Using modified Lasso regression to learn large undirected graphs in a probabilistic framework

Fan Li
LTI, SCS, Carnegie Mellon University
4502 NSH, 5000 Forbes Ave.
Pittsburgh, PA 15213
hustlf@cs.cmu.edu

Yiming Yang
LTI, SCS, Carnegie Mellon University
4502 NSH, 5000 Forbes Ave.
Pittsburgh, PA 15213
yiming@cs.cmu.edu

Abstract

Learning the structures of large undirected graphs with thousands of nodes from data has been an open challenge. In this paper, we use graphical Gaussian model (GGM) as the underlying model and propose a novel ARD style Wishart prior for the precision matrix of the GGM, which encodes the graph structure we want to learn. With this prior, we can get the MAP estimation of the precision matrix by solving (a modified version of) Lasso regressions and achieve a sparse solution. We use our approach to learn genetic regulatory networks from genome-wide expression microarray data and protein-binding location analysis data. Evaluated on the basis of consistency with the GO annotations, the experiments show that our approach has a much better performance than the clustering-based approaches and BN learning approaches in discovering gene regulatory modules.

Key words: data mining, machine learning, Bayesian networks.

Introduction

Learning the structures of large undirected graphs with thousands of nodes from data is an open challenge in machine learning and has many potential applications. One recent promising direction is to use this technique to induct gene networks from high-throughput microarray data ((Gustafsson, Hornquist, & Lombardi 2003), (Schafer & Strimmer 2004),(Wille *et al.* 2004)).

Graphical Gaussian model (GGM) ((Lauritzen 1996)) is a popular undirected graphical model that has been frequently used in recent years. 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 (in other words, the partial correlation between these two variables is not zero). Matrix $\Omega = \Sigma^{-1}$ is often called the Precision matrix. The non-zero entries in matrix Ω are corresponding 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 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 will be extremely large. Thus directly learning a GGM [1] with thousands of nodes in this way is very difficult. *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, (Wille *et al.* 2004) used GGM to model gene networks and used order-limited significance test to identify edges (only first order partial correlations are considered). (Schafer & Strimmer 2004) did a similar work but used multiple testing procedures to identify edges in GGM. While these constraint based approaches are interesting and helpful, they lack an explicit objective function and do not try to directly find the GGM with maximal likelihood. Thus they do not fit into the probabilistic framework as well as score based approaches.

Several researchers 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 are explained as edges in the graph. Lasso regression ((Tibshirani 1996)) is often preferred in recent years because it gives exactly zero regression coefficients to most irrelevant or redundant variables (thus leading to a sparse graph structure). Furthermore, theoretical analysis in (Ng 2003) and (Meinshausen & Buhlmann 2004) 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. This conclusion indicates that L1 regularized

[1] 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.

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). (Meinshausen & Buhlmann 2004) used Lasso regression to estimate undirected graph structures and (Gustafsson, Hornquist, & Lombardi 2003) used similar approaches to recover large scale transcriptional regulatory network from expression microarray data. However, the network estimated by using *ad hoc* regressions does not correspond to a likelihood-based estimator of the GGM. From existing approaches based on the Lasso regressions, there has been few attempts to explicitly identify the exact global objective function and the probabilistic framework for the objective. [2].

(Dobra *et al.* 2004) proposed an alternative way to learn GGM using score based approach. The idea is that, 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 Gaussian distributions using chain rule). Thus, instead of learning the undirected graph directly, we 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 can be estimated. This precision matrix will be exactly the one in the GGM we are trying to learn. (Dobra *et al.* 2004) defined a prior penalizing the number of edges in the DAG and they heuristically searched an order of variables in the DAG so that the MAP estimations of the DAG parameters have the largest posterior probability. Our work follows the basic idea of learning GGM by DAGs but uses a very different strategy to achieve a sparse estimation of the precision matrix in GGM.

Note that the ultimate goal of standard Bayesian network (BN) learning is to find a DAG that best explains the data. However, the ultimate goal in this paper is to learn a sparse undirected graph and DAG learning in our approach is only an intermediate step. Thus there are some important differences between our way learning the DAG and the standard BN learning algorithms.

- In our model, the priors of the DAG parameters are derived from a unified global prior for the undirected graph. Notice that there is not a simple mapping between the sparsity of an undirected graph and the sparsity of its corresponding DAGs. The topologies of an undirected graph and its corresponding DAGs may be very different. An undirected graph can correspond to many DAGs with variables in different orders: some DAGs may be sparse and others may be dense. In fact, some undirected graphs only have corresponding DAGs with many more edges, no matter how the variables are ordered. A loop with more than three nodes is a simple example of that. Thus, conceptually, it is not always a good strategy to enforce sparsity on the corresponding DAGs in order to obtain the sparsity in the ultimate undirected graph. A more natural and consistent approach (and our approach) is to define a sparsity prior for the precision matrix that encodes the undirected graph, and then derive the DAG priors correspondingly, instead of defining the DAG priors directly, which has been typically used in BN learning algorithms.
- In our model, we do not need to search the order of variables in the DAG because the finial MAP estimation of the precision matrix does not depend on the variable order in the DAG. In standard BN learning, the ultimate goal is to learn the topology of the BN, which will be influenced by the order of the variables. Thus this order should be learned automatically from the data. However, in our model, the ultimate goal is to learn the undirected graph structure, which is determined by the precision matrix of the multivariate Gaussian distribution. A multivariate Gaussian distribution can be re-written as the product of a set of one-dimensional Gaussian distributions using chain rule in an arbitrary order. This implies that, for DAGs with different variable orders, as far as the DAG parameters are all learned using MAP estimations and the priors of the DAG parameters are all derived from a unified global prior (not depending on the variable order) for the precision matrix, the finial recovered joint distributions of the variables in these DAGs should all be the same. In other words, these DAGs with different variable orders should all correspond to the same undirected graph, even if the structures of these DAGs are very different (e.g. some are sparse while some are dense). Thus, in principle, we can use an arbitrary variable order in our model. [3] Notice that if the DAG parameter priors are not corresponding to a unified global prior of the precision matrix (one example is the prior directly penalizing the number of edges in the DAG), then the MAP estimation of the precision matrix will become dependent on the variable order in the DAG.

In this paper, we propose an Automatic Relevance Determination (ARD) style Wishart prior for the precision matrix Ω of the GGM. All the DAG priors are derived from this unified prior. Thus our approach is conceptually cleaner and does not depend on the variable order of the corresponding DAG. We show this prior for the precision matrix is equivalent to the Laplace priors for the regression coefficients in the corresponding DAGs. Thus, the MAP estimation of the precision matrix can be found efficiently by solving (a modified version of) Lasso regressions using fast grafting algorithm (proposed in (Simon, Kevin, & James 2003)). In this way, we have a consistent probabilistic model to incorporate Lasso regressions in undirected graph structure learning and we can share all the benefits of Lasso regression discussed before. Another important advantage of using 1-norm regularizer (implied by Lasso regression in our approach) than zero-norm regularizer (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

[2] In fact, *ad hoc* regressions will not lead to a consistent estimation of the GGM because the regression coefficient of variable i on j may be zero while the regression coefficient of variable j on i may be non-zero.

[3] In the real implementation, a sparse DAG still has some advantages since it requires less computational cost and less space to store its parameters.

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.

Method

We attempt to learn the GGM with maximal posterior probability from training data. For simplicity, we assume all the variables have been preprocessed so that each of them follows a standard normal distribution. Thus the precision matrix of the GGM is all we want to learn. Since there must be a DAG with exactly the same joint distribution as the GGM, our task is equivalent to learning the DAG with maximal posterior probability.

For any DAG, there is a specific ordering of variables, which we take here for discussion as simply 1,2,...,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 $p(Data|\beta) = \prod_{i=1}^{p} P(x_i|X_{(i+1):p}), \beta)$. 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 written in a matrix form $X = \Gamma X + \epsilon \quad and \quad \epsilon \sim N_p(0, \psi)$, where $\epsilon = (\epsilon_1, ..., \epsilon_p)'$, $\psi = diag(\psi_1, ..., \psi_p)$ 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.

prior for Ω and prior for β

Now we are trying to find out 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) determines regression parameter β (and of cause, β also determines Σ and Ω). Thus the prior over Σ (or Ω) defines consistent priors 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 $T = diag(\theta_1, ..., \theta_p)$. Here θ_i is a positive hyper prior and satisfy

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

This kind of hyper prior distributions have been suggested by (Figueiredo & Jain 2001) 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 (Geiger & D.Heckerman 2002), we know the associated prior for β_i is $P(\beta_i|\psi_i, \theta_{(i+1):p}) = N_{p-i}(0, T_i\psi_i)$. Thus

$$P(\beta_{ji}|\psi_i, \theta_j) = N(0, \theta_j\psi_i) \tag{3}$$

From the derivations in (Geiger & D.Heckerman 2002), we also know the associated prior for ψ_i is

$$P(\psi_i^{-1}|\theta_i) = Gamma(\frac{\delta + p - i}{2}, \frac{\theta_i^{-1}}{2}) \tag{4}$$

Following the idea in (Figueiredo & Jain 2001), we can integrate out the hyper parameter θ from prior distribution of β_{ji} and get

$$\begin{aligned} P(\beta_{ji}|\psi_i) &= \int_0^\infty P(\beta_{ji}|\psi_i, \theta_j)P(\theta_j)d\theta_j \\ &= \frac{1}{2}\sqrt{\frac{\gamma}{\psi_i}} exp(-\sqrt{\frac{\gamma}{\psi_i}}|\beta_{ji}|) \end{aligned} \tag{5}$$

Let's suppose there are K samples and we use k to index them. x_{ki} represents the value of the ith variable in the kth sample. Thus

$$\begin{aligned} &P(\beta_i|\psi_i, Data) \propto P(x_i|X_{(i+1):p}, \beta_i, \psi_i)P(\beta_i|\psi_i) \\ &\propto exp(\frac{\sum_k (x_{ki} - \sum_{j=i+1}^{p} \beta_{ji}x_{kj})^2 + \sqrt{\gamma\psi_i}\sum_{j=i+1}^{p} |\beta_{ji}|}{-\psi_i}) \end{aligned} \tag{6}$$

and

$$\begin{aligned} &P(\psi_i^{-1}|\theta_i, \beta_i, Data) = \\ &Gamma(\frac{\delta + p - i + K}{2}, \frac{\theta_i^{-1} + \sum_k (x_{ki} - \sum_{j=i+1}^{p} \beta_{ji}x_{kj})^2}{2}) \end{aligned} \tag{7}$$

Parameter estimation of the modified Lasso regression

From formula 6, we see the MAP estimation of β_i is

$$\hat{\beta}_i = argmin \sum_k (x_{ki} - \sum_{j=i+1}^{p} \beta_{ji}x_{kj})^2 + \sqrt{\gamma\psi_i} \sum_{j=i+1}^{p} |\beta_{ji}| \tag{8}$$

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

However, we cannot solve formula 8 as 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 in formula 8, we need to first estimate the regularization parameter $\sqrt{\gamma\psi_i}$.

Formula 6 and 8 provide the posterior distribution of β_i conditioned on ψ_i while formula 7 provides us the posterior distribution of ψ_i conditioned on β_i (the effect of θ_i in formula 7 can be numerically integrated out by sampling method [4]). Thus we can use an iterative procedure to estimate $\hat{\beta}_i$ and $\hat{\psi}_i$. We start from an initial guess $\hat{\psi}_i = 0$. Using

[4] Ideally, we'd like to multiply formula 2 and formula 7 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 formula 2 and use these values to simulate the integration operation. Since θ_i (with different i values) all follow the same distribution in formula 2, the sampled θ_i values can be used repeatedly with different i values.

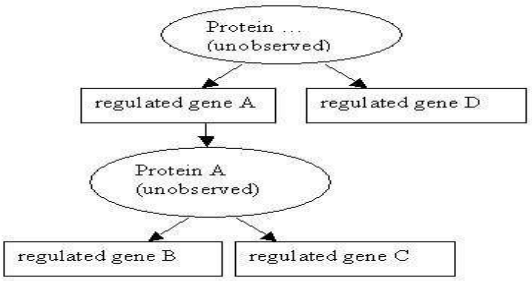

Figure 1: A typical sample of genetic regulatory network

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 $\hat{\beta}_i$ do not change. This process converges quickly in our experiments. 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 can incorporate the iterative procedure into the fast grafting algorithm that is used to solve Lasso regression [5]. Thus the computational cost can be significantly reduced.

We further note that from formula 8, for an arbitrary variable x_q that $q > i$, we can guarantee $2|\sum_k (x_{ki} - \sum_{j=i+1}^{p} \beta_{ji} x_{kj}) x_{kq}| \leq \sqrt{\gamma \psi_i}$ where equality only holds when variable q has a non-zero weight (see (Simon, Kevin, & James 2003)). Note we have pre-processed data so that x_i has a standard normal distribution. This implies that $\text{cor}(r_i, x_q) \leq \frac{\sqrt{\gamma}}{2K}$ where $\text{cor}(r_i, x_q)$ means the correlation coefficient between r_i (the residual of variable i) and x_q. This gives us an intuitive way when we want to decide the value of the hyper prior γ

Experiments

Discovering regulatory networks is an important application of undirected graph learning. We first give a brief introduction in this field before we compare the empirical results by our approach and by other approaches.

Figure 1 shows an example of a small genetic regulatory network. Microarray data provides the mRNA levels of

[5]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 (Simon, Kevin, & James 2003). In order to incorporate the iterative procedure adjusting the regularization parameter into the fast grafting algorithm, our implementation is a little different from (Simon, Kevin, & James 2003). 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.

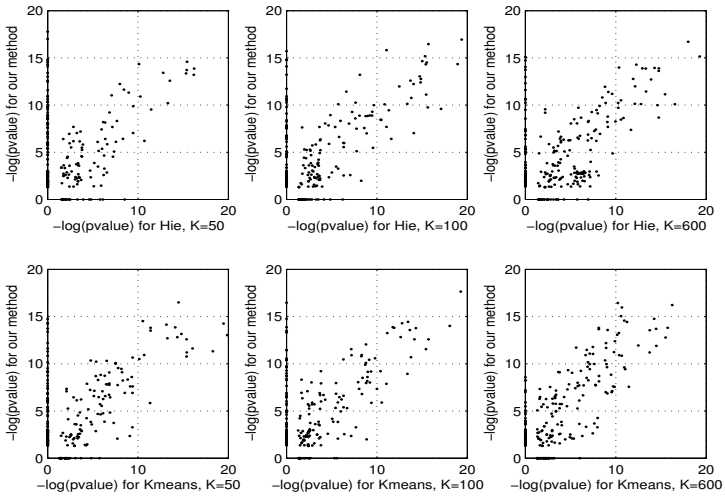

Figure 2: Comparison between our approach and Hierarchy clustering (in the first row) and K-means clustering (in the second row), with k=50 (first column), 100 (second column), 600 (third column) respectively. In the up left figure, there are 109 dots on the Y axis and 22 dots on the X axis. In the up 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.

thousands of genes under various conditions. If two genes are co-regulated by the same proteins (the genes encoding these proteins are called regulator genes and the genes regulated by these protein are called regulatee genes), they will tend to have similar patterns of mRNA levels in different conditions. Thus it is possible to estimate the gene regulatory network from microarray data if we treat each gene as a variable and each condition as a sample.

Hierarchical and K-means clustering algorithms have been used to find co-regulated genes from microarray data. (Bar-Joseph *et al.* 2003) and (Segal, Yelensky, & Koller 2003) developed more advanced clustering-based approaches to discover regulatory networks from multiple data sources. A major problem of clustering approaches is that they cannot model the partial correlations between variables.

Bayesian networks (BNs) have also been used to analyze microarray data((Friedman *et al.* 1996)). One problem of modeling a regulatory network as a BN lies on the fact that the protein levels are unobservable from microarray data. 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. Thus the estimated BN may confound the regulator-regulatee dependencies and co-regulated dependencies among genes. (Hartemink *et al.* 2001) and (Bernard & Hartemink 2005) have used genome wide location analysis data [6] as priors to constrain the structures of BNs so that they could filter out the edges corresponding to co-regulated dependencies among genes. However, they still have to assume (relatively strong) correlations between mRNA levels of the regulator genes and the regulatee genes. Furthermore, they did not make use of the co-regulated dependencies among genes, which are 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. Instead of directly modeling the regulator-regulatee dependencies using BNs, we only attempt to model the co-regulated dependencies using GGM in the first stage. The structure of the undirected graph is learned from microarray data and each edge in the undirected graph corresponds to a co-regulated gene pair. In the second stage, we developed a post-processing algorithm to identify the co-regulated gene modules and their corresponding regulator genes from the estimated undirected graph with the help of location analysis data. The advantage of our approach lies on the fact that the co-regulated dependencies are often much stronger than the regulator-regulatee dependencies in microarray data. Thus by using the co-regulated dependency information, which is not used in BNs, we may discover more regulatory patterns.

We applied our method on the Yeast Saccharomyces cerevisiae dataset. The expression microarray data we used comes from (Spellman *et al.* 1998). The mRNA expression levels of 6177 genes are measured under 76 conditions. The location analysis data we used comes from (Lee *et al.* 2002). It gives the p-values of genes regulated by each of the 106 regulators. When a p-value is lower than 0.05, we would assume the gene is regulated by the regulator. The value of the hyperprior γ is assigned to be 2.31 so that the correlation coefficients between zero weighted variables and the residuals are no larger than 0.01.

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((Ashburner *et al.* 2000)). 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 ((Segal, Yelensky, & Koller 2003)).

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

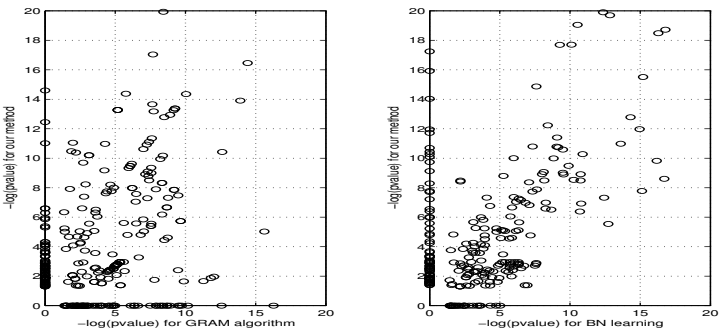

Figure 3: The left graph is the comparison between our method and Gram algorithm. There are 93 dots on the Y axis and 61 dots on the X axis. The right graph is the comparison between our method and BN learning algorithm. There are 88 dots on the Y axis and 29 dots on the X axis.

The details of the evaluation strategy are 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 (by SGD Term Finder Toolkit((Ashburner *et al.* 2000)), and took p-value < 0.05 to be significant. We compared the enrichment of modules for GO annotations between results achieved by our approach and results achieved by other baseline approaches. We only consider GO annotations in "Process" category.

(Bar-Joseph *et al.* 2003) has used GRAM algorithm, which can be thought as a kind of clustering algorithm, 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. We will use their result as a baseline result. 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 (Bar-Joseph *et al.* 2003). In this way, we can compare our results with (Bar-Joseph *et al.* 2003) conveniently.

We also compared our results to the results achieved by hierarchical clustering (agglomerative average linkage version) and k-means clustering algorithms (with 5 random restarts). We tried several settings (k=50, 100, and 600) for the number of clusters in these two algorithms. We can 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) so that we can

directly use our post-processing algorithm to combine the clustering results with location analysis data. 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.

We further compared our approach with the BN learning approach that used the location analysis data as priors to constrain the BN structure. We used sparse candidate hill climbing as the search algorithm and only allowed an edge from a regulator gene to a regulatee gene when the p-value between them is lower than 0.05 in location analysis data. We tuned the weight of the penalizer for the number of edges in the BN so that it also generated 106 regulatory modules.

The results are shown in figure 2 and 3. 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 2 and 3, which implies our method achieved much better performance. Notice the microarray data used in (Bar-Joseph *et al.* 2003) contains richer information than our data, but their results are still not as good as ours, as reflected in figure 3.

Conclusions

In this paper, we propose a new approach to learn large undirected graphs in GGM. By introducing an ARD style Wishart prior for the precision matrix, the MAP estimation of the precision matrix, which encodes the sparse graph structure we want to learn, can be solved efficiently by modified Lasso regressions. We used our approach to learn genetic regulatory networks from microarray data and location analysis data. An evaluation on the basis of consistency with the GO annotations shows that our approach significantly outperforms clustering-based approaches and BN learning approaches in discovering regulatory modules.

Acknowledgments

We thank Eric P. Xing for immensely helpful discussions. This research work is supported by National Science Foundation Information Technology Research grant number 0225656.

References

Ashburner, M.; Ball, C.; Blake, J.; Botstein, D.; and Butler, H. 2000. Gene ontology: tool for the unification of biology. *Nature Genetics* 25:25–29.

Bar-Joseph, Z.; Gerber, G.; Lee, T.; and Rinaldi, N. 2003. Computational discovery of gene module and regulatory networks. *Nature Biotechnology* 21(11):1337–42.

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

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

Figueiredo, M., and Jain, A. 2001. Bayesian learning of sparse classifiers. In *CVPR*, 35–41.

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

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

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

Hartemink, A.; Gifford, D.; Jaakkola, T.; and Young, R. 2001. Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. In *Pacific Symposium on Biocomputing*.

Lauritzen, S. 1996. *Graphical Models*. Oxford: Clarendon Press.

Lee, T.; Rinaldi, N.; Robert, F.; and Odom, D. 2002. Transcriptional regulatory networks in saccharomyces cerevisiae. *Science* 298:799–804.

Meinshausen, N., and Buhlmann, P. 2004. 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.

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

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

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

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

Spellman, P.; Sherlock, G.; Zhang, M.; and Iyer, V. 1998. Compehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. molecular. *Biology of the Cell* 9:3273–3297.

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

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