambda_{zz}}}^{-1}]_{i,i}\big)$, where each element $[\mathbf{B_{xz}}]_{i,j}$ represents the overall influence of SNP i on phenotype j mediated by the gene network in $\mathbf{\Lambda_{yy}}$ and other phenotypes connected to phenotype j in $\mathbf{\Lambda_{zz}}$.

- **SNP effects on clinical phenotypes mediated by a gene module:** The overall SNP effects on phenotypes in $\mathbf{B_{xz}}$ above can be decomposed into the SNP effects on phenotypes mediated by each gene module. Let M be a gene module that consists of a subset of the q genes whose expression levels were modeled in $\mathbf{\Lambda_{yy}}$ (yellow and orange gene modules in Figure 3.3B). Then, the effects of SNPs on phenotypes mediated by the genes in module

M can be obtained as follows:

$$\mathbf{B}_{\mathbf{xz}}^{M} = \sum_{k \in M} [\mathbf{B}_{\mathbf{xy}}]_{:,k} [\mathbf{B}_{\mathbf{yz}}]_{k,:},$$

where $[\mathbf{B}_{\mathbf{xy}}]_{:,a}$ represents the ath row of $\mathbf{B}_{\mathbf{xy}}$ and $[\mathbf{B}_{\mathbf{yz}}]_{b,:}$ represents the bth column of $\mathbf{B}_{\mathbf{yz}}$. In the above equation, $[\mathbf{B}_{\mathbf{xz}}^{M}]_{i,j}$ quantifies the effect of SNP i on phenotype j through the expression levels of genes in module M. If $M_1, \ldots, M_s$ are disjoint subsets of q genes, where $\cup_{m=1,\ldots,s} M_m$ is the full set of q genes, we have the following decomposition:

$$\mathbf{B}_{\mathbf{xz}} = \sum_{m=1}^{s} \mathbf{B}_{\mathbf{xz}}^{M_m}. \tag{3.23}$$

- **Inferred dependencies among genes after seeing phenotype data:** $\mathbf{\Lambda_{y|x,z}} = \mathbf{\Lambda_{yy}} + \mathbf{\Theta_{yz}} \mathbf{\Lambda_{zz}}^{-1} \mathbf{\Theta_{yz}}^{T}$ represents gene network $\mathbf{\Lambda_{yy}}$ augmented with the component $\mathbf{\Theta_{yz}} \mathbf{\Lambda_{zz}}^{-1} \mathbf{\Theta_{yz}}^{T}$ introduced through dependencies in phenotype network $\mathbf{\Lambda_{zz}}$ (blue dashed edge in Figure 3.3B). In this augmented network, additional edges are introduced between two genes if their expression levels influence the same trait or if they both affect traits that are connected in the phenotype network $\mathbf{\Lambda_{zz}}$. The posterior gene network $\mathbf{\Lambda_{y|x,z}}$, which contains the dependencies among expression levels after taking into account phenotype data, can be obtained by inferring the posterior distribution given phenotypes from the estimated Gaussian chain graph model as follows:

$$p(\mathbf{y}|\mathbf{x}, \mathbf{z}) = N\Big(- \big(\mathbf{z}^T \mathbf{\Theta}_{\mathbf{yz}}^T + \mathbf{x}^T \mathbf{\Theta_{xy}}\big) \mathbf{\Lambda}_{\mathbf{y|x,z}}^{-1},\ \mathbf{\Lambda}_{\mathbf{y|x,z}}^{-1} \Big),$$

where

$$\mathbf{\Lambda_{y|x,z}} = \mathbf{\Lambda_{yy}} + \mathbf{\Theta_{yz}} \mathbf{\Lambda_{zz}}^{-1} \mathbf{\Theta}_{\mathbf{yz}}^T.$$

The inferred network $\mathbf{\Lambda_{y|x,z}}$ can also be seen by inferring from the estimated model the joint distribution

$$p(\mathbf{z}, \mathbf{y}|\mathbf{x}) = N\Big(- \mathbf{\Lambda}_{(\mathbf{zz,yy})}^{-1} \mathbf{\Theta}_{(\mathbf{yz,xy})}^T \mathbf{x}, \mathbf{\Lambda}_{(\mathbf{zz,yy})}^{-1} \Big),$$

where $\mathbf{\Theta_{(yz,xy)}} = (\mathbf{0}_{p \times r}, \mathbf{\Theta_{xy}})$ with $p \times r$ matrix of 0's and $\mathbf{\Lambda_{(zz,yy)}} = \begin{pmatrix} \mathbf{\Lambda_{zz}} & \mathbf{\Theta}_{\mathbf{yz}}^T \\ \mathbf{\Theta_{yz}} & \mathbf{\Lambda_{y|x,z}} \end{pmatrix}$. This joint distribution is an alternative representation of the same Gaussian chain graph model in Eq. (3.4) and corresponds to another sparse CGGM over $\mathbf{y}$ and $\mathbf{z}$ conditional on $\mathbf{x}$. This process of introducing the additional dependencies via $\mathbf{\Theta_{yz}} \mathbf{\Lambda_{zz}}^{-1} \mathbf{\Theta}_{\mathbf{yz}}^T$ in this new sparse CGGM, which is equivalent to the original chain graph model, is also known as moralization in the probabilistic graphical model literature [58].

3.3.3 Flow of SNP perturbation effects through the gene network onto traits

Motivated by SNP-trait regression decomposition (Eq. 3.23) we define a network flow tensor representing how SNP perturbations effect traits via the gene network. Score functions defined

as summations over the tensor summarize the role of genes or gene modules in mediating the effect of SNPs on traits or trait groups. Each coefficient in the tensor, corresponding to SNP s, gene g, and trait t, captures the mediating role of gene g in propagating perturbations of s onto t.

Various different order-3 network flow tensors can be chosen, which represent the flow of either direct or indirect perturbations through 3-layer chain model.

- **Indirect perturbation effects on gene levels and indirect effects of gene levels on clinical traits**

  The network flow tensor is computed using $\mathbf{B_{xy}}$, which represents indirect SNP perturbation effects, and $\mathbf{B_{yz}}$, which represents indirect gene effects on traits:

  $$T_{sgt} = [\mathbf{B_{xy}}]_{sg}[\mathbf{B_{yz}}]_{gt}. \tag{3.24}$$

  Each coefficient T_{sgt} measures the contribution of gene g in the decomposition of $\mathbf{B_{xz}}$, shown in Eq. (3.23).

  Inspired by this decomposition of indirect SNP-trait effects into indirect SNP-gene effects and indirect gene-trait effects, we propose using other additive decompositions involving direct effects, as below.

- **Indirect perturbation effects on gene levels and direct effects of gene levels on clinical traits**

  The network flow tensor is now computed using $\mathbf{B_{xy}}$, which represents indirect SNP perturbation effects, and $\mathbf{\Theta_{yz}}$, which represents direct gene effects on traits, as follows:

  $$U_{sgt} = [\mathbf{\Theta_{xy}}]_{sg}[\mathbf{B_{yz}}]_{gt}. \tag{3.25}$$

- **Direct perturbation effects on gene levels and indirect effects of gene levels on clinical traits**

  The network flow tensor is computed using $\mathbf{\Theta_{xy}}$, which represents direct SNP perturbation effects, and $\mathbf{B_{yz}}$, which represents indirect gene effects on traits, as follows:

  $$V_{sgt} = [\mathbf{B_{xy}}]_{sg}[\mathbf{\Theta_{yz}}]_{gt}. \tag{3.26}$$

- **Direct perturbation effects on gene levels and direct effects of gene levels on clinical traits**

  The network flow tensor is computed from $\mathbf{\Theta_{xy}}$, which represents direct SNP perturbation effects, and $\mathbf{\Theta_{yz}}$, which represents direct gene effects on traits, as follows:

  $$W_{sgt} = [\mathbf{\Theta_{xy}}]_{sg}[\mathbf{\Theta_{yz}}]_{gt}. \tag{3.27}$$

Questions about how SNPs and gene modules work in concert to effect phenotypes can be answered via score functions operating on these network flow tensors. We can rank the importance of SNPs and genes (or gene modules) by their score function output, selecting the SNPs and genes which maximize the score function output for a particular trait or trait group. Each score function can be computed for a network flow tensor $F \in \{T, U, V, W\}$, depending on whether we want to base our analysis on the flow of either direct or indirect relationships for SNP-gene perturbations and gene-trait effects.

- **Scoring genes and gene modules by their mediation of a SNP on a trait group**
  For each SNP s and trait group $\mathcal{T}$, we define a score function with query input $\mathcal{G}$, which is either a single gene or gene module, as follows:

$$c_{s,\mathcal{T}}(\mathcal{G}; F) = \sum_{t \in \mathcal{T}} \sum_{g \in \mathcal{G}} |F_{sgt}|. \tag{3.28}$$

- **Scoring SNPs by how much they influence a trait group through the gene network**
  For a trait group $\mathcal{T}$, we define a score function with query input SNP s as follows:

$$b_{\mathcal{T}}(s; F) = \sum_{t \in \mathcal{T}} \sum_{g} |F_{sgt}|. \tag{3.29}$$

- **Scoring genes or modules by their mediation of all genetic variation on a trait group**
  For a trait group $\mathcal{T}$, we define a score with input $\mathcal{G}$, a gene or gene module, as follows:

$$d_{\mathcal{T}}(\mathcal{G}; F) = \sum_{s} \sum_{t \in \mathcal{T}} \sum_{g \in \mathcal{G}} |F_{sgt}|. \tag{3.30}$$

3.4 Experiments

In this section, we empirically demonstrate that CGGMs are more effective components for sparse Gaussian chain graph models than linear regression for various tasks, using synthetic and real-world genomic datasets. We used the sparse three-layer structure for $p(\mathbf{z}, \mathbf{y}|\mathbf{x})$ in all our experiments.

3.4.1 Simulation Study

In simulation study, we considered two scenarios for true models, CGGM-based and linear-regression-based Gaussian chain graph models. We evaluated the performance in terms of graph structure recovery and prediction accuracy in both supervised and semi-supervised settings.

In order to simulate data, we assumed the problem size of J=500, K=100, and L=50 for $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$, respectively, and generated samples from known true models. Since we do not model $p(\mathbf{x})$, we used an arbitrary choice of multinomial distribution to generate samples for $\mathbf{x}$. The true parameters for CGGM-based simulation were set as follows. We set the graph structure in $\mathbf{\Lambda_{yy}}$ to a randomly-generated scale-free network with a community structure [96] with six communities. The edge weights were drawn randomly from a uniform distribution [0.8, 1.2]. We then set $\mathbf{\Lambda_{yy}}$ to the graph Laplacian of this network plus small positive values along the diagonal so that $\mathbf{\Lambda_{yy}}$ is positive definite. We generated $\mathbf{\Lambda_{zz}}$ using a similar strategy, assuming four communities. $\mathbf{\Theta_{xy}}$ was set to a sparse random matrix, where 0.4% of the elements have non-zero values drawn from a uniform distribution [-1.2,-0.8]. $\mathbf{\Theta_{yz}}$ was generated using a similar strategy, with a sparsity level of 0.5%. We set the sparsity pattern of $\mathbf{\Theta_{yz}}$ so that it roughly respects the functional mapping from communities in $\mathbf{y}$ to communities in $\mathbf{z}$. Specifically, after reordering the variables

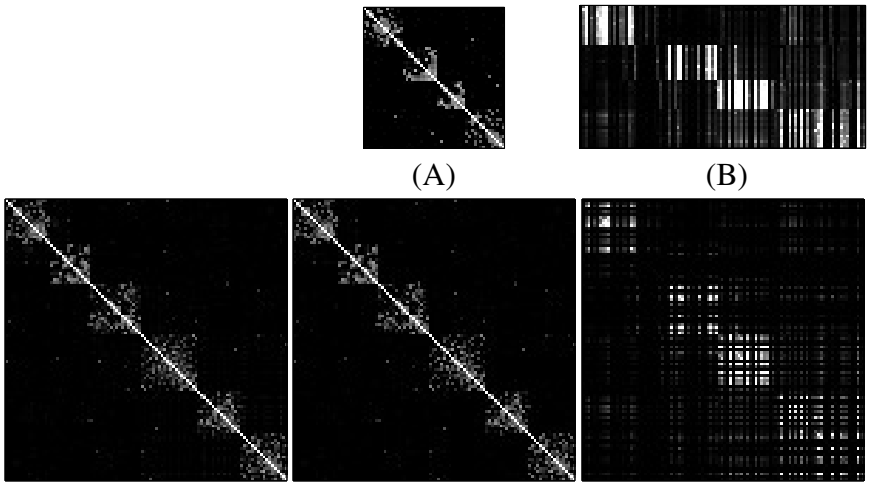


Figure 3.4: Illustration of the structured sparsity recovered by the model with CGGM components, simulated dataset. (A) $\mathbf{\Lambda_{zz}}$. (B) $\mathbf{B_{yz}} = -\mathbf{\Lambda_{zz}^{-1}\Theta_{yz}^{T}}$ shows the effects of marginalization (white vertical bars). The effects of moralization are shown in (C) $\mathbf{\Lambda_{yy}} + \mathbf{\Theta_{yz}\Lambda_{zz}^{-1}\Theta_{yz}^{T}}$, and its decomposition into (D) $\mathbf{\Lambda_{yy}}$ and (E) $\mathbf{\Theta_{yz}\Lambda_{zz}^{-1}\Theta_{yz}^{T}}$.

in $\mathbf{y}$ and $\mathbf{z}$ by performing hierarchical clustering on each of the two networks $\mathbf{\Lambda_{yy}}$ and $\mathbf{\Lambda_{zz}}$, the non-zero elements were selected randomly around the diagonal of $\mathbf{\Theta_{yz}}$.

We set the true parameters for the linear-regression-based models using the same strategy as the CGGM-based simulation above for $\mathbf{\Lambda_{yy}}$ and $\mathbf{\Lambda_{zz}}$. We set $\mathbf{B_{xy}}$ so that 50% of the variables in $\mathbf{x}$ have non-zero influence on five randomly chosen variables in $\mathbf{y}$ in one randomly chosen community in $\mathbf{\Lambda_{yy}}$. We set $\mathbf{B_{yz}}$ in a similar manner, assuming 80% of the variables in $\mathbf{y}$ are relevant to eight randomly-chosen variables in $\mathbf{z}$ from a randomly-chosen community in $\mathbf{\Lambda_{zz}}$.

Each dataset consisted of 600 samples, of which 400 and 200 samples were used as training and test sets. To select the regularization parameters, we estimated a model using 300 samples, evaluated prediction errors on the other 100 samples in the training set, and selected the values with the lowest prediction errors. We used the optimization methods in [100] for CGGM-based models and the MRCE procedure [77] for linear-regression-based models.

Figure 3.4 illustrates how the model with CGGM chain components can be used to discover the structured sparsity via inference and moralization. In each panel, black and bright pixels correspond to zero and non-zero values, respectively. While Figure 3.4A shows how variables in $\mathbf{z}$ are related in $\mathbf{\Lambda_{zz}}$, Figure 3.4B shows $\mathbf{B_{yz}} = -\mathbf{\Lambda_{zz}^{-1}\Theta_{yz}^{T}}$ obtained via marginalization within the CGGM $p(\mathbf{z}|\mathbf{y})$, where functional mappings from variables in $\mathbf{y}$ to multiple related variables in $\mathbf{z}$ can be seen as white vertical bars. In Figure 3.4C, the effects of moralization $\mathbf{\Lambda_{yy}} + \mathbf{\Theta_{yz}\Lambda_{zz}^{-1}\Theta_{yz}^{T}}$ are shown, which further decomposes into $\mathbf{\Lambda_{yy}}$ (Figure 3.4D) and $\mathbf{\Theta_{yz}\Lambda_{zz}^{-1}\Theta_{yz}^{T}}$ (Figure 3.4E). The additional edges among variables in $\mathbf{y}$ in Figure 3.4E correspond to the edges introduced via moralization and show the groupings of the variables $\mathbf{y}$ as the block structure along the diagonal. By examining Figures 3.4B and 3.4E, we can infer a functional mapping from modules in $\mathbf{y}$ to modules in $\mathbf{z}$.

In order to systematically compare the performance of CGGM-based and linear-regression-

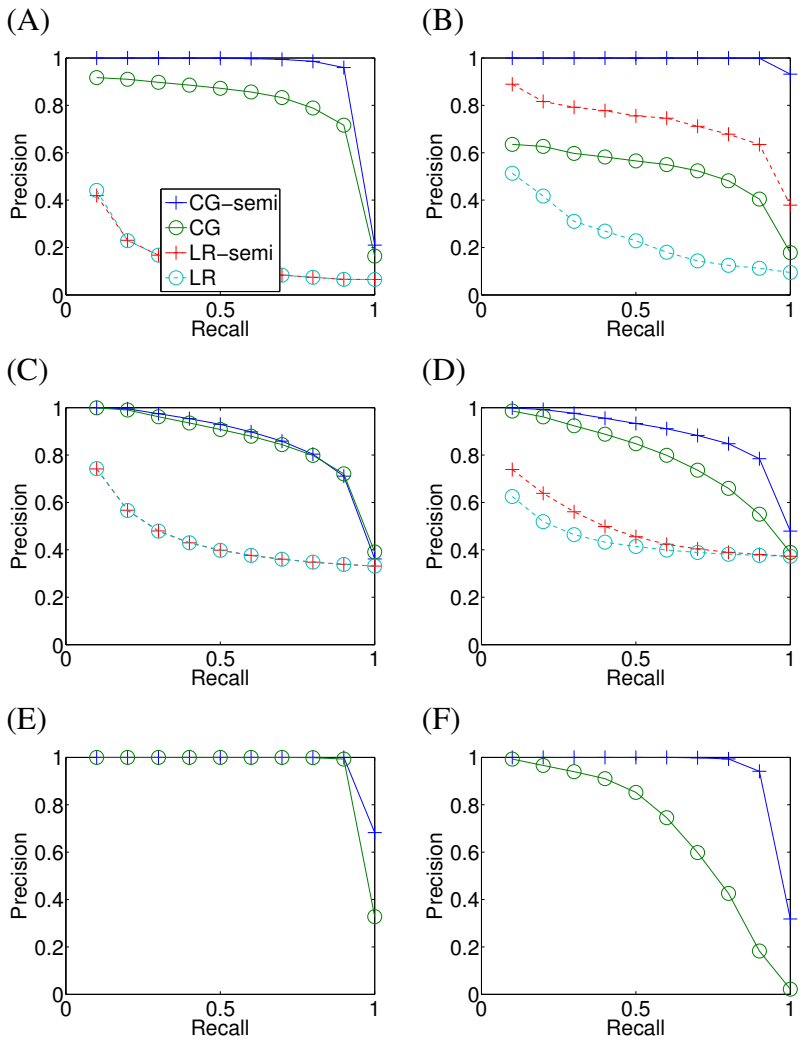


Figure 3.5: Precision/recall curves for graph structure recovery in CGGM-based simulation study. (A) $\mathbf{\Lambda_{yy}}$, (B) $\mathbf{\Lambda_{zz}}$, (C) $\mathbf{B_{xy}}$, (D) $\mathbf{B_{yz}}$, and (E) $\mathbf{\Theta_{xy}}$ and (F) $\mathbf{\Theta_{yz}}$. (CG: CGGM-based models with supervised learning, CG-semi: CG with semi-supervised learning, LR: linear-regression-based models with supervised learning, LR-semi: LR with semi-supervised learning.)

based models, we examined the average performance over 30 randomly-generated datasets. We considered both supervised and semi-supervised settings. Assuming that 200 samples out of the total 400 training samples were missing data for $\mathbf{y}$, for supervised learning, we used only those samples with complete data; for semi-supervised learning, we used all samples, including partially-observed cases.

The precision/recall curves for recovering the true graph structures are shown in Figure 3.5, using datasets simulated from the true models with CGGM components. Each curve was obtained as an average over 30 different datasets. We observe that in both supervised and semi-supervised settings, the models with CGGM components outperform the ones with linear regression components. In addition, the performance of the CGGM-based models improves sig-

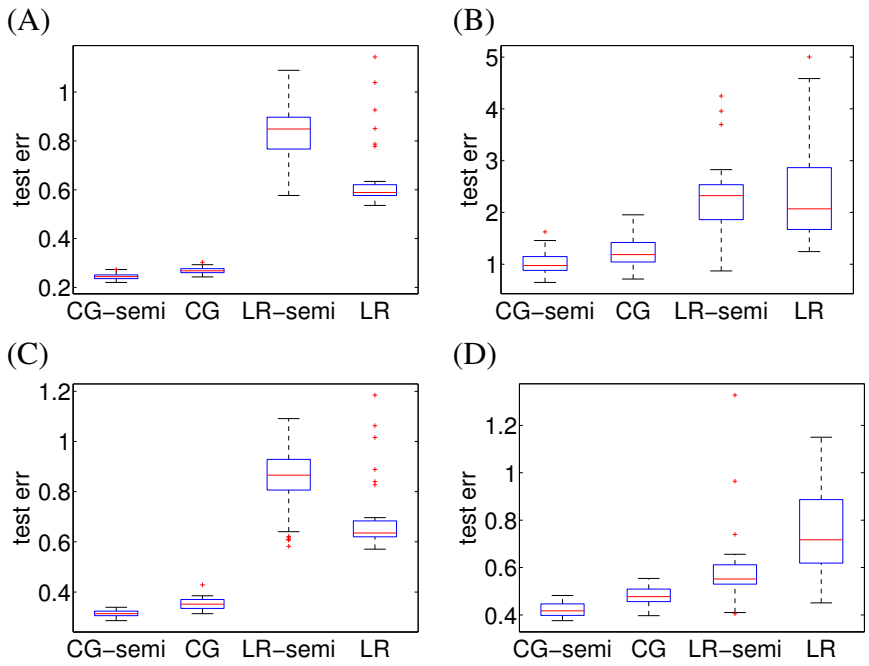


Figure 3.6: Prediction errors in CGGM-based simulation study. The same datasets and estimated models in Figure 3.5 were used to predict (A) $\mathbf{y}$ given $\mathbf{x}, \mathbf{z}$, (B) $\mathbf{z}$ given $\mathbf{x}$, (C) $\mathbf{y}$ given $\mathbf{x}$, and (D) $\mathbf{z}$ given $\mathbf{y}$.

nificantly, when using the partially-observed data in addition to the fully-observed samples (the curve for CG-semi in Figure 3.5), compared to using only the fully-observed samples (the curve for CG in Figure 3.5). This improvement from using partially-observed data is substantially smaller for the linear-regression-based models. The average prediction errors from the same set of estimated models in Figure 3.5 are shown in Figure 3.6. The CGGM-based models outperform in all prediction tasks, because they can leverage the underlying structure in the data and estimate models more effectively.

For the simulation scenario using the linear-regression-based true models, we show the results for precision/recall curves and prediction errors in Figures 3.7 and 3.8, respectively. We find that even though the data were generated from chain graph models with linear regression components, the CGGM-based methods perform as well as or better than the other models.

We also examined the performance of the CGGM-based models, as we varied the number of samples with missing observations for $\mathbf{y}$. Figure 3.9 shows precision/recall curves averaged over 30 datasets, using datasets simulated from CGGM-based true models. We find that the accuracy for recovering the true graph structures improves, as more data are available for $\mathbf{y}$, although the recovery of $\mathbf{\Lambda_{zz}}$ remains unaffected by the number of samples with missing data.

3.4.2 Integrative Genomic Data Analysis

We applied the two types of three-layer chain graph models to single-nucleotide-polymorphism (SNP), gene-expression, and phenotype data from the pancreatic islets study for diabetic mice

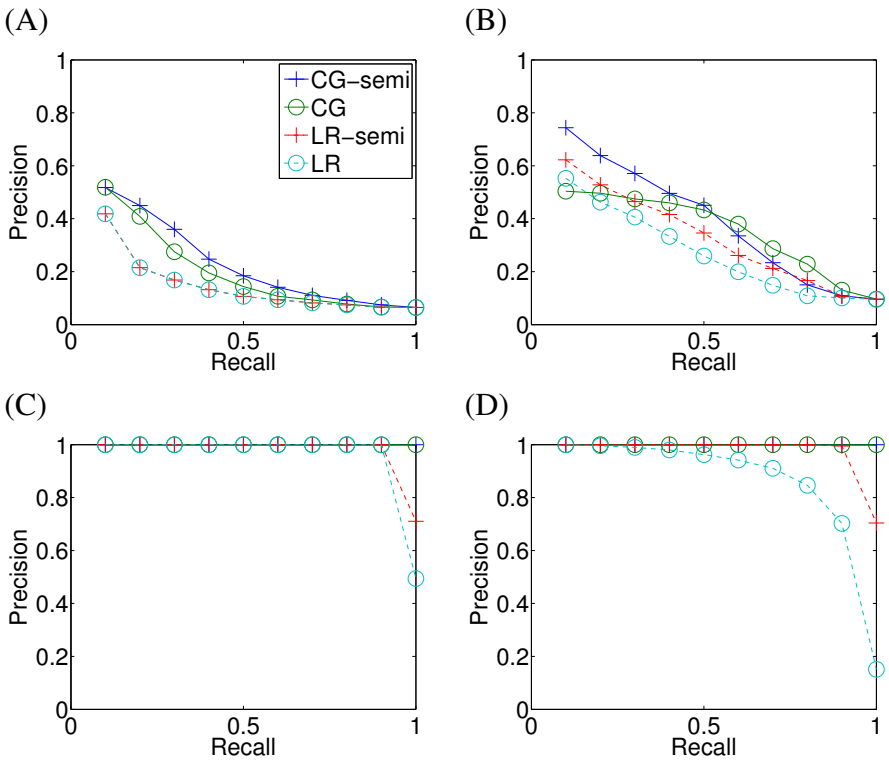


Figure 3.7: Performance for graph structure recovery in linear-regression-based simulation study. Precision/recall curves are shown for (A) $\mathbf{\Lambda_{yy}}$, (B) $\mathbf{\Lambda_{zz}}$, (C) $\mathbf{B_{xy}}$, and (D) $\mathbf{B_{yz}}$.

[91]. We selected 200 islet gene-expression traits after performing hierarchical clustering to find several gene modules. Our dataset also included 1000 SNPs and 100 pancreatic islet cell phenotypes. Of the total 506 samples, we used 406 as training set, of which 100 were held out as a validation set to select regularization parameters, and used the remaining 100 samples as test set to evaluate prediction accuracies. We considered both supervised and semi-supervised settings, assuming gene expressions are missing for 150 mice. In supervised learning, only those samples without missing gene expressions were used.

As can be seen from the prediction errors in Table 3.1, the models with CGGM chain components are more accurate in various prediction tasks. In addition, the CGGM-based models can more effectively leverage the samples with partially-observed data than linear-regression-based models.

Table 3.1: Prediction errors, mouse diabetes data

Task	CG-semi	CG	LR-semi	LR
$\mathbf{y} \mid \mathbf{x}, \mathbf{z}$	0.9070	0.9996	1.0958	0.9671
$\mathbf{z} \mid \mathbf{x}$	1.0661	1.0585	1.0505	1.0614
$\mathbf{y} \mid \mathbf{x}$	0.8989	0.9382	0.9332	0.9103
$\mathbf{z} \mid \mathbf{y}$	1.0712	1.0861	1.1095	1.0765

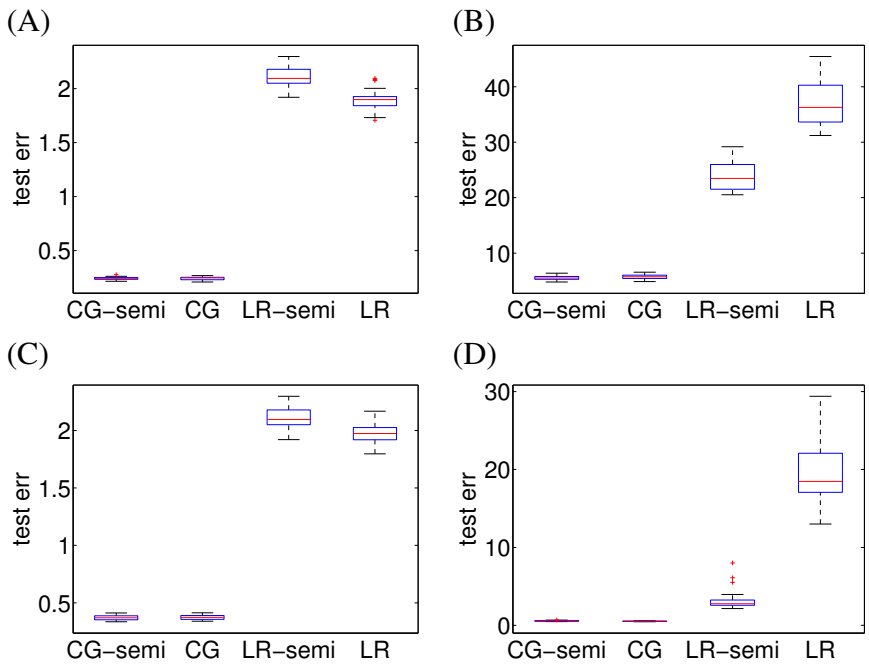


Figure 3.8: Prediction errors in linear-regression-based simulation study. The same estimated models in Figure 3.7 were used to predict (A) $\mathbf{y}$ given $\mathbf{x}, \mathbf{z}$, (B) $\mathbf{z}$ given $\mathbf{x}$, (C) $\mathbf{y}$ given $\mathbf{x}$, and (D) $\mathbf{z}$ given $\mathbf{y}$.

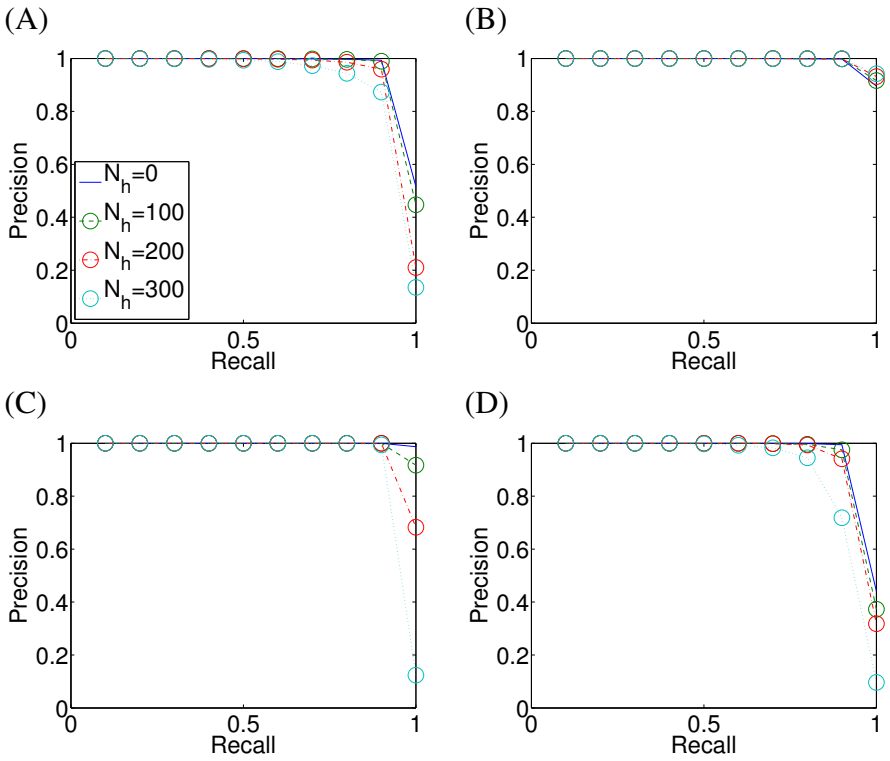


Figure 3.9: Performance in graph structure recovery in semi-supervised learning in simulation study. Using different numbers of partially-observed samples N_h, precision/recall curves for CGGM-based chain graphs are shown for (A) $\mathbf{\Lambda_{yy}}$, (B) $\mathbf{\Lambda_{zz}}$, (C) $\mathbf{\Theta_{xy}}$, and (D) $\mathbf{\Theta_{yz}}$.

3.5 Conclusions for Sparse Gaussian Chain Graph Models

In this chapter, we addressed the problem of learning the structure of Gaussian chain graph models in a high-dimensional space. We argued that when the goal is to recover the model structure, using sparse CGGMs as chain component models has many advantages such as recovery of structured sparsity, computational efficiency, globally-optimal solutions for parameter estimates, and superior performance in semi-supervised learning.

Chapter 4

Asthma PerturbNet Analysis

4.1 Preparation of asthma dataset

We applied our method to a dataset comprising genotype, gene expression, and clinical phenotype data, collected from asthma patients participating in CAMP study [20, 21, 67]. We used 174 non-Hispanic Caucasian subjects for whom both genotype and clinical phenotype data were available. For a subset of 140 individuals, gene expression data from primary peripheral blood CD4+ lymphocytes were also available. After removing SNPs with minor allele frequency less than 0.1 and those with missing reference SNP ids, we obtained 495,597 SNPs for autosomal chromosomes. We then imputed missing genotypes using fastPHASE [81]. Given expression levels for 22,184 mRNA transcripts profiled with Illumina HumanRef8 v2 BeadChip arrays [67], we removed transcript levels with expression variance less than 0.01, which resulted in a set of 11,598 expression levels to be used in our analysis. Then, we converted the expression values to their z-scores. The clinical phenotype data comprised 35 phenotypes (Table 4.1), including 25 features related to lung function and 10 features collected via blood testing. The clinical phenotypes were converted to their z-scores within each phenotype so that all phenotypes have equal variance. We then imputed missing values using low-rank matrix completion [17], then repeated the procedure for centering and scaling features to their z-scores.

Table 4.1: Description of asthma clinical phenotypes in CAMP data.

Phenotype Name	Phenotype Group	Description
PREFVCPP	lung	Baseline FVC % predicted
HPRFVCPP	lung	Hankinson pre BD FVC % predicted
HPRFVCPC	lung	Hankinson pre BD FVC percentile
HPRFVCZS	lung	Hankinson pre BD FVC z-score
PREFEVPP	lung	Pre BD FEV % predicted
HPRFEVPP	lung	Hankinson pre BD FEV % predicted
HPRFEVPC	lung	Hankinson pre BD FEV percentile
HPRFEVZS	lung	Hankinson pre BD FEV z-score
PO1FEVPP	lung	1st Post BD FEV % predicted
HPO1FEPP	lung	1st Hankinson post BD FEV % predicted
HPO1FEPC	lung	1st Hankinson post BD FEV percentile
HPO1FEZS	lung	1st Hankinson post BD FEV z-score
PREFF	lung	Pre BD FEV/FVC ratio (%)
HPRFFPC	lung	Hankinson pre BD FEV/FVC percentile
HPRFFZS	lung	Hankinson pre BD FEV/FVC z-score
MC928	lung	Baseline (pre-diluent) FEV1/FVC ratio
PREFVC	lung	Pre BD FVC
PREFEV	lung	Pre BD FEV
PO1FEV	lung	1st Post BD FEV
MC935b	lung	FEV1 15 minutes after 2puffs albuterol
PREPF	lung	Pre BD peak flow
POSPF	lung	Post BD peak flow
lnpc20	lung	Airway responsiveness to methacholine
rescueBD7day	lung	Rescue BD use last 7 days
preventBD7day	lung	Preventative BD use last 7 days
HEMOG	blood	Hemoglobin
WBC	blood	white blood cell count
MONOPCT	blood	Monocytes %
NEUTPCT	blood	Neutrophils (segs,polys) %
LYMPHPCT	blood	Lymphocytes %
BASOPCT	blood	Basophils %
TOTEOSP	blood	Total eosinophils count by % of WBC
EOSPCT	blood	Eosinophils %
log10ige	blood	Logarithm transformed IgE level
log10eos	blood	Logarithm transformed eosinophil level

4.2 Comparison of the scalability of Mega-sCGGM and other methods

In order to compare the computation of different algorithms, we used the following software and hardware setup. For Lasso, we used the implementation in GLMNET [74] with a backend written in Fortran. For Newton coordinate descent, which is the previous state-of-the-art approach for optimizing sparse CGGMs, we took the implementation written in C++ provided by the authors [100] and sped up this implementation with the Eigen matrix library, by employing low-rank matrix representations and using sparse matrix multiplications. For all methods, the code was compiled and run with OpenMP multi-threading enabled on the same machines with 20Gb of memory and 16 cores. We used the same regularization parameters for our method and the previous method for sparse CGGM optimization, so the resulting solutions were identical with the same sparsity levels. For Lasso, we chose the regularization parameters so that the L_1-norm of the regression matrix roughly matched that of our inferred indirect SNP effects.

In all data points, the number of non-zeros in the Lasso was less than that of our model (counting both $\mathbf{\Lambda_{yy}}$ and $\mathbf{\Theta_{xy}}$). For example, for our experiment including all 495,597 SNPs in the model, there were 226,400 non-zero parameters in our model versus 162,836 in the Lasso model.

We assess the scalability of Mega-sCGGM and other previous algorithms on the expression measurements of 11,598 genes and the genotypes of 495,597 SNPs for 140 subjects from the CAMP data. We estimated sparse CGGMs, using both our new method and the previous state-of-the-art method based on the Newton coordinate descent method [100]. Since the sparse CGGM optimization problem is convex with a single globally optimal solution, both our and previous methods obtain the same parameter estimates, although the computation time differs between the two methods. We also obtained the computation time of Lasso implemented in GLMNET [74, 90], the well-known computationally efficient algorithm for learning a simple but less powerful regression model. Although the sparse multivariate regression with covariance estimation [78] has also provided a methodology that could be used for learning a gene network influenced by SNPs, this approach has been found to take days to learn a model from a small dataset of only 1,000 SNPs and 500 gene expression levels [104], so we did not include it in our experiment. All of the optimization methods were run on the same hardware setup with comparable software implementations.

In our comparison of different methods, our algorithm significantly outperformed the previous state-of-the-art method for learning a sparse CGGM in terms of both computation time and memory requirement and scaled similarly to Lasso (Figure 4.1). In comparison of our method with Lasso on datasets with 40,056 SNPs from chromosome 1, 21,757 SNPs for chromosomes 1 through 6, and 495,597 SNPs from all autosomal chromosomes and all expression measurements, our method was not substantially slower than Lasso, even though our method learns a more expressive model than Lasso. The previous CGGM optimization algorithm ran out of memory even on the smallest dataset above with SNPs only from chromosome 1, so we compared the two algorithms on a much smaller dataset with 1,000 and 10,000 SNPs. On 10,000 SNPs, the previous algorithm required more than four hours, whereas in less than four hours, our algorithm was able to run on all 495,597 SNPs.

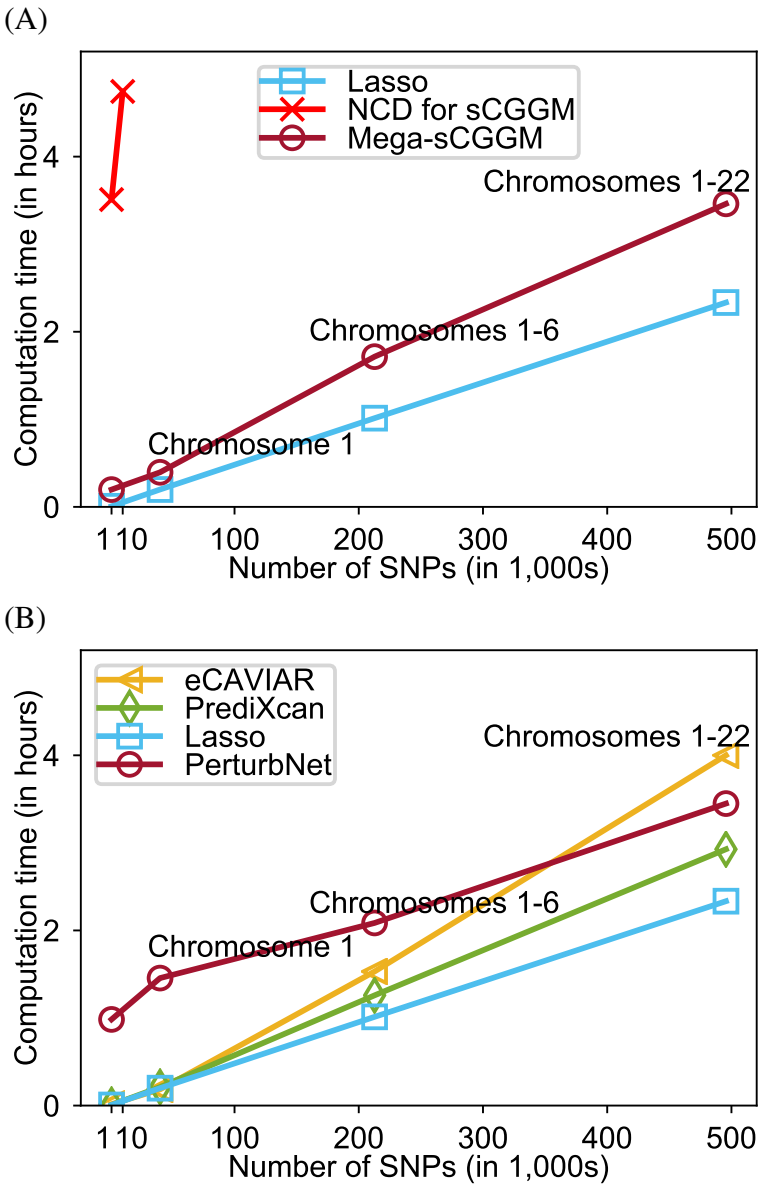


Figure 4.1: Comparison of computation time of different methods. (A) The computation time of our Mega-sCGGM is compared with that of previous learning algorithm for sparse CGGMs and Lasso. We applied all methods to all expression data and genotype data from chromosome 1, chromosomes 1-6, chromosomes 1-16, and chromosomes 1-22. The previous algorithm for sparse CGGMs ran out of memory at chromosome 1, so we obtained its computation time with much smaller datasets with 1,000 and 10,000 SNPs. (B) Computation time for analysis of SNP, expression, and clinical trait data from the CAMP study. The expression levels of 11,598 genes, 35 traits, and varying numbers of SNPs were used. We use runtimes for the PerturbNet method with the EM algorithm for semi-supervised learning.

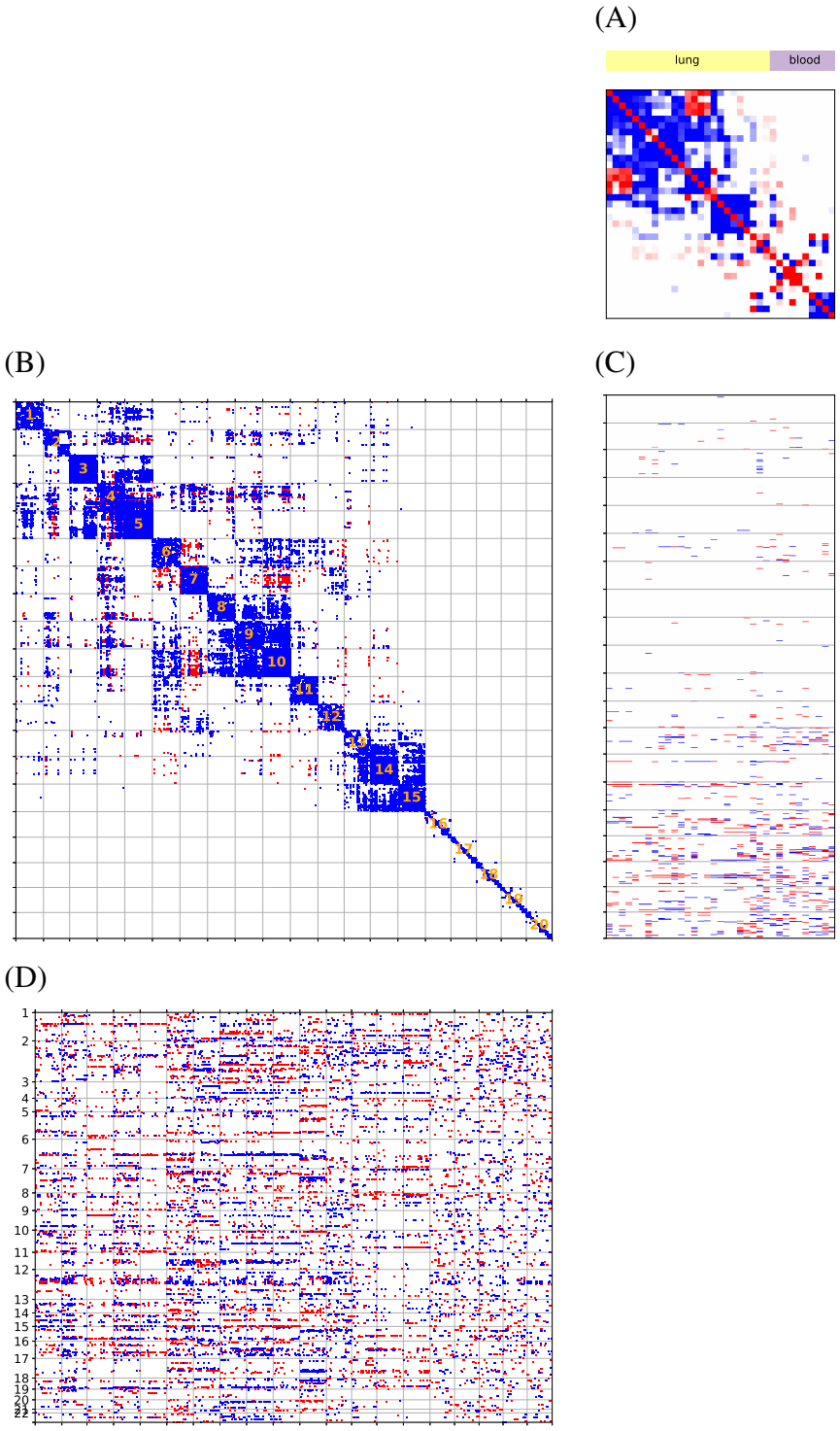


Figure 4.2: The PerturbNet model estimated from asthma data. The parameters of the sparse Gaussian chain graph model estimated from the asthma data are shown. (A) Asthma phenotype network $\mathbf{\Lambda_{zz}}$. The phenotypes were ordered by hierarchical clustering applied to within each of the two groups of phenotypes, lung function traits (yellow) and blood test traits (purple). (B) Gene network $\mathbf{\Lambda_{yy}}$. The gene network is annotated with 20 modules obtained from applying a network clustering algorithm METIS [53] to $\mathbf{\Lambda_{yy}}$. (C) The influence of gene expression levels on phenotypes $\mathbf{\Theta_{yz}}$. (D) SNP perturbation of gene expression levels $\mathbf{\Theta_{xy}}$ for the top 1,000 eQTL hotspots, ordered by genomic location and labeled by chromosomes. In each panel, non-zero elements of the estimated parameters are shown as blue for positive interactions and red for negative interactions.

4.3 Analysis of asthma data

We now fit a sparse Gaussian chain graph model to the genotype, expression, clinical phenotype data gathered from participants in the Childhood Asthma Management Program (CAMP) [20, 21, 67]. After preprocessing the data, we applied our method to the data from 140 subjects for whom all data were available for 495,597 SNPs on 22 autosomal chromosomes, 11,598 gene expression levels, and 35 phenotypes (Table 4.1) and 34 additional subjects for whom data were available only for genotypes and phenotypes but not for expression levels. Below we perform a detailed analysis of the estimated model.

4.3.1 Overview of the PerturbNet model for asthma

We first examined the overall estimated model for the module structures in the phenotype and gene networks (Figure 4.2). To see the structure in the phenotype network $\mathbf{\Lambda_{zz}}$, we reordered the nodes of the network by applying hierarchical clustering to each set of the lung function and blood test phenotypes. This revealed the dense connectivities within the two known groups of phenotypes and the two sub-clusters within the group of lung function phenotypes (Figure 4.2A).

The gene network $\mathbf{\Lambda_{yy}}$ also showed a clear module structure (Figure 4.2B). To find the module structure in the network, we identified the genes that are connected to at least one other gene in the network $\mathbf{\Lambda_{yy}}$ and partitioned the network over those genes into 20 subnetworks with roughly equal number of nodes, using the network clustering algorithm METIS [53]. Out of 11,598 genes, 6,102 genes were connected to at least one other gene in the network. For the rest of our analysis, we focus on the network and modules over the 6,102 genes, since these genes are likely to form modules for pathways with a functional impact on asthma phenotypes. Modules 1-15 were densely connected clusters of co-expressed genes, suggesting those modules are likely to consist of a functionally coherent set of genes, whereas modules 16-20 had relatively fewer edge connections within each cluster.

Next, we considered the effects of the gene modules on the lung and blood phenotypes in $\mathbf{\Theta_{yz}}$ and the SNP perturbations of the gene modules in $\mathbf{\Theta_{xy}}$. Modules 1-12 had relatively small effects on the phenotypes despite their dense connectivities, whereas modules 13-20 appeared to have stronger effects on both groups of phenotypes (Figure 4.2C). The SNP effects on the modules in $\mathbf{\Theta_{xy}}$ for the top 1000 eQTL hotspots, determined by overall SNP effects on all genes ($\sum_j |[\mathbf{\Theta_{xy}}]_{i,j}|$ for each SNP i), showed that many of these hotspots perturb the expression of genes in the same module in the gene network (Figure 4.2D). Given these observations from the visual inspection of $\mathbf{\Theta_{yz}}$ and $\mathbf{\Theta_{xy}}$, we summarized $\mathbf{\Theta_{yz}}$ and $\mathbf{\Theta_{xy}}$ at module level and compared the module-level summaries across modules. To quantify the module-level influence of expression levels on each group of phenotypes, from the direct influence $\mathbf{\Theta_{yz}}$ and indirect influence $\mathbf{B_{yz}}$ we computed the overall effect sizes of all genes in the given gene module on all phenotypes in each of the lung and blood phenotype groups ($\sum_{i\in M, j\in K} |[\mathbf{\Theta_{yz}}]_{i,j}|$ and $\sum_{i\in M, j\in K} |[\mathbf{B_{yz}}]_{i,j}|$ for each gene module M and phenotype group K). Similarly, from $\mathbf{\Theta_{xy}}$ and $\mathbf{B_{xy}}$ we computed the overall SNP effect sizes on all genes in the given module ($\sum_{i,j\in M} |[\mathbf{\Theta_{xy}}]_{i,j}|$ and $\sum_{i,j\in M} |[\mathbf{B_{xy}}]_{i,j}|$ for each SNP i and module M).

Among the 20 gene modules, modules 13-20 overall had stronger influence on phenotypes than the other gene modules (Figures 4.3A), although SNP perturbations were found across all

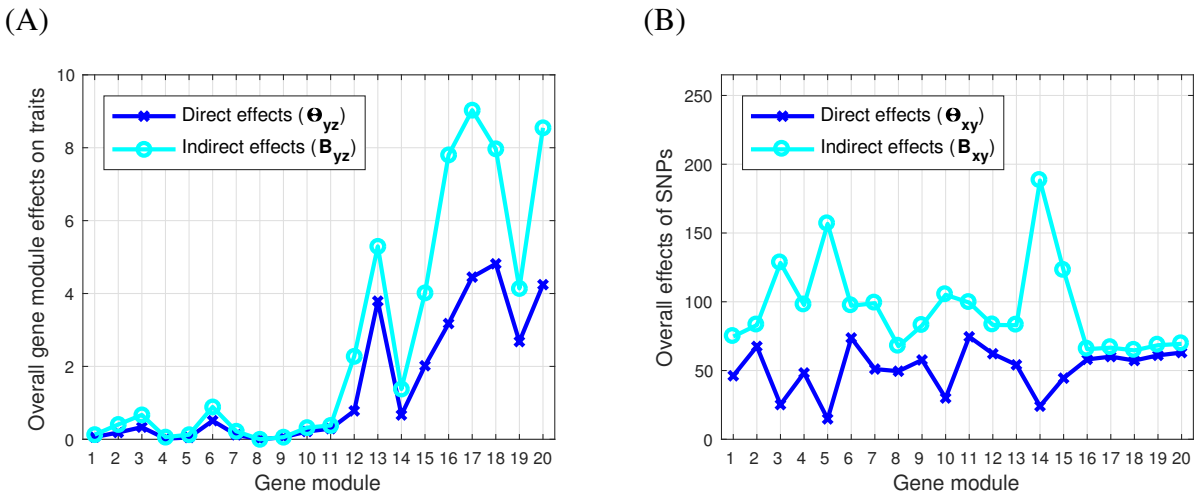


Figure 4.3: SNP effects on gene modules and gene-module effects on phenotypes. (A) Given the estimated $\mathbf{\Theta_{yz}}$ and inferred $\mathbf{B_{yz}}$ for gene-expression effects on phenotypes from the PerturbNet model, we show the gene module effects on phenotypes, computed as the sum of absolute effect sizes across all genes within the module and across all phenotypes. (B) Given the estimated $\mathbf{\Theta_{xy}}$ and inferred $\mathbf{B_{xy}}$ for SNP effects on gene network, we show the SNP effects on each gene module, summarized as the sum of absolute effect sizes across all SNPs and all genes within the module.

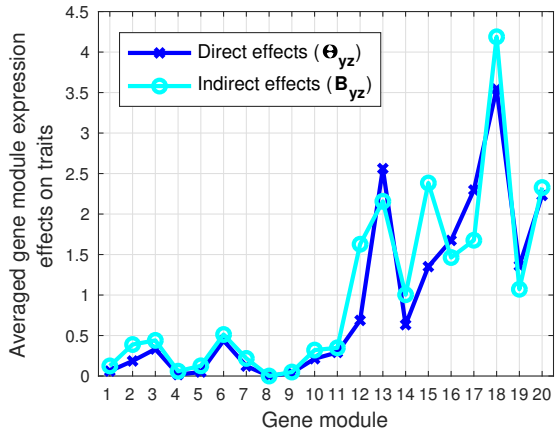


Figure 4.4: Effects of average gene module expression levels on phenotypes. (A) Given the estimated $\mathbf{\Theta_{yz}}$ and inferred $\mathbf{B_{yz}}$ for gene-expression effects on phenotypes from the PerturbNet model, we show the gene module effects on phenotypes, computed as the sum of effect sizes across all genes within the module and across all phenotypes.

gene modules without any preference to those modules with stronger influence on phenotypes (Figure 4.3B). On the other hand, the overall SNP effects were similar across all gene modules for both the direct and indirect SNP effects (Figure 4.3B). The overall indirect SNP effects were larger for some modules (e.g., module 14), but this was largely because of the substantially stronger edge connectivities in that module, which led to stronger propagation of the direct SNP perturbation effects.

We also plot the effects of $\mathbf{\Theta_{yz}}$ and $\mathbf{B_{yz}}$, if we avoid taking absolute values before summing across genes within each module. By incorporating the signs of the gene effects rather than taking absolute values before summing, we instead measure the effect of the average expression level of each module on the phenotypes. Comparing Figures 4.3A and 4.4, we see that the effect of modules 13 and 18 are still large when summing the signed gene effects, showing that the average expression level of those modules are influencing traits. Meanwhile, the effects of modules 16, 17, and 20 are smaller in Figure 4.4, showing that the genes in these modules have effects on traits which cancel each other out, so that the average gene expression level for the entire module is less related to the clinical traits.

4.3.2 Gene modules that influence phenotypes are enriched for immune genes

To determine the functional role of the gene modules, we performed gene ontology (GO) gene set enrichment analysis [8, 35]. For each module, we performed a Fisher's exact test to find the significantly enriched GO categories in biological processes (p-value < 0.05 after Bonferroni correction for multiple testing), using the GO database with annotations for 21,002 genes.

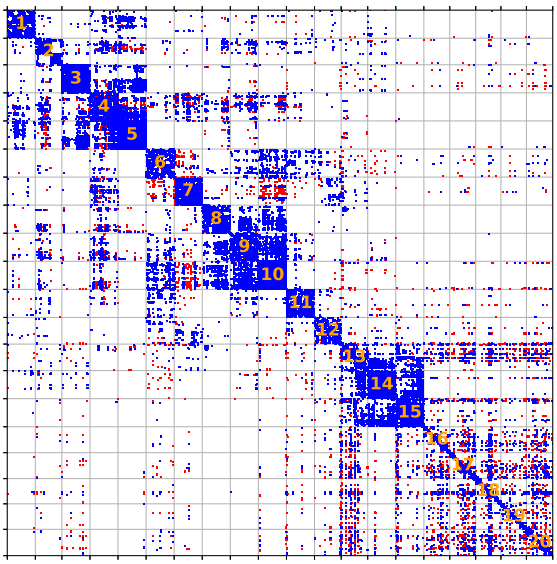


Figure 4.5: Asthma posterior gene network. The posterior gene network $\mathbf{\Lambda_{y|x,z}}$ after taking into account the phenotype data.

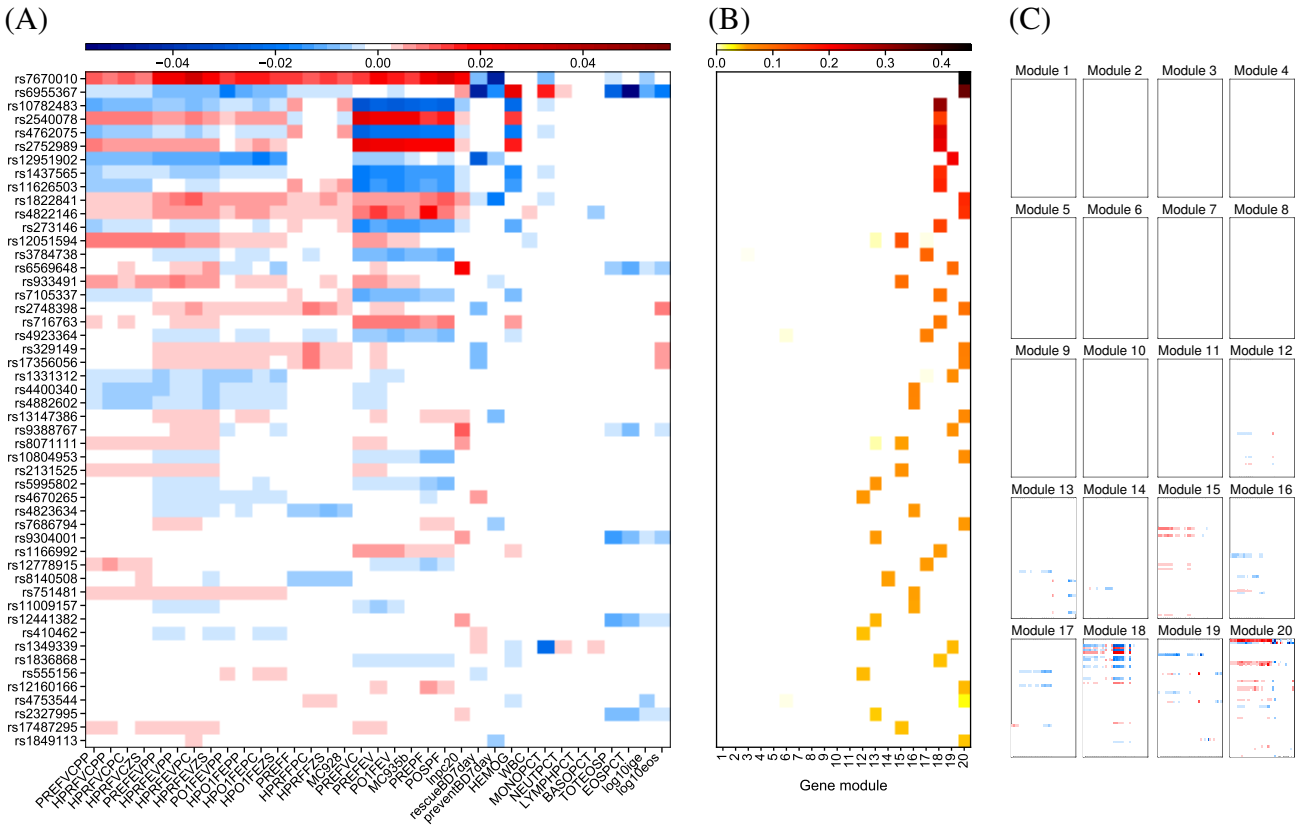


Figure 4.6: Top 50 SNPs perturbing phenotypes and their perturbations effects on phenotypes mediated by gene modules. For the top 50 SNPs perturbing lung phenotypes, we show (A) their effect sizes on phenotypes $\mathbf{B_{xz}}$ and (C) the decomposition of $\mathbf{B_{xz}}$ into component effects $\mathbf{B}_{\mathbf{xz}}^{M_1}, \ldots, \mathbf{B}_{\mathbf{xz}}^{M_{20}}$ mediated by each of the 20 gene modules. The sum over all component effects in Panel (B) is equal to the overall effects in Panel (A). (B) We summarize each component SNP effect $\mathbf{B}_{\mathbf{xz}}^{m}$ for module m in Panel (C) as a row-wise sum of $\mathbf{B}_{\mathbf{xz}}^{m}$, shown as the mth column in the figure. The SNPs are ordered according to their overall effect sizes on the phenotypes.

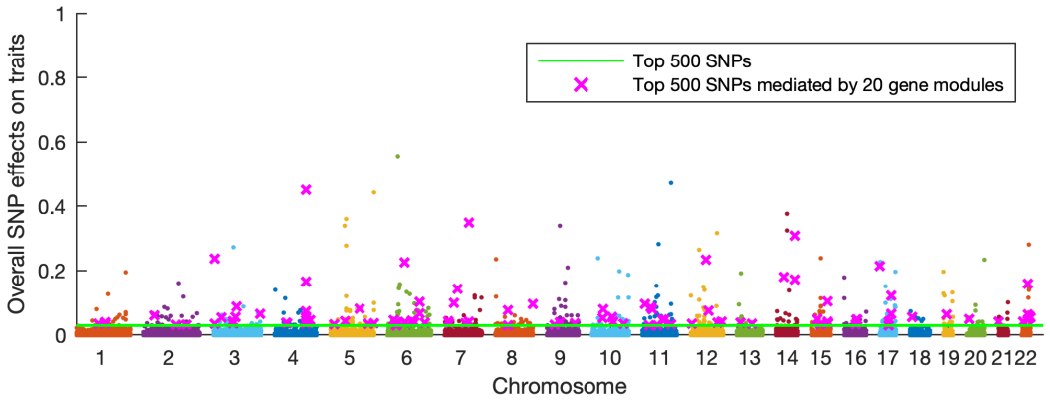


Figure 4.7: Manhattan plots for overall SNP effects on asthma phenotypes determined by PerturbNet. The overall SNP effects on phenotypes from PerturbNet are shown for all SNPs across the 22 autosomal chromosomes. Top 500 SNPs with the strongest effect sizes on phenotypes are shown as SNPs above the green line.

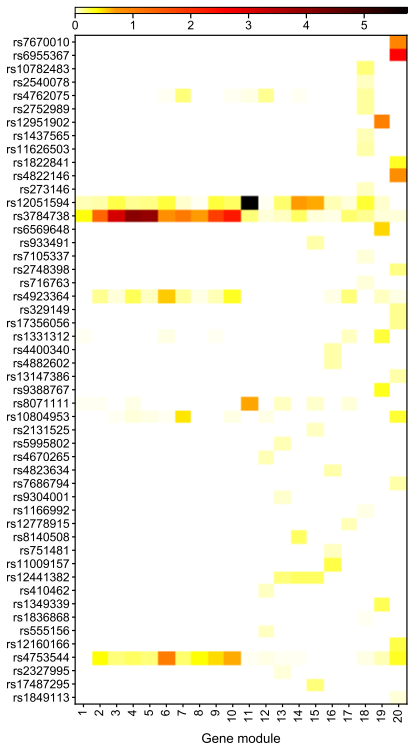


Figure 4.8: PerturbNet asthma SNPs as eQTLs. For each asthma SNP in Figure 4.6, its overall effect on each gene module in the estimated PerturbNet model is shown.

Table 4.2: GO categories enriched in gene modules in the estimated asthma gene network

	Size	Biological Process Pathway	P-value	Overlap*
1	314	Cellular macromolecule metabolic process	2.94×10^{-12}	159 / 7006
2	297	Cellular nitrogen compound metabolic process	1.64×10^{-4}	112 / 5164
3	314	Nucleobase-containing compound metabolic process	5.62×10^{-13}	123 / 4538
4	314	Organelle organization	2.02×10^{-4}	79 / 3167
5	314	Nucleobase-containing compound metabolic process	4.58×10^{-4}	106 / 4538
6	314	Unclassified	NA	NA
7	314	Cellular localization	1.80×10^{-4}	60 / 2287
8	314	Cellular metabolic process	1.73×10^{-4}	178 / 9003
9	314	Cellular metabolic process	8.39×10^{-6}	185 / 9003
10	314	Macromolecule metabolic process	8.73×10^{-5}	159 / 7749
11	314	Heterocycle metabolic process	7.63×10^{-3}	103 / 4715
12	298	Translation	1.65×10^{-9}	27 / 383
13*	297	Immune system process	3.66×10^{-12}	83 / 2552
		Response to stimulus	1.70×10^{-9}	162 / 8009
		Response to stress	8.43×10^{-9}	90 / 3333
14*	314	Immune response	3.96×10^{-38}	106 / 1673
		Leukocyte activation involved in immune response	9.04×10^{-31}	62 / 607
		Granulocyte activation	1.24×10^{-26}	53 / 495
15*	313	Cell activation in immune response	4.95×10^{-32}	65 / 611
		Myeloid leukocyte activation	5.28×10^{-32}	63 / 566
		Immune system process	1.01×10^{-31}	124 / 2552
16	290	Cellular process	3.73×10^{-3}	132 / 15013
17	290	Regulation of macromolecule metabolic process	1.58×10^{-3}	74 / 6142
18	282	Organonitrogen compound metabolic process	1.97×10^{-2}	60 / 5523
19	285	Cell cycle	1.50×10^{-8}	37 / 1355
20	295	Cellular component organization or biogenesis	4.34×10^{-3}	66 / 5525

* The number of genes in the overlap / the total number of genes in the GO category

Among all 20 modules, modules 13-15 had a statistically significant enrichment of GO terms related to immune system function, which also corresponded to the most significant enrichments across all modules (Table 4.2). Even though modules 16-20 did not have any significant enrichment of asthma-related GO categories, many of the genes in these modules were connected to genes in modules 13-15 in the posterior gene network $\mathbf{\Lambda}_{\mathbf{y}|\mathbf{x},\mathbf{z}}$ (Figure 4.5), and thus this subset of genes in modules 16-20 may be also involved with immune system function. To see if this is indeed the case, we obtained the significantly enriched GO categories in the 374 genes in modules 16-20 that are connected to modules 13-15 in the posterior network $\mathbf{\Lambda}_{\mathbf{y}|\mathbf{x},\mathbf{z}}$ (p-value $<$ 0.05 after Bonferroni correction). This set of genes was significantly enriched for several GO categories related to immune system processes, including cellular response to stress (p-value = 2.90×10^{-2} with overlap of 35 genes out of 1599 genes in the category), regulation of defense

response to virus (p-value = 4.36 $\times 10^{-2}$ with overlap of 6 genes out of 71 genes in the category), and regulation of immune effector process (p-value = 4.42 $\times 10^{-2}$ with overlap of 14 genes out of 409 genes in the category).

Thus, all of the modules that influence phenotypes, modules 13-20, showed enrichments in immune-related genes, with significant enrichment for modules 13-15 and weaker but still significant enrichment for modules 16-20. Since asthma is an immune disorder, the enrichment of immune-related genes in the trait-perturbing modules provides evidence that these modules are likely to play an important role in asthma patients.

Another unique feature of PerturbNet is its inference methods for characterizing the role of the gene network in modulating SNP effects on traits. To see if modules 13-20 are indeed the key mediators of the genetic effects on asthma traits as suggested above, we applied two of the PerturbNet inference procedures to the model estimated from the CAMP data. First, we inferred the genetic effects on traits mediated by the network, identifying top 500 SNPs with the largest overall effects on traits (Figure 4.7). Next, we inferred the decomposition of these SNP effects into component effects mediated by different parts of the gene network (Figure 4.6). This decomposition showed that for 99 out of 500 SNPs, the primary mediators of the genetic effects were the gene modules rather than single isolated genes in the network. For all of those 99 SNPs, modules 13-20 mediated nearly all of the genetic effects (Figure 4.6), providing further evidence that these modules are implicated in asthma. Additionally, for nearly all of the 99 SNPs, only one of modules 13-20 served as a mediator (Figure 4.6B), suggesting that a SNP influences traits via localized perturbation of a gene subnetwork. These trait-perturbing SNPs were also eQTLs for modules 13-20, suggesting co-localization between eQTLs and trait-perturbing SNPs (Figure 4.8). Our results demonstrate that PerturbNet inference methods can identify the gene modules that are mediators of SNP effects on traits, providing insights into the functional roles of genetic variants in the disease process.

4.3.3 SNPs perturbing asthma phenotypes overlap with SNPs perturbing immune modules

The SNP perturbation of the gene modules above (Figure 4.3A) may or may not result in a change in phenotypes. To see if the SNPs perturbing each gene module have an impact on the lung and blood phenotypes, we compared the top module-specific eQTLs in $\mathbf{\Theta_{xy}}$ with the SNPs with the strongest effects on the lung or blood phenotypes in $\mathbf{B_{xz}}$ inferred from our sparse Gaussian chain graph model. The SNPs with the strongest effects on the lung (or blood) phenotypes were determined based on the sum over the SNP effects on all lung (or blood) phenotypes in $|\mathbf{B_{xz}}|$. Similarly, the top module-specific eQTLs were determined based on the sum over the SNP effect sizes on all expression levels in each module in $|\mathbf{\Theta_{xy}}|$. We obtained the overlap between the SNPs perturbing the phenotype network and the SNPs perturbing the gene network, considering the top 100 and 200 module-specific eQTLs and top 200 SNPs perturbing each phenotype group (the cutoff for top 200 SNPs shown as the magenta line at SNP effect size 0.01325 for lung traits in Figure 4.9A and at SNP effect size 0.00375 for blood traits in Figure 4.9B). Using Fisher's exact test, we also assessed the significance of these overlaps within the set of SNPs with non-zero effects in $\mathbf{\Theta_{xy}}$.

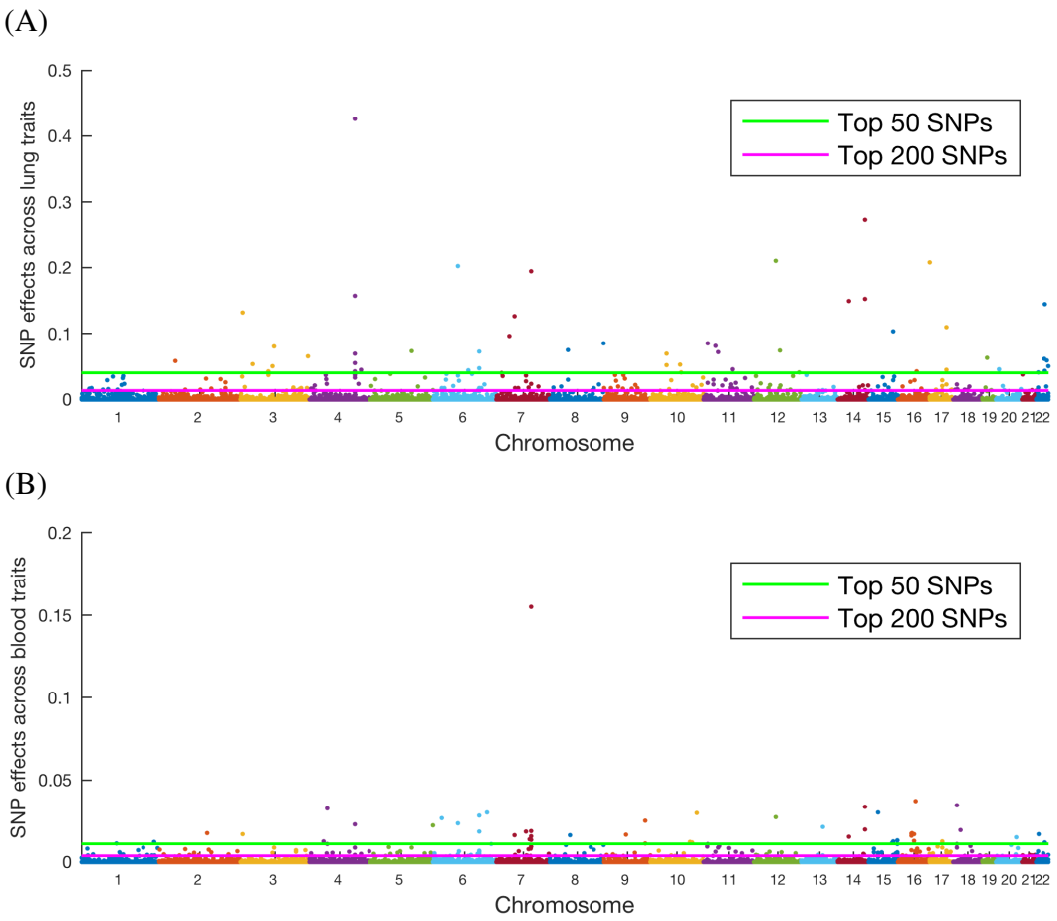


Figure 4.9: Module-mediated SNP effects on asthma phenotypes. The SNP effects on lung and blood phenotypes in $\mathbf{B_{xz}}$ inferred from our estimated model are shown for all SNPs across the 22 autosomal chromosomes. For each SNP, we summarize its effect as the sum of absolute values of effect sizes in $\mathbf{B}_{\mathbf{xz}}^{m}$ across gene modules and across (A) lung phenotypes and (B) blood phenotypes. Top 50 and 200 SNPs with the strongest effect sizes on each group of phenotypes are shown as SNPs above the green and magenta lines respectively.

In our comparison, only a subset of the eQTLs influenced phenotypes, but the eQTLs perturbing the immune modules, modules 13-20, were more likely to perturb the phenotypes than the eQTLs for the other modules (Figures 4.10A and 4.10B). Among the top 100 module-specific eQTLs, only a fraction of those SNPs overlapped with top 200 SNPs perturbing phenotypes (ranging from 0% to 18% of eQTLs across modules for an overlap with SNPs perturbing lung phenotypes and ranging from 2% to 20% of eQTLs across modules for an overlap with SNPs perturbing blood phenotypes). These fractions increased as we considered more eQTLs as in top 200 and all module-specific eQTLs. This matches with the observations from previous studies that not all of the eQTLs affect higher-level phenotypes [73] and that trait-associated SNPs are likely to be eQTLs [69]. However, in our analysis, the eQTLs for immune-related modules, modules 13-20, tended to have larger overlaps than the other modules (Figures 4.10A and 4.10B).

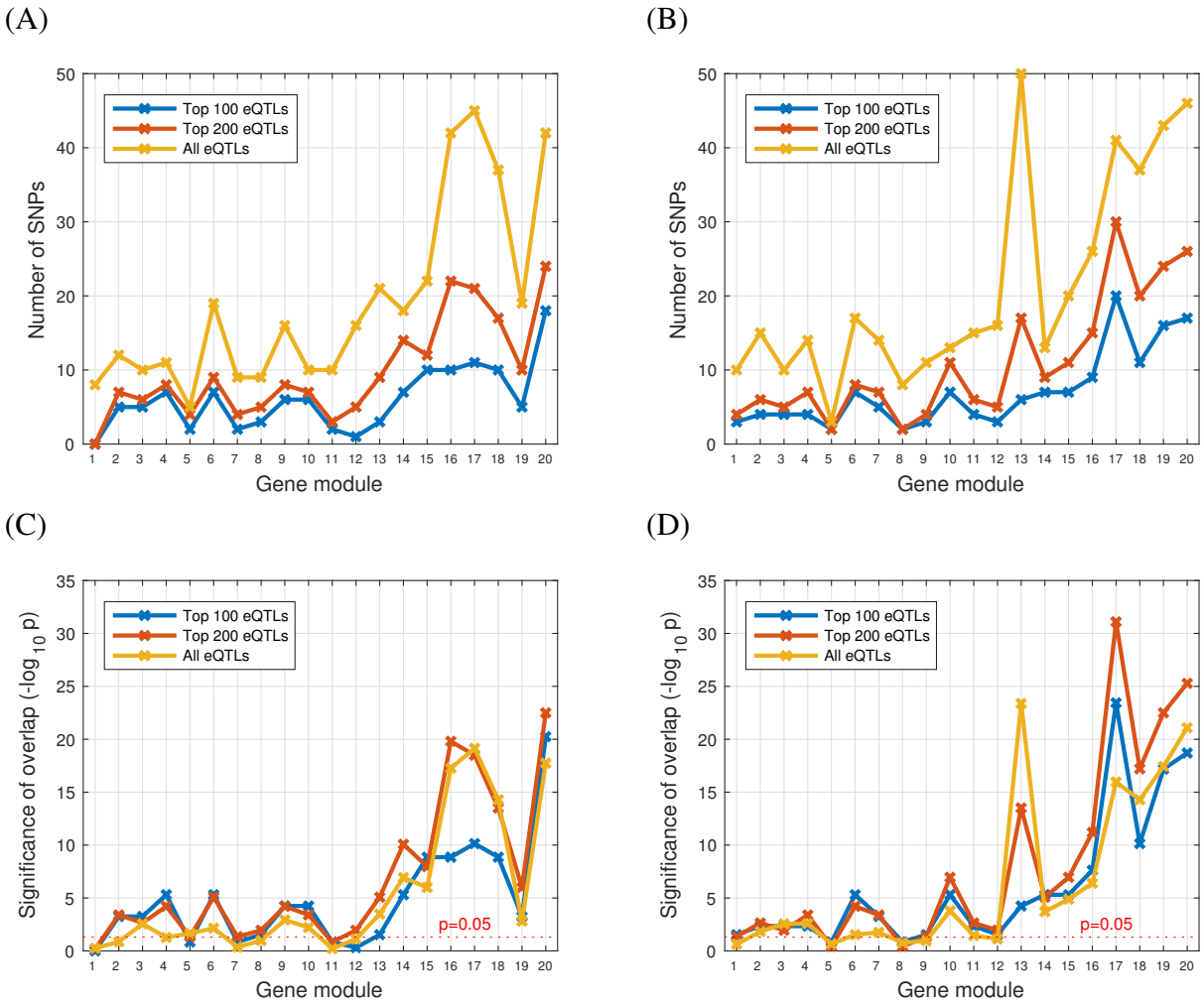


Figure 4.10: Overlap between SNPs perturbing phenotype network and SNPs perturbing gene network. For each gene module and each group of lung and blood phenotypes, we found the overlap between the top 200 SNPs perturbing the phenotype subnetwork and each of the top 100, 200, and all eQTLs perturbing the gene module. The number of SNPs in the overlap is shown for (A) lung phenotypes and (B) blood phenotypes. Statistical significance of the overlap is shown for (C) lung phenotypes and (D) blood phenotypes.

Furthermore, we found these overlaps are statistically significant for all of the immune modules, modules 13-20, but not for all of the other modules, and the most statistically significant overlaps were from the immune modules (Figures 4.10C and 4.10D). This suggests that eQTLs that perturb the modules that influence phenotypes are more likely to perturb phenotypes than eQTLs that perturb other gene modules.

4.3.4 The immune modules mediate SNP perturbation of phenotypes

To understand the molecular mechanisms that underlie the SNPs perturbing phenotypes beyond the simple overlap of SNPs perturbing the phenotype network and SNPs perturbing the gene network, we used the PerturbNet inference procedure to obtain the decomposition of the SNP effects on phenotypes $\mathbf{B_{xz}}$ into the component SNP effects on phenotypes $\mathbf{B}_{\mathbf{xz}}^{M_1}, \ldots, \mathbf{B}_{\mathbf{xz}}^{M_{20}}$ mediated by each of the 20 gene modules. We examined this decomposition for the 50 SNPs with the strongest effects on each group of lung and blood phenotypes (the cutoff for top 50 SNPs is shown as the green line at SNP effect size 0.04 for the lung phenotypes in Figure 4.9A and at SNP effect size 0.011 on blood phenotypes in Figure 4.9B).

For each set of 50 SNPs with the strongest perturbation effects on lung or blood phenotypes, nearly all of their effects on phenotypes were mediated by modules 12 through 20. The decomposition of the SNP effects on lung phenotypes (Figure 4.11A) into the 20 components (Figure 4.11C) shows that only the components for modules 12 through 20 contain non-zero SNP effects on the lung phenotypes, except for module 6, which mediates the effects of SNP rs1008932. We further summarized the component SNP effects by summing across all lung phenotypes for each SNP ($\sum_{j \in \text{Lung}} |[\mathbf{B}_{\mathbf{xz}}^{M}]_{i,j}|$ for module M and SNP i; Figure 4.11B). In Figure 4.11C, for 45 out of the 50 SNPs the SNP effect on the lung phenotypes is mediated by a single module from modules 12-20. For the other 5 SNPs, although their effects on phenotypes were mediated by two or three modules, the module with the strongest mediator effect had effect size at least 5 times as large as the other modules. Although only 20 SNPs overlapped between the two sets of top 50 SNPs for lung and blood phenotypes, the SNP effects on blood phenotypes were also mediated by modules 12-20 (Figure 4.12). This indicates that modules 12-20 can potentially explain the molecular mechanisms behind the SNP perturbations of asthma phenotypes.

The effect of each SNP on the lung phenotypes is mediated by a single module for 45 out of these 50 SNPs. For the 5 SNPs with effects mediated by multiple modules, the primary module has effect size at least 5x as large as the minor mediating modules, with all secondary mediating modules having effects sizes ¡0.01. Furthermore, aside from rs1008932 which is mediated through module 6, all SNPs are mediated through modules 12-20, and only modules 12-20 have any non-zero effects > 0.05 for these top 50 lung SNPs. These observations provide evidence that modules 12-20 provide insights on the molecular mechanisms behind SNPs influencing phenotypes.

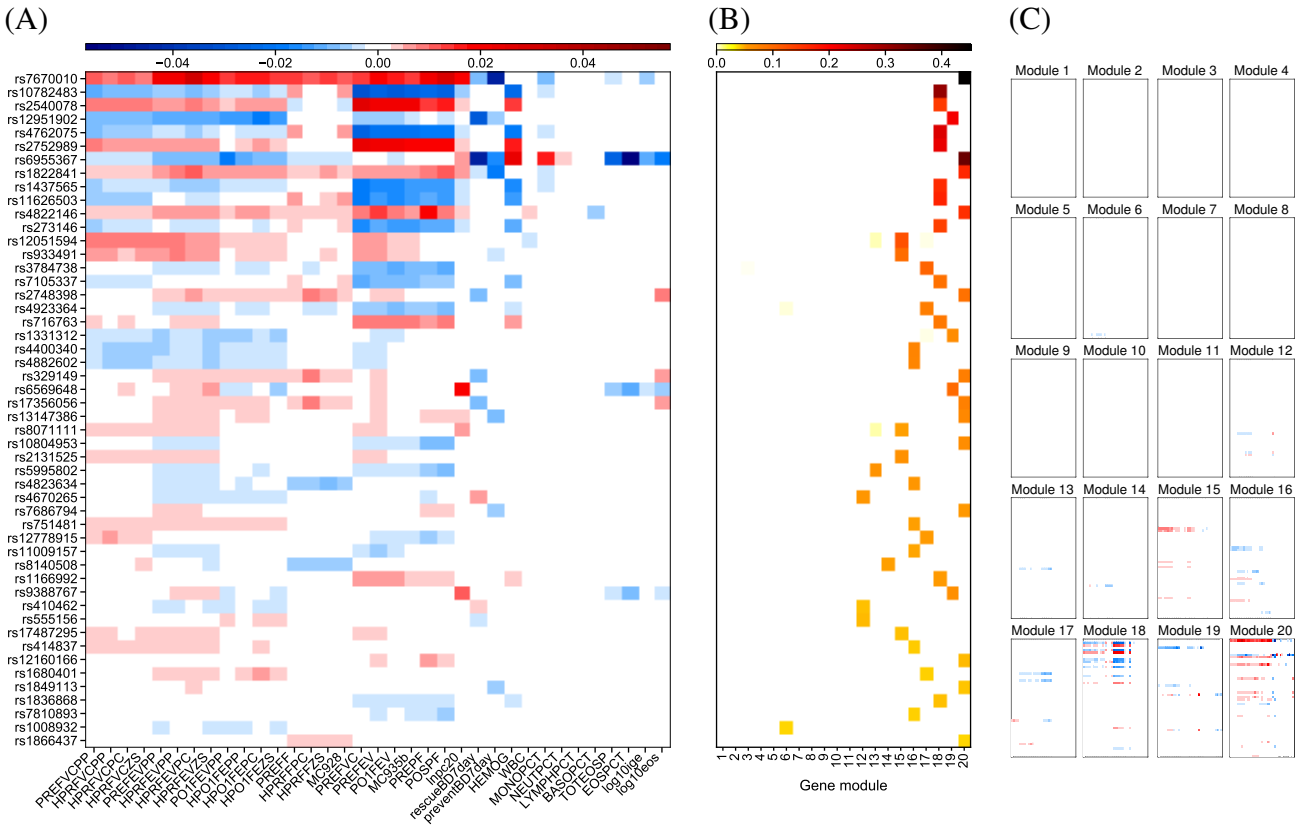


Figure 4.11: Top 50 SNPs perturbing lung phenotypes and their perturbations effects on phenotypes mediated by gene modules. For the top 50 SNPs perturbing lung phenotypes, we show (A) their effect sizes on phenotypes $\mathbf{B_{xz}}$ and (C) the decomposition of $\mathbf{B_{xz}}$ into component effects $\mathbf{B}_{\mathbf{xz}}^{M_1}, \ldots, \mathbf{B}_{\mathbf{xz}}^{M_{20}}$ mediated by each of the 20 gene modules. The sum over all component effects in Panel (B) is equal to the overall effects in Panel (A). (B) We summarize each component SNP effect $\mathbf{B}_{\mathbf{xz}}^{m}$ for module m in Panel (C) as a row-wise sum of $\mathbf{B}_{\mathbf{xz}}^{m}$, shown as the mth column in the figure. The SNPs are ordered according to their overall effect sizes on the lung phenotypes.

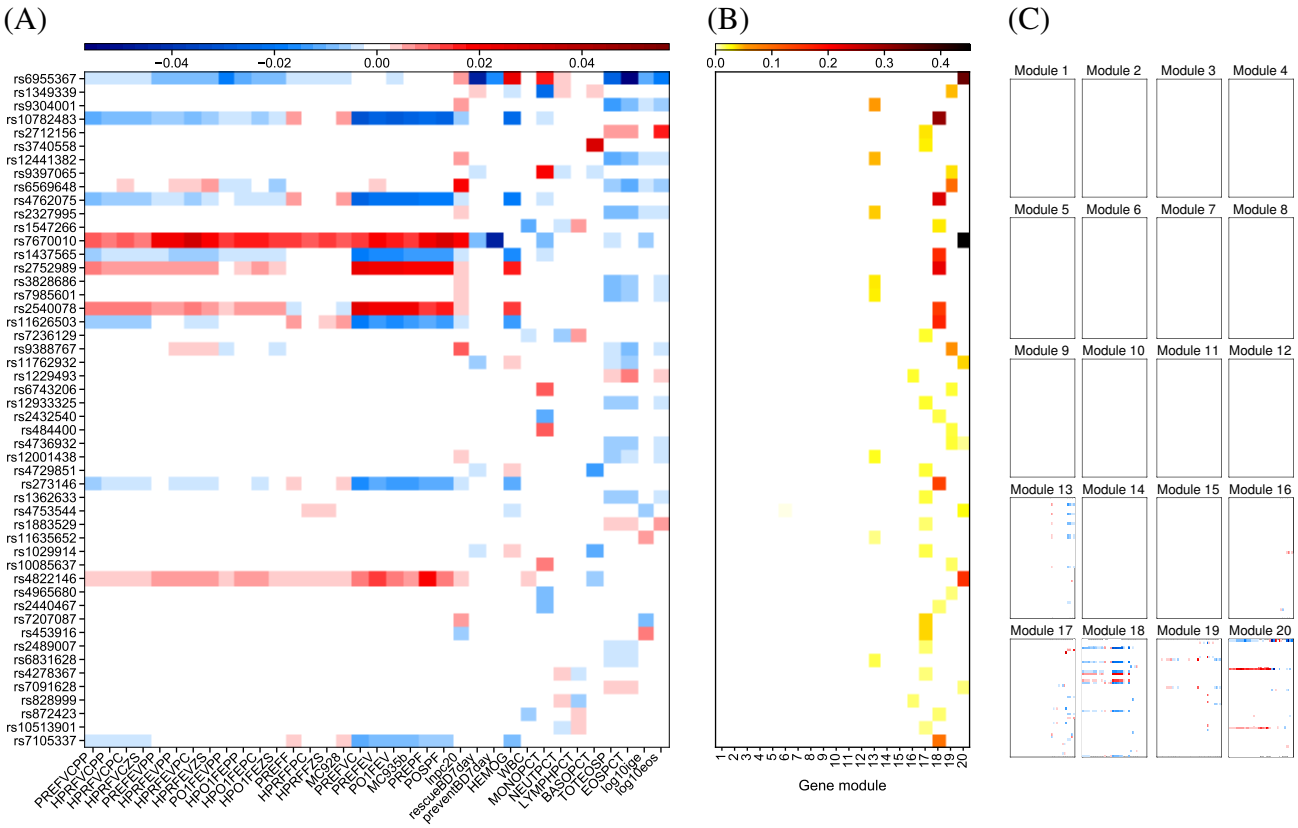


Figure 4.12: Top 50 SNPs perturbing blood phenotypes and their perturbations effects on phenotypes mediated by gene modules. For the top 50 SNPs perturbing lung phenotypes, we show (A) their effect sizes on phenotypes $\mathbf{B_{xz}}$ and (C) the decomposition of $\mathbf{B_{xz}}$ into component effects $\mathbf{B}_{\mathbf{xz}}^{M_1}, \ldots, \mathbf{B}_{\mathbf{xz}}^{M_{20}}$ mediated by each of the 20 gene modules. The sum over all component effects in Panel (B) is equal to the overall effects in Panel (A). (B) We summarize each component SNP effect $\mathbf{B}_{\mathbf{xz}}^{m}$ for module m in Panel (C) as a row-wise sum of $\mathbf{B}_{\mathbf{xz}}^{m}$, shown as the mth column in the figure. The SNPs are ordered according to their overall effect sizes on the blood phenotypes.

4.3.5 Module 13 explains the molecular mechanism of the previously known association between SNP rs12441382 and asthma susceptibility

We examined module 13 that mediates the perturbation of asthma traits by SNP rs12441382, This SNP is within the top 500 SNPs of our model, and the 41st largest among the 99 SNPs in Figure 4.6. SNP rs12441382, located in genomic region 15q21.2, is 17kb from rs1841128, the closest SNP listed in the National Human Genome Research Institute (NHGRI) GWAS catalog [12] as being associated with a lung function trait FVC in asthma patients though little is known about the underlying molecular mechanism [56]. This pair of SNPs has normalized LD coefficient $D' = 1.0$ for the CEU population in the 1000 Genomes Project [1]. PerturbNet found SNP rs12441382 directly perturbs only two genes in module 13, *ATF3* as the top perturbant and *EGR2* as the second-to-the-top perturbant, both of which have been previously linked with allergic asthma. *ATF3* is a known negative regulator of allergic asthma and of IFN-gamma, a cytokine involved in asthmatic inflammation [38, 76], and was recently proposed to be a hub of the cellular adaptive-response network, playing a key role in immune diseases [42]. The perturbation of *ATF3* by this SNP is also reflected in statistically significant association in univariate regression analysis (FDR $q = 6.851 \times 10^{-05}$). *EGR2* has been linked to migration of CD4+ T cells to the lung, and to blood eosinophil levels in asthma [9, 88, 98]. The effect of SNP rs12441382 on the traits is also mediated by other indirectly perturbed genes near *ATF3* and *EGR2* in the network: *C2*, *SERPING1*, *ZG16B*, and *CEACAM3*. *C2* has not been directly associated with asthma, but has been linked to immune balance and auto-immune diseases [52, 54, 55]. *ZG16B* has been linked with severe asthma [63]. *SERPING1* has been found to play an important role in hereditary angiodema [16]. *CEACAM3*, though not directly associated with asthma, is solely expressed in phagocytes and plays a role in immune response [14]. Our network analysis suggests that the 15q21.2 locus underlies asthmatic inflammation via *ATF3*-centered pathway, providing new insights into the molecular characterization of the previously known GWAS SNP with little known functional role.

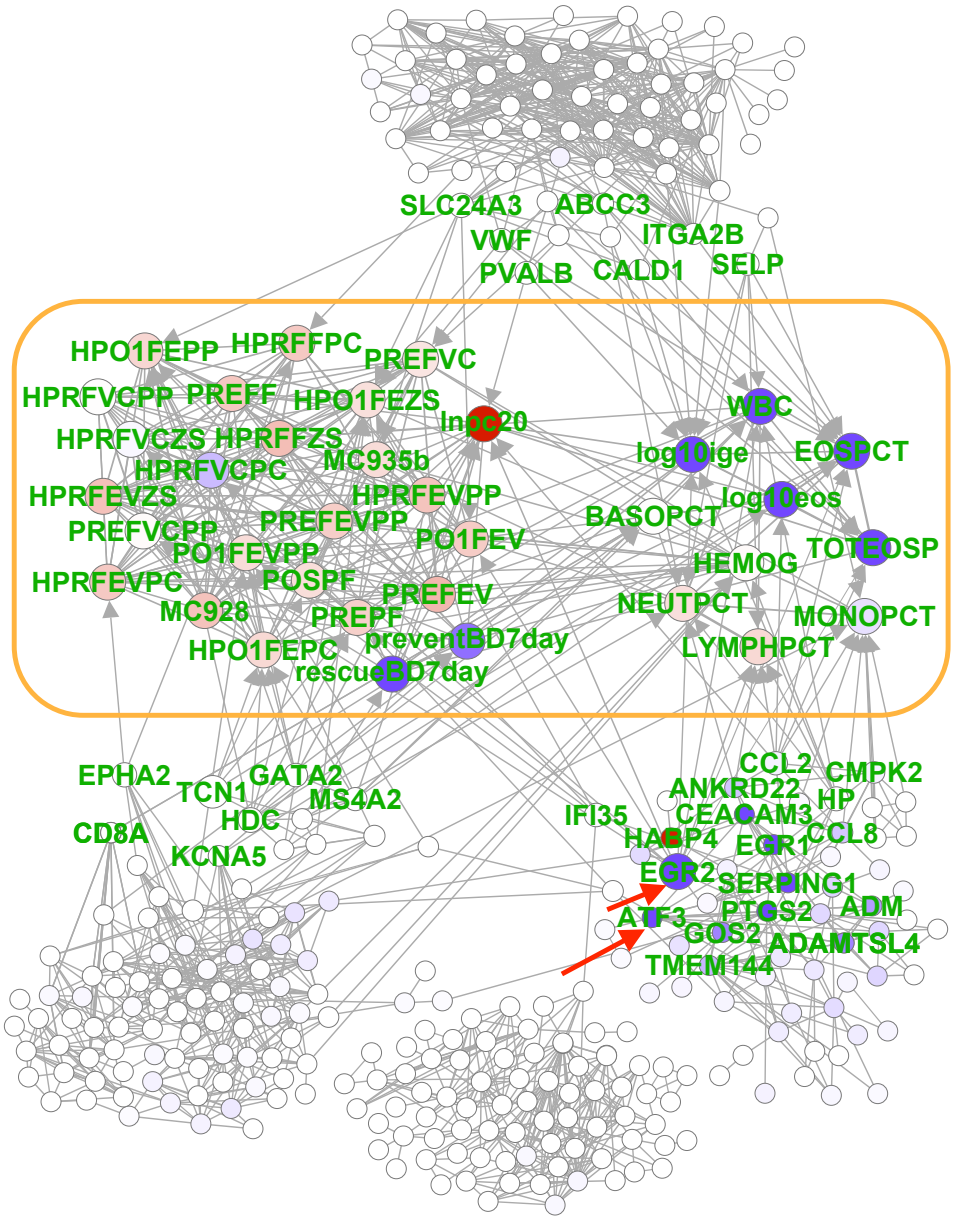


Figure 4.13: The gene network for module 13, its influence on asthma phenotypes, and its perturbation by SNP rs12441382. The asthma phenotype network $\mathbf{\Lambda_{zz}}$ in our estimated model is shown in the green box and the gene network $\mathbf{\Lambda_{yy}}$ for module 13 is shown outside of the green box. The edges across the two networks correspond to direct influence of expression levels on phenotypes $\mathbf{\Theta_{yz}}$. The two genes (*ATF3* and *EGR2*) whose expression is directly perturbed by SNP rs12441382 are labeled with arrows, colored red to indicate positive eQTL effects. Node colors depict the indirect effects of this eQTL on gene expression levels $\mathbf{B_{xy}}$ and phenotypes $\mathbf{B_{xz}}$, with red for up-regulation and blue for down-regulation. Node size of genes depicts the component of the eQTL effects on phenotypes mediated by the given gene, the row of $\mathbf{B^m_{xz}}$ for SNP rs12441382 and for gene m in module 13 summed across all phenotypes.

4.3.6 Module 13 mediates the effect of genetic variation on blood traits

We analyzed the mediatory roles of gene modules using their scoring functions (Eq. 3.30), separately for lung and blood trait groups. It was previously seen in Figure 4.10 that blood trait SNPs had greater overlap with module 13 eQTLs, while overlaps for modules 16-20 were even between blood and lung traits. We now check whether, in addition to this colocalization between module 13 eQTLs and blood traits, whether module 13 genes also play a mediating role primarily for blood traits. The scores of gene modules for both trait groups are shown in Figure 4.14. Module 13 plays a leading role in propagating genetic variation effects onto blood traits (Figure 4.14B) rather than lung phenotypes (Figure 4.14A).

4.3.7 Significance analysis of SNP effect sizes shows little to no confounding from population stratification

We estimated the null distribution of effect sizes for PerturbNet on our asthma dataset using a permutation test with 500 random permutations. Following the conservative approach in genetical genomics [11] to handle correlated gene expressions and traits, in each permutation we randomly shuffle separately each of the genotypes while leaving the gene expression and phenotype datasets intact. After creating each of the 500 shuffled datasets, we ran PerturbNet using the same regularization parameters used on the real data, which also produces conservative p-values. The distribution of p-values are shown as QQ-plots, both for SNP-gene (Figure 4.15) and SNP-trait associations (Figure 4.16). The distribution of p-values is uniform, except at extremely small values. This indicates that SNP effect sizes are not distorted by confounding, and that substantial population stratification was not present in our study. After multiple-testing correction, our study found 161 significant SNP-gene associations with FDR < 0.05. For SNP-trait associations, none of the coefficients in $\mathbf{B_{xz}}$, nor any overall SNP effects across all traits was found to be significant at FDR < 0.05. However, aggregating SNP-trait effects separately for lung and blood traits revealed two significant associations for lung traits and 17 for blood traits at FDR < 0.05. Our significance testing procedure is conservative, and a less overly-conservative approach to getting SNP effect p-values may reveal more statistically-significant non-zero effect sizes.

4.3.8 Local SNP perturbations and direct SNP perturbations have larger eQTL effect sizes

Next, we examined the effect sizes of eQTLs identified by our method as perturbing the gene network. Since the indirect perturbations represent the direct perturbation effects propagating to other parts of the network, the direct SNP perturbation effects in $\mathbf{\Theta_{xy}}$ may be stronger than the indirect SNP perturbation effects in $\mathbf{B_{xy}}$. We compared the distribution of SNP perturbation effect sizes between $\mathbf{\Theta_{xy}}$ and $\mathbf{B_{xy}}$. We obtained the distribution of perturbation effect sizes as a histogram of non-zero elements in $\mathbf{\Theta_{xy}}$ and in $\mathbf{B_{xy}}$, after normalizing the frequencies with the total number of non-zero elements (99,055 in $\mathbf{\Theta_{xy}}$ and 326,455 in $\mathbf{B_{xy}}$ after thresholding $\mathbf{B_{xy}}$ at 0.001 to rule out the perturbations with small effect sizes). The histograms in Fig 4.17A confirm our hypothesis that overall, direct SNP perturbation effects are stronger than indirect perturbation effects (rank sum test p-value = 0 within machine precision; 1,656 and 12,049 elements in $\mathbf{\Theta_{xy}}$

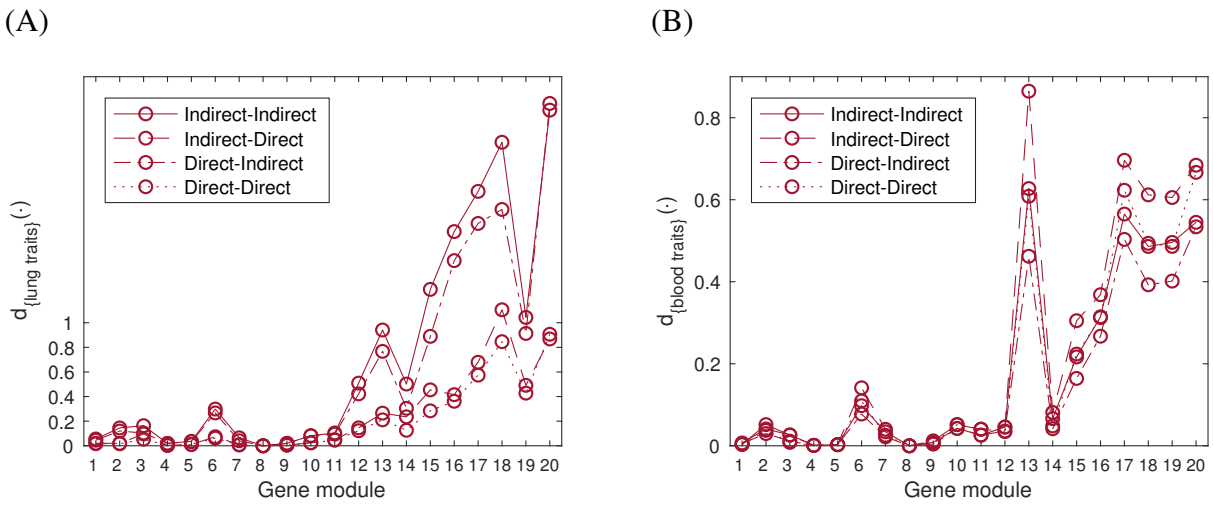


Figure 4.14: Scores of gene modules 1-20 on (A) lung phenotypes and (B) blood phenotypes, for mediating all genetic variation on these two trait groups. Scores are shown for the flow of all combinations of direct and indirect effects onto and from the gene modules.

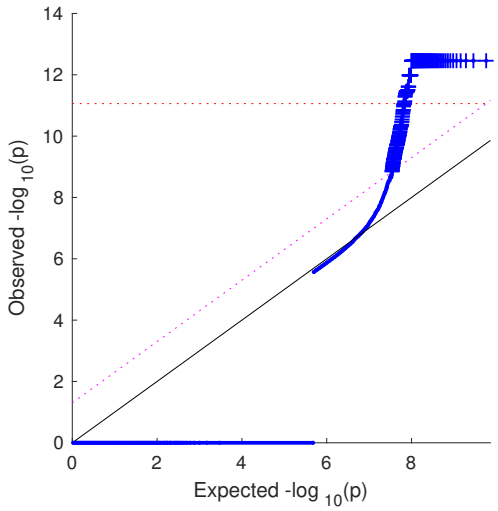


Figure 4.15: QQ-plot for p-values of SNP-gene perturbation effect sizes. The distribution of effect size significances are shown for all direct effects in $\mathbf{\Theta_{xy}}$. The dotted red line is the FWER threshold at 0.05, while the dotted pink line is the FDR threshold at 0.05.

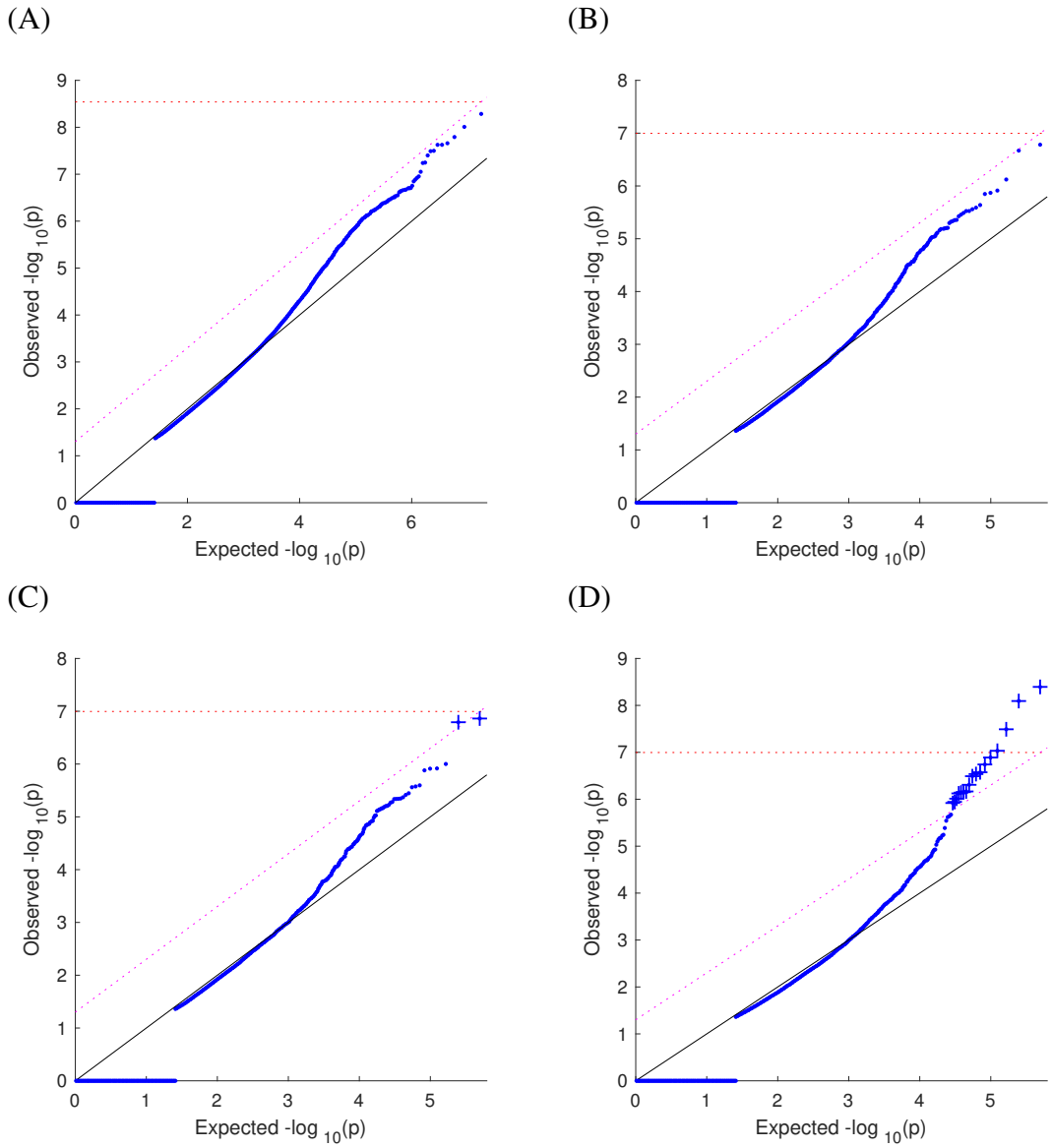


Figure 4.16: QQ-plots for p-values of SNP-trait effect sizes. The distribution of effect size significances are shown for (A) all coefficients in $\mathbf{B_{xz}}$, (B) overall effects of SNPs across all traits from $\mathbf{B_{xz}}$, (C) overall effects of SNPs across lung traits from $\mathbf{B_{xz}}$, and (D) overall effects of SNPs across blood traits from $\mathbf{B_{xz}}$. The dotted red line is the FWER threshold at 0.05, while the dotted pink line is the FDR threshold at 0.05.

with effect sizes greater than 0.1 and 0.05, and 0 and 9 elements in $\mathbf{B_{xy}}$ in the corresponding ranges).

Among the direct perturbations in $\mathbf{\Theta_{xy}}$, the perturbations with stronger effects may be more likely to be local eQTLs. We compared the distributions of the effect sizes of local and distal eQTLs in $\mathbf{\Theta_{xy}}$, where the local eQTLs are defined as eQTLs located within 50kb from the coding regions of perturbed genes, after normalizing the frequencies with the total number of eQTLs in each category (279 for local eQTLs and 98,776 for distal eQTLs). The histograms in Fig 4.17B show that overall, local eQTLs have larger effect sizes than distal eQTLs (rank sum test p-value=2.0930×10^{-43}), even though the number of local eQTLs with large effect sizes was smaller than that of distal eQTLs (43 local eQTLs and 163 distal eQTLs with effect size>0.2, 11 local eQTLs and 42 distal eQTLs with effect size>0.4). Out of the 279 local eQTLs above, 65 eQTLs were found to affect the expression of the 20 gene modules in our gene network $\mathbf{\Lambda_{yy}}$ but belong to one of the 20 modules in Table 4.2 These module-perturbing eQTLs are shown in Table 4.3). Overall, our results indicate that strong perturbation effects tend to arise from local eQTLs.

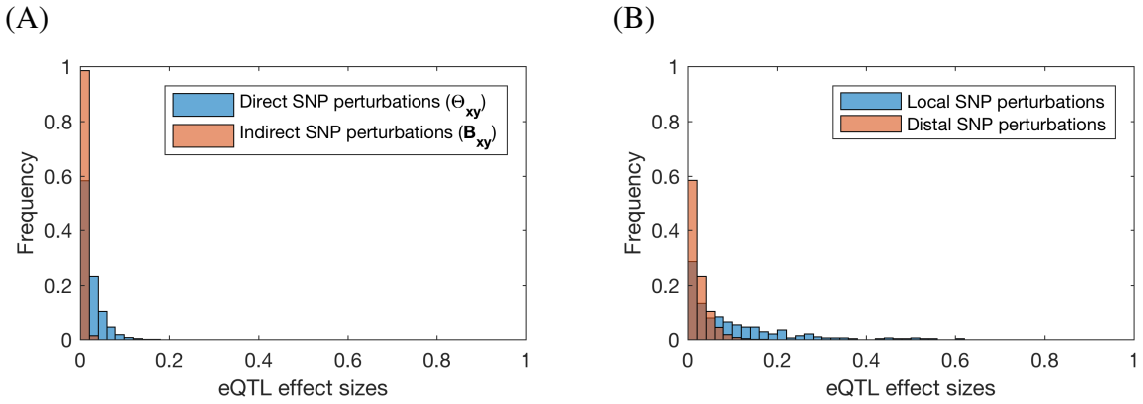


Figure 4.17: SNP perturbations of the asthma gene network. (A) Comparison of the distributions of the effect sizes for direct and indirect SNP perturbations of the gene network in the sparse Gaussian chain graph model estimated from the asthma data. Histograms of the effect sizes in the estimated $\mathbf{\Theta_{xy}}$ and the inferred $\mathbf{B_{xy}}$ are shown after the frequencies were normalized by the total numbers of SNP perturbations in each perturbation type. (B) Comparison of the distributions of local and distal eQTL effect sizes among the direct perturbations in $\mathbf{\Theta_{xy}}$. The local eQTLs are defined as 50kb from the coding region of the gene for the corresponding expression trait.

4.3.9 *Trans* eQTLs are more likely to perturb gene modules than *cis* eQTLs

We find that the *trans* eQTLs discovered by PerturbNet tend to effect gene modules, whereas *cis* eQTLs tend to perturb individual singleton genes unconnected to other genes in the network. Out of the direct eQTLs found in $\mathbf{\Theta_{xy}}$ (Table 4.4), roughly a fourth of the *cis* eQTLs directly perturb genes in modules 1-20, with the majority going to singleton genes. Meanwhile, almost half of the *trans* eQTLs directly perturb genes within the gene modules. This association is even more stark among indirect eQTLs recovered in $\mathbf{B_{xy}}$, with less than a third of *cis* eQTLs indirectly perturbing

RS id	Region	Gene	Module	Found in Murphy?
rs12951902	17p13.3	C17orf97	19	
rs7556463	1q24.2	ATP1B1	18	
rs12568757	1q21.3	CTSK	6	
rs13166314	5p15.31	MTRR	1	
rs2495396	1q21.3	TDRKH	6	
rs12536500	7p12.1	GRB10	2	Yes
rs10797432	1p36.32	TNFRSF14	11	
rs1040404	1q24.2	TIPRL	16	Yes
rs4729851	7q22.1	POLR2J3	17	
rs2432540	16q12.2	AMFR	18	
rs13102358	4p16.3	ZNF721	1	
rs1029914	7q22.1	POLR2J3	17	
rs9395049	6p21.1	SUPT3H	16	
rs1680401	3p14.1	SLC25A26	17	
rs2440467	16q12.2	AMFR	18	
rs2712156	4p11	OCIAD2	17	
rs152164	16q22.1	NAE1	19	
rs1781423	1q21.3	TDRKH	6	
rs7810893	7p21.3	ARL4A	16	Yes
rs9393708	6p22.2	BTN3A2	18	
rs1885498	9p24.1	RLN2	16	
rs4387211	1q32.1	DDX59	16	
rs1262184	6p25.2	SERPINB1	14	
rs4974608	4p16.3	CTBP1	18	
rs7132019	12p13.31	LRRC23	16	
rs2281014	1q24.2	TIPRL	16	
rs2568065	11p15.4	AKIP1	16	
rs1362633	16q12.1	HEATR3	17	
rs4925041	17p11.2	ALDH3A2	17	
rs17422760	6p21.1	SUPT3H	16	
rs12933325	16q12.1	HEATR3	17	
rs2025126	9p13.3	TPM2	19	
rs593713	18p11.21	MPPE1	16	Yes
rs7522061	1q23.1	FCRL3	18	Yes
rs10840160	11p15.4	AKIP1	16	
rs13406184	2p22.2	FEZ2	14	Yes
rs2038760	6p25.2	MYLK4	12	
rs10189344	2p22.2	FEZ2	14	
rs710415	12p13.31	LRRC23	16	Yes
rs7029205	9p22.1	HAUS6	3	
rs13216116	6p21.1	SUPT3H	16	Yes
rs2295797	9p13.3	TPM2	19	Yes
rs10797802	1q25.3	LAMC1	15	
rs8051216	16q12.1	HEATR3	17	Yes
rs9358945	6p22.2	BTN3A2	18	
rs7314535	12p13.31	LRRC23	16	
rs1978	6p22.2	BTN3A2	18	
rs649537	10p15.1	ANKRD16	18	Yes
rs3770833	2p22.2	FEZ2	14	
rs12315364	12p13.31	LRRC23	16	

Table 4.3: Top 50 non-singleton *cis* eQTLs, found by L_1-norm of row in $\mathbf{\Theta_{xy}}$. The right-most column indicates whether the SNP-gene mapping in the PerturbNet asthma model matches the eQTL mapping results from the earlier analysis on this same dataset [67].

Table 4.4: Association between whether $\mathbf{\Theta_{xy}}$ eQTLs are *cis* vs *trans* and whether the eQTLs perturb modules vs singleton genes. Based on various distance thresholds, eQTLs are classified as either *cis* or *trans*. We show the percentage of *cis* and *trans* eQTLs which go to singleton genes rather than gene modules 1-20, as well as the resulting odds ratios and p-values.

Distance (in kb)	Percent of *cis* eQTLs to modules	Percent of *trans* eQTLs to modules	Odds ratio	p-value
10	24.62%	47.93%	2.8	1.03e-24
20	25.82%	47.99%	2.7	1.21e-34
50	25.28%	48.07%	2.7	1.57e-51
100	25.97%	48.10%	2.6	3.66e-56
200	27.29%	48.11%	2.5	2.46e-54

Table 4.5: Association between whether $\mathbf{B_{xy}}$ eQTLs are *cis* vs *trans* and whether the eQTLs perturb modules vs singleton genes. Based on various distance thresholds, eQTLs are classified as either *cis* or *trans*. We show the percentage of *cis* and *trans* eQTLs which go to singleton genes rather than gene modules 1-20, as well as the resulting odds ratios and p-values.

Distance (in kb)	Percent of *cis* eQTLs to modules	Percent of *trans* eQTLs to modules	Odds ratio	p-value
10	26.74%	96.45%	74.4	4.94e-324
20	28.02%	96.46%	70.1	2.96e-323
50	28.07%	96.48%	70.2	3.95e-323
100	29.96%	96.49%	64.2	4.94e-323
200	33.47%	96.49%	54.7	1.48e-323

genes in modules, but more than 95% of *trans* eQTLs indirectly perturbing genes within gene modules. This suggest that PerturbNet's approach, which takes the gene network into account when finding eQTLs, is better at recovering *trans* eQTLs, which would otherwise be hard to find without network information, and that these *trans* eQTLs are especially important to correctly model the genetic regulation of modules in the regulatory network.

4.4 Comparison with other methods

We compared PerturbNet with eCAVIAR, PrediXcan, and two-layer Lasso on the CAMP data by evaluating top 500 SNPs identified by each method as perturbing both gene expression and asthma traits based on the evidence provided by external sources: DNase-seq hypersensitivity sites from the ENCODE B-lymphoblastoid cell-line, the RegulomeDB annotations, and the NHGRI GWAS database. We also compared our method with the two-layer Lasso both quantitatively by assessing the predictive power of different methods and qualitatively by visual inspection of the estimated parameters.

We compared PerturbNet with PrediXcan, eCAVIAR, and two-layer Lasso on the asthma data. We applied PerturbNet with semi-supervised learning to all samples, including the sam-

ples with only genotype and clinical trait data. With PrediXcan, we mimiced semi-supervised learning by fitting elastic net to the samples with both expression and genotype data, imputing the missing expression data, and using all samples to perform association tests between the imputed gene expression values and traits. To obtain top 500 trait-associated SNPs, we scored each SNP by first scoring SNP-trait pairs as in the simulation study above and summing these scores across all traits. For eCAVIAR, GWAS SNPs were found using all samples, while eQTLs were found using only the samples with expression data. Then, top 500 asthma-associated SNPs were found by summarizing CLPP scores of all triplets into scores of each SNP by taking maximum over all mediator genes and summing over all traits. We applied the two-layer Lasso only to the samples with all data and obtained top SNPs perturbing asthma traits using the same strategy as in PerturbNet.

4.4.1 Comparison using gene expression and clinical data simulated from CAMP genotype data

Using simulated data and patient cohort data from the Childhood Asthma Management Program (CAMP) [20, 21, 67], we benchmarked PerturbNet against eCAVIAR [46], PrediXcan [33], and two-layer Lasso. We demonstrate PerturbNet model and inference methods provide higher sensitivities for detecting genetic variants that affect clinical traits and for identifying genes that mediate these genetic effects on traits, and offer deeper insight on the molecular mechanism underlying the SNP perturbations of disease phenotypes.

Using data simulated from real genotypes in the CAMP study, we evaluated PerturbNet on the accuracy of detecting trait-perturbing SNPs and mediator genes and on the accuracy of gene network recovery. To assess PerturbNet on the recovery of trait-associated SNPs and mediator genes, we assumed two types of ground-truth models: a two-layer sCGGM as in PerturbNet to mimic the SNP perturbations of clinical traits modulated by gene networks and a two-layer linear regression model to mirror the assumptions of mediator genes acting independently of other genes as in the existing methods. To assess the accuracy of gene network learning, we again used two types of ground-truth models: a two-layer sCGGM with a gene network perturbed by SNPs and a two-layer Gaussian graphical model (GGM) with no perturbations of networks. For each type of models, we obtained sensitivities at different false discovery rates (FDRs) averaged over 10 simulated datasets, each with $5{,}000$ gene expression levels and 100 clinical trait values generated from $10{,}000$ SNPs from chromosome 1 of 540 non-Hispanic Caucasians from the CAMP study.

PerturbNet had the highest sensitivities across all FDRs for detecting SNPs affecting traits (**Fig. 4.18A**) and identifying mediator genes (**Figs. 4.18B-C**), regardless of the model types used to simulate data. This suggests whether a gene is a mediator acting independently or cooperating with other genes in the network, PerturbNet can correctly identify SNP effects on traits and their mediator genes. Unlike other methods, PerturbNet has the ability to distinguish between direct and indirect perturbation of a network, which further enables a categorization of the role of the mediator gene into four possible combinations of direct vs. indirect perturbations the mediator gene receives from SNPs and passes onto clinical trait network. PerturbNet identified these categories for each mediating gene with high accuracy (**Fig. 4.18B**). On gene network

recovery, PerturbNet had higher accuracy on data simulated with SNP perturbations and even on data simulated without SNP perturbations the accuracy of PerturbNet did not suffer significantly (**Fig. 4.18D**). Our results suggest that PerturbNet achieves higher accuracy and provides richer information on the regulatory roles of SNPs and mediator genes.

4.4.2 Comparison with SNPs in the NHGRI GWAS catalog

We compared the asthma SNPs found by each method with the previously reported asthma-associated SNPs in the NHGRI GWAS catalog [12]. Out of 557 SNPs in the NHGRI GWAS catalog for asthma, we examined how many of these SNPs are within 10, 20, 50, and 100kb of top k SNPs from each method, k ranging from 1 to 500. To assess the statistical significance of the overlap between two sets of SNPs, we performed a permutation test: for top k SNPs from the given method, we generated 10,000 random sets of k SNPs from the SNPs employed in the analysis to find the distribution of overlaps under the null hypothesis and reported the overlaps with p-value < 0.05.

In comparison with the 557 SNPs annotated as asthma-associated in the NHGRI GWAS catalog, 11 SNPs were within 20kb of the 500 SNPs found by PerturbNet (with enrichment p-value = 0.0575), while only 2 NHGRI SNPs overlapped for eCAVIAR (p-value = 0.950), 6 for two-layer Lasso (p-value = 0.408), and 3 SNPs for PrediXcan (p-value = 0.861). Statistically significant enrichment held up to the top $K = 400$ SNPs for PerturbNet, where $K = 100, \ldots, 500$, but for none of the top SNP sets from eCAVIAR, Lasso, or PrediXcan.

For the 32 NHGRI SNPs overlapping at a distance of 100kb of the top PerturbNet SNPs, we examined the 12 NHGRI SNPs which were primarily mediated through the 20 gene modules, as opposed to singleton genes. For each of these NHGRI SNPs, we found the PerturbNet SNPs within 100kb and recorded the gene modules which they perturbed according to our model. These mediating modules for each of the NHGRI SNPs are shown in Fig. 4.20. The NHGRI SNPs accounted for by our model, and not primarily mediated by singleton genes, are all mediated by modules 13-20.

4.4.3 Functional annotations of asthma SNPs

We examined the functional annotations of the top 500 SNPs identified as asthma-associated by each method, using two external sources of annotations: the ENCODE DNase hypersensitivity sites (GSM1008572) from GM12878 B-lymphoblastoid cell line [28], a blood cell type as in CD4+ T lymphocytes in the CAMP study; and RegulomeDB [10] integrating ENCODE and other sources of annotations for a diverse set of cell types. For comparison with the DNase hypersensitivity sites, we used the UES tool [44] to identify the sites enriched in top asthma SNPs from each method and to compute the enrichment p-values for these overlaps with 100 Monte Carlo simulations. We used RegulomeDB [10] to score the top 500 asthma SNPs from each method into six categories: category 1 for overlaps with previously reported eQTLs with additional functional annotation on TF binding; categories 2-5 for overlaps with TF binding sites in ChIP-seq, DNase-seq, and motif hits, where lower scores indicate stronger evidence for being functional; and category 6 for little evidence of being functional.

PerturbNet SNPs had the largest overlap with the DNase-seq hypersensitivity sites from the ENCODE B-lymphoblastoid cell-line and PerturbNet along with two-layer Lasso were the only two methods with statistically significant overlaps across top SNPs of different sizes, indicating that PerturbNet SNPs were supported with the strongest evidence that they are likely to play a role in gene regulation (Fig. 4.21A). Compared against the functional annotation of variants in RegulomeDB, PerturbNet SNPs again had the largest overlap, especially with SNPs in the RegulomeDB category of "1", which corresponds to the strongest evidence of being located in the functional region and includes SNPs previously identified as eQTLs (Fig. 4.21B).

4.4.4 Comparison of prediction accuracy

We use the estimated PerturbNet model and the results of probabilistic inference on this model to make predictions on previously unseen patients. From each of the two component sparse CGGMs in our model, we make the following predictions:

- $\hat{\mathbf{y}}_{\text{new}}|\mathbf{x}_{\text{new}} = \mathbf{B}_{\mathbf{xy}}^T\mathbf{x}_{\text{new}}$ for predicting the expression levels $\hat{\mathbf{y}}_{\text{new}}$ given the genotypes $\mathbf{x}_{\text{new}}$ of a new patient
- $\hat{\mathbf{z}}_{\text{new}}|\mathbf{y}_{\text{new}} = \mathbf{B}_{\mathbf{yz}}^T\mathbf{y}_{\text{new}}$ for predicting the phenotypes $\hat{\mathbf{z}}_{\text{new}}$ given the expression levels $\mathbf{y}_{\text{new}}$ of a new patient

From the full sparse Gaussian chain graph model, we make the following predictions:

- $\hat{\mathbf{z}}_{\text{new}}|\mathbf{x}_{\text{new}} = \mathbf{B}_{\mathbf{xz}}^T\mathbf{x}_{\text{new}}$ for predicting the phenotypes $\hat{\mathbf{z}}_{\text{new}}$ given the genotypes $\mathbf{x}_{\text{new}}$ of a new patient
- $\hat{\mathbf{y}}_{\text{new}}|\mathbf{x}_{\text{new}}, \mathbf{z}_{\text{new}} = -\Big(\mathbf{z}_{\text{new}}^T\mathbf{\Theta}_{\mathbf{yz}}^T + \mathbf{x}_{\text{new}}^T\mathbf{\Theta}_{\mathbf{xy}}\Big)\mathbf{\Lambda}_{\mathbf{y}|\mathbf{x},\mathbf{z}}^{-1}$ for predicting the gene expression levels $\hat{\mathbf{y}}_{\text{new}}$ given the genotypes $\mathbf{x}_{\text{new}}$ and the phenotypes $\mathbf{x}_{\text{new}}$ of a new patient

Two-layer Lasso for comparison with PerturbNet We compare the performance of our method with that of Lasso [90, 97], a popular statistical method based on linear regression models for studying the associations among SNPs, expression measurements, and phenotypes. We begin by setting up a two-layer multivariate regression model for genotypes $\mathbf{x} \in \{0, 1, 2\}^p$, expression measurements $\mathbf{y} \in \mathbb{R}^q$, and phenotypes $\mathbf{z} \in \mathbb{R}^r$ as follows:

$$\begin{aligned}\mathbf{y} &= \mathbf{A}_{\mathbf{xy}}^T\mathbf{x} + \epsilon_{\mathbf{y}}, \quad \epsilon_{\mathbf{y}} \sim \mathcal{N}(\mathbf{0}_q, \mathbf{\Omega}_{\mathbf{y}}),\\ \mathbf{z} &= \mathbf{A}_{\mathbf{yz}}^T\mathbf{y} + \epsilon_{\mathbf{z}}, \quad \epsilon_{\mathbf{z}} \sim \mathcal{N}(\mathbf{0}_r, \mathbf{\Omega}_{\mathbf{z}}),\end{aligned}$$

where $\mathbf{A}_{\mathbf{xy}} \in \mathbb{R}^{p\times q}$ and $\mathbf{A}_{\mathbf{yz}} \in \mathbb{R}^{q\times r}$ are regression coefficients, $\epsilon_{\mathbf{y}} \in \mathbb{R}^q$ and $\epsilon_{\mathbf{z}} \in \mathbb{R}^r$ are noise distributed with zero means and diagonal covariances $\mathbf{\Omega}_{\mathbf{y}} = \text{diag}(\sigma_{\mathbf{y}_1}^2, \ldots, \sigma_{\mathbf{y}_q}^2)$ and $\mathbf{\Omega}_{\mathbf{z}} = \text{diag}(\sigma_{\mathbf{z}_1}^2, \ldots, \sigma_{\mathbf{z}_q}^2)$.

Given genotype data $\mathbf{X} \in \{0, 1, 2\}^{n\times p}$ for n samples and p SNPs, expression data $\mathbf{Y} \in \mathbb{R}^{n\times q}$ for q genes, and phenotype data $\mathbf{Z} \in \mathbb{R}^{n\times r}$ for r phenotypes, we obtain a Lasso estimate of the regression coefficients by minimizing L_1-regularized negative log-likelihood as follows:

$$\begin{aligned}&\min_{\mathbf{A}_{\mathbf{xy}}} \frac{1}{n}\,\text{tr}\left(\big(\mathbf{Y} - \mathbf{X}^T\mathbf{A}_{\mathbf{xy}}\big)\big(\mathbf{Y} - \mathbf{X}^T\mathbf{A}_{\mathbf{xy}}\big)^T\right) + \gamma_1||\mathbf{A}_{\mathbf{xy}}||_1,\\ &\min_{\mathbf{A}_{\mathbf{yz}}} \frac{1}{n}\,\text{tr}\left(\big(\mathbf{Z} - \mathbf{Y}^T\mathbf{A}_{\mathbf{yz}}\big)\big(\mathbf{Z} - \mathbf{Y}^T\mathbf{A}_{\mathbf{yz}}\big)\right)^T + \gamma_2||\mathbf{A}_{\mathbf{yz}}||_1.\end{aligned}$$

Using the Lasso estimate of the regression coefficients $\mathbf{A_{xy}}$ and $\mathbf{A_{yz}}$, we compute predictions for this model analogously to our sparse Gaussian chain graph model.

- $\hat{\mathbf{y}}_{\text{new}}|\mathbf{x}_{\text{new}} = \mathbf{A}_{\mathbf{xy}}^T\mathbf{x}_{\text{new}}$
- $\hat{\mathbf{z}}_{\text{new}}|\mathbf{y}_{\text{new}} = \mathbf{A}_{\mathbf{yz}}^T\mathbf{y}_{\text{new}}$
- $\hat{\mathbf{z}}_{\text{new}}|\mathbf{x}_{\text{new}} = \mathbf{A}_{\mathbf{xz}}^T\mathbf{z}_{\text{new}}$, where $\mathbf{A_{xz}} = \mathbf{A_{xy}}\mathbf{A_{yz}}$.
- $\hat{\mathbf{y}}_{\text{new}}|\mathbf{x}_{\text{new}}, \mathbf{z}_{\text{new}} = \left(\mathbf{z}_{\text{new}}^T\mathbf{\Omega_z}\mathbf{A}_{\mathbf{yz}}^T + \mathbf{x}_{\text{new}}^T\mathbf{A_{xy}}\mathbf{\Omega_z}\right)\mathbf{\Omega}_{\mathbf{y}|\mathbf{x},\mathbf{z}}$, where $\mathbf{\Omega}_{\mathbf{y}|\mathbf{x},\mathbf{z}} = \left([\mathbf{\Omega_y}]^{-1} + \mathbf{A_{yz}}[\mathbf{\Omega_z}]^{-1}\mathbf{A_{yz}}^T\right)^{-1}$. For this prediction task, we estimate the variances as follows [75]:

$$\sigma^2_{\mathbf{y}_i} = \frac{1}{n - s_{\mathbf{y}_i}}\left([\mathbf{Y}]_{:,i} - \mathbf{X}^T[\mathbf{A_{xy}}]_{:,i}\right)^T\left([\mathbf{Y}]_{:,i} - \mathbf{X}^T[\mathbf{A_{xy}}]_{:,i}\right), \quad \text{for } i = 1, \ldots, q,$$
$$\sigma^2_{\mathbf{z}_i} = \frac{1}{n - s_{\mathbf{z}_i}}\left([\mathbf{Z}]_{:,i} - \mathbf{Y}^T[\mathbf{A_{yz}}]_{:,i}\right)^T\left([\mathbf{Z}]_{:,i} - \mathbf{Y}^T[\mathbf{A_{yz}}]_{:,i}\right) \quad \text{for } i = 1, \ldots, r,$$

where $s_{\mathbf{y}_i}$ and $s_{\mathbf{z}_i}$ are the numbers of non-zero entries in $[\mathbf{A_{xy}}]_{:,i}$ and $[\mathbf{A_{yz}}]_{:,i}$ respectively.

Table 4.6: Prediction errors of different methods on asthma test set

Prediction task	Lasso	Our Model	Our Model with Semi-supervised Learning
$\mathbf{y}|\mathbf{x}$	0.76494	0.75322	$\mathbf{0.75318}$
$\mathbf{z}|\mathbf{y}$	1.03486	0.97068	$\mathbf{0.89317}$
$\mathbf{y}|\mathbf{x}, \mathbf{z}$	0.78161	0.75346	$\mathbf{0.75324}$
$\mathbf{z}|\mathbf{x}$	0.85785	0.85795	$\mathbf{0.85709}$

Prediction comparison results We assess the ability to make predictions about new asthma patients based on the estimated PerturbNet model and compare the results with those from the two-layer Lasso. We split the data into train and test sets and obtained the prediction accuracy using the test set after training a model on the train set. We set aside 25 samples as a test set and used the remaining 115 fully-observed samples and 34 partially observed samples to train a sparse Gaussian chain graph model with our semi-supervised learning method. We also trained a model, using only the 115 fully observed samples with the supervised learning method, and compared the results from the two-layer Lasso, also trained from the fully observed samples. Given the estimated models, we performed prediction tasks and obtained the prediction error as the squared difference between the observed and predicted values averaged across samples in test set.

The PerturbNet model estimated from all data had the smallest prediction error for all of the prediction tasks (Table 4.6). In particular, our model with semi-supervised learning performed better than our model with supervised learning, demonstrating that leveraging partially observed data can help learn a model with greater predictive power. For supervised learning, our model outperformed Lasso. This demonstrates that taking into account the network structure in expression levels and clinical phenotypes increases the performance on prediction tasks.

4.4.5 Visual comparison of the PerturbNet and two-layer Lasso models

We compared the results from our approach and the two-layer Lasso by visually inspecting the estimated SNP effects on gene modules and the estimated gene module effects on phenotypes. For the top 50 SNPs perturbing the lung phenotypes (Figure 4.11), we examined the overall SNP effects on each gene module based on $\mathbf{\Theta_{xy}}$ and $\mathbf{B_{xy}}$ from our model and $\mathbf{A_{xy}}$ from the two-layer Lasso ($\sum_{j \in M} |[\mathbf{\Theta_{xy}}]_{i,j}|$, $\sum_{j \in M} |[\mathbf{B_{xy}}]_{i,j}|$, and $\sum_{j \in M} |[\mathbf{A_{xy}}]_{i,j}|$ for SNP i and module M). To see how each gene module influences phenotypes, we computed the magnitudes of overall gene module effects on each phenotype from $\mathbf{\Theta_{yz}}$ and $\mathbf{B_{yz}}$ in our model and $\mathbf{A_{yz}}$ in the two-layer Lasso ($\sum_{j \in M} |[\mathbf{\Theta_{yz}}]_{j,k}|$, $\sum_{j \in M} |[\mathbf{B_{yz}}]_{j,k}|$, and $\sum_{j \in M} |[\mathbf{A_{yz}}]_{j,k}|$ for module M and phenotype k).

Unlike the PerturbNet model, the two-layer Lasso does not model direct and indirect perturbation effects separately but attempts to capture both types of effects in a single set of parameters. Thus, the perturbation effects captured by the two-layer Lasso appeared to be a compromise between the direct and indirect perturbation effects captured by PerturbNet (Figure 4.22). However, the SNP effects appeared to be similar across $\mathbf{\Theta_{xy}}$, $\mathbf{B_{xy}}$, and $\mathbf{A_{xy}}$ in the module-level summaries (Figures 4.22A-4.22C), because the direct SNP perturbation effects tended to propagate to other genes only within each module, but not to genes in other modules. On the other hand, the module effects on phenotypes showed a distinct pattern across $\mathbf{\Theta_{yz}}$, $\mathbf{B_{yz}}$, and $\mathbf{A_{yz}}$ (Figures 4.22D-4.22F), because in our model, the direct influence of gene expression levels on a phenotype induces the indirect influence on other correlated phenotypes, whereas the Lasso parameter tries to capture both types of information in a single parameter. The Lasso model similarly tries to capture both direct and indirect influences of averaged gene module expression levels on traits, compared to PerturbNet which distinguishes between them, as shown in Figure 4.23.

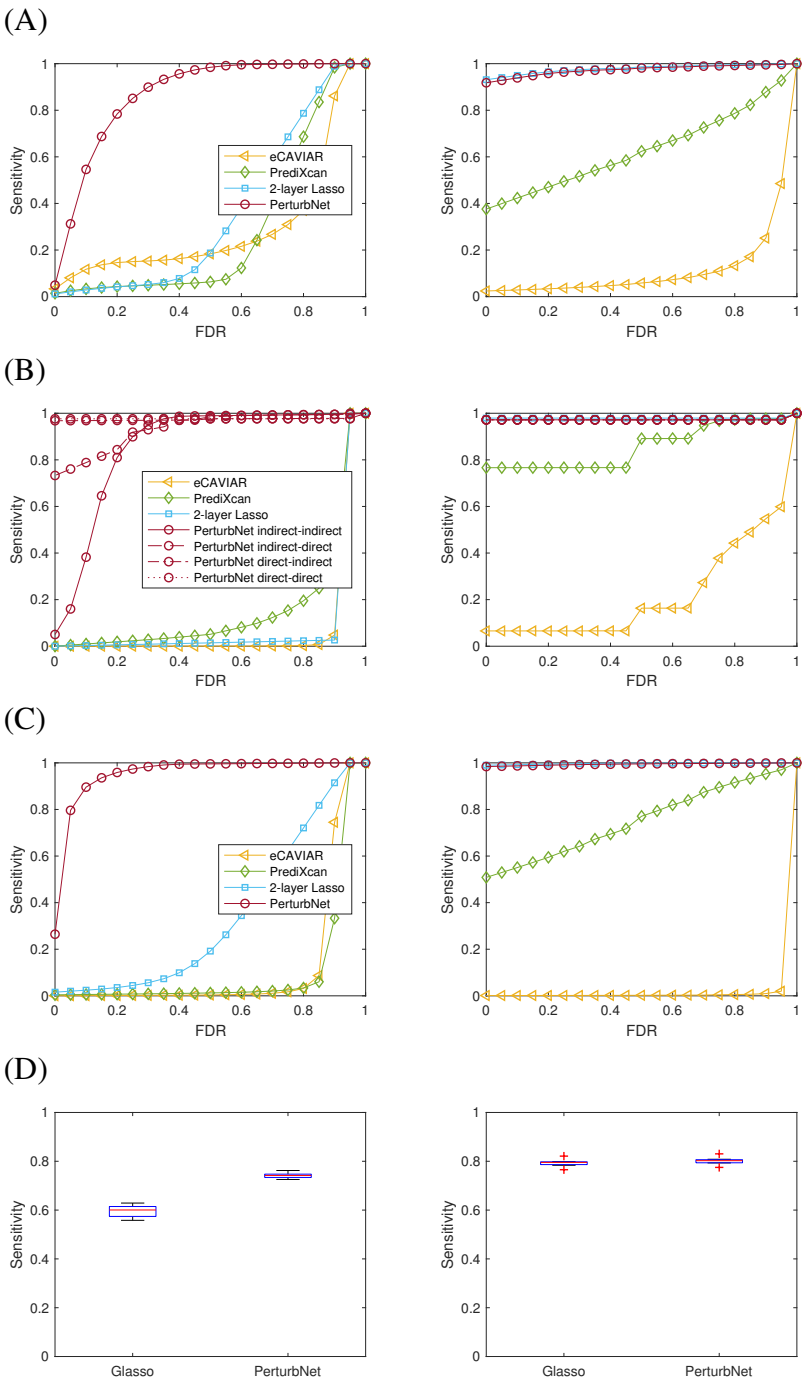


Figure 4.18: Results on simulated asthma data. Comparison of PerturbNet with eCAVIAR, PrediXcan, and two-layer Lasso. The accuracy of each method is shown for (A) the recovery of SNPs perturbing traits, (B) the recovery of genes mediating the perturbation effect of each SNP on each trait, and (C) the recovery of genes mediating the overall SNP effects on each trait. Ground-truth models with networks over genes and traits (left column) and no network (right column) were used. (D) Comparison of network learning methods on simulated data. PerturbNet and sparse GGMs were fit to data simulated from networks with SNP perturbations (left) and without SNP perturbations (right). Sensitivities at FDR=0.05 are shown.

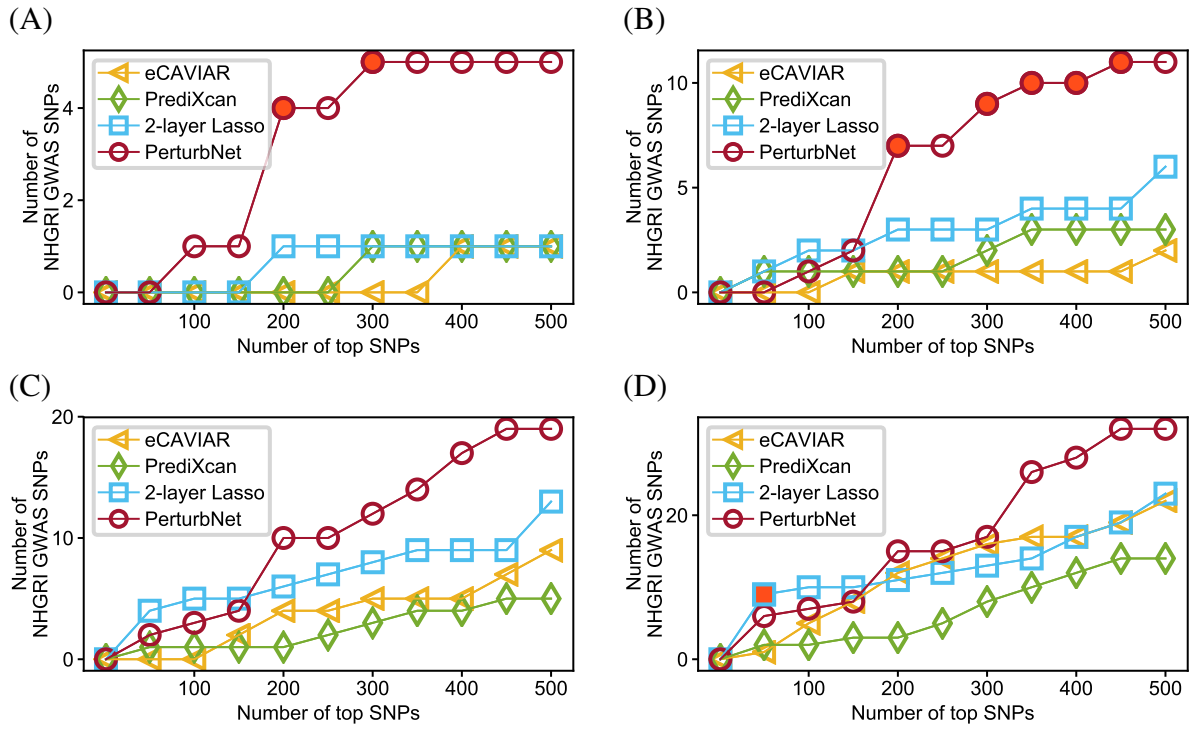


Figure 4.19: Overlaps with asthma SNPs in the NHGRI GWAS catalog. An overlap is defined as the SNPs in the GWAS catalog within (A) 10kb, (B) 20kb, (C) 50kb, and (D) 100kb of top k SNPs from each method. Statistically significant enrichments (p-value < 0.05) are highlighted.

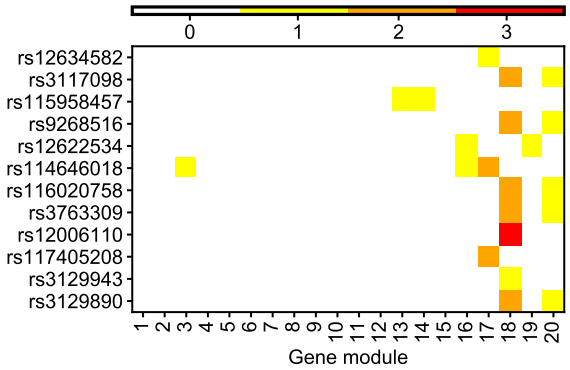


Figure 4.20: PerturbNet gene modules mediating the effects of the NHGRI GWAS SNPs on asthma traits. Out of 32 NHGRI SNPs within 100kb of our top 500 SNPs, 12 SNPs, shown in rows, had most of their effects on traits mediated by gene modules that are not singletons. The colorbar indicates the number of SNPs found by PerturbNet that are located in the region of each SNP in the GWAS catalog.

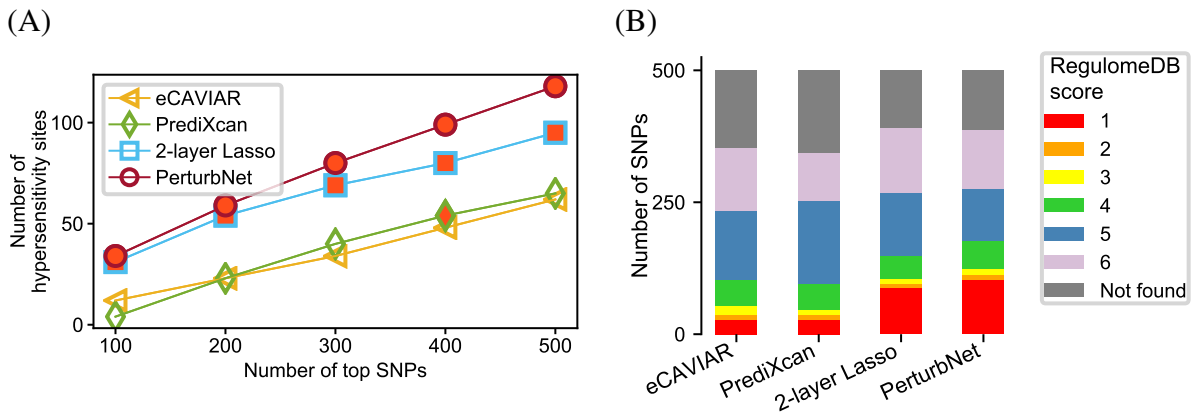


Figure 4.21: Overlaps with ENCODE DNase hypersensitivity sites and functionally annotated SNPs in RegulomeDB. (A) Overlaps with the ENCODE DNase hypersensitivity sites for B-lymphoblastoid cell line. Statistically significant enrichments (p-value < 0.05) are highlighted. Enrichment p-values were computed with the UES software package [44]. (B) Overlaps with functionally annotated SNPs in RegulomeDB [10]. Lower scores indicate more likely to be functional. RegulomeDB scores rank genomic locations from 1 (strong evidence for being functional) to 6 (minimal evidence).

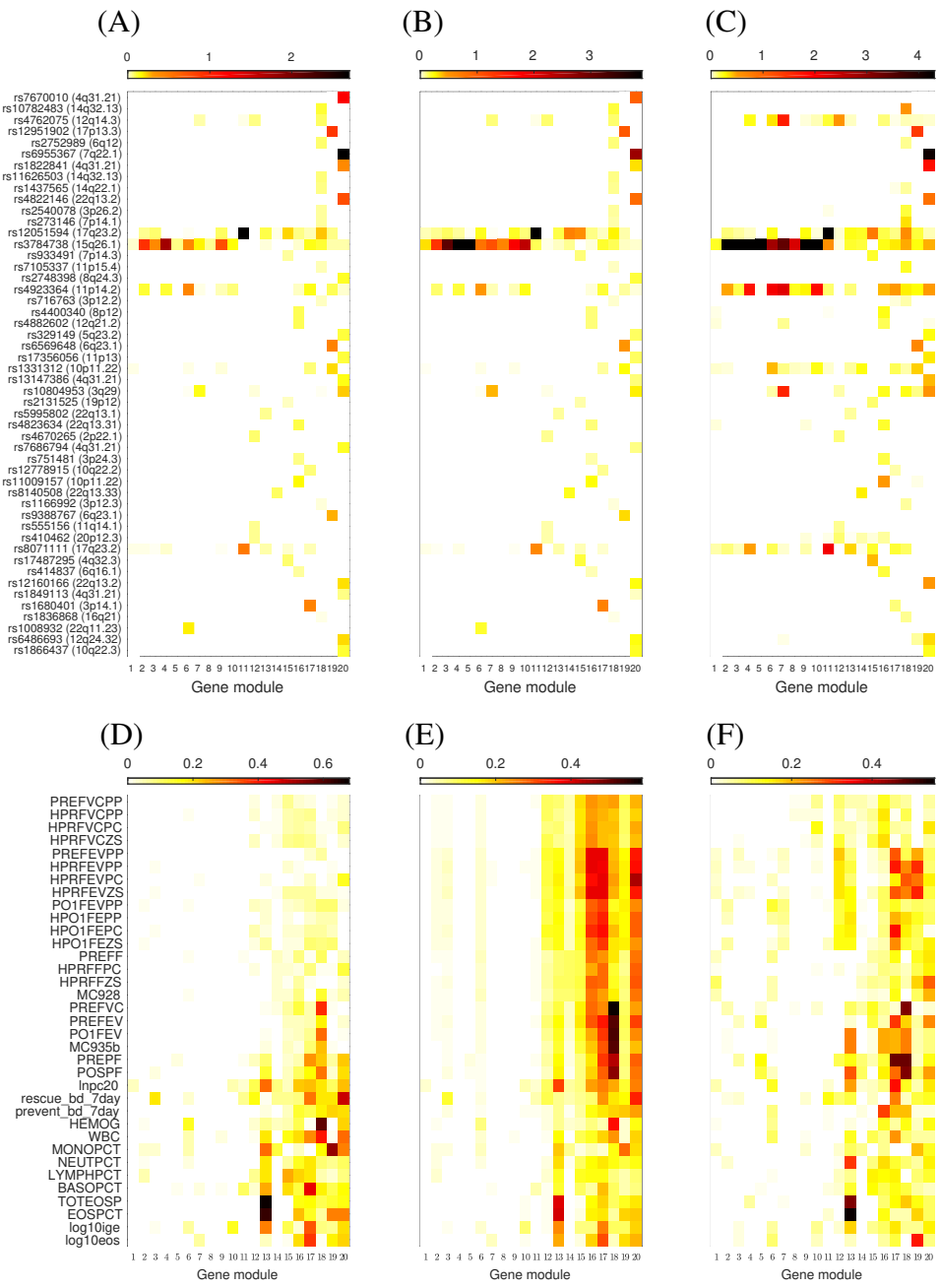


Figure 4.22: Comparison of PerturbNet and Lasso for learning the cascaded influence of SNPs to gene modules to phenotypes. For the top 50 SNPs perturbing lung phenotypes (Figure 4.11), the effects of these SNPs on each of the gene modules are shown for (A) $\mathbf{\Theta_{yz}}$ from our model, (B) $\mathbf{B_{yz}}$ inferred from our model, and (C) $\mathbf{A_{yz}}$ from the two-layer Lasso. The effects of the expression levels in each gene module on lung phenotypes are shown for (D) $\mathbf{\Theta_{yz}}$ from our model, (E) $\mathbf{B_{yz}}$ inferred from our model, and (F) $\mathbf{A_{yz}}$ from the two-layer Lasso. The effect sizes in each model parameter matrix above were summed across all genes within each module after taking absolute values.

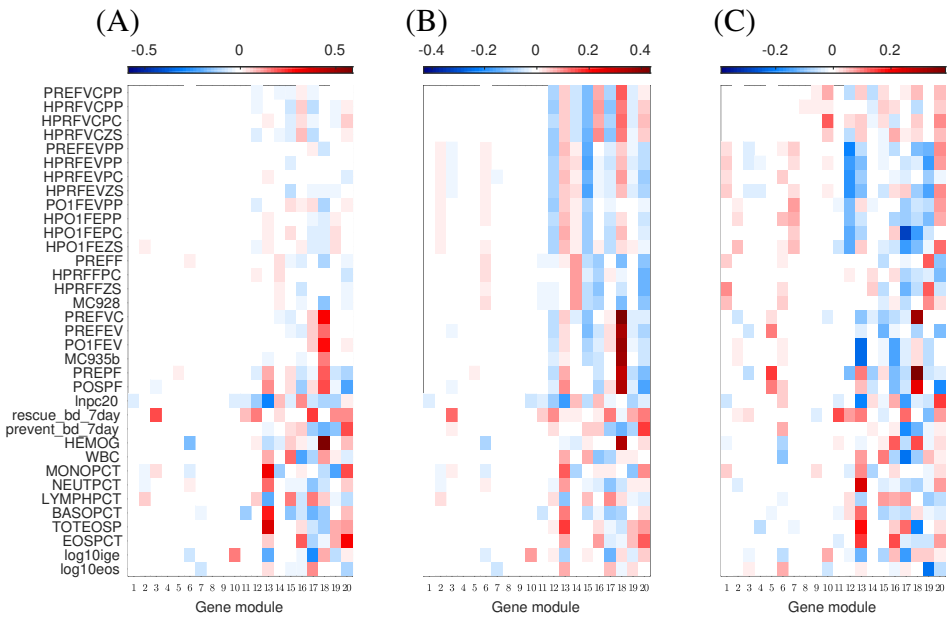


Figure 4.23: Association between clinical traits and aggregated expression level of each gene module, compared between PerturbNet and Lasso. (a) Direct edges in $\mathbf{\Theta_{yz}}$ (b) regression coefficients $\mathbf{B_{yz}}$, (c) regression coefficients from Lasso model $\mathbf{A_{yz}}$.

4.5 Discussion of PerturbNet for Asthma

Our results from applying PerturbNet to asthma data confirmed the observations from the previous studies, including GWAS, eQTL mapping, and gene network modeling [69, 79]. Our results confirmed the finding from previous studies on combining the results of GWAS and eQTL mapping [69] that there is a partial overlap between SNPs perturbing expression levels and SNPs perturbing phenotypes. In addition, this overlap was more significant for eQTLs that perturb trait-associated modules than eQTLs that perturb other parts of the gene network, as was previously reported [73].

The analysis of the asthma data with PerturbNet provided new insights. PerturbNet was able to systematically reveal the gene network that lies between the SNPs and phenotypes and to uncover how different parts of this gene network modulate the SNP effects on phenotypes in a statistically principled manner. Often, there are genetic loci that have been previously known to be linked to the disease susceptibility, though little is known about the underlying molecular mechanism. In such cases, the PerturbNet analysis of asthma data demonstrated the potential to reveal the molecular pathway that are perturbed by previously known trait-associated loci.

Chapter 5

Discussion and Conclusion

5.1 Discussion

In this thesis, we have introduced a new machine learning framework called PerturbNet for learning a gene network underlying phenotypes using SNP perturbations and for identifying SNPs that perturb this network, given population genotype, expression, and phenotype data. As part of our framework, we have proposed new statistical models, combined with scalable learning algorithms and intuitive inference procedures and scoring functions to make using the framework simple and useful for systems biologists working with human genomic datasets.

Compared to many of the previous methods that focused on the co-localization of eQTLs and genetic association signals for phenotypes [33, 36, 45], using multi-stage methods [27, 79, 80], our approach combines all available data in a single statistical analysis and directly models the multiple layers of a biological system with a cascade of influence from SNPs to expression levels to phenotypes, while modeling each layer as a network. Our probabilistic graphical model framework allows to model eQTLs with or without an impact on phenotypes for an investigation of co-localization of SNPs perturbing expression levels and SNPs perturbing phenotypes and to extract rich information on the molecular mechanisms that explains the influence of SNPs on phenotypes. We developed fast learning algorithms called Fast-sCGGM and Mega-sCGGM for learning sparse CGGM components of the PerturbNet model, which serve as the key subroutine of our PerturbNet learning method, to enable analysis of human genome scale data within a few hours.

5.2 Future Directions

We conclude with directions for future study.

5.2.1 Alternative threadings of sCGGM components for more complex integrative analyses

PerturbNet provides a flexible tool that can be extended in several different ways in a straightforward manner. Because the sparse Gaussian chain graph model in PerturbNet uses sparse CGGMs

as building blocks, the sparse CGGM component models can be threaded in different ways to form sparse Gaussian chain graph models with different structures. For example, if expression data from multiple tissue types are available for a patient cohort along with genome sequence and phenotype data, a sparse Gaussian chain graph model can be set up with multiple component sparse CGGMs, each corresponding to the gene network under SNP perturbation in each tissue type, linked to another sparse CGGM for modeling expression levels influencing phenotypes. Models like this can reveal SNPs that perturb phenotypes through different tissue types and through different modules in each tissue-specific gene network. Another possible extension is to thread more than two component sparse CGGMs within a sparse Gaussian chain graph model to model more than two layers in a biological system, including epigenomes, metabolomes, and proteomes.

5.2.2 Mixed effects modeling extensions of sparse CGGMs

One advantage of PerturbNet over single-gene and single-trait methods such as Lasso, is that our network-based method takes advantage of correlations among gene expressions and among clinical traits to increase statistical power despite noisy gene expression levels and clinical traits measurements. However, another type of correlation can potentially provide misleading results due to confounding: correlations among subjects. In our analysis of CAMP asthma data, we sidestepped this problem by simply excluding from our analysis all the data from individuals who were not non-Hispanic Caucasians. Extending sparse CGGMs with a mixed effects approach would allow scientists to improve the power of their studies with more data while avoiding confounding due to population stratification.

Bibliography

[1] 1000 Genomes Project Consortium. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491(7422):56, 2012. 4.3.5

[2] F. Abegaz and E. Wit. Sparse time series chain graphical models for reconstructing genetic networks. *Biostatistics*, pages 586–599, 2013. 3

[3] Frank Wolfgang Albert, Joshua S Bloom, Jake Siegel, Laura Day, and Leonid Kruglyak. Genetics of trans-regulatory variation in gene expression. *eLife*, 7:e35471, 2018. 1.4.2

[4] AF Maarten Altelaar, Javier Munoz, and Albert JR Heck. Next-generation proteomics: towards an integrative view of proteome dynamics. *Nature Reviews Genetics*, 14(1):35, 2013. 1.1

[5] S. Andersson, D. Madigan, and D. Perlman. An alternative Markov property for chain graphs. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, pages 40–48. Morgan Kaufmann, 1996. 3.1.1, 3.1.3

[6] S. Andersson, D. Madigan, and D. Perlman. Alternative Markov properties for chain graphs. *Scandinavian Journal of Statistics*, 28:33–85, 2001. 3.1.1, 3.1.3

[7] Larry Armijo et al. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1):1–3, 1966. 2.1.1

[8] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25, 2000. 4.3.2

[9] Cindy Barnig, Ghada Alsaleh, Nicolas Jung, Doulaye Dembélé, Nicodème Paul, Anh Poirot, Béatrice Uring-Lambert, Philippe Georgel, Fréderic de Blay, and Seiamak Bahram. Circulating human eosinophils share a similar transcriptional profile in asthma and other hypereosinophilic disorders. *PloS One*, 10(11):e0141740, 2015. 4.3.5

[10] Alan P Boyle, Eurie L Hong, Manoj Hariharan, Yong Cheng, Marc A Schaub, Maya Kasowski, Konrad J Karczewski, Julie Park, Benjamin C Hitz, Shuai Weng, et al. Annotation of functional variation in personal genomes using RegulomeDB. *Genome Research*, 22 (9):1790–1797, 2012. 4.4.3, 4.21

[11] Rainer Breitling, Yang Li, Bruno M Tesson, Jingyuan Fu, Chunlei Wu, Tim Wiltshire, Alice Gerrits, Leonid V Bystrykh, Gerald De Haan, Andrew I Su, et al. Genetical genomics: spotlight on qtl hotspots. *PLoS genetics*, 4(10):e1000232, 2008. 4.3.7

[12] Annalisa Buniello, Jacqueline A L MacArthur, Maria Cerezo, Laura W Harris, James

Hayhurst, Cinzia Malangone, Aoife McMahon, Joannella Morales, Edward Mountjoy, Elliot Sollis, et al. The NHGRI-EBI GWAS Catalog of published genome-wide association studies, targeted arrays and summary statistics 2019. *Nucleic Acids Research*, 47(D1): D1005–D1012, 2018. 4.3.5, 4.4.2

[13] W. Buntine. Chain graphs for learning. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, pages 46–54. Morgan Kaufmann, 1995. 3

[14] Alexander Buntru, Kathrin Kopp, Maike Voges, Ronald Frank, Verena Bachmann, and Christof R Hauck. Phosphatidylinositol 3-kinase activity is critical for initiating the oxidative burst and bacterial destruction during ceacam3-mediated phagocytosis. *Journal of Biological Chemistry*, 286(11):9555–9566, 2011. 4.3.5

[15] T Tony Cai, Hongzhe Li, Weidong Liu, and Jichun Xie. Covariate-adjusted precision matrix estimation with an application in genetical genomics. *Biometrika*, page ass058, 2012. 1.4.2

[16] Delphine Charignon, Denise Ponard, Christian de Gennes, Christian Drouet, and Arije Ghannam. SERPING1 and F12 combined variants in a hereditary angioedema family. *Annals of Allergy, Asthma & Immunology*, 121(4):500–502, 2018. 4.3.5

[17] Minming Chen, Zhouchen Lin, Yi Ma, and Leqin Wu. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. Technical report, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, 2009. 4.1

[18] X. Chen, X. Shi, X. Xu, Z. Wang, R. Mills, C. Lee, and J. Xu. A two-graph guided multi-task lasso approach for eQTL mapping. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 16. JMLR W&CP, 2012. 3.2.1

[19] Y. Chen, J. Zhu, P.K. Lum, X. Yang, S. Pinto, D.J. MacNeil, C. Zhang, J. Lamb, S. Edwards, S.K. Sieberts, et al. Variations in DNA elucidate molecular networks that cause disease. *Nature*, 452(27):429–35, 2008. 1.4.3, 3, 3.1.4

[20] Childhood Asthma Management Program Research Group. The childhood asthma management program (CAMP): design, rationale, and methods. *Controlled Clinical Trials*, 20 (1):91–120, 1999. 1.1, 4.1, 4.3, 4.4.1

[21] Childhood Asthma Management Program Research Group. Long-term effects of budesonide or nedocromil in children with asthma. *New England Journal of Medicine*, 343(15): 1054–1063, 2000. 1.1, 4.1, 4.3, 4.4.1

[22] Ross E Curtis, Junming Yin, Peter Kinnaird, and Eric P Xing. Finding genome-transcriptome-phenome association with structured association mapping and visualization in genamap. In *Biocomputing 2012*, pages 327–338. World Scientific, 2012. 1.1

[23] Darren A Cusanovich, Bryan Pavlovic, Jonathan K Pritchard, and Yoav Gilad. The functional consequences of variation in transcription factor binding. *PLoS Genet*, 10(3): e1004226, 2014. 1.1

[24] Patrick Danaher, Pei Wang, and Daniela M Witten. The joint graphical lasso for inverse covariance estimation across multiple classes. *Journal of the Royal Statistical Society:*

Series B (Statistical Methodology), 76(2):373–397, 2014. 1.4.2

[25] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, 39:1–38, 1977. 3.2.2

[26] M. Drton and M. Eichler. Maximum likelihood estimation in Gaussian chain graph models under the alternative Markov property. *Scandinavian Journal of Statistics*, 33:247–57, 2006. 3, 3.1.1, 3.1.1

[27] Valur Emilsson, Gudmar Thorleifsson, Bin Zhang, Amy S Leonardson, Florian Zink, Jun Zhu, Sonia Carlson, Agnar Helgason, G Bragi Walters, Steinunn Gunnarsdottir, et al. Genetics of gene expression and its effect on disease. *Nature*, 452:423–428, 2008. 1.1, 5.1

[28] ENCODE Project Consortium. An integrated encyclopedia of DNA elements in the human genome. *Nature*, 489(7414):57, 2012. 4.4.3

[29] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–41, 2008. 3, 3.2.1

[30] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008. 1.3, 1.3.1, 1.4.2

[31] Benjamin Frot, Luke Jostins, and Gilean McVean. Graphical model selection for Gaussian conditional random fields in the presence of latent variables. *Journal of the American Statistical Association*, accepted(accepted), 2018. 1.1

[32] M. Frydenberg. The chain graph Markov property. *Scandinavian Journal of Statistics*, 17: 333–53, 1990. 3, 3.1, 3.1.3

[33] Eric R Gamazon, Heather E Wheeler, Kaanan P Shah, Sahar V Mozaffari, Keston Aquino-Michaels, Robert J Carroll, Anne E Eyler, Joshua C Denny, Dan L Nicolae, Nancy J Cox, et al. A gene-based association method for mapping traits using reference transcriptome data. *Nature Genetics*, 47(9):1091, 2015. 1.1, 1.4.3, 4.4.1, 5.1

[34] Daniel M Gatti, Andrey A Shabalin, Tieu-Chong Lam, Fred A Wright, Ivan Rusyn, and Andrew B Nobel. Fastmap: fast eqtl mapping in homozygous populations. *Bioinformatics*, 25(4):482–489, 2008. 1.4.1

[35] Gene Ontology Consortium. The Gene Ontology (GO) database and informatics resource. *Nucleic Acids Research*, 32(suppl_1):D258–D261, 2004. 4.3.2

[36] Claudia Giambartolomei, Damjan Vukcevic, Eric E Schadt, Lude Franke, Aroon D Hingorani, Chris Wallace, and Vincent Plagnol. Bayesian test for colocalisation between pairs of genetic association studies using summary statistics. *PLoS Genetics*, 10(5):e1004383, 2014. 1.1, 1.4.3, 3.1.4, 5.1

[37] Greg Gibson, Joseph E Powell, and Urko M Marigorta. Expression quantitative trait locus analysis for translational medicine. *Genome Medicine*, 7(1):60, 2015. 1.1

[38] Mark Gilchrist, William R Henderson, April E Clark, Randi M Simmons, Xin Ye, Kelly D Smith, and Alan Aderem. Activating transcription factor 3 is a negative regulator of allergic pulmonary inflammation. *Journal of Experimental Medicine*, 205(10):2349–2357,

2008. 4.3.5

[39] Maxim Grechkin, Maryam Fazel, Daniela M Witten, and Su-In Lee. Pathway graphical lasso. 2015. 1.4.2

[40] Danielle M Greenawalt, Radu Dobrin, Eugene Chudin, Ida J Hatoum, Christine Suver, John Beaulaurier, Bin Zhang, Victor Castro, Jun Zhu, Solveig K Sieberts, et al. A survey of the genetics of stomach, liver, and adipose gene expression from a morbidly obese cohort. *Genome Research*, 21(7):1008–1016, 2011. 1.1

[41] GTEx Consortium. The genotype-tissue expression (GTEx) pilot analysis: multi-tissue gene regulation in humans. *Science*, 348(6235):648–660, 2015. 1.1, 1.4.1

[42] Tsonwin Hai, Christopher C Wolford, and Yi-Seok Chang. ATF3, a hub of the cellular adaptive-response network, in the pathogenesis of diseases: is modulation of inflammation a unifying component? *Gene Expression*, 15(1):1–11, 2010. 4.3.5

[43] David Hallac, Youngsuk Park, Stephen Boyd, and Jure Leskovec. Network inference via the time-varying graphical lasso. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 205–213. ACM, 2017. 1.4.2

[44] James E Hayes, Gosia Trynka, Joseph Vijai, Kenneth Offit, Soumya Raychaudhuri, and Robert J Klein. Tissue-specific enrichment of lymphoma risk loci in regulatory elements. *PloS One*, 10(9):e0139360, 2015. 4.4.3, 4.21

[45] Xin He, Chris K Fuller, Yi Song, Qingying Meng, Bin Zhang, Xia Yang, and Hao Li. Sherlock: detecting gene-disease associations by matching patterns of expression QTL and GWAS. *The American Journal of Human Genetics*, 92(5):667–680, 2013. 1.1, 1.4.3, 5.1

[46] Farhad Hormozdiari, Martijn van de Bunt, Ayellet V Segrè, Xiao Li, Jong Wha J Joo, Michael Bilow, Jae Hoon Sul, Sriram Sankararaman, Bogdan Pasaniuc, and Eleazar Eskin. Colocalization of gwas and eqtl signals detects target genes. *The American Journal of Human Genetics*, 99(6):1245–1260, 2016. 1.4.3, 3.1.4, 4.4.1

[47] Cho-Jui Hsieh, Inderjit S. Dhillon, Pradeep K. Ravikumar, and Mátyás A. Sustik. Sparse inverse covariance matrix estimation using quadratic approximation. In *Advances in Neural Information Processing Systems 24*, pages 2330–2338, 2011. 1.3.1, 1.3.2, 2.1.1

[48] Cho-Jui Hsieh, Matyas A Sustik, Inderjit S Dhillon, Pradeep K Ravikumar, and Russell Poldrack. Big & quic: Sparse inverse covariance estimation for a million variables. In *Advances in Neural Information Processing Systems 26*, pages 3165–3173, 2013. 1.3.1, 2, 2.3, 2.3.1, 2.4.1

[49] C.J. Hsieh, M. Sustik, I. Dhillon, and P. Ravikumar. Sparse inverse covariance matrix estimation using quadratic approximation. In *Advances in Neural Information Processing Systems (NIPS) 24*, 2011. 3.2.1

[50] L. Jacob, G. Obozinski, and J. Vert. Group lasso with overlap and graph lasso. In *Proceedings of the 26th International Conference on Machine Learning*, 2009. 3, 3.2.1

[51] Ritsert C Jansen. Studying complex biological systems using multifactorial perturbation. *Nature Reviews Genetics*, 4(2):145, 2003. 1.1

[52] Göran Jönsson, AG Sjöholm, Lennart Truedsson, AA Bengtsson, Jean Henrik Braconier, and Gunnar Sturfelt. Rheumatological manifestations, organ damage and autoimmunity in hereditary c2 deficiency. 0, 2007. 4.3.5

[53] George Karypis and Vipin Kumar. METIS - unstructured graph partitioning and sparse matrix ordering system, version 2.0. Technical report, University of Minnesota, Department of Computer Science and Engineering, Army HPC Research Center, Minneapolis, MN, 1995. 2.3.1, 4.2, 4.3.1

[54] AB Kay, GD Bacon, BarbaraA Mercer, H Simpson, and JW Crofton. Complement components and IgE in bronchial asthma. *The Lancet*, 304(7886):916–920, 1974. 4.3.5

[55] Mohammad Afzal Khan, Abdullah Mohammed Assiri, and Dieter Clemens Broering. Complement mediators: key regulators of airway tissue remodeling in asthma. *Journal of Translational Medicine*, 13(1):272, 2015. 4.3.5

[56] Gleb Kichaev, Gaurav Bhatia, Po-Ru Loh, Steven Gazal, Kathryn Burch, Malika K Freund, Armin Schoech, Bogdan Pasaniuc, and Alkes L Price. Leveraging polygenic functional enrichment to improve GWAS power. *The American Journal of Human Genetics*, 104(1):65–75, 2019. 4.3.5

[57] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009. 3, 3.1, 3.1.3

[58] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT Press, 2009. 3.3.2

[59] Nicole Krämer, Juliane Schäfer, and Anne-Laure Boulesteix. Regularized estimation of large-scale gene association networks using graphical gaussian models. *BMC bioinformatics*, 10(1):384, 2009. 1.4.2

[60] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, 2001. 3, 3.1

[61] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, volume 951, pages 282–289, 2001. 1.3.2

[62] S.L. Lauritzen and N. Wermuth. Graphical models for associations between variables, some of which are qualitative and some quantitative. *The Annals of Statistics*, 17(1): 31–57, 1989. 3, 3.1, 3.1.2

[63] Wei Luo, Maen Obeidat, Antonio Fabio Di Narzo, Rong Chen, Don D Sin, Peter D Paré, and Ke Hao. Airway epithelial expression quantitative trait loci reveal genes underlying asthma and other airway diseases. *American Journal of Respiratory Cell and Molecular Biology*, 54(2):177–187, 2016. 4.3.5

[64] Patricia Menéndez, Yiannis AI Kourmpetis, Cajo JF ter Braak, and Fred A van Eeuwijk. Gene regulatory networks from multifactorial perturbations using graphical lasso: application to the dream4 challenge. *PloS one*, 5(12):e14147, 2010. 1.4.2

[65] Miriam F Moffatt, Ivo G Gut, Florence Demenais, David P Strachan, Emmanuelle Bouzigon, Simon Heath, Erika Von Mutius, Martin Farrall, Mark Lathrop, and William OCM Cookson. A large-scale, consortium-based genomewide association study of asthma. *New England Journal of Medicine*, 363(13):1211–1221, 2010. 1.1

[66] Karthik Mohan, Palma London, Maryam Fazel, Daniela Witten, and Su-In Lee. Node-based learning of multiple gaussian graphical models. *The Journal of Machine Learning Research*, 15(1):445–488, 2014. 1.4.2

[67] Amy Murphy, Jen-Hwa Chu, Mousheng Xu, Vincent J Carey, Ross Lazarus, Andy Liu, Stanley J Szefler, Robert Strunk, Karen DeMuth, Mario Castro, et al. Mapping of numerous disease-associated expression polymorphisms in primary peripheral blood cd4+ lymphocytes. *Human molecular genetics*, 19(23):4745–4757, 2010. 1.1, 4.1, 4.3, 4.3, 4.4.1

[68] Bernard Ng, Rafeef Abugharbieh, Gael Varoquaux, Jean Baptiste Poline, and Bertrand Thirion. Connectivity-informed fmri activation detection. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2011*, pages 285–292. Springer, 2011. 1.3.1

[69] Dan L Nicolae, Eric Gamazon, Wei Zhang, Shiwei Duan, M Eileen Dolan, and Nancy J Cox. Trait-associated snps are more likely to be eQTLs: annotation to enhance discovery from GWAS. *PLoS genetics*, 6(4):e1000888, 2010. 4.3.3, 4.5

[70] G. Obozinski, M.J. Wainwright, and M.J. Jordan. High-dimensional union support recovery in multivariate regression. In *Advances in Neural Information Processing Systems 21*, 2008. 3, 3.2.1

[71] Yongjin Park, Abhishek Sarkar, Liang He, Jose Davilla-Velderrain, Philip L De Jager, and Manolis Kellis. Causal gene inference by multivariate mediation analysis in alzheimer's disease. *bioRxiv*, page 219428, 2017. 1.4.3

[72] Miguel Pérez-Enciso, José R Quevedo, and Antonio Bahamonde. Genetical genomics: use all data. *BMC genomics*, 8(1):69, 2007. 1.4.2

[73] Lauren A Peters, Jacqueline Perrigoue, Arthur Mortha, Alina Iuga, Won-min Song, Eric M Neiman, Sean R Llewellyn, Antonio Di Narzo, Brian A Kidd, Shannon E Telesco, et al. A functional genomics predictive network model identifies regulators of inflammatory bowel disease. *Nature genetics*, 49(10):1437, 2017. 1.1, 4.3.3, 4.5

[74] J Qian, T Hastie, J Friedman, R Tibshirani, and N Simon. Glmnet for matlab. *Accessed: Nov*, 13:2017, 2013. 4.2, 4.2

[75] Stephen Reid, Robert Tibshirani, and Jerome Friedman. A study of error variance estimation in lasso regression. *Statistica Sinica*, 26:35–67, 2016. 4.4.4

[76] Carrie M Rosenberger, April E Clark, Piper M Treuting, Carrie D Johnson, and Alan Aderem. Atf3 regulates mcmv infection in mice by modulating ifn-γ expression in natural killer cells. *Proceedings of the National Academy of Sciences*, 105(7):2544–2549, 2008. 4.3.5

[77] A. Rothman, E. Levina, and J. Zhu. Sparse multivariate regression with covariance estimation. *Journal of Computational and Graphical Statistics*, 19(4):947–962, 2010. 3.1.1,

3.4.1

[78] Adam J Rothman, Elizaveta Levina, and Ji Zhu. Sparse multivariate regression with covariance estimation. *Journal of Computational and Graphical Statistics*, 19(4):947–962, 2010. 1.4.2, 4.2

[79] Eric E Schadt. Molecular networks as sensors and drivers of common human diseases. *Nature*, 461:218–223, 2009. 1.1, 4.5, 5.1

[80] Eric E Schadt, Cliona Molony, Eugene Chudin, Ke Hao, Xia Yang, Pek Y Lum, Andrew Kasarskis, Bin Zhang, Susanna Wang, Christine Suver, et al. Mapping the genetic architecture of gene expression in human liver. *PLoS Biology*, 6(5):e107, 2008. 1.1, 5.1

[81] Paul Scheet and Matthew Stephens. A fast and flexible statistical model for large-scale population genotype data: applications to inferring missing genotypes and haplotypic phase. *The American Journal of Human Genetics*, 78(4):629–644, 2006. 4.1

[82] Andrey A Shabalin. Matrix eqtl: ultra fast eqtl analysis via large matrix operations. *Bioinformatics*, 28(10):1353–1358, 2012. 1.4.1

[83] Ophir Shalem, Neville E Sanjana, and Feng Zhang. High-throughput functional genomics using CRISPR–Cas9. *Nature Reviews Genetics*, 16(5):299, 2015. 1.1

[84] Teppei Shimamura, Seiya Imoto, Rui Yamaguchi, and Satoru Miyano. Weighted lasso in graphical gaussian modeling for large gene network estimation based on microarray data. In *Genome Informatics 2007: Genome Informatics Series Vol. 19*, pages 142–153. World Scientific, 2007. 1.4.2

[85] Vladimir Shulaev. Metabolomics technology and bioinformatics. *Briefings in bioinformatics*, 7(2):128–139, 2006. 1.1

[86] K.A. Sohn and S. Kim. Joint estimation of structured sparsity and output structure in multiple-output regression via inverse-covariance regularization. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 16. JMLR W&CP, 2012. 1.3, 1.3.2, 2.1.1, 3, 3.1.2, 3.1.2, 3.1.2

[87] Kyung-Ah Sohn and Seyoung Kim. Joint estimation of structured sparsity and output structure in multiple-output regression via inverse-covariance regularization. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1081–1089. JMLR W&CP, 2012. 1.1, 3.3.1

[88] Roger Temple, Elizabeth Allen, Jeremy Fordham, Simon Phipps, Hans-Christoph Schneider, Klaus Lindauer, Ian Hayes, Jacqui Lockey, Kenny Pollock, and Ray Jupp. Microarray analysis of eosinophils reveals a number of candidate survival and apoptosis genes. *American Journal of Respiratory Cell and Molecular Biology*, 25(4):425–433, 2001. 4.3.5

[89] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of Royal Statistical Society, Series B*, 58(1):267–288, 1996. 2.1.1

[90] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996. 1.3, 1.3.1, 4.2, 4.4.4

[91] Z. Tu, M.P. Keller, C. Zhang, M.E. Rabaglia, D.M. Greenawalt, X. Yang, I.M. Wang, H. Dai, M.D. Bruss, P.Y. Lum, Y.P. Zhou, D.M. Kemp, C. Kendziorski, B.S. Yandell,

A.D. Attie, E.E. Schadt, and J. Zhu. Integrative analysis of a cross-loci regulation network identifies app as a gene regulating insulin secretion from pancreatic islets. *PLoS Genetics*, 8(12):e1003107, 2012. 1.4.3, 3, 3.1.4, 3.4.2

[92] Peter M Visscher, Naomi R Wray, Qian Zhang, Pamela Sklar, Mark I McCarthy, Matthew A Brown, and Jian Yang. 10 years of gwas discovery: biology, function, and translation. *The American Journal of Human Genetics*, 101(1):5–22, 2017. 1.4.1

[93] Wellcome Trust Case Control Consortium. Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls. *Nature*, 447(7145):661, 2007. 1.1

[94] Xiaoquan Wen, Roger Pique-Regi, and Francesca Luca. Integrating molecular qtl data into genome-wide genetic association analysis: Probabilistic assessment of enrichment and colocalization. *PLoS genetics*, 13(3):e1006646, 2017. 1.4.3, 3.1.4

[95] Ernst C Wit and Antonino Abbruzzo. Inferring slowly-changing dynamic gene-regulatory networks. *BMC bioinformatics*, 16(6):S5, 2015. 1.4.2

[96] J. Wu, Z. Gao, and H. Sun. Cascade and breakdown in scale-free networks with community structure. *Physical Review*, 74:066111, 2006. 3.4.1

[97] Tong Tong Wu, Yi Fang Chen, Trevor Hastie, Eric Sobel, and Kenneth Lange. Genome-wide association analysis by lasso penalized logistic regression. *Bioinformatics*, 25(6): 714–721, 2009. 1.4.1, 4.4.4

[98] Zhengli Wu, Adam J MacNeil, Robert Junkins, Bo Li, Jason N Berman, and Tong-Jun Lin. Mast cell FcεRI-induced early growth response 2 regulates CC chemokine ligand 1–dependent CD4+ T cell migration. *The Journal of Immunology*, 190(9):4500–4507, 2013. 4.3.5

[99] Matt Wytock and J Zico Kolter. Large-scale probabilistic forecasting in energy systems using sparse gaussian conditional random fields. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 1019–1024. IEEE, 2013. 1.3.2

[100] Matt Wytock and J.Zico Kolter. Sparse Gaussian conditional random fields: algorithms, theory, and application to energy forecasting. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28. JMLR W&CP, 2013. 1.1, 1.3, 1.3.2, 2, 2.1, 2.1.1, 2.2.1, 2.2.2, 3, 3.1.2, 3.1.2, 3.1.2, 3.2.1, 3.4.1, 4.2, 4.2

[101] J. Yin and H. Li. A sparse conditional Gaussian graphical model for analysis of genetical genomics data. *The annals of applied statistics*, 5(4):2630, 2011. 3

[102] Jianxin Yin and Hongzhe Li. A sparse conditional gaussian graphical model for analysis of genetical genomics data. *The annals of applied statistics*, 5(4):2630, 2011. 1.4.2

[103] Xiao-Tong Yuan and Tong Zhang. Partial gaussian graphical model estimation. *IEEE Transactions on Information Theory*, 60(3):1673–1687, 2014. 1.3.2, 2.1.1

[104] Lingxue Zhang and Seyoung Kim. Learning gene networks under SNP perturbations using eQTL datasets. *PLoS Computational Biology*, 10(2):e1003420, 2014. 1.1, 1.3.2, 1.4.2, 3.3.1, 4.2