DISCOVERING CONDITION-DEPENDENT BAYESIAN NETWORKS FOR GENE REGULATION

Antti Ajanki, Janne Nikkilä, and Samuel Kaski

Adaptive Informatics Research Centre & Helsinki Institute for Information Technology
Laboratory of Computer and Information Science
Helsinki University of Technology
P.O. Box 5400, FI-02015 TKK, Finland
antti.ajanki@tkk.fi, janne.nikkila@tkk.fi, samuel.kaski@tkk.fi

ABSTRACT

Among the main interests in many biological studies are the structure of gene regulatory network, and in particular differences in the regulatory interactions between different conditions. However, since the number of available samples is always very small and estimating the network structure is extremely hard, most current algorithms have to assume that the gene regulation does not change between conditions. We propose a new Bayesian network algorithm which (i) utilizes all the samples for estimating regulatory relations that remain the same across conditions, and (ii) explicitly searches for regulatory relationships that are active only in one of the conditions. The result is an easily interpretable map of changes in regulation in several conditions.

1. INTRODUCTION

A Bayesian network is a graphical representation of a multivariate joint probability distribution. The nodes are the random variables and edges encode the dependency relations between them. They are a commonly used for modeling the regulation of gene expression. Value of a node in the network corresponds to the expression of a gene and the edges describe the regulatory relationships in which the protein product of the parent genes binds to the child gene.

Most previously introduced Bayesian network models for modeling the regulation of gene expression assume that the regulation does not change between the conditions. There have been some methods which model context specific regulation, but they have done so in an unsupervised manner by clustering similarly regulated genes together (see for example [1]).

In this paper, we will introduce a method that is able to discover differences in regulation if explicit *classes*, for example the measurement *conditions*, are available. This is achieved by a new representation for the conditional probability distributions for the nodes on a Bayesian network. We introduce also a structure search algorithm that takes advantage of the new representation.

The algorithm takes genes' expression profiles and their classes as input and fits a special Bayesian network to them. Only a subset of the edges in the resulting network are associated with all classes; the rest are present in only a subset of the classes. This makes it straightforward to interpret differences in the regulation between the classes.

Using toy data we show that our new method will recover a greater proportion of correct edges than a state-of-the-art method trained separately for each class, especially when only a small number of training examples are available. We apply the method to discover the differences in regulation of *Saccharomyces cerevisiae* yeast between normal and stressful growth conditions.

2. RELATED WORK

Most genes in a cell are naturally divided into functional modules which are co-regulated and co-expressed. It is important to take this organization into account when modeling the activity of the cell. Module networks [1] have been previously proposed as a way to predict the regulatory network and, at the same time, group the genes into biologically plausible modules.

A module network is a Bayesian network where the nodes are gathered into modules and the values of the nodes inside one module follow the same distribution. Particularly, this means that all nodes in a module have the same parent nodes. The module assignment is found during the optimization of the network. The optimization is based on the *Bayesian score*, or the logarithm of the posterior probability of the dependency structure $\mathcal{S}$ and assignment $\mathcal{A}$ of the nodes in the modules given the observed values D of the nodes:

$$score(\mathcal{A}, \mathcal{S}|D) = \log p(D|\mathcal{A}, \mathcal{S}) + \log p(\mathcal{A}, \mathcal{S}) \quad (1)$$

The optimization is done by alternating two greedy steps. In the structure search step the score is computed for every possible local modification i.e. all additions and deletions of single edge in the current network. The modification which results in the highest increase in the score is selected for the next iteration step. In the module assignment search step a better assignment for the nodes is sought by iterating over the nodes and trying to move each node at a time to the other modules. If the new module

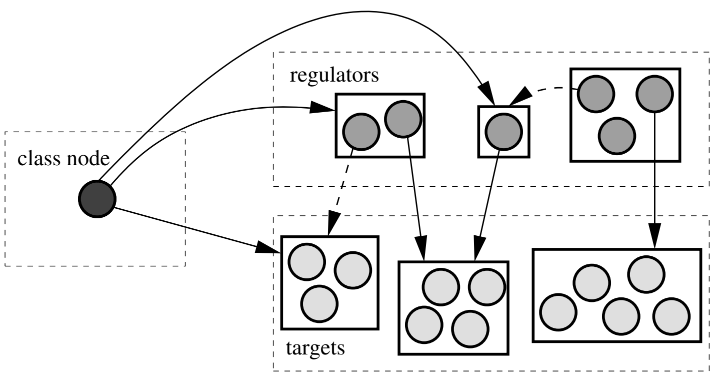

Figure 1. In a condition-dependent network the nodes are divided into regulator modules and target modules. The dashed edges from regulators to targets modules, which have the class node as the additional parent, are condition-dependent, i.e. may be present only in some classes.

assignment leads to a higher score the move is accepted, otherwise the node is returned to the original module. This step is continued until no assignment change improves the score. The two steps are repeated until convergence. During the optimization, care must be taken not to introduce any cycles in the network.

The score can be decomposed into terms, each of which depends only on the observations of nodes inside one module. This means that computing the score changes during the algorithm is very efficient; only scores for the modules which are affected by the local modification need to be recomputed.

The nodes in a module network can include regression trees with separate probability distributions associated with each leaf. The probability distribution for a node is chosen by traveling through the tree along the path defined by the values of the parent nodes. Some combinations of the parent values may lead to the same distribution. In Section 3 we will introduce a similar scheme for sharing distributions between different parent combinations, but our approach will include a class variable explicitely unlike the earlier module network algorithm.

3. CONDITION-DEPENDENT NETWORKS

3.1. Class dependency in the distributions of the nodes

By the term *condition-dependent* we mean the kind of dependency where the node depends on a parent in one class but not in an other class. For example, the values of the node X may depend on the values of the both A and B on class 1 but only on the value of A on class 2:

$$\begin{cases} p(X|C=1,A,B) & = p_1(X|A,B) \\ p(X|C=2,A,B) & = p_2(X|A). \end{cases}$$

Because we want to explicitly model condition dependencies we add a new discrete-valued *class node* to the network to stand for the class of the observation. The children of the class node have separate distributions, which may depend on different parent nodes, for each class.

A more general framework for selecting a distribution based on the values of the parent nodes have been suggested previously for regularizing the model [2]. Our motivation is to make the model easier to interpret.

3.2. Condition-dependent network structure

Because the gene expression data sets typically contain maximally some dozens of measurements of the expression for thousands of genes, it is not possible to learn a very detailed Bayesian network. Therefore, we choose to bring prior knowledge into the model by constraining the network structure in two ways. First, the nodes are divided into modules like in the module network described in Section 2, but we further constrain the model by having two types of modules. The genes which are suspected to function as regulators in some condition must belong to *regulator modules*, the other genes must belong to *target modules*. Like in the original module networks, we assume that a list of candidate regulators is available. Only they may belong to regulatory modules. The class node is always kept in isolation in its own module. Secondly, only the regulator modules are allowed to depend freely on the class node; the target modules may depend on the class only when they have a regulator as a parent as well. This constrains the model to explain the class-dependent changes in the expression of target genes through regulatory relationships.

In Subsection 3.1 it was mentioned that the child modules of the class node may have different parent nodes in different classes. The edges that may be present only in a subset of classes are called *condition-dependent regulatory edges*. These edges are dashed in the example network in Figure 1.

3.3. Training algorithm

The basic idea of the training algorithm for condition-dependent networks is similar to that of the training of the module network (see Section 2). The iteration alternates between structure search and assignment optimization. The first step tries to add or remove edges in the network and the second step moves nodes from one module to another. In both steps the criterion for acceptance is that the proposed modification must increase the Bayesian

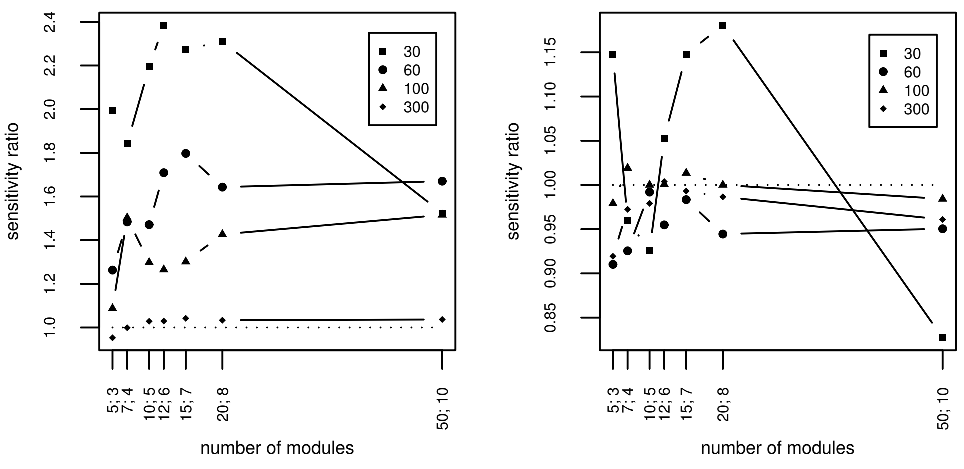

Figure 2. Sensitivity ratio between the condition-dependent network and the multinet (on the left) and between the condition-dependent network with and without condition-dependent edges (on the right). Values above 1 indicate that the regular condition-specific network is better. The X-axis shows the number of the target (the first figure) and regulator modules (the second figure) in the learned network. The curves correspond to different sample sizes used in training.

score. Similarly to the module network approach, the number of modules must be fixed in advance.

The Bayesian score for the case when the distribution of the nodes may depend on the class is

$$score(\mathcal{A}, \mathcal{S}, \mathcal{L}|D) = \log p(D|\mathcal{A}, \mathcal{S}, \mathcal{L}) + \log p(\mathcal{L}|\mathcal{A}, \mathcal{S}) + \log p(\mathcal{A}, \mathcal{S}), \quad (2)$$

where $\mathcal{L}$ encodes the class dependencies. Except for the middle term, the class dependence prior, this is similar to (1), the Bayesian score of a module network. We choose to use a Minimum Description Length-based formulation that has been proposed for $\log p(\mathcal{L}|\mathcal{A}, \mathcal{S})$ before [2]. The score for a module that does not depend on the class is computed from all observations. The score for a class dependent module is sum of scores of the individual classes.

The potential condition-dependency of the distributions requires special consideration in the structure search. Whenever the optimization algorithm is considering adding or removing a parent for a child of the class node it must re-select best parents for each class. This requires evaluating the score function for all combinations of class values and parent subsets. While the cost of this step is exponential in the number of parent nodes we can still control it by restricting the maximum number of parents for a node to be smaller than some constant.

The assignment optimization step is almost identical to the similar step in training of the module networks. The only difference is that the candidate regulators and the target genes are not allowed to mix in the same module.

4. EXPERIMENTS

4.1. Toy data

We generated a random artificial network of 400 nodes which were divided into 5 regulator modules and 10 target modules. A random subset of edges, but not the node assignments, were modified to get a network for the second class. Sets of 30, 60, 100 and 300 random samples were generated from the probability distributions of these two networks to be used in the training.

Next, we trained a condition-dependent network using the simulated samples with the assumption that in this simple setup the training algorithm should quite well be able to find the correct modules and identify which of the edges are condition-dependent. We used several different numbers of modules close to the correct module count.

We had two comparison network models which were trained using the same samples. The first was a multinet which consists of two separate Bayesian networks (without the class node), each trained with only the samples from one class. The networks were otherwise like normal module networks but the nodes were divided into regulators and targets as described in Section 3.2. Because only a subset of the samples was used for training each of the networks, one could assume that this model may be more likely to overfit than the condition-dependent network. The second comparison model was similar otherwise in structure to the condition-dependent network but the parents were forced to be the same in all classes. It is much harder to interpreted this kind of a network because the condition-dependent edges are not clearly labeled.

The learned networks were compared to the true generating networks by computing sensitivity and specificity of the edges averaged over the two classes. Sensitivity is the proportion of edges that are recovered by the training and specificity is the percentage of missing links in the original network that are also missing in the learned network.

For the smallest training set of 30 samples the average sensitivity of the condition-dependent network with correct number of modules in 10 trials was 27%, and for the training set of 300 samples the sensitivity was about 95%. The specificities for different sample sizes ranged from

90% to 99%. The sensitivities of the multinet were lower especially for small sample sizes; the left-hand side in the Figure 2 plots the sensitivity of the condition-specific network divided by the sensitivity of the multinet for different sample sizes and numbers of modules in the learned network. The results show that the condition-dependent network is better at avoiding overfitting and achieves higher sensitivity with a small number of samples. When plenty of samples are available the models have similar performance.

The right plot in the Figure 2 shows the sensitivity ratio between the condition-dependent network with and without class-specific distributions. The sensitivities are quite similar for both models, which means that permitting the condition-dependent edges in the model does not affect the performance very much but does make it a lot easier to find the dependency differences. The wild oscillations with the smallest samples size are probably due to overfitting. The specificities are above 90% for all tested models with the differences between the proposed and comparison models being less than 5 percent (data not shown).

4.2. Regulation of stress in yeast

We tested the method by searching for differences in regulation in the *Saccharomyces cerevisiae* yeast using gene expression measurements from stressful and normal growth conditions. Finding new regulatory mechanisms for stress might help to characterize the stress response in yeast. The stress samples are measured by Gasch et al. [3] and the samples of the normal conditions are the cell cycle samples by Spellman et al. [4]. In total, we had 80 stress samples and 53 cell cycle samples. After leaving out genes which showed negligible variance 1268 genes remained. We used a list of candidate regulators composed by Segal et al. [1], which included 69 of the genes we had selected.

We trained a condition-dependent network with 15 regulator modules and 25 target modules, which translates to 50 genes per target module on average. This is a biologically plausible figure. The optimization resulted in a network having 143 edges (excluding the outbound edges from the class node), of which 25 were associated with only the stress class, 3 with only the cell cycle class and the rest with both classes. To get some indication on the validity of the results we compared the predicted regulator genes of the condition-dependent edges, to information available at the *Saccharomyces Genome Database* [5]. All 3 of the predicted cell cycle regulators have been previously annotated as regulators of the cell cycle, and 20 out of 25 predicted stress regulators have been annotated to operate as regulators in stressful environmental conditions.

5. CONCLUSION

We have proposed a Bayesian network structure which is targeted towards discovering condition-specific gene regulatory relationships and derived an optimization algorithm which can identify condition-dependent edges. The condition-dependent edges are easier to interpret in our model that in previous models.

Using a toy example we have shown that the new model is able to find larger proportion of correct edges than a multinet model, especially when the number of observations is low. We also used the model to discover regulation differences in yeast, between stressful and normal growth conditions, with real gene expression data. The results seem promising as the majority of the edges which the method associated with only one of the conditions have been annotated to the same condition in the existing literature.

6. ACKNOWLEDGMENTS

This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. Part of the work was done at the Department of Computer Science, University of Helsinki, under a grant from the University's Research Funds. This publication only reflects the authors views. All rights are reserved because of other commitments.

7. REFERENCES

[1] E. Segal, M. Shapira, A. Regev, D. Pe'er, D. Botstein, D. Koller, and N. Friedman, "Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data," *Nature Genetics*, vol. 34, pp. 166–176, 2003.

[2] N. Friedman and M. Goldszmidt, "Learning Bayesian networks with local structure," in *Learning in Graphical Models*, M. I. Jordan, Ed., pp. 421–459. MIT Press, Cambridge, MA, USA, 1999.

[3] A. P. Gasch, P. T. Spellman, C. M. Kao, O. Carmel-Harel, M. B. Eisen, G. Storz, D. Botstein, and P. O. Brown, "Genomic expression programs in the response of yeast cells to environmental changes," *Molecular Biology of the Cell*, vol. 11, pp. 4241–4257, 2000.

[4] P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher, "Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization," *Molecular Biology of the Cell*, vol. 9, pp. 3273–97, 1998.

[5] S. S. Dwight, R. Balakrishnan, K. R. Christie, M. C. Costanzo, K. Dolinski, S. R. Engel, B. Feierbach, D. G. Fisk, J. Hirschman, E. L. Hong, L. Issel-Tarver, R. S. Nash, A. Sethuraman, B. Starr, C. L. Theesfeld, R. Andrada, G. Binkley, Q. Dong, C. Lane, M. Schroeder, S. Weng, D. Botstein, and J. M. Cherry, "Saccharomyces genome database: Underlying principles and organisation," *Briefings in Bioinformatics*, vol. 5, no. 1, pp. 9–22, 2004.