Inferring regulatory networks using a hierarchical Bayesian graphical Gaussian model

Fan Li **Yiming Yang** **Eric Xing**

March 2005
CMU-ML-06-117

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

In this paper, we propose a new formalism based on graphical Gaussian model (GGM) to infer genetic regulatory networks. A hierarchical Bayesian prior for the precision matrix of the GGM is introduced to impose a bias toward sparse graph structure. We show that the MAP estimation of the undirected graph can be readily obtained by a variant of the well-known Lasso regression algorithm. Then we integrate the estimated graph with the "CHIp-Chip" protein-binding location data to infer the regulatory networks using a post-processing algorithm. Compared to extant Bayesian network (BN) models for similar tasks, our formalism captures statistical dependencies among genes that are more prevalent and plausible in the biological system. Our approach is also capable of modeling partial correlations between mRNA levels and therefore goes beyond clustering-based approaches. We applied our method to an expression microarray data (more than 6000 genes) together with a genome-wide location analysis data (more than 100 TFs). Evaluated on the consistency with the GO annotations, our method achieves a significantly better performance than clustering and BN learning algorithms in discovering genetic regulatory modules.

Keywords: Lasso, structure learning, graphical Gaussian model

1 Background

A *gene regulatory network* usually refers the ensemble of DNA segments (e.g., genes, motifs) and proteins in a cell and the causal schemes (e.g., regulatory dependencies) according these elements interacting with each other and with other substances in the cell, thereby governing the rates at which genes in the network are transcribed into mRNA (Fig. 1). Automated induction of large genetic regulatory networks has been an open challenge in machine learning and computational biology. In recent years, advances in microarray technology have made possible the simultaneous monitoring of the mRNA levels of tens of thousands of genes from the entire genome under various conditions. If two genes are co-regulated by the same Transcriptional Factors (TFs), they will tend to have similar patterns of mRNA levels in different conditions. Thus it is possible to infer the structure of the gene regulatory network from microarray data if we treat each gene as a variable and treat the expression levels of all studied genes under a single condition as a (multivariate) sample.

Clustering-based algorithms are widely applied in microarray data analysis. The two most popular clustering methods, hierarchical clustering algorithm ([5]) and K-means clustering algorithm ([6]), have been used to find biologically meaningful gene clusters from expression data by many previous researchers. [2] and [17] have developed more advanced clustering-based approaches to discover genetic regulatory networks from multiple data sources.

Although clustering analysis is effective and straightforward, it has several limitations. It is often tricky to determine the number of clusters and the partial correlation (conditional independence relationship) between mRNA levels of genes cannot be modeled.

One solution to overcome this is to use a Bayesian network (BN), which is defined as a directed acyclic graph associated with local conditional distributions, as the model of gene network ([8], [11], [3]). Each node in a BN represents a gene and each directed edge represents an interaction between two genes. Several algorithms (usually called "structure learning algorithms"), like Sparse Candidate Hill Climbing and Grow-Shrinkage algorithm, have been developed to learn the structures of BNs with thousands of nodes from data.

One problem of modeling a regulatory network as a BN lies on the fact that the protein levels are unobservable from microarray data (Fig. 1). Most BNs proposed so far only include mRNA levels of genes as nodes, but not protein levels. While the mRNA levels of co-regulated genes are often correlated, the mRNA levels of the regulated genes and regulator genes may not always correlate to each other because the mRNA expression levels of regulator genes are not always good indicators of their protein levels and activities. Even when the mRNA levels of regulator genes and regulated genes do correlate, the mRNA levels of co-regulated genes are generally not conditionally independent to each other given the mRNA levels of the regulator genes. Generally speaking, there are two kinds of dependencies among mRNA levels of genes in microarray data. One is the dependency between regulator genes and regulatee genes. The other is the dependency among co-regulated genes. A BN directly inferred from mRNA profiles may confound these two kinds of dependencies.

[11] and [3] have used genome wide location analysis data [1] as priors to constrain the structures

[1] Location analysis identifies physical interactions between regulators and motifs on the regulatory regions of the

of BNs so that the edges corresponding to the dependencies of co-regulated genes are filtered out. Such an approach implicitly assumes the presence of (relatively strong) correlations between mRNA levels of the regulator genes and the regulated genes, whereas ignores the dependencies among genes that are co-regulated, which are often the strongest signals in microarray data and may be very useful.

In this paper, we integrate the microarray data and location analysis data in a different way. Our approach learns the genetic regulatory network in two steps. In the first step, instead of directly modeling the regulator-regulatee dependencies as in BNs, we only attempt to model the dependencies among co-regulated genes using graphical Gaussian model (GGM). Each edge of the undirected graph encoded by the GGM corresponds to the dependencies between a pair of co-regulated genes. Fig 2 shows an example of such a model. We have proposed a novel hierarchical Bayesian prior for the GGM so that the MAP estimation of the undirected graph has a sparse structure and can be learned efficiently by adjusted Lasso regression from microarray data. Of course in practice our algorithm can not completely exclude edges that may correspond to the dependencies between regulators and regulatees. In this case, we rely on the location analysis data to distinguish the two kinds of edges in the next stage. In the second step, we take the learned undirected graph together with a genome-wide location analysis dataset (i.e., the ChIp-ChIp data) as the input. A post-processing algorithm is developed to calculate the probability that genes connected in the estimated undirected graph have some shared TF set in location analysis data by chance. If this probability is less than a p-value threshold, we would infer these genes be regulated by the shared TF set.

The advantage of our approach lies on the fact that the dependencies among co-regulated genes are often much stronger and more robust than the dependencies between regulator genes and regulatee genes in microarray data. Thus by exploiting the co-regulation dependency information, which is not used in BNs, we may discover more regulatory patterns and we no longer need to assume the presence of detectable statistical correlations between mRNA levels of regulator genes and regulated genes.

2 Related work in structure learning for undirected graphs

Graphical Gaussian model ([12]) defines a joint distribution associated with a set of variables related in accordance to an undirected graphical model. Let P be the number of nodes in the graph. Then in GGM, the vector of the P nodes $X = (x_1, ..., x_P)$ is assumed to follow multivariate normal distribution with covariance matrix Σ. An edge between two nodes (variables) in the undirected graph encoded by the GGM implies that the two variables are NOT conditionally independent to each other given the rest variables. Matrix $\Omega = \Sigma^{-1}$ is often called the *precision matrix*. The non-zero entries in matrix Ω correspond to the edges of the undirected graph.

There are two basic categories of approaches for structure learning: score-based approaches and constraint-based approaches. *Score based approaches* perform a search through the space of

genes. However, physical binding is not equivalent to regulatory interaction. Furthermore, location analysis itself is also very noisy. Thus, by combining these two data sources, we may recover modules more accurately than using either one alone.

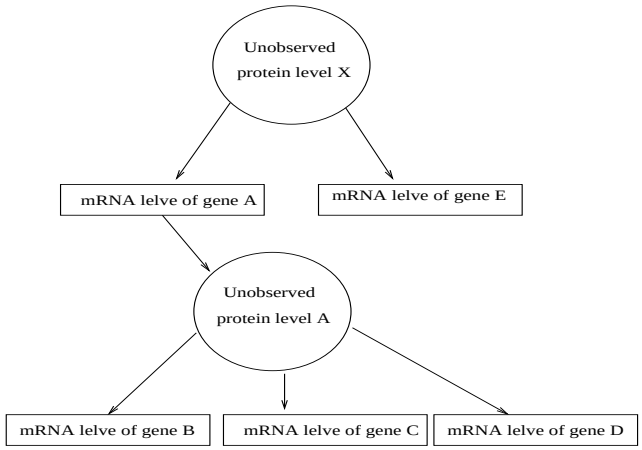

Figure 1: A typical sample of gene regulatory network

possible structures and attempt to identify the most probable graph given the data and the prior knowledge. When the graph is not decomposable, the space of possible structures that needs to be explored is super exponential to the number of nodes. Thus directly learning a GGM [2] with thousands of nodes in this way is very difficult. For decomposable graphs (such as BNs), several large scale score-based structure learning algorithms (like Sparse Candidate Hill Climbing) are already available. *Constraint-based approaches* use conditional independence tests to tell whether an edge between two variables exists or not. When the sample size is small (which is always the case in microarray data), it is difficult to test the partial correlation between two variables given all the other variables. Several possible solutions to circumvent this problem have been proposed. For example, [21] used GGM to model gene networks and used order-limited significance tests to identify edges (only first order partial correlations are considered). [16] proposed multiple testing procedures to identify edges in GGM. While constraint-based approaches are easy to implement, they lack an explicit objective function and do not try to directly find the GGM with maximal likelihood. Therefore, they are unable to directly produce a consistent joint distribution of the data.

Several studies have also used regression-based approaches to identify structures of undirected graphs. The idea is to apply regression analysis for each variable on the rest variables and the regression coefficients with relatively large absolute values correspond to edges in the graph. Lasso regression ([20]) is often preferred because it gives exactly zero regression coefficients to most irrelevant or redundant variables (thus leading to a sparse graph structure). Furthermore, theoretical analysis in [15] and [14] shows that using L1 regularization, the sample complexity (i.e., the number of training examples required to learn "well") grows only logarithmically in the number of irrelevant features. These results suggest that L1 regularized Lasso regression can be effective even if there are exponentially many irrelevant features as there are training examples (which is exactly the case in microarray data). [14] used Lasso regression to estimate undirected graph structures

[2] In this paper, when we say "learn a GGM" or "learn a DAG", we mean learning the parameters of a GGM (precision matrix) or a DAG (regression coefficients). The graph structures are encoded by these parameters. The non-zero entries in the precision matrix correspond to the edges in the undirected graph and the non-zero regression coefficients correspond to the edges in the DAG.

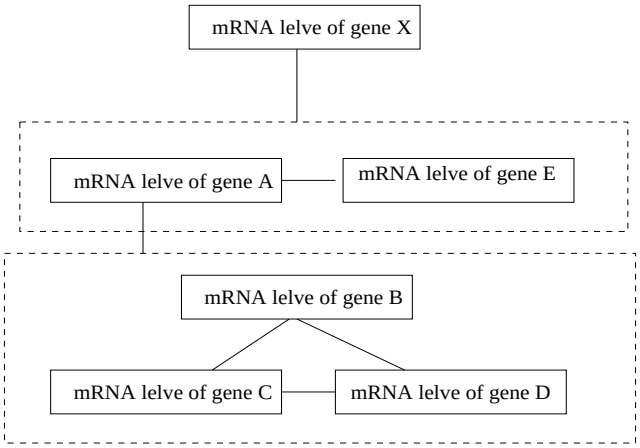

Figure 2: An undirected graph learned in our approach corresponding the regulatory network in Fig 1. The dot-lined boxes represent modules of co-regulated genes. The gene connecting to a module encodes the TF that regulates the genes in that module.

and [10] used similar approaches to recover large scale transcriptional regulatory network from expression microarray data. It is worth noting that the networks estimated using *ad hoc* regression approaches usually do not correspond to a likelihood-based estimator of the GGM. From extant approaches based on the Lasso regressions, to our knowledge, there have been few attempts to explicitly identify the exact global objective function underlying the algorithm and the probabilistic underpinning of such objectives. Indeed, our analysis in the sequel suggests that an *ad hoc* Lasso regression will not lead to a consistent estimation of the GGM (i.e., the regression coefficient of variable i on j may be zero while the regression coefficient of variable j on i is non-zero).

Recently, [4] proposed an alternative way to learn GGM by using score based approach, motivated by the following well-known facts about GGM. For any undirected graph whose variables (nodes) follow a joint multivariate Gaussian distribution, there must be a directed acyclic graph (DAG) whose variables have exactly the same joint distribution (because a multivariate Gaussian distribution can be always decomposed into a set of one-dimensional conditional Gaussian distributions using chain rule). Thus, instead of learning the undirected graph directly, one can first learn a DAG with the largest posterior probability. Once this DAG is found, the joint multivariate Gaussian distribution of the variables in this DAG and its associated precision matrix follow suite. This precision matrix will be exactly the one in the GGM we are trying to learn. To favor sparsity, [4] introduced a prior penalizing the number of edges in the DAG and used a heuristic search to find the MAP estimations of the DAG parameters. Our work follows this basic idea (learning GGM through DAGs) but uses a very different, arguably more natural and consistent strategy, to achieve a sparse estimation of the precision matrix in GGM.

Notice that since our ultimate goal is to learn a sparse undirected graph and DAG learning is only an intermediate step, there are some important differences between our way learning the DAG and the standard Bayesian network (BN) learning algorithms.

- There is no simple mapping between the sparsity of an undirected graph and the sparsity

of its corresponding DAGs. An undirected graph corresponds to many DAGs with different variable orders, some may be sparse and some may be dense. Thus the topology of undirected graphs and their corresponding DAGs may be very different (the relationship is shown in formula 1). In fact, some undirected graphs only have corresponding DAGs with many more edges, no matter which variable order we use (a loop with more than three nodes is a simple example). Thus, conceptually, it is NOT necessarily the best idea to enforce sparsity on the corresponding DAGs in order to achieve the sparsity of the undirected graph. A more natural way is to define a unified sparsity prior for the precision matrix and derive the corresponding DAG priors, rather than define the DAG priors directly.

We propose an Automatic Relevance Determination (ARD) style Wishart prior for the precision matrix Ω of the GGM. This kind of ARD priors have been introduced in [7] in a single regression model. In this paper, we extend it with the addition of an exponential hyper prior for the Wishart parameters and apply it to structure learning of a graph. We show that such a hierarchical prior leads to a proper prior for the regression coefficients that define the DAG subject to sparsity bias, which is skin to but richer than the Laplace priors corresponding to the Lasso regression algorithm. The MAP estimation of the precision matrix of GGM under this setting can be found efficiently by solving (a modified version of) Lasso regressions using fast grafting algorithm (proposed in [18]). Our approach produces a consistent probabilistic model over a high-dimensional variable space and naturally leads to a bias toward sparse topology of the corresponding undirected graphical model. It also enjoys the robustness and computational efficiency due to the Lasso regression procedure. Another important merit of using 1-norm regularizer (implied by Lasso regression in our approach) compared to the zero-norm regularizer scheme (implied by the term penalizing the number of edges in the DAG) is that, in each regression model in our approach, the solution space is convex and we can find the global optimal point efficiently, while the solution space in the zero-norm regularizer case is very bumpy. We further designed a method to discover genetic regulatory networks with more than 6000 nodes from real biological data based on our undirected graph learning algorithm. Evaluated on the basis of GO annotations, our results are significantly better than the state-of-art baseline results.

3 Hierarchical Bayesian learning of the GGM

3.1 The model

We attempt to learn the GGM with maximal posterior probability from training data. For simplicity, we assume that all the variables have been preprocessed and each of them follows a standard normal distribution. Thus the joint distribution of all the variables is a multivariate normal distribution. Since the topology of the undirected graph encoded by the GGM is completely determined by the precision matrix of the corresponding multivariate normal distribution, structural learning of a GGM is equivalent to learning the precision matrix of the multivariate normal distribution. In particular, we seek to learn the precision matrix that maximize the product of its prior probability and the probability of the training data under the GGM.

Given an ordering of the variables to be modeled, using chain rule, any high-dimensional multivariate distribution can be factored into a sequence of conditional distributions (i.e., $P(x_1, x_2, \ldots, x_n) = P(x_1)P(x_2|x_1)P(x_3|x_2, x_1)\ldots$), and therefore can be equivalently represented by a DAG built on edges from the parental nodes (corresponding to the variables been conditioned on) to the children nodes (corresponding to the conditioning nodes) induced by the variable ordering and the chain rule. In presence of conditional independences between certain variables, such a DAG may be *sparse*. Since learning a high-dimensional precision matrix of the GGM directly may be hard, especially when certain bias (e.g., sparsity) is to be introduced, it is possible and potentially advantageous to solve the structural learning problem for GGM indirectly by first learning its corresponding DAGs and then recover the precision matrix of the GGM. In the following, we formalize these ideas.

For any DAG, there is a specific ordering of variables, which we take here for discussion as simply $1, 2, \ldots, p$ without loss of generality. For any variable x_i, only variables whose indexes are larger than i could be considered as x_i's parents. We use $X_{(i+1):p}$ to represent these variables. Let's use β to represent the regression coefficients of the DAG. Notice once β is known, the DAG is determined. The probability of data given the DAG parameter β is

$$p(X|\beta) = \prod_{i=1}^{p} P(x_i|X_{(i+1):p)}, \beta).$$

For each x_i, we can have

$$x_i = \sum_{j=(i+1):p} \beta_{ji} x_j + \epsilon_i,$$

where $\epsilon_i \sim N(0, \psi_i)$. This can be further summarized in a matrix form

$$X = \Gamma X + \epsilon \quad and \quad \epsilon \sim N_p(0, \psi).$$

Where $\epsilon = (\epsilon_1, ..., \epsilon_p)'$, $\psi = \text{diag}(\psi_1, ..., \psi_p)$ and Γ is the $p \times p$ upper triangular matrix with zero diagonal elements and upper triangular, non-diagonal entries $\Gamma_{ij} = \beta_{ji}, (j > i)$.

It is easy to show that the joint distribution of variables in this DAG is a Gaussian distribution with precision matrix

$$\Omega = (I - \Gamma)'\psi(I - \Gamma). \tag{1}$$

Notice this precision Ω is just the precision matrix of the undirected graph we are trying to learn and it encodes the structure of the undirected graph.

3.2 Hierarchical Bayesian priors for GGM and connections to Bayesian regression

Now our goal reduces to estimating the DAG parameters β so that $P(\beta|Data) \propto P(Data|\beta)P(\beta)$ is maximized. Notice that covariance matrix Σ (or precision matrix Ω) together with an ordering of the variables determines the regression parameter β in the DAG (and of course, β also determines

the Σ and Ω). Thus a prior over Σ (or Ω) also translates to a consistent prior on any regression parameters $\beta_{ji}|j > i$.

Here, for precision matrix Ω, we define a Wishart prior $\Omega \sim W_p(\delta, T)$ with δ degrees of freedom and diagonal scale matrix

$$T = \text{diag}(\theta_1, ..., \theta_p)$$

The reason we want to use a diagonal matrix as the scale matrix is that we want to encourage a sparse precision matrix. In other words, without seeing the data, we assume the off-diagonal elements of the precision matrix are all zeros, which means that there are no edges between different nodes. Furthermore, we introduce an exponential hyper-prior for the Wishart parameters θ_i:

$$P(\theta_i) = \frac{\gamma}{2}\exp(\frac{-\gamma\theta_i}{2}). \tag{2}$$

This type of hyper prior distributions have been suggested by [7] in a single regression model.

Let's use β_i to represent the $(p-i)\times 1$ matrix $\beta_i = (\beta_{(i+1)i}, ...\beta_{pi})'$. Let's use T_i to represent the sub-matrix of T corresponding to variables $X_{(i+1):p}$. From the derivations in [9] [3], we obtain the marginal prior for β_i (which is a row in the Γ matrix in Eq. (1)):

$$P(\beta_i|\psi_i, \theta_{(i+1):p}) = N_{p-i}(0, T_i\psi_i). \tag{3}$$

Thus

$$P(\beta_{ji}|\psi_i, \theta_j) = N(0, \theta_j\psi_i). \tag{4}$$

Also following [9], we can derive the prior for ψ_i:

$$P(\psi_i^{-1}|\theta_i) = \text{Gamma}(\frac{\delta+p-i}{2}, \frac{\theta_i^{-1}}{2}). \tag{5}$$

As done in [7], we can integrate out the hyper parameter θ from prior distribution of β_{ji} and get

$$\begin{aligned} P(\beta_{ji}|\psi_i) &= \int_0^\infty P(\beta_{ji}|\psi_i, \theta_j)P(\theta_j)d\theta_j \\ &= \frac{1}{2}\sqrt{\frac{\gamma}{\psi_i}}exp(-\sqrt{\frac{\gamma}{\psi_i}}|\beta_{ji}|). \end{aligned} \tag{6}$$

Now, given K samples $\{x_1, \ldots, x_K\}$, and let x_{ki} represents the value of the ith variable in the kth sample. Using the Bayes rule, we can derive the posterior distribution of the regression coefficients and the variance of each regression function:

$$\begin{aligned} &P(\beta_i|\psi_i, X) \propto P(x_i|X_{(i+1):p}, \beta_i, \psi_i)P(\beta_i|\psi_i) \\ &\propto \exp(-\frac{\sum_k(x_{ki} - \sum_{j=(i+1)}^p \beta_{ji}x_{kj})^2 + \sqrt{\gamma\psi_i}\sum_{j=i+1}^p|\beta_{ji}|}{\psi_i}). \end{aligned} \tag{7}$$

[3] Notice the different notations in this paper and in [9]. The scale matrix T in this paper is detonated as T^{-1} in their work. The second parameter in normal distribution represents the covariance matrix in this paper and precision matrix in their work.

and

$$P(\psi_i^{-1}|\theta_i, \beta_i, X) = \\ \text{Gamma}(\frac{\delta + p - i + K}{2}, \frac{\theta_i^{-1} + \sum_k (x_{ki} - \sum_{j=i+1}^{p} \beta_{ji} x_{kj})^2}{2}). \tag{8}$$

These regression functions determine the conditional Gaussian distributions defined on the DAG.

3.3 The adjusted Lasso regression

From Eq. (7), we see the MAP estimation of β_i is

$$\hat{\beta}_i = \arg\min \quad \sum_k (x_{ki} - \sum_{j=(i+1)}^{p} \beta_{ji} x_{kj})^2 \\ + \sqrt{\gamma \psi_i} \sum_{j=(i+1)}^{p} |\beta_{ji}|. \tag{9}$$

Thus $\hat{\beta}_i$ is the solution of a Lasso regression.

However, we cannot solve Eq. (9) using a standard Lasso regression. The reason is that ψ_i with different i values are generally not the same. Thus the regularization parameter $\sqrt{\gamma \psi_i}$ can not be treated as a single parameter. In order to estimate β_i using Lasso regression, we need to first estimate the regularization parameter $\sqrt{\gamma \psi_i}$. Thus we refer to the regression problem defined by Eq. (9) as "adjusted Lasso regression".

Eq. (7) provides the posterior distribution of β_i conditioned on ψ_i while Eq. (8) provides the posterior distribution of ψ_i conditioned on β_i. This essentially defines a fixed point for our Bayesian regression problem Eq. (9). Note that θ_i in Eq. (8) could be numerically integrated out by sampling method [4]. An iterative procedure is used to estimate $\hat{\beta}_i$ and $\hat{\psi}_i$. We start from an initial guess $\hat{\psi}_i = 0$. Using this $\hat{\psi}_i$ value, we can estimate $\hat{\beta}_i$. Then using the estimated $\hat{\beta}_i$, we estimate $\hat{\psi}_i$ again. This procedure is iterated until the estimation $\hat{\psi}_i$ and $\hat{\beta}_i$ converge. Once $\hat{\psi}_i$ and $\hat{\beta}_i$ are known, we can estimate Ω using formula 1 easily. Thus the graph structure is learned.

In the real implementation, we incorporate the iterative procedure into the *fast grafting* algorithm that is used to solve Lasso regression. Fast grafting is a kind of stage-wise gradient descent algorithm. It uses the sparseness property of Lasso regression to accelerate the optimization. The algorithm starts with two feature sets: free feature set F and fixed zero feature set Z. Then it gradually moves features from Z to F while training a predictor model only using features in F. The details of this algorithm are referred to [18]. In order to incorporate the iterative procedure adjusting the regularization parameter into the fast grafting algorithm, our implementation is a little different from [18]. We initialize ψ_i as zero value. In the end of each iteration in the fast grafting algorithm, ψ_i and the regularization parameter are re-estimated based on the current regression coefficients β_i. With the fast grafting algorithm, the computational cost for graph learning can be significantly reduced.

[4] Ideally, we'd like to multiply Eq. (2) and Eq. (8) and analytically integrate out θ_i. However, it's unclear to us whether θ_i can be integrated out in a close form. Thus we just sample θ_i values under the distribution in Eq. (2) and use these values to simulate the integration operation. Since θ_i (with different i values) all follow the same distribution in Eq. (2), the sampled θ_i values can be used repeatedly with different i values.

3.4 More on the hyper parameter γ

In Lasso regressions, most variables would be assigned zero weights (i.e., $\beta_{ji} = 0$). In particular, for the ith adjusted Lasso regression model defined in Eq. 9, for an arbitrary variable x_q that $q > i$, the solution of Lasso regression would satisfy: (For details, see [18].)

$$2|\sum_k (x_{ki} - \sum_{j=i+1}^{p} \beta_{ji} x_{kj}) x_{kq}| \leq \sqrt{\gamma \psi_i},$$

where equality only holds when variable x_q has a non-zero weight.

Let $r_{ki} = x_{ki} - \sum_{j=i+1}^{p} \beta_{ji} x_{kj}$, We treat $r_{1i}, r_{2i}, ..., r_{Ki}$ to be samples of random variable r_i (representing the residual of variable i). Similarly, we treat $x_{1q}, ..., x_{Kq}$ to be samples of random variable f_q (representing variable q). Notice that all the variables have been preprocessed so that they all follow standard normal distribution. We use cor(a, b) to represent the correlation coefficient between variable a and b. Thus we have

$$\text{cor}(r_i, f_q) \leq \frac{\sqrt{\gamma}}{2K},$$

where again equality only holds for variable x_q's with non-zero weights.

This implies that, given the non-zero weighted variables, the correlation coefficient between left (zero weighted) variables and the residue is no larger than $\frac{\sqrt{\gamma}}{2K}$. This gives us an intuitive way when we want to decide the value of the hyper prior γ.

4 Identifying regulators and co-regulated modules from GGM

So far, we have learned an undirected graph encoded by the GGM from microarray data. The edges in the undirected graph are supposed to correspond to the co-regulated gene pairs. In this section, we outline a method to identify the regulator genes and the modules of co-regulated genes from the estimated GGM with the help of genome-wide location analysis data (i.e., the ChIp-ChIp data).

Location analysis identifies physical interactions between regulators and motifs on the regulatory regions of the genes. It consists of a matrix whose rows are all the possible regulatee genes and columns are all the possible regulators. The elements of the matrix are the P-values reflecting the chance that a physical interaction happens between the regulator and regulatee gene.

For any gene x_i, let's assume there are K genes connecting to it in the previously estimated undirected graph. Let's suppose in these K genes, there are L genes that share some regulator set with gene x_i in the location analysis data. We use R to represent the shared regulator set. Now we want to calculate the probability that the $L + 1$ genes (L genes plus gene x_i itself) share the same regulator set R by chance. In this paper, we use a binomial distribution to approximate this probability.

Let's use N to represent the total number of genes. Let's use M to represent the number of genes which are possibly regulated by the regulator set R in the location analysis data. Then, the probability of a randomly selected gene being regulated by regulator set R is $\frac{M}{N}$.

Thus, in the $K+1$ genes (K genes connecting to gene x_i plus gene x_i itself), the probability that $L+1$ or more than $L+1$ genes share the same regulator set R by chance is

$$score = \sum_{j=L+1}^{K+1} \frac{(K+1)!}{(K+1-j)!j!}(\frac{M}{N})^j(1-\frac{M}{N})^{K+1-j}. \quad (10)$$

If this score is lower than a pre-defined p-value threshold (we use 0.05 in this paper), we will infer regulator-regulatee dependencies between regulator set R and these $L+1$ genes.

The pseudo code of the post-processing algorithm is shown as Algorithm 1.

Algorithm 1 Pseudo code of the post-processing algorithm

1. For each gene x_i
   (a) Get all the genes connecting to gene x_i in the estimated undirected graph. Use Neighbor(x_i) to represent this gene subset. Suppose the size of Neighbor(x_i) is K.
   (b) From the protein-binding location analysis data, find all the TFs that are shared by x_i and one or more genes in Neighbor(x_i). For each shared TF set R
      i. Suppose there are N genes in total. Suppose there are M genes regulated by TF set R. Suppose in Neighbor(x_i), there are L genes regulated by TF set R. Use $S(R, \text{Neighbor}(x_i))$ to represent these L genes. Use formula 10 to calculate a p-value. If this p-value is less than a p-value threshold, then we would infer that the gene x_i and genes in $S(R, \text{Neighbor}(x_i))$ form a co-regulated module, which is regulated by genes in TF set R

5 The algorithm

Our approach discovers regulatory networks in two steps. In the first step we infer an undirected graph from microarray data using the adjusted Lasso regression approach for hierarchical Bayesian learning of GGM . Each edge in the learned undirected graph represents a co-regulated gene pair. We first use the heuristic search described in [4] to find an order for the DAG. Then we use adjusted Lasso regressions described in section 3.3 to estimate the DAG parameters. With these estimated DAG parameters, the precision matrix, which encodes the structure of the undirected graph, can be calculated using Eq. 1. In the second step we take the estimated undirected graph together with the protein-binding location analysis data as the input, and use post-processing algorithm to identify the regulators and the modules of co-regulated genes from the estimated undirected graph.

The overall pseudo code of our approach is given as following:

6 Experiments

We applied our method on the Yeast Saccharomyces cerevisiae dataset. The expression microarray data we used comes from [19]. The mRNA expression levels of 6177 genes are measured under

Algorithm 2 Overall pseudo code

1. Use a heuristic search to find an order for the DAG
2. Estimate the regression coefficients β and the variances ψ in the DAG using adjusted Lasso regressions in formula 9.
3. Calculate the MAP estimation of the precision matrix according to formula 1, which encodes the undirected graph structure.
4. Use the post-procession algorithm in section 4 to infer the co-regulated gene modules and their TFs.

76 conditions. The location analysis data we used comes from [13]. It gives the p-values of genes regulated by each of the 106 TFs. When a p-value is lower than 0.05, we would assume it is possible that the gene is regulated by the TF.

In our experiment, the value of the hyper-prior γ is assigned to be 2.31 so that the correlation coefficients between zero weighted variables and the regression residue are no larger than 0.01. Another parameter is the p-value threshold of the post-processing algorithm, which can be used to control the number of output gene modules. Here we also set the p-value threshold to be 0.05.

Directly evaluating the estimated gene regulatory network is difficult because the true gene network is usually unknown. One possible solution is that, instead of directly evaluating the estimated gene regulatory network, we evaluate the gene regulatory modules in the estimated gene network. A gene regulatory module is defined to be a set of co-regulated genes sharing the same set of regulator genes. Genes in the same module are likely to attend the same cellular processes and have similar biological functions. Most yeast genes have been annotated with certain cellular processes or functions in GO database([1]). Thus it is possible to use this information as an external evidence to evaluate the estimated gene regulatory modules. In fact, such evaluation strategies have been widely used in recent years ([17]).

The details of the evaluation strategy are summarized as following. For each gene module and each GO annotation, we calculated the fraction of genes in that module associated with that GO annotation and used the hypergeometric distribution to calculate a p-value for this fraction (which can be done using SGD Term Finder Toolkit([1]), and took p-value < 0.05 to be significant. We compared the enrichment of modules for GO annotations between results achieved by our undirected graph structure learning algorithm and results achieved by other baseline algorithms. In this paper, we only consider GO annotations in "Process" category.

6.1 Co-regulated gene modules recovered by our method

We list the regulatory modules found by our approach with at least 2 TFs and five regulated genes in table 1.

Each line in table 1 represents a regulatory module found by our approach. The first column gives the regulator set of this module. The second column gives the number of regulated genes with annotations in GO database VS the number of all the regulated genes in this module. The third

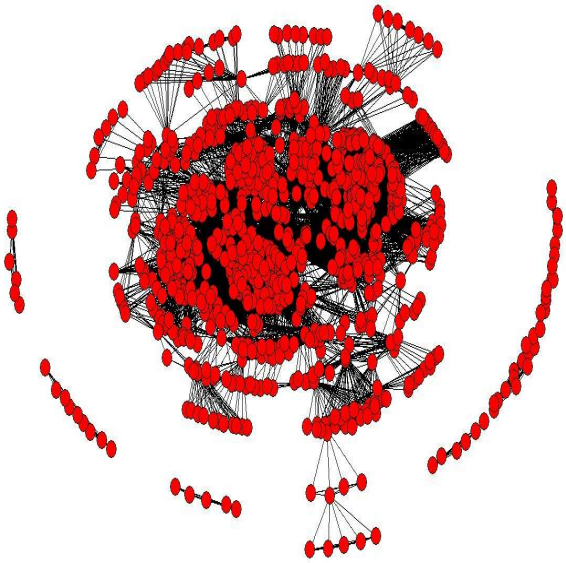

Figure 3: This is an undirected graph that is corresponding to all the gene regulatory modules(whose sizes are required to be no less than five) found by our algorithm. Each node represents a gene and each edge represents a co-regulated relationship between two genes. Each regulatory module corresponds to a clique. Notice that one gene can be in multiple regulatory modules. That's why many cliques are connected in the figure. The graph clustering coefficient is 0.433.

column gives the previous reference of the interactions among regulator genes in this module. The last column gives the GO annotations of processes shared by the regulated genes in this module (the names of the regulated genes are not shown to save space). We also give the P-values for these GO annotations, which are calculated by SGD Gene Ontology Term Finder Toolkit([1]). This value reflects the probability that the genes in the module share the GO annotation by chance. Since there are generally multiple GO annotations shared by a gene cluster, we only show the GO annotation with the smallest P-value for each module in the last column. We can see many of the co-regulators in our recovered modules have already been reported as co-TFs or have interactions by previous references. Some well studied co-TFs like (MBP1, SWI6) and (FKH2, MCM1, NDD1) can also be found in table 1.

There are a lot of other interesting patterns found by our approach that are not listed in table 1. For example, our approach found the module with regulator set (CAD1,PHO4) and the regulated gene set (CUP1-1,CUP1-2). This small module corresponds to the process "response to copper ion" on a significance level with $pvalue = 3.02e - 07$ (both CUP1-1 and CUP1-2 belong to this process, while in all of the 7276 annotated genes in the GO database, only 4 genes belong to this process).

In figure 3, we plot the undirected graph that is corresponding to all the gene regulatory modules(whose sizes are required to be no less than five) found by our algorithm. There are 957 nodes in this figure. Each node represents a gene and each edge represents a co-regulated relationship between two genes. Thus each regulatory module corresponds to a clique. Notice that one gene can

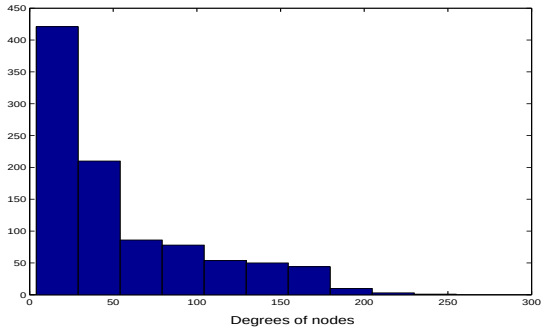

Figure 4: This is the histogram that shows the distribution of node degrees in fi gure 3. The X axis represents the degree of nodes. The Y axis shows the number of nodes with such degrees in each bin. We can see that the histogram approximately follows power law distribution.

Table 1: **modules with multiple regulators and more than 5 regulated genes.**

regulator set	annotated/ALL	reference of co-regulators	common processes or function of regulated genes
ACE2,SWI5	6/6		4/6 cell proliferation, p=0.023
ASH1,FKH2	4/5		2/4 cell wall organization and biogenesis, p=0.002
ASH1,SWI4	5/5		3/5 cell organization and biogenesis(p=5.94e-05),
CIN5,MET4	6/8		2/6 copper ion import, p=0.0002
DIG1,STE12	10/10	functional and physical	10/10 cellular process(p=1.45e-05),
FHL1,RAP1	100/100	share motif(Davide et al)	100/100 structural constituent of ribosome, $p < 1e-4$
FKH1,FKH2	4/5	co-TFs(Kato et al 2004)	2/4 covalent chromatin modifi cation, p=0.0003
FKH2,MCM1,NDD1	12/12	co-TFs(CYGD)	6/12 cell proliferation, p=0.01
GAT3,RGM1	3/13		2/3 telomerase-independent telomere maintenance, p=9.56e-06
GCR1,GCR2,RAP1	6/6		6/6 energy pathways, p=1.65e-08
HIR1,HIR2	6/6	co-TFs(Spector et al 1997)	6/6 chromatin assembly or disassembly, p=1.23e-14
HSF1,MSN4	6/8	co-TFs(Jeffrey et al, 2002)	4/6 response to stress, p=9.62e-05
INO2,INO4	6/7	co-TFs(BRIAN et al 1995)	2/6 translation elongation factor activity, p=4.76e-05
MAL13,MSN4,RGM1	5/7		3/5 telomerase-independent telomere maintenance, p=1.05e-6
MBP1,SWI4	8/8		3/8 microtubule cytoskeleton organization and biogenesis, p=0.005
MBP1,SWI6	32/37	functional and physical interaction(CYGD)	21/32 cell cycle, p=4.74e-17
MET31,MET4	8/8	in the same complex(Pierre-Louis et al 1998)	6/8 sulfur metabolism, 9.78e-11
MET4,RAP1	6/6		6/6 macromolecule biosynthesis, p=4.24e-06
NDD1,SKN7	5/5		5/5 cell organization and biogenesis, p=9.35e-05
NDD1,SWI4	5/5		2/5 cytoskeleton organization and biogenesis, p=0.012
SWI4,SWI6	15/19	functional and physical interaction (CYGD)	7/15 cell cycle p=3.57e-05

be in multiple regulatory modules. That's why many cliques are connected. The graph clustering coefficient is also given in figure 3. In figure 4, we plot the histogram that shows the distribution of node degrees for the undirected graph in figure 3. We can see that the histogram approximately follows power law distribution.

6.2 Comparison between our approach and other approaches

[2] has used GRAM algorithm, which can be thought as a kind of clustering algorithms, to extract gene regulatory modules from expression microarray data and location analysis data. In particular, they have used the same genome-wide location analysis data as the one we used. However, their expression microarray data is over 500 experimental conditions and our microarray data, with 76 conditions, is only a subset of theirs. [5] We will use their result as a baseline result.

[5] [2] combined several microarray data sources in different conditions to get a big microarray data over 500 conditions in their paper. Thus the cell cycle microarray data from [19], which we used in our paper, is a only a subset of it.

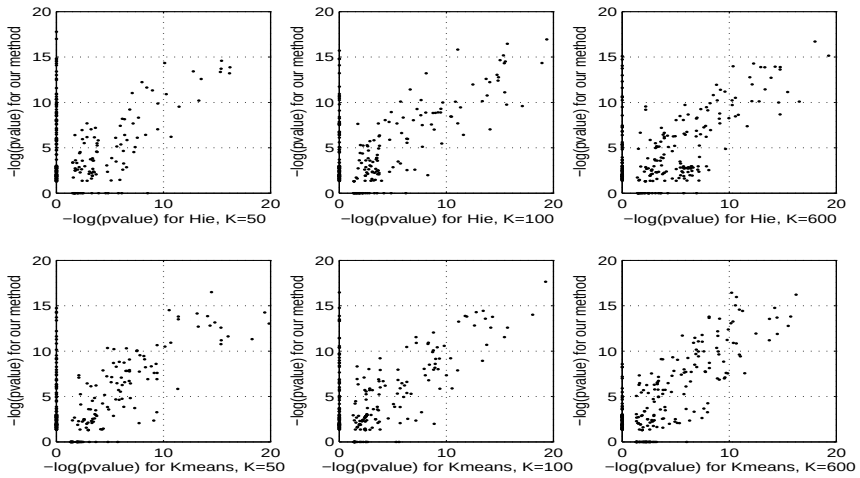

Figure 5: Comparison between our approach and Hierarchy clustering (in the first row) and K-means clustering (in the second row), with k=50 (first column), 100 (second column), 600 (third column) respectively. In the up left figure, there are 109 dots on the Y axis and 22 dots on the X axis. In the up middle figure there are 121 dots on the Y axis and 19 dots on the X axis. In the up right figure, there are 91 dots on the Y axis and 22 dots on the X axis. In the down left figure, there are 119 dots on the Y axis and 29 dots on the X axis. In the down middle figure, there are 112 dots on the Y axis and 27 dots on the X axis. In the down right figure, there are 92 dots on the Y axis and 39 dots on the X axis.

We set the requirement that the size of a gene module should be no less than five. The p-value threshold of our post-processing algorithm, which can be used to control the number of output gene modules, is set to be 0.04 so that the number of output gene modules is 106, which is exactly the same as the number of gene modules in the results reported by [2]. In this way, we can compare our results with [2] conveniently.

We also compared our results to the results achieved by hierarchical clustering and k-means clustering algorithms. Since the number of clusters is a parameter that needs to be pre-defined in these two clustering algorithms, we tried several settings (k=50, 100, and 600) for this parameter. For the Hierarchical clustering method, we used the agglomerative average linkage version. For the K-means clustering method, we tried 5 random restarts and only reported the best result. Notice that clustering algorithms can be only applied on expression microarray data to group correlated genes. They cannot make use of location analysis data themselves. In order to be fair in comparing our method with these two clustering methods, we need some post-processing algorithms that can combine the clustering results with location analysis data. In fact, we can simply treat the clustering results as a disjoint undirected graph. Genes in same clusters form complete sub-graphs and genes in different clusters are disconnected with each other. Thus we can directly use the post-processing algorithm we described before to post-process clustering results. In order to be fair in comparing the two module extraction methods using this evaluation strategy, the number of gene regulatory modules found by these two methods should be the same. Thus for the two clustering methods, we tuned the p-value threshold of the post-processing algorithm so that it also returns 106 modules. In this way, we can give a fair comparison.

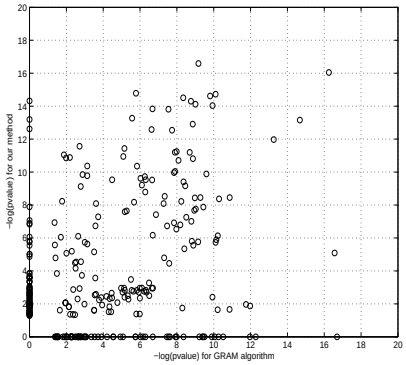

Figure 6: Comparison between our method and Gram algorithm in GRAM algorithm. There are 93 dots on the Y axis and 61 dots on the X axis.

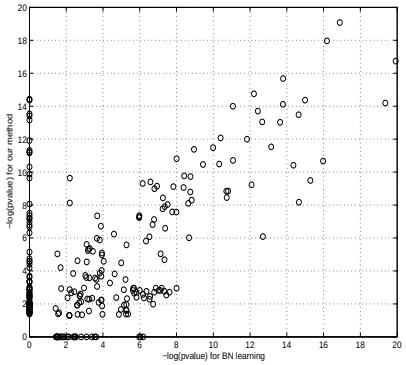

Figure 7: Comparison between our method and BN learning algorithm. There are 88 dots on the Y axis and 29 dots on the X axis.

We further compared our approach with the BN learning approaches that used the location analysis data as priors to constrain the BN structures. We used sparse candidate hill climbing as the search algorithm for BN learning and only allow edges from a regulator gene to a regulatee gene when the p-value between these two genes is lower than 0.05 in location analysis data. We tune the weight of the penalizer for the number of edges in the BN so that it will also generate 106 regulatory modules.

The results are shown in figure 5, 6 and 7. In these figures, each dot represents a GO annotation. If it is on the Y-axis, it means that the GO annotation has been enriched in our results but not in the baseline results. If it is on the X-axis, it means that the GO annotation has been enriched in the baseline results but not in our results. We can see that there are many more dots on the Y-axis than on the X-axis in figure 5, 6 and 7, which implies our method achieved a much better performance. Notice the microarray data used in [2] contains richer information than our data, but their results are still not as good as ours, as reflected in figure 6.

7 Conclusions

In this paper, we proposed a new formalism based on graphical Gaussian models to learn regulatory networks from microarray data and location analysis data. By introducing a hierarchical Bayesian prior for the precision matrix of the GGM, the MAP estimation of the precision matrix, which encodes the undirected graph structure to be inferred, is sparse and can be solved efficiently by adjusted Lasso regressions. Then in the second stage, a post-processing algorithm is used to identify the regulators and the co-regulated gene modules from the estimated GGM, with the help of location analysis data. Overall, our approach provides an efficient and biological more informative and comprehensible (i.e., yielding sparse network and gene modules) solution to the problem of gene network inference, doing so in a well-founded statistical framework that provides easy-to-understand global cost function (i.e., data likelihood) and consistent joint distribution of the data. We applied our approach to a real microarray dataset and a protein-binding location analysis dataset. An evaluation on the basis of consistency with the GO annotations shows our results significantly better than the results of clustering-based methods and BN learning methods.

References

[1] M. Ashburner, CA. Ball, JA. Blake, D. Botstein, and H. Butler. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

[2] Z. Bar-Joseph, GK. Gerber, T. Lee, and NJ. Rinaldi. Computational discovery of gene module and regulatory networks. *Nature Biotechnology*, 21(11):1337–42, 2003.

[3] A. Bernard and AJ. Hartemink. Informative structure priors: Joint learning of dynamic regulatory networks from multiple types of data. In *Pacific Symposium on Biocomputing*, Hawaii, 2005.

[4] A. Dobra, C. Hans, B. Jones, J.R. Nevins, G. Yao, and M. West. Sparse graphical models for exploring gene expression data. *J. Mult. Analysis*, 90:196–212, 2004.

[5] Brown PO. Eisen MB., Spellman PT. and Botstein D. Cluster analysis and display of genome-wide expression patterns. *Proc. Nat. Acad. Sci.*, 95:14863–14868, 1998.

[6] Tavazoie et al. Systematic determination of genetic network architecture. *Nat Genetics*, 22:281–285, 1998.

[7] M. Figueiredo and AK. Jain. Bayesian learning of sparse classifiers. In *CVPR*, pages 35–41, 2001.

[8] N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using bayesian network to analyze expression data. *Jounnal of Computational Biology*, 7(2002):175–186, 1996.

[9] D. Geiger and D.Heckerman. Parameter priors for directed acyclic graphical models and the characterization of several probability distributions. *Annals of Statistics*, 5:1412–1440, 2002.

[10] M. Gustafsson, M. Hornquist, and A. Lombardi. Large-scale reverse engineering by the lasso. In *ICSB 2003 Poster*, 2003.

[11] AJ. Hartemink, DK. Gifford, TS. Jaakkola, and RA. Young. Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. In *Pacific Symposium on Biocomputing*, Hawaii, 2001.

[12] S. Lauritzen. *Graphical Models*. Clarendon Press, Oxford, 1996.

[13] TI. Lee, NJ. Rinaldi, FD Robert, and DT Odom. Transcriptional regulatory networks in saccharomyces cerevisiae. *Science*, 298:799–804, 2002.

[14] N. Meinshausen and P. Buhlmann. Consistent neighbourhood selection for sparse high-dimensional graphs with lasso. Technical Report Seminar for statistic, ETH 123, Institute for the Study of Learning and Expertise, 2004.

[15] A. Ng. Feature selection, l1 vs l2 regularization, and rotational invariance. In *ICML*, 2003.

[16] J. Schafer and K. Strimmer. An empirical bayes approach to inferring large-scale gene association networks. *Bioinformatics in press*, 2004.

[17] E. Segal, R. Yelensky, and D. Koller. Genome-wide discovery of transcriptional modules from dna sequence and gene expression. *Bioinformatics*, 19:273–82, 2003.

[18] P. Simon, L. Kevin, and T. James. Grafting: Fast,incremental feature selection by gradient descent in function space. *JMLR*, pages 1333–1356, 2003.

[19] PT. Spellman, G. Sherlock, MQ. Zhang, and VR. Iyer. Compehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. molecular. *Biology of the Cell*, 9:3273–3297, 1998.

[20] R. Tibshirani. Optimal reinsertion:regression shrinkage and selection via the lasso. *J.R.Statist*, Soc. B(1996), 58,No.1:267–288, 1996.

[21] A. Wille, P. Zimmermann, E. Vranov, and A. Frholz. Sparse graphical gaussian modeling of the isoprenoid gene network in arabidopsis thaliana. *Genome Biology*, 5:92, 2004.