Report: Probabilistic Boolean Networks

David Kaltenpoth

July 23, 2017

1 Introduction

With the ubiquitous availability of *omics data, and the shortcoming of classical methods in dealing with these sources, systems biology has come to the forefront of biological inquiry. Within systems biology, different modelling approaches try to infer the cellular networks underlying the data generating process in a variety of ways.

One of these methods is logical modelling, where the state of each gene g takes values in $\{0, 1\}$, and is regulated by a number of other genes $g_1, \ldots, g_m$.

2 Probabilistic Boolean Networks

For Boolean Networks (BNs) [7, 8], we consider n genes $g_1, \ldots, g_n \in \{0, 1\}$ and logical functions $f_i : \{0, 1\}^n \to \{0, 1\}$ so that

$$g_i(t+1) = f_i(g_1(t), ..., g_m(t)).$$

The set G_i of genes on which f_i is not constant are the *regulators* of g_i. Hence, we can also understand f_i as a function on $\{0, 1\}^{|G_i|}$.

Since cells are noisy, it makes sense to allow for perturbations. These models are called Boolean Networks with perturbations (BNps). In this case

$$g_i(t+1) = f_i(g_1(t), ..., g_m(t)) \oplus z_i$$

where the $z_i \sim \text{Bern}(p)$ are for convenience taken to be i.i.d. Of course, different noise levels could be used for different genes.

To arrive at Probabilistic Boolean Networks (PBNs) [13, 12], we allow each gene g_i to be regulated by a set of functions, $\left\{f_i^{(l)} : l = 1, ..., m_i\right\}$, each picked with probability $c_i^{(l)}$ such that

$$g_i(t+1) = f_i^{(l)}\left(g_1(t), ..., g_m(t)\right) \oplus z_i \text{ with probability } c_i^{(l)}.$$

2.1 Threshold PBNs

Since in general there are far too many Boolean functions f_i on m inputs and therefore the model class is too rich, we restrict ourselves to a more restricted set of PBNs, called *Threshold PBNs.* Here, the Boolean functions $f_i^{(l)}$ are of the form

$$f_i^{(l)}\left(g_1, ..., g_m\right) = \begin{cases} 1, & \sum_{j=1}^m a_{ij}^{(l)} g_j > 0 \\ 0, & \sum_{j=1}^m a_{ij}^{(l)} g_j < 0 \\ g_i, & \text{otherwise} \end{cases}$$

These restrictions are also quite plausible, and additive effects of regulators along with threshold dynamics are commonly used in systems biology [1, 9].

3 Method

3.1 The loss function

In the following, all logarithms are taken to the base two.

Given a timeseries of gene measurements $D = \{g_i(t) : t = 1, ..., T, i = 1, ..., n\}$, our goal is to find that PBN $N = \left\{f_i^{(l)}, c_i^{(l)}\right\}$ which minimizes

$$-\log P(N|D) = -\log P(D|N) - \log P(N).$$

Here, the probability of the data under the network is given by

$$\begin{aligned} P(D|N) &= \prod_{t=2}^{T}\prod_{i=1}^{m} P\left(g_i(t)|G(t-1), N\right) \\ &= \prod_{t=2}^{T}\prod_{i=1}^{m}\left(p\sum_{l=1}^{m_i} c_i^{(l)} 1_{f_i^{(l)}(G(t-1))=g_i(t)} + (1-p)\sum_{l=1}^{m_i} c_i^{(l)} 1_{f_i^{(l)}(G(t-1))\neq g_i(t)}\right), \end{aligned}$$

where $G(t-1) = \{g_i(t-1) : i = 1, ..., n\}$. Here we used that the process generated is a first order Markov chain and that $G(t)$ is independent given $G(t-1)$.

Furthermore, for the prior $P(N)$ we use an MDL approach as follows.

Generally, we encode each $f_i^{(l)}$ independently, i.e. we do not assume that different functions share any input or structure worth exploiting.

Hence, we assume that $-\log P(N)$ is of the form

$$-\log P(N) = \sum_{i,l} L(f_i^{(l)})$$

where $L(f)$ is an encoding of the function f.

First, for each function $f_i^{(l)}$ we have to encode how many inputs $k_{i,l}$ it takes, taking $L_{\mathbb{N}}(k_{i,l})$ bits, where $L_{\mathbb{N}}(n) = \log n + \log\log n + \log\log\log n + ... + \log c_0$ where sum goes only over positive terms, and $c_0 \approx 2.865$ is such that $\sum_n 2^{-L_{\mathbb{N}}(n)} = 1$. This is a universal code for the integers [5].

We also need to encode which genes are inputs to the function. We can do using $\log \binom{n}{k_{i,l}}$ bits. Last, we need to encode the parameters $a_i \in \{-1, 1\}$ which requires $\log 2^{k_{i,l}} = k_{i,l}$ bits.

The overall prior for N is therefore

$$-\log P(N) = \sum_{i=1}^{m} \left(\sum_{l=1}^{m_i} L_{\mathbb{N}}(k_{i,l}) + \log \binom{n}{k_{i,l}} + k_{i,l} \right)$$

In total, we therefore need to minimize the overall loss function

$$\begin{aligned} & -\log P(D|N) - \log P(N) \\ = & \prod_{t=2}^{T} \prod_{i=1}^{m} \left(p \sum_{l=1}^{m_i} c_i^{(l)} 1_{f_i^{(l)}(\{g_j(t-1)\}) = g_i(t)} + (1-p) \sum_{l=1}^{m_i} c_i^{(l)} 1_{f_i^{(l)}(\{g_j(t-1)\}) \neq g_i(t)} \right) \\ & + \sum_{i=1}^{m} \left(\sum_{l=1}^{m_i} L_{\mathbb{N}}(k_{i,l}) \log \binom{n}{k_{i,l}} + k_{i,l} \right) \end{aligned}$$

with respect to $\left\{ a_{ij}^{(l)}, c_i^{(l)} \right\}$.

3.2 Minimizing the Loss

It shouldn't be surprising that global optimization of this loss function is not feasible. We therefore use a greedy bottom-up approach to find a PBN which explains the data well.

Further, since the optimizations for every gene are independent, we will focus only on the optimization for $g = g_1$ across all time steps, and drop all unnecessary indices.

Starting from the empty net N_0, we start by building a single BN as follows: Given distinct $G_k := g_{j_1}, ..., g_{j_k}$, $A_k := a_{j_1}, ..., a_{j_k}$, find

$$g_{j_{k+1}}, A_{k+1} := \text{argmin}_{g',A'} \left(-\log P\left(g | f_{(G_k,g'),A'}\right) + L(f_{(G_k,g'),A'}) \right),$$

where $f_{G,A}$ is the threshold function with inputs G and parameters A. The term $-\log P(g|f_{G,A})$ should be understood as $-\sum_{t=2}^{T} \log P(g(t)|f_{G,A}, G(t-1))$.

That is, at every step we add a single gene to the regulatory set, and do a joint optimization over all parameters again. This joint optimization has been found to perform much better than picking only a single new $a_{j_{k+1}}$ while leaving the previous parameters A_k fixed.

New genes g' are added until the loss function $-\log P(g|f) + L(f)$ doesn't improve any longer by adding more genes to the inputs of f. In practice, limiting the number of potential inputs to $k = 3, 5$ tends to be a good idea to improve running time without hurting performance too much.

Once we have found $f^{(1)}, ..., f^{(L-1)}$, we try to find a new $f^{(L)}$ as follows.

For lack of a better heuristic, start by setting $c^{(L-1)}, c^{(L)} \leftarrow c^{(L-1)}/2$. Given c, $g_{j_1}^{(L)}, ..., g_{j_k}^{(L)}$, find $g_{j_{k+1}}^{(L)}$ and $A_{k+1}^{(L)}$ as before

$$g_{j_{k+1}}^{(L)}, A_{k+1}^{(L)} :=$$
$$\text{argmin}_{g',A'} \left(-\log P\left(g| \left\{ f^{(l)}, f_{(G_k^{(L)},g'),A'}^{(L)} \right\}\right) - \log P\left(\left\{ f^{(l)}, f_{(G_k^{(L)},g'),A'}^{(L)} \right\}\right)\right),$$

i.e. we optimize the inputs and parameters for the new function given all other functions in the same way as we did when there were no other functions. It should be noted that this is not at all similar to finding simply a number of functions which all do a reasonably good job of explaining the temporal evolution by themselves and then using a weighted combination of these to create the PBN. Instead, each new regulatory function has to explain parts of the data different from previous inferred functions to be worth the penalty incurred for putting it in the PBN.

Then, given all the $\left\{G_k^{(l)}, A_k^{(l)}\right\}$, we find

$$c := \operatorname{argmin}_{c'} \left(-\log P\left(g | \left\{f^{(i)}\right\}, c'\right) - \log P\left(\left\{f^{(i)}\right\}, c'\right)\right),$$

i.e. we find the best probabilities given all the other parameters.

The procedure stops when no new genes are found.

After all $f^{(i)}, c^{(i)}$ have been computed, we can prune all functions for which $c^{(i)} < t$ where t is an arbitrary threshold. If no pruning is desired, $t = 0$.

A high-level overview of the algorithm is given in...

4 Results

All experiments are performed with a cap of 3 inferred inputs per regulatory function, and a maximum of 3 inferred regulatory functions per gene.

To evaluate the networks, we compare the inferred network to the true network both in terms of the loss as well as the F-score in finding the correct edges of the true network.

To compare the loss functions, we generate 50 new time series à 10 points and compute the total loss for both the inferred as well as the true network.

These losses are shown in Figure 1 for the true network (red) and the inferred network (blue) for sizes 5, 10, 20, 50 and 5 inputs per gene. The inferred network never does much (more than a factor 2, say,) worse than the true network. However, it is not clear if something similar can be said for much larger network sizes.

This is all despite the case that we have put a hard constraint on the number of inputs per function which is lower than the true number.

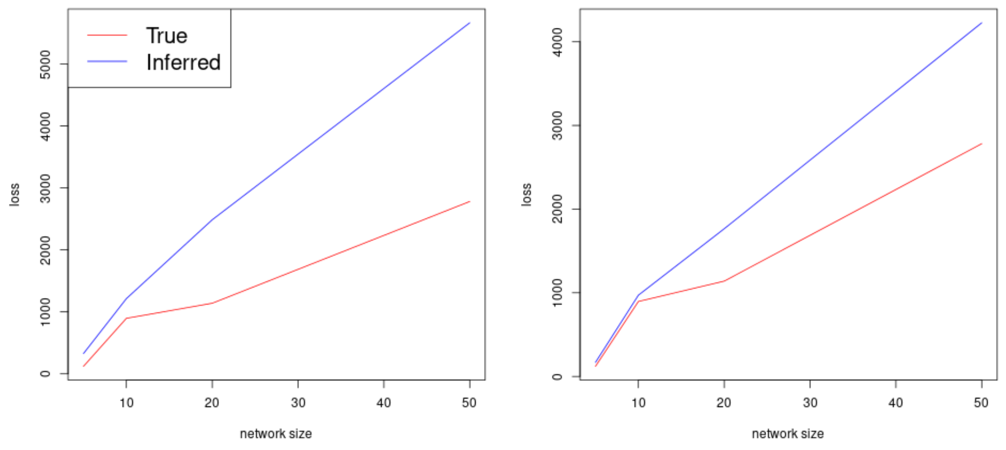

Figure 1: Losses for true (red) and inferred (blue) networks for $n = 5, 10, 20, 50$ genes using a scalefree topology with exponent 2.5. Left: 20 timeseries à 10 points. Right: 50 timeseries à 10 points.

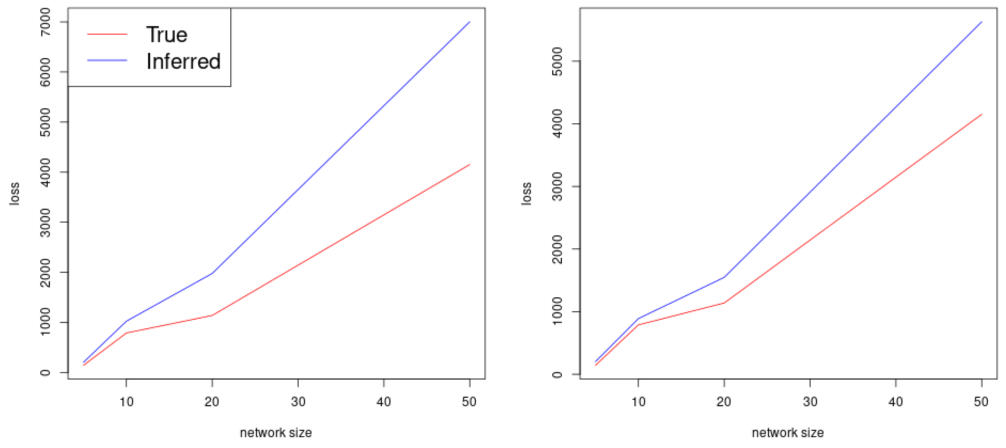

Figure 2: Losses for true (red) and inferred (blue) networks for $n = 5, 10, 20, 50$ genes using a scalefree topology with exponent 2.5. Left: 20 timeseries à 10 points. Right: 50 timeseries à 10 points.

Similar, in Figure 2 we find similar results for scale-free networks with indegrees distributed according to $d^{-\gamma}$ with $\gamma = 2.5$.

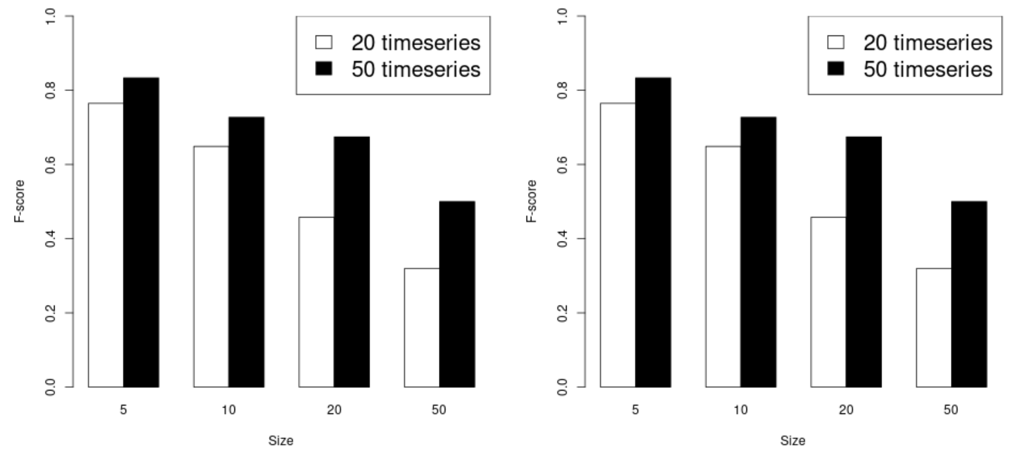

Figure 3: F-score of the inferred network for sizes $n = 5, 10, 20, 50$. Left: Homogeneous topology. Right: Scalefree topology.

Note that while both true and inferred networks do worse here, the inferred network doesn't do any worse relative to the true network despite the fact that our constraint on using only few inputs should hurt us even more in this case.

In Figure 3 the F-scores for networks of different size and using different number of experiments are shown.

Further, in Figure 4, we see that for homogeneous topology on the network, the runtime scales roughly quadratically with the size of the networks. This is exactly what we would have expected simply from the description of the algorithm above.

Further comparing the two plots it appears like the time taken for the inference is roughly linear in the number of parallel time series we are running. This also isn't surprising.

As the amount of time taken for networks with scalefree topology is virtually indistinguishable from those shown in Figure 4, they are not shown additionally.

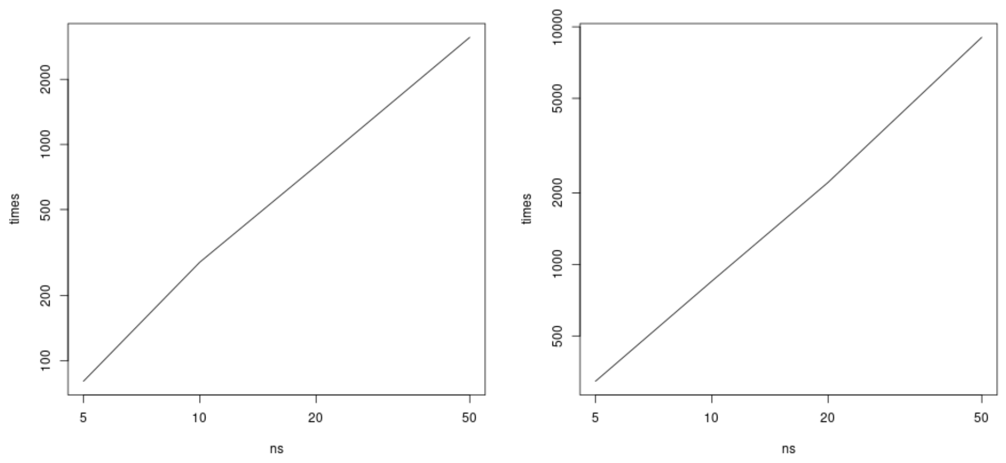

Figure 4: Times for network inference in seconds for $n = 5, 10, 20, 50$ genes using homogeneous topology. Left: 20 timeseries à 10 points. Right: 50 timeseries à 10 points.

5 Further Ideas

From a coding perspective, instead of computing $P(D|N)$ by summing over probabilities of networks predicting the correct outcome, one could also use $H\left(\left\{c_j^{(l)} : l = 1, ..., L\right\}\right)$ bits to encode which network is the "correct one" at every time step and try to find that model which requires the least number of bits to encode the outcome.

Furthermore, since the network inference process quickly becomes expensive the more data is used, it might be a good idea to use subsampling methods.

Another way to reduce the computational load here would be to consider only those genes which show a high level of (lagged) mutual information with the target genes.

6 Related research

Full Bayesian Inference for BNs have been made use of only quite recently [6]. Previously, the MDL principle has been used for inferring gene regulatory networks in [14, 15, 17, 4, 2, 3]. However, these approaches were always applied to finding a single model like a BNp rather than a combination of BNs as in a PBN. Previous attempts at inferring PBNs from timeseries data were [10, 11], trying to make use of context switches between different PBNs and inferring one BN for every single such context. However, they required timeseries consisting of hundreds of contiguous measurements and were found to have problems finding context switches even then. Another approach by [16] uses logic optimization, with a naive 0-1-loss.

References

[1] U. Alon. *An introduction to systems biology: design principles of biological circuits.* CRC press, 2006.

[2] V. Chaitankar, C. Zhang, P. Ghosh, E. J. Perkins, P. Gong, and Y. Deng. "Gene Regulatory Network Inference Using Predictive Minimum Description Length Principle and Conditional Mutual Information." In: 2009, 487–490.

[3] V. Chaitankar, P. Ghosh, E. J. Perkins, P. Gong, Y. Deng, and C. Zhang. A novel gene network inference algorithm using predictive minimum description length approach. *BMC Systems Biology* **4** (2010), S7.

[4] J. Dougherty, I. Tabus, and J. Astola. Inference of Gene Regulatory Networks Based on a Universal Minimum Description Length. *EURASIP Journal on Bioinformatics and Systems Biology* **2008**.1 (2008), 482090.

[5] P. D. Grünwald. *The minimum description length principle.* MIT press, 2007.

[6] S. Han, R. K. W. Wong, T. C. M. Lee, L. Shen, S.-Y. R. Li, and X. Fan. A Full Bayesian Approach for Boolean Genetic Network Inference. *PLOS ONE* **9** (2014), e115806.

[7] S. A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of theoretical biology* **22** (1969), 437–467.

[8] S. A. Kauffman. *The origins of order: Self-organization and selection in evolution.* Oxford University Press, USA, 1993.

[9] E. Klipp, W. Liebermeister, C. Wierling, A. Kowald, and R. Herwig. *Systems biology: a textbook.* John Wiley & Sons, 2016.

[10] S. Marshall, L. Yu, Y. Xiao, and E. R. Dougherty. "Temporal Inference of Probabilistic Boolean Networks." In: *Genomic Signal Processing and Statistics, 2006. GENSIPS'06. IEEE International Workshop On*. IEEE, 2006, 71–72.

[11] S. Marshall, L. Yu, Y. Xiao, and E. R. Dougherty. Inference of a Probabilistic Boolean Network from a Single Observed Temporal Sequence. *EURASIP Journal on Bioinformatics and Systems Biology* **2007**.1 (2007), 32454.

[12] I. Shmulevich and E. R. Dougherty. *Probabilistic Boolean networks: the modeling and control of gene regulatory networks*. SIAM, 2010.

[13] I. Shmulevich, E. R. Dougherty, S. Kim, and W. Zhang. Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks. *Bioinformatics* **18**.2 (2002), 261–274.

[14] I. Tabus and J. Astola. On the use of MDL principle in gene expression prediction. *EURASIP Journal on Applied Signal Processing* **2001** (2001), 297–303.

[15] I. Tabus, J. Rissanen, J. Astola, W. Zhang, and I. Shmulevich. Normalized Maximum Likelihood Models for Boolean Regression with Application to Prediction and Classification in Genomics. *Statistics and Computing* (2003), 173–189.

[16] P. Trairatphisan, A. Mizera, J. Pang, A. A. Tantar, and T. Sauter. optPBN: An Optimisation Toolbox for Probabilistic Boolean Networks. *PLOS ONE* **9** (2014), e98001.

[17] W. Zhao, E. Serpedin, and E. R. Dougherty. Inferring Gene Regulatory Networks from Time Series Data Using the Minimum Description Length Principle. *Bioinformatics* **22**.17 (2006), 2129–2135.