Reverse Engineering of Genetic Networks with Bayesian Networks

Dirk Husmeier
Biomathematics and Statistics Scotland (BioSS)
JCMB, The King's Buildings
Edinburgh EH9 3JZ, United Kingdom
Email: dirk@bioss.ac.uk

July 14, 2003

Introduction

There have been several approaches to the reverse engineering of genetic regulatory networks from gene expression data. At the most refined level of detail is a mathematical description of the biophysical processes in terms of a system of coupled differential equations [1], which, however, is restricted to very small systems. At the other extreme is the coarse-grain approach of clustering [2, 3]. While clustering provides a computationally cheap way to extract useful qualitative information about co-expression of genes from large-scale expression data sets, it does not lead to a fine resolution of the interaction processes between the genes. A promising compromise between these two extremes is the approach of Bayesian networks, which were first applied to gene expression data by Friedman et al. [4]. Bayesian networks are interpretable and flexible models for representing conditional dependence relations between multiple interacting quantities, and their probabilistic nature is capable of handling noise inherent in both the biological processes and the microarray experiments. However, the inference problem is particularly hard in that interactions between hundreds of genes have to be learned from very small data sets, typically containing only a few dozen time points during a cell cycle. The objective of the present study is to test the viability of the Bayesian network paradigm in a simulation study where the objective is to learn an *a priori* known network structure from sparse training sets.

Bayesian networks

A Bayesian network is defined by a graphical structure, a family of (conditional) probability distributions, and their parameters, which together specify a joint distribution over a set of random variables of interest. The graphical structure consists of a set of *nodes* and a set of *directed edges*. The nodes represent random variables, while the edges indicate conditional dependence relations. In applying this method to the inference of genetic networks, we associate nodes with genes and their expression levels, while edges indicate interactions between the genes. A Bayesian network offers a simple and unique rule for expanding the joint probability in terms of simpler conditional probabilities. The advantage of this decomposition is that a complex system of interacting quantities can be visualized as being composed of simpler subsystems, which facilitates system interpretation and comprehension. An example is given in Figure 1. The top subfigure shows the subgraph of a Bayesian network with several directed edges leading from gene SLT2 to a group of low-osmolarity response genes. This network is part of a larger network inferred by Pe'er et al. [9] from gene expression data measured during the yeast (*S. cerevisiae*) cell cycle. The bottom of Figure 1 shows a known biological pathway: SLT2 encodes the enzyme MAP kinase, which post-translationally activates two transcription factors, which in turn activate several low-osmolarity response genes. The inferred Bayesian network thus captures the essential feature of this pathway, namely, that a group of low-osmolarity response genes is regulated by a common regulator. This provides a finer resolution than available from most clustering techniques, which merely tend to group co-regulated genes together in a monolithic block. On the other hand, the Bayesian network approach does not model the biophysical details of the regulatory pathway, which would require a more detailed mathematical description in terms of a system of differential equations.

Reverse engineering

We would like to extract genetic regulatory interactions from noisy gene expression data in the absence of general theories. This is the objective of reverse engineering, which aims to learn the network structure from the data automatically through a process of inference and learning from examples. Denote by $\mathcal{M}$ the structure of a Bayesian network, and by $\mathcal{D}$ the data. For large data sets, the objective of learning is to find the network structure that is most supported by the data $\mathcal{D}$, that is, the mode of the posterior probability $P(\mathcal{M}|\mathcal{D})$. The computation of $P(\mathcal{M}|\mathcal{D})$ involves an integral over the network parameters, which becomes analytically tractable when certain regularity

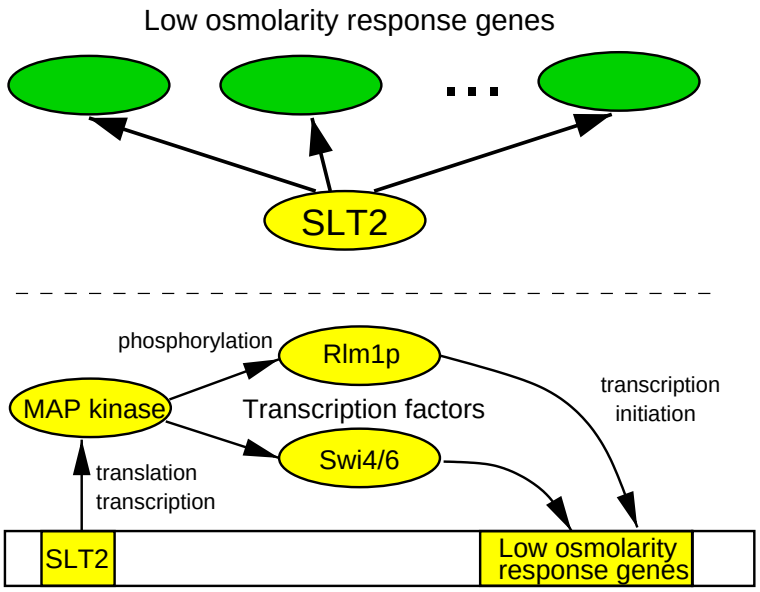

Figure 1:

conditions are satisfied [6]. Unfortunately, this closed-form solution to $P(\mathcal{M}|\mathcal{D})$ does not imply a straightforward solution to the optimization problem: the number of network structures increases super-exponentially with the number of nodes, and the optimization problem is known to be NP-hard. Moreover, gene expression data are usually sparse, with typically only a few dozen measurements during a cell cycle. This implies that the posterior distribution over structures, $P(\mathcal{M}|\mathcal{D})$, is likely to be diffuse. Consequently, $P(\mathcal{M}|\mathcal{D})$ will not be adequately represented by a single optimal structure, and it is more appropriate to sample networks from the posterior distribution $P(\mathcal{M}|\mathcal{D})$ so as to obtain a representative sample of high-scoring network structures, that is, structures that offer a good explanation of the data. Again, a direct approach is impossible due to the NP-hardness of the problem, and we therefore have to resort to a numerical approximation, using Markov chain Monte Carlo (MCMC) [5, 8].

Reliability of inference

To evaluate the performance of the inference procedure on sparse data sets, we can proceed as shown in the top of Figure 2. Synthetic data $\mathcal{D}$ are generated from a known Bayesian network. Then, new networks are sampled from the posterior distribution $P(\mathcal{M}|\mathcal{D})$ with MCMC. From a comparison between the sampled networks and the true network, we can estimate the reliability of the inference procedure as follows. First, compute the marginal posterior probabilities of all the edges from the MCMC sample.

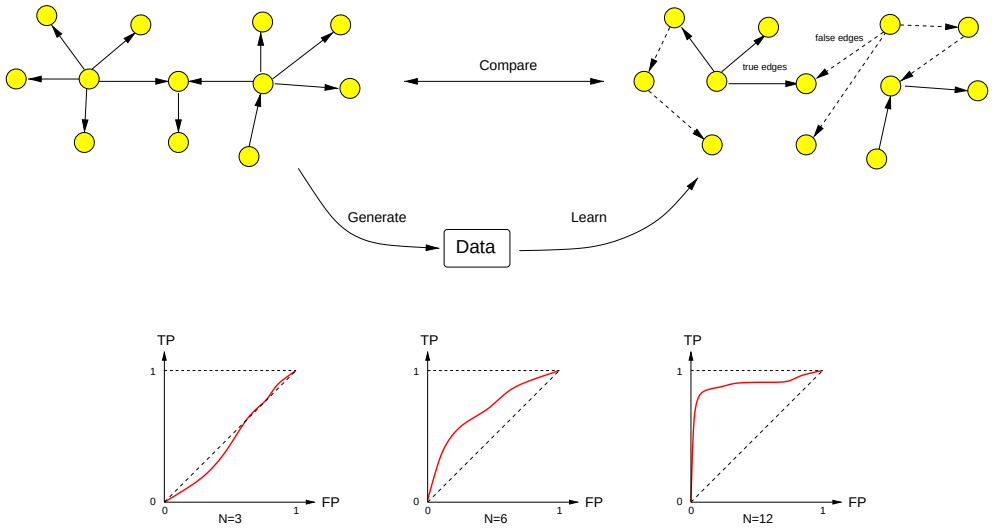

Figure 2:

Then, apply a threshold between 0 and 1, and discard all edges with posterior probability below that threshold. Next, determine the number of true and false positive edges from the resulting network. Finally, repeat this procedure for different threshold values. The results can be plotted as *receiver operator characteristics* (ROC) curves, as shown in the bottom of Figure 2, where the proportion of true edges (TP) is plotted against the proportion of false edges (FP). The diagonal dashed line indicates the expected ROC curve for a random predictor. A ROC curve that follows the left vertical axis and then runs parallel to the horizontal axis at a value of 1 indicates a perfect retrieval of all true edges without incurring any spurious edges. In general, ROC curves are between these two extremes, with a larger *area under the ROC curve* indicating a better performance.

Results

Binary data (corresponding to up- and down-regulation of genes) were generated from a Bayesian network[1] of 12 nodes, whose structure is shown in Figure 2, and whose conditional probabilities associated with the edges were binomial distributions. The generated training sets were sparse, containing only 3, 6, and 12 exemplars. The resulting ROC curves are shown in the bottom of Figure 2. A training set of size 3 gives a ROC curve similar to that of a random predictor, indicating that no real structures of the true network have been learned. For a training set of size 6, the leading edges of the true network can be learned, but they are obscured by a considerable amount of false

[1] To avoid ambiguity in the edge directions, a dynamic Bayesian network was used. See [7] for details.

edges. A training set of size 12, however, allows a considerable amount of true edges to be recovered without incurring any notable contamination by spurious edges.

Discussion

Synthetic simulations provide important clues about whether it is meaningful to try and infer complex network structures from sparse training sets. However, the results obtained are over-optimistic because the same model is used for data generation and inference. When trying to infer genetic networks from real expression data, the inherent mismatch between the underlying data-generating process and the model used for inference is likely to render the inference problem harder and, therefore, to lead to less favourable results.

In an attempt to achieve a more realistic estimation, several authors have tested their inference methods on real microarray data, testing if *a priori* known gene interactions (reported in the biological literature) could be recovered with their learning algorithms. This approach suffers from the absence of known gold standards: when predicting a gene interaction that is not supported by the literature, it is impossible to decide, without further expensive interventions in the form of multiple gene knock-out experiments, whether the algorithm has discovered a new, previously unknown interaction, or whether it has flagged a false edge.

A better approach would be to test the performance of the inference scheme on realistic simulated data, for which the true network is known and the data-generating processes are similar to those found in real biological systems. Space restrictions do not allow this approach to be discussed in the present paper. The interested reader is referred to [7], where first results can be found.

Acknowledgements

This work was funded by the Scottish Executive Environmental and Rural Affairs Department (SEERAD).

References

[1] T. Chen, H. L. He, and G. M. Church. Modeling gene expression with differential equations. *Pacific Symposium on Biocomputing*, 4:29–40, 1999.

[2] P. D'haeseleer, S. Liang, and R. Somogyi. Genetic network inference: from co-expression clustering to reverse engineering. *Bioinformatics*, 16(8):707–726, 2000.

[3] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences of the United States of America*, 95:14863–14868, 1998.

[4] N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. *Journal of Computational Biology*, 7:601–620, 2000.

[5] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109, 1970.

[6] D. Heckerman. A tutorial on learning with Bayesian networks. In M. I. Jordan, editor, *Learning in Graphical Models*, Adaptive Computation and Machine Learning, pages 301–354, Cambridge, Massachusetts, 1999. MIT Press.

[7] D. Husmeier. Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics*, 2003. In press.

[8] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[9] D. Pe'er, A. Regev, G. Elidan, and N. Friedman. Inferring subnetworks from perturbed expression profiles. *Bioinformatics*, 17:S215–S224, 2001.