

Recovering Genetic Regulatory Networks from Micro-Array Data and Location Analysis Data

Fan Li

hustlf@cs.cmu.edu

Yiming Yang

yiming@cs.cmu.edu

LTI, School of Computer Science, Carnegie Mellon Univ., 4502 Newell Simon Hall,
5000 Forbes Ave, Pittsburgh, PA 15213, USA

Abstract

Learning large network (with hundreds of variables) is gaining interest of many researchers with the emergence of high-throughput biological data sources such as micro-array data. In this paper, we investigated the two popular large scale network structure learning algorithms, sparse candidate hill climbing (SCHC) and Grow-Shrinkage(GS) algorithm. The experiments show that in fact both of them have serious effectiveness problems when the number of variables(genes) is large compared to the number of instances(experimental conditions), which is a common case in micro-array data. We further propose a new large scale structure learning algorithm based on Lasso regression. Theoretical analysis in [10] suggested that the L1-norm in lasso regression could make our algorithm especially suitable in the cases that the number of variables and instances is unbalance. Our algorithm achieves much better results than SCHC and GS on the synthetic data. We also show the effectiveness of our algorithm by learning genetic regulatory network modules from a real micro-array data (with more than 6000 genes), combined with the genome-wide location analysis data. The learned results are consistent well with biological knowledge.

Keywords: large scale network structure learning, regulatory network, Lasso, transcription factor

1 Introduction

Learning large genetic regulatory network has been an area of intense research in recent years with the emergence of high-throughput biological data. However, many interesting datasets contain large number of variables and small number of instances. This big unbalance makes it difficult for large scale structure learning algorithms to achieve effective performance, even if they are efficient enough. For example, the yeast cell cycle dataset contains 6177 mRNA levels of open reading frames of the yeast genome (features) measured under 76 conditions (training examples) [13]. Several researchers have applied large scale structure learning algorithms on such genetic datasets([3]), but a systematic evaluation of the results (the number of missing edges and false positive edges) has not been reported(since the ground truth is unknown), leaving the effectiveness of the algorithms uninvestigated to some extent.

There are two basic categories of approaches for network structure learning: constraint-based approaches and score-based approaches. Both categories have structure learning algorithms especially designed to be scalable for large number of variables.

- Score based methods perform a search(say, hill-climbing) through the space of possible structures and attempt to identify the most probable network given the data and the prior knowledge. However, when the number of variables is large(> 100), simple hill-climbing search may become un-scalable. [3] proposed Sparse Candidate hill climbing(SCHC) to solve this problem. SCHC first estimates the possible candidate parent set for each variable and then use this set to constraint the hill-climbing search. On the other hand, the structure returned by the search can be in turn used to estimate the possible candidate parent set for each variable in the next loop. The time cost of SCHC is $O(p^2)$ where p is the total number of variables.

- Constraint-based approaches directly use conditional independence tests to tell whether an edge between two variables exists or not. The PC algorithm ([14]) is the most popular constraint-based approach, which tries to d-separate all the variable pairs with all the possible conditional sets whose sizes are lower than a given threshold. However, the number of possible conditional sets increases quickly with the number of variables, which makes the PC algorithm not scalable when the number of variables is large (> 100). [8] has proposed the Grow-Shrinkage algorithm to solve this problem. In GS, two phases are used to identify the estimated Markov blanket $\hat{M}(V_j)$ for variable V_j . In the “grow” phase, variables are added to the $\hat{M}(V_j)$ sequentially using a forward feature selection procedure and we are supposed to get a superset of the real Markov blanket. In the “shrinkage” phase, variables are deleted from the $\hat{M}(V_j)$ if they are independent from the target variable conditioned on a subset of other variables in $\hat{M}(V_j)$. Then the algorithm tries to identify the parents and children for each variable from the estimated Markov blanket and tries to orient the edges. In sparse network ($|\hat{M}(V_j)|$ is small), The time cost of GS is $O(p^2)$.

Although the SCHC and GS algorithms can be applied on large scale datasets efficiently, their effectiveness still need be investigated. In [3], SCHC has been applied on text data and gene micro-array data with 500 to 800 variables. The BDEU score was used as the evaluation measure. However, although the BDEU score¹ is a good measure when we have sufficient training examples (in other words, consistent), it may not be the best choice when the number of training examples is small compared to the number of variables (which is almost always the case in large scale structure learning problems). In fact, in structure learning tasks, we hope to find a structure with the least structure errors (the sum of missing edges and the false positives). Higher BDEU scores may not lead to a smaller number of structure errors when the number of training examples is insufficient.

[8] has applied GS algorithm on several synthetic datasets and compared the results with the PC and hill-climbing algorithm. However, the number of variables in those datasets is not very large (less than 50) thus the effectiveness of GS on large scale problems is still unknown. The second problem is that, [8] only evaluated the results using BDEU scores, leaving the structure errors unreported. Our experiments show that when the number of variables is very large compared to the number of instances, the structure returned by GS would contain many spurious relationships between two variables.

In this paper, we proposed a new structure learning algorithm based on Lasso regression, which is scalable for large number of variables and much more effective than the two existing algorithms. The spirit of the algorithm is similar to GS algorithm. However, we use Lasso regression instead of greedy forward feature selection to estimate $\hat{M}(V_j)$. We compared it empirically with the other two large scale structure learning algorithms (SCHC and GS) on two synthetic datasets (a moderate network and a large scale network) and experimental results show that our algorithm has outperformed the other two significantly. We also notice the interesting work recently done by [10]. They show 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 Lasso regression can be effective even if there are exponentially many irrelevant features as there are training examples. which gives theoretical support to our structure learning algorithm². These are the major contribution of this paper.

We also applied our algorithm on the real micro-array data together with the genome-wide location

¹BDEU score of structure G reflects the Bayesian estimation of the probability of the observed data generated by structure G under an uniform priors over equivalent structures. Another popular score metric is the Bayesian Information Criterion (BIC) score. It describes the maximal likelihood estimation of the probability of the observed data generated by structure G and adds a penalization term to the complexity of the model. BDEU and BIC will be equivalent with infinite number of samples and [6] shows that in fact these two score metrics become very close with only 40 samples. Details of these score metrics are referred to [5]. Because we are using continuous variables which makes it difficult to get exact Bayesian estimations, we use BIC score (as a supplementary evaluation measure in addition to structure error measure) in our experiments.

²[10]’s derivation is based on logistic regression with L1 norm, but it can be easily generalized to lasso regression (least square regression with L1 norm), which is also mentioned by the author.

analysis data to learn genetic regulatory network in cell cycles. The reason we use multiple data sources is that: micro-array data only provide the expression level of mRNAs of genes, leaving protein levels unobservable. Since the regulatory interactions can only happen through the proteins (figure 4 shows an example), it is sometimes very difficult to recover regulatory network from micro-array data alone. In fact, the expression profile of a transcription factor and its regulated genes do not often show obvious correlation, although regulated genes often show strong correlation among themselves. In this case, learning correct regulatory network is impossible if micro-array is the only data source. However, by combining the location analysis data³ with micro-array data in the learning process, this problem could be solved. [4] has used location analysis information to constrain the search of regulatory networks. However, they still use the assumption that there is (relatively strong) correlation between transcription factors and the regulated genes. Thus their method may still have problems when such assumptions fail. Further more, their structure learning algorithm is not scalable to hundreds of variables. [2] also combined location analysis data and micro-array data. But they used clustering instead of structure learning algorithm to find the genetic regulatory modules, which makes it more difficult to identify partial correlation(conditional independence) relationships. Our algorithm has overcome the above difficulties.

The organization of the remaining parts of this paper is as follows: Section 2 describes how to use Lasso regression in large scale structure learning problems. Section 3 conducts experiments and compares our algorithm with PC, hill-climbing, SCHC and GS algorithm on two synthetic networks(with moderate or large number of nodes). We also applied our algorithm on a real micro-array data (together with the location analysis data) and found some interesting genetic regulatory network modules. Section 4 gives conclusions.

2 Network Structure Learning Using Lasso Regression

2.1 The Basic Idea of Using Lasso Regression to Learn Structures

The basic idea of our method is similar to the GS algorithm. We first estimate the Markov blanket for each variable. Then we identify the parent and child variables from the estimated Markov blanket and try to orient the edges. This procedure is described in the following pseudo code.

Algorithm 1 Main framework of structure learning procedure.

1. For each variable V_j , estimate V_j 's Markov Blanket $\hat{M}(V_j)$
 2. For each variable V_j , identify the variables in $\hat{M}(V_j)$ which are V_j 's parent or child (deleting V_j 's spouses)
 3. Try to orient all the undirected edges.
-

The most important (and the most difficult) step is to estimate $\hat{M}(V_j)$, which is also the step for which we are different from GS algorithm. For step 2 and step 3, we used exactly the same method GS has used(details are referred to [8]). We only focus on step 1 in this paper.

Unlike GS algorithm, we use Lasso regression (proposed in [15]) to estimate $\hat{M}(V_j)$ in step 1. This implies that the edges in the network are assumed to be linear(such assumptions are popular in

³The genome-wide location analysis data provide the information that which genes may be binded by each transcription factor. However, physical binding is not equivalent to regulatory interaction. Furthermore location analysis itself also has noise. Thus by combining location analysis data with the micro-array data, we may benefit more than using any single data source.

biological data such as micro-array data). The objective function of Lasso regression is

$$L = \sum_i (y_i - \sum_j \beta_j x_{ij})^2 + \lambda \sum_j |\beta_j| \quad (1)$$

where y_i is the value of target variable in the i -th training example and x_{ij} represents the value of the j -th variable in the i th training example. β_j is the regression coefficient assigned to feature j . λ is the regularization parameter. In Lasso regression, many features would be assigned a exact zero β_j value. And the solution would satisfy(for the details, see [12])

$$\left| \sum_{i=1}^n (y_i - \sum_{j=1}^p \beta_j x_{ij}) x_{ik} \right| \leq \lambda$$

and equality only holds for non-zero weighted features k . (2)

Now we rewrite the above formula using vector notation. Let $r_i = y_i - \sum_{j=1}^p \beta_j x_{ij}$, We use $\vec{r} = (r_1, r_2, \dots, r_n)$ to represent the vector of residues. We use $\vec{f}_k = (x_{1k}, \dots, x_{nk})$ to represent the vector of feature f_k (over examples). We get

$$|\vec{r} \bullet \vec{f}_k| \leq \lambda$$

and equality only hold for non-zero weighted feature k . (3)

This implies that, given the non-zero weighted features, the left (zero weighted) features can be made linearly independent from the residues to a certain degree(decided by λ). Thus we use the non-zero weighted features in Lasso regression to simulate $\hat{M}(V_j)$.

2.2 Using the Fast Grafting Algorithm to Solve Lasso Regression

Here we used the fast grafting algorithm (a kind of stage-wise gradient descent algorithm) proposed in [12] to solve Lasso regression . This algorithm uses the property that there are only few non-zero weighted features(if λ is large enough) in Lasso regression to accelerate the optimization. It scales lineally in the total number of features(if there are only a few non-zero weighted features). 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 . For the details of this algorithm and the proof that it works, we refer to [12]. Here we just list the brief pseudo code of this algorithm:

Algorithm 2 **Input:**data and λ **Output:**coefficients β

1. let $C = \sum_i (y_i - \sum_j \beta_j x_{ij})^2$
2. set Free feature set F and Zero feature set Z . Assign all the features to Z and let F be the empty set.
3. Use gradient descent algorithm to minimize C with respect to the features in F . Notice $\frac{\partial L}{\partial w_j} = \frac{\partial C}{\partial w_j} \pm \lambda$ and the sign of λ is decided by the sign of w_j . if $w_j = 0$, then if $\frac{\partial C}{\partial w_j} > \lambda$, we should take -1 ; If $\frac{\partial C}{\partial w_j} < -\lambda$, we should take 1 . if $-\lambda \leq \frac{\partial C}{\partial w_j} \leq \lambda$, then w_j should keep zero.
4. Move the zero weighted features from free feature set F to Z .
5. For each feature w_j in Zero feature set Z , compute $cor = |\sum_i r_i w_{ij}|$ where r_i are the residues in the previous regression model. Choose the feature f_l with the largest cor value.

$$\left\{ \begin{array}{l} \text{If this largest } cor > \lambda, \text{ move this feature from } Z \text{ set to } F \text{ set and Goto step 3} \\ \text{If this largest } cor < \lambda, \text{ exit;} \end{array} \right.$$

Since we need do Lasso regression for every variable, the total structure learning scales $O(p^2)$

2.3 How to Choose the λ Value

In Lasso regression, the value of λ controls the number of non-zero weights. Thus λ plays a very important role in Lasso regression based structure learning, controlling the number of edges. In the GS algorithm, the corresponding parameter is the P-value threshold (which is often set to 0.05 or 0.01). In SCHC, this problem does not exist because the algorithm will just choose the structure with the highest score.

Here we use a strategy to choose λ value automatically so that all the non-zero features are guaranteed to be not independent with the target feature at a certain significance level. In other words, we use the p-value instead of λ as the parameter to control the number of edges. The reason for doing this is that it is much more convenient to assign a fixed p-value (such as 0.01) manually rather than assigning the λ value manually. In our experiments, we always use p-value 0.01 as our threshold.

The pseudo code is described as follows. The bold codes are the new codes for tuning the λ value.

Algorithm 3 Input: data and pvalue Output: coefficients β

1. let $C = \sum_i (y_i - \sum_j \beta_j x_{ij})^2$.
 2. set Free feature set F and Zero feature set Z. Assign all the features to Z and let F be the empty set. **set λ to be a small initial value (say, 0.1).**
 3. Use gradient descent algorithm to minimize C with respect to the features in F. Notice $\frac{\partial L}{\partial w_j} = \frac{\partial C}{\partial w_j} \pm \lambda$ and the sign of λ is decided by the sign of w_j . if $w_j = 0$, then if $\frac{\partial C}{\partial w_j} > \lambda$, we should take -1 ; If $\frac{\partial C}{\partial w_j} < -\lambda$, we should take 1. if $-\lambda \leq \frac{\partial C}{\partial w_j} \leq \lambda$, then w_j should keep zero.
 4. Move the zero weighted features from free feature set F to Z.
 5. For each feature w_j in Zero feature set Z, compute $corr = |\sum_i r_i w_{ij}|$ where r_i are the residues in the previous regression model. Choose the feature f_l with the largest $corr$ value.

$\left\{ \begin{array}{l} \text{If this largest } cor > \lambda, \text{ move this feature from Z set to F set.} \\ \text{Conduct a Fisher's Z test between } f_l \text{ and y conditioned on other features in F).} \\ \text{if calculated pvalue} < \text{the given pvalue, then } \lambda = \lambda + \Delta \text{ where } \Delta \text{ is a small number(0.1).} \\ \text{Goto step 3.} \\ \text{If this largest } cor < \lambda, \text{ exit;} \end{array} \right.$
-

3 Experimental Results

3.1 Experiments on Moderate and Large Synthetic Datasets

We first do experiments with moderate number of variables so that we can compare the large scale structure learning algorithms with other baseline structure learning algorithms which are not scalable for large number of variables. We investigate five algorithms: GS, SCHC, Lasso regression, PC, Hill-climbing (with 3 restarts). In our experiments, HC and SCHC use BIC score to guide their heuristic search. For the GS, PC and SCHC algorithm, we used the code which is available online in [1]. We tried three pvalue thresholds 0.05, 0.01 and 0.005 for GS and PC algorithm. Since GS and PC have the least number of structure errors with pvalue thresholds 0.01, we only reported their results in this setting. For the Hill-climbing algorithm, we used the BNT code which is also available online. We wrote the code for Lasso regression for structure learning using C++. All the features are centered and normalized before being fed into the Lasso regression module. We always use p-value 0.01 as the threshold for our Lasso based structure learning algorithm in the experiment.

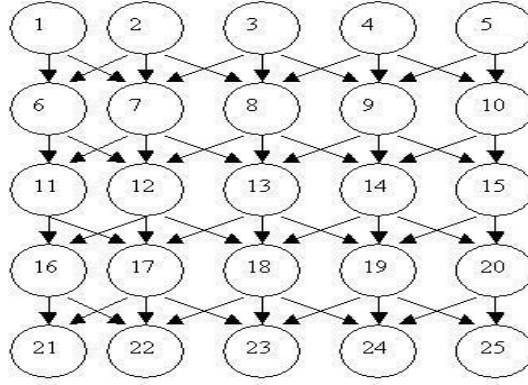


Figure 1: A typical synthetic network.

The first synthetic dataset was generated from the structure in Figure 1 (the same structure was used in structure learning research in [8]). Linear coefficients with values between 0.1 and 0.9 were randomly assigned to each directed edge in the graph. Using joint normal distribution on the variables of zero in-degree, simulated data was generated. Since we are focusing on the case that the number of examples is seriously insufficient compared to the number of variables, we only draw 100 samples from the structure as our dataset. This kind of experiment was repeated ten times to get an averaged result and to plot the error bars.

Here we use the edge existence errors as our measurement. We neglect the edge direction errors for two reasons. First, there are equivalent classes, which makes some direction difference meaningless. Second, in many real applications, if we can find an edge between two variables, even if the direction of the edge is wrong, the information will still be very valuable. This is exactly the case in gene regulatory network structure learning, by which our research was motivated. The results are shown in Figure 2.

We also computed the Bayesian Information Criterion (BIC) scores of the structures found by different algorithms.

PC and GS can not always assign a direction for every edge. In this case, we use hill climbing to orient such undirected edge. The results are also shown in Figure 2.

From Figure 2, we can see that Lasso regression is doing slightly better than PC and significantly better than GS and hill-climbing. One interesting observation from Figure 2 is that although hill-climbing is doing significantly worse than any other algorithms, it has achieved the highest BIC score. BIC score is a consistent measure and show good performance when we have sufficient training examples. However, our experiments seem to suggest that when the number of instances is insufficient, higher BIC score may not correspond to smaller number of structure errors. This may be the reason of the inferior performance of hill-climbing algorithm.

We also investigated applying the three large scale structure learning algorithms(SCHC,GS, Lasso regression) on the dataset with several hundred variables. We extended the network shown in Figure 1 by adding more variables in each layer. The structure we used here has the same kind of shape as Figure 1, but containing 300 nodes (thus there are 60 nodes in each layer). Linear coefficients and a simulated data with a sample size of 100 were generated in the same way as before. The three large scale algorithms are used to learn structures from the data. This kind of experiment was repeated ten times to get a averaged result and to plot the error bars, which are shown in Figure 3.

In our experiments, GS with fixed threshold($p=0.05$ or $p=0.01$) has very bad performance. There are several thousand false positive edges. Because this number is too large, we did not show the error bars of GS on Figure 3. Instead, we plot the error bars of “GS(up-bound)” on Figure 3. The result of

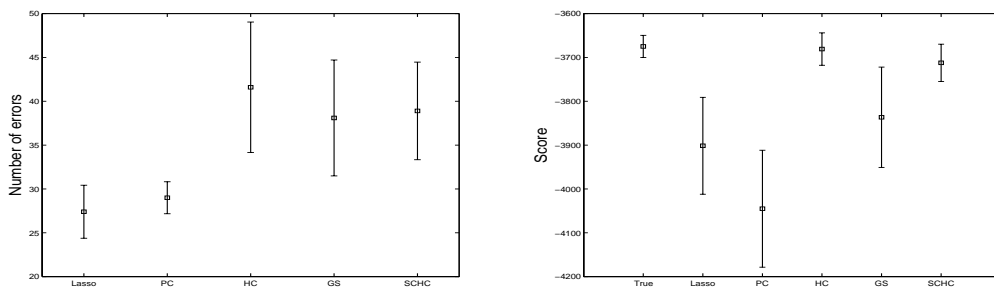


Figure 2: Result on moderate Synthetic dataset. The left sub-graph shows the number of errors and the right sub-graph shows the BIC score for each method.

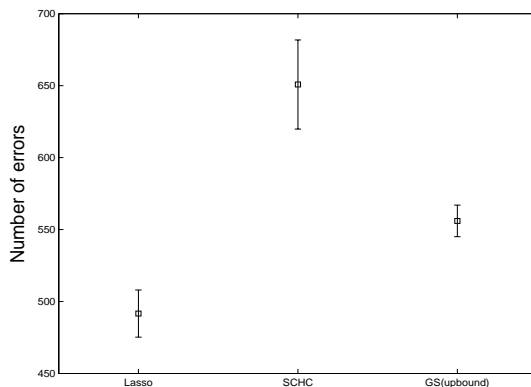


Figure 3: Results of large synthetic dataset, with 300 variables.

GS(up-bound) is obtained by tuning the p-value threshold using the information from ground truth (p-value was tuned to be 0.000005). In real applications, we can not tune the threshold in this way because we don't know the ground truth. We can see that even GS(up-bound) is no better than Lasso regression. SCHC is also significantly worse than Lasso regression in our experiments.

3.2 Experiments on Real Genetic Regulatory Network Data

We also applied the lasso based structure learning algorithm on real genetic regulatory network data. If a set of genes share the same one or more regulators, we call these regulators and their regulated genes a genetic regulatory module. Our task is to find the genetic regulatory modules from the 6177 genes.

As we have mentioned before, we can not simply run the structure learning algorithm on micro-array data because protein levels are unobservable (thus even gene A regulate gene B and gene C, the mRNA of gene B and C are NOT conditionally independent given mRNA of gene A). As a solution, we combine the micro-array data together with the location analysis data to learn regulatory network.

The microarray data we use comes from [13]. The mRNA expression level of 6177 genes are measured under 76 conditions. The location analysis data we use comes from [7]. It gives the pvalues of genes not being binded by each of the 106 regulators.

Figure 4 shows a typical genetic regulatory network. If we can directly measure protein A, the marokov blanket of gene B would only be protein A. However, if protein A can not be observed and is not treated as a variable, the markov blanket of gene B would become gene C and gene A. Recall that regulator and regulated genes may or may not have strong correlation. Thus in the estimated markov blanket of gene B, we may or may not see gene A. However, since the genes regulated by the same regulators tend to have correlation, we are likely to find gene C in gene B's estimated markov

Table 1: 14 modules related to SWI4, SWI5 and SWI6 found by the structure learning algorithm. The number in the first column means the number of regulated genes. For example, SWI4 (5) means there are 5 genes regulated by the single regulator SWI4.

regulator set	reference of co-regulators
ACE2,SWI5 (2)	physical interaction (Doolin <i>et al.</i> , 2001)
FKH2,MBP1,SWI6 (4)	FKH2 and (MBP1,SWI6) are cooperative TFs (Nilanjala 2004)
FZF1,SRD1,SWI6 (2)	NA
GAT3,MSN4,PDR1,RAP1,RGM1,SWI5,YAP5 (2)	(GAT3,MSN4) (Msn4,Yap5) are cooperative TFs (Nilanjana,2003)
GAT3,SWI5,YAP5 (2)	NA
MBP1,STB1,SWI4,SWI6 (2)	same sub-motif ACGCGA (Kato, 2004)
MBP1,SWI4,SWI6(3)	functional and physical interaction between (MBP1,SWI6) and (SWI4,SWI6). (CYGD database)
MBP1,SWI5 (3)	NA
MBP1,SWI6 (17)	functional and physical interaction between MBP1 and SWI6(CYGD database)
PHD1,SWI4 (2)	PHD1 encode a protein with homology to SWI4. Gimeno CJ, 1994
SKN7,SWI4 (2)	functional interaction between SKN7 and SWI4(Morgan <i>et al.</i> , 1995; Cui <i>et al.</i> , 2002)
SWI4 (5)	single regulator
SWI4,SWI6 (2)	physical interaction between SWI4 and SWI6(CYGD database)
SWI5,YAP5 (2)	NA

blanket. Thus, in the results of structure learning algorithm, if we find gene C is in gene B's estimated markov blanket, and if we know gene C and gene B have the common possible regulator A from location analysis data, we could infer that gene B and gene C are indeed regulated by A in the current micro-array conditions.

We give the pseudo code of the approach as following:

Algorithm 4 Using lasso based structure learning algorithm to find real genetic regulatory network from micro-array data and location analysis data.

1. For each regulator T we are interested in, find the gene subset $G(T)$ which may be binded by this regulator(we choose genes with pvalue less than 0.01 in location analysis data)
2. For each genes g_i in $G(T)$, use the lasso based structure learning algorithm to learn its esitimated markov blanket from micro-array data. (This step selected K genes from 6177 genes as g_i 's estimated markov blanket $\hat{M}(g_i)$. The size of $\hat{M}(g_i)$ is controled by the pvalue threshold of our structure learning algorithm. We use 0.05 in our experiment)
3. For each gene g_j in $\hat{M}(g_i)$, check

- $$\left\{ \begin{array}{l} \text{whether it is has common possible regulator(s) T' that } g_i \text{ has from location analysis data}^4 \\ \text{whether it is the possible regulator of } g_i \text{ from location analysis data} \\ \text{whether it may be regulated by } g_i \text{ (if } g_i \text{ itself is a regulator) from location analysis data} \end{array} \right.$$

If any of the above conditions is satisfied, put the found (regulator, regulated gene) in the output results.

Because of space limitation, we only show the genetic regulatory modules involved with gene SWI4, SWI5 and SWI6 (since we know these three genes are important and active regulators in cell cycles). Table 1 listed the 14 modules found by our algorithm. Each module is represented using its regulator set

It is very interesting to investigate the modules with more than one regulators. In fact, 13 of the 14 modules found have more than one regulators, which may suggest that most obvious regulation often involve multiple regulators. If we check the co-regulators, we can find interactions among these co-regulators reported in previous literatures in most cases. This is summarized in the second column in Table 8.

Because of space limitation, we can not list all the regulated genes(and their GO annotations) in each module. Here we only plot the module with regulator (MBP1,SWI4,SWI6) as an example (in Figure 5). It can be validated from Kegg database that CLB5 and CLB6 are indeed regulated by these

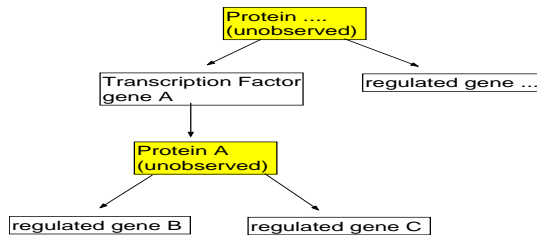


Figure 4: A typical sample of genetic regulatory network.

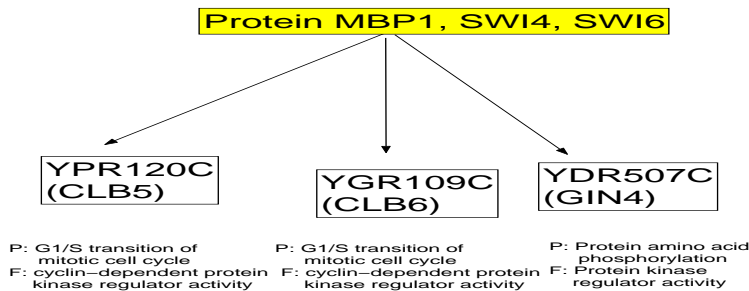


Figure 5: Modulal (MBP1,SWI4,SWI6) learned by the structure learning algorithm. P means procedure annotation and F means function annotation in GO.

regulators. GIN4 did not appear in Kegg cell cycle database. However, it shares the same functional GO annotation “protein kinase activity” with CLB5 and CLB6, which may suggest that it could be in the same regulation module with CLB5 and CLB6.

4 Conclusion

In this paper, we presented a new method in large scale structure learning using Lasso regression and compared it with the other two existing large scale structure learning algorithms. Our main research findings are:

- In large scale network structure learning, the number of instances is often insufficient compared to the number of variables. Using bayesian score as the searching strategy may lead to a solution with high score but many structure errors. This may be the reason that SCHC is not doing very well in our experiments.
- GS algorithm with fixed p-value threshold(such as p=0.01 or 0.05) has very low performance in our large scale structure learning experiment. Even if we tune its threshold using the ground truth information, its performance is still worse than our algorithm’s.
- We have used Lasso regression in large scale network structure learning and achieved much better performance than SCHC and GS algorithm consistently on two synthetic datasets(a moderate network and a large scale network). The success may come from the the L1 regularizer of Lasso regression and the theoretical support could be found in [10]. We also applied our Lasso based structure learning algorithm on a real micro-array data combing with location analysis data. The learned genetic regulatory network modules are consistent well with the biological knowledge.

References

- [1] Aliferis, C., Tsamardinos, I., Statnikov, A., and Brown, L.E., Causal explorer: A causal probabilistic network learning toolkit for biomedical discovery, *METMBS*, 371–376, 2003.
- [2] Bar-Joseph, Z., Gerber, G.K., Lee, T.I., Rinaldi, N.J., Yoo, J.Y., Robert, F., Gordon, D.B., Fraenkel, E., Jaakkola, T.S., Young, R.A., Gifford, D.K., Computational discovery of gene modules and regulatory networks, *Nature Biotechnology*, 21(11):1337–1342, 2003.
- [3] Friedman, N., Nachman, D., and Peier D., Learning bayesian network structure from massive datasets: The sparse candidate algorithm, *In Proc. Fifteenth Conf. on Uncertainty in Artificial Intelligence (UAI)*, 206–215, 1999.
- [4] Hartemink, A., Gifford, D., Jaakkola, T., and Young, R., Combining location and expression data for principled discovery of genetic regulatory networks, *Pac. Symp. Biocomput.*, 437-449, 2002.
- [5] Heckerman, D., A tutorial on learning Bayesian networks, *Technical Report MSR-TR-95-06*, Microsoft Research, 1996.
- [6] Kevin, P.M., *The Bayes Net Toolbox for Matlab*, Computing Science and Statistics, 2001.
- [7] Lee, T.I., Rinaldi, N.J., Robert, F., Odom, D.T., Bar-Joseph, Z., Gerber, G.K., Hannett, N.M., Harbison, C.T., Thompson, C.M., Simon, I., Zeitlinger, J., Jennings, E.G., Murray, H.L., Gordon, D.B., Ren, B., Wyrick, J.J., Tagne, J.B., Volkert, T.L., Fraenkel, E., Gifford, D.K., and Young, R.A., Transcriptional regulatory networks in *Saccharomyces cerevisiae*, *Science*, 298:799–804, 2002.
- [8] Margaritis, D. and Thrun, S., Bayesian network induction via local neighborhoods, *Advances in Neural Information Processing Systems 12 (NIPS 1999)*, 505–511, 1999.
- [9] Nair, M., McIntosh, P.B., Frenkiel, T.A., Kelly, G., Taylor, I.A., Smerdon, S.J., and Lane, A.N., NMR structure of the DNA-binding domain of the cell cycle protein Mbp1 from *Saccharomyces cerevisiae*, *Biochemistry*, 42:1266–1273, 2003.
- [10] Ng, A., Feature selection, L1 vs L2 regularization, and rotational invariance, *ICML 2004*, in press.
- [11] Simon, I., Barnett, J., Hannett, N., Harbison, C.T., Rinaldi, N.J., Volkert, T.L., Wyrick, J.J., Zeitlinger, J., Gifford, D.K., Jaakkola, T.S., and Young, R.A., Serial regulation of transcriptional regulators in the yeast cell cycle, *Cell*, 106:697–708, 2001.
- [12] Simon, P., Kevin, L., and James, T., Grafting: Fast, incremental feature selection by gradient descent in function space, *J. Machine Learning Res.*, 3:1333–1356, 2003.
- [13] Spellman, P.T., Sherlock G., and Zhang, M.Q., Iyer, V.R., Anders, K., Eisen, M.B., Brown, P.O., Botstein, D., and Futcher, B., Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization, *Mol. Biol. Cell*, 9:3273–3297, 1998.
- [14] Spirtes, P., Glymour, C., and Scheines, R., *Causation, Prediction, and Search*, Second edition, Cambridge, Massachusetts, The MIT Press, London, 2000.
- [15] Tibshirani, R., Optimal reinsertion: Regression shrinkage and selection via the lasso, *J. R. Statist. Soc. B*, 58(1):267–288, 1996.
- [16] Tsalenko, A., Ben-Dor, A., Cox, N., Yakhini, Z., Methods for analysis and visualization of SNP genotype data for complex diseases, *Pac. Symp. Biocomput.*, 548–561, 2003.
- [17] Wille, A., Hoh, J., and Ott, J., Sum statistics for the joint detection of multiple disease loci in case-control association studies with SNP markers genetic epidemiology, *Genet. Epidemiol.*, 25(4):350–359, 2003.