Fast inference and sampling of NEMs?

Jack Kuipers jack.kuipers@bsse.ethz.ch
D-BSSE, ETH Zurich
Mattenstrasse 26
4058 Basel, Switzerland

Abstract

Faster method to estimate NEMs from interventional data?

Keywords: NEM, Bayesian Networks, Structure Learning, MCMC.

1. Introduction

NEMs (Markowetz et al., 2007) are a class of probabilistic graphical models with two types of nodes: the n signalling nodes which when knocked down affect all their descendants, and m effect nodes whose state depends on the state of its parent signalling node.

The space of NEMs is essentially built on DAGs – officially some kind of cycles can exist, but all nodes along a cycle can be collapsed into one node to return to the space of DAGs. We will focus on the DAG case.

Since a knockdown affects all descendants, this is usually represented by transitively closing the DAG. Then the state of each signalling node is ON if itself or any of its parents are ON.

The effects nodes are attached to the signalling nodes as an additional layer. We can represent their parents as an attachment vector (like we do for the single-cell trees, which are actually transitively closed trees with attached single cells) $\boldsymbol{\sigma}$ with each element the parent node of that effect node. An effect node can also be disconnected (or we connect it to a dummy node with label $(n+1)$ say).

Say we take the example in Figure 1 and attachment vector $\boldsymbol{\sigma} = (1, 4, 3, 2, 5, 5, 2, 1)$ for the 8 effect nodes. If we knockdown signalling node 3 say, then we would expect to observe the following effect node vector $(0, 1, 1, 1, 0, 0, 1, 0)$ while if we knockdown signalling node 1 we would expect to observe $(1, 0, 0, 1, 0, 0, 1, 1)$.

By setting the false positive rate to α and the false negative rate to β we compute the likelihood of the observed data for the given DAG and our set of N knockdown experiments. See Markowetz et al. (2007) and related papers for details/examples.

Here let's say we observe the matrix

	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8
K_1	1	1	0	1	0	0	1	1
K_3	0	0	1	1	0	0	1	0

(1)

Then the likelihood would be

$$\alpha(1-\alpha)^7\beta(1-\beta)^7 \tag{2}$$

since we have one error in each direction for E_2.

This can be adapted to continuous data for the effect nodes, and can be more efficiently computed by calculating the relative likelihoods to the case where all the effect nodes are

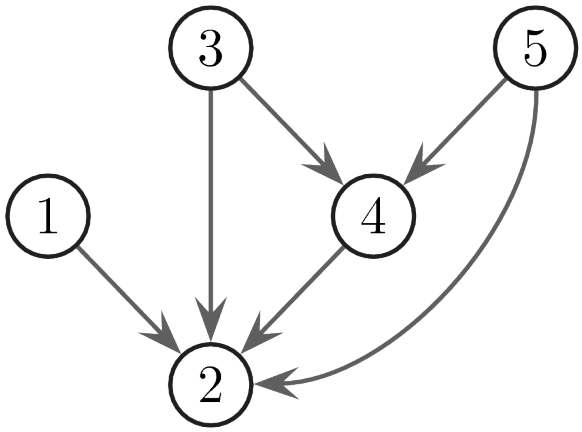

Figure 1: An example transitively closed DAG

disconnected (and we would always expect their state to stay 0) or where we perfectly explain the data (in which case we only count differences) – see Tresch and Markowetz (2008). For example, the log score from before is simply

$$\log\left(\frac{\alpha}{1-\beta}\right) + \log\left(\frac{\beta}{1-\alpha}\right) + C \tag{3}$$

we let

$$A = \log\left(\frac{\alpha}{1-\beta}\right), \qquad B = \log\left(\frac{\beta}{1-\alpha}\right) \tag{4}$$

for later.

2. Outline

The main problem with learning NEMs is that we require the DAGs to be transitively closed. This is a global property and so we cannot perform the local decomposition we use for fast inference and sampling of BNs using our order and partition MCMC schemes. The second issue is the marginalisation over the attachment of effect nodes. This massively speeds up structure based searches by reducing the search space, but interferes even more with the local decomposition. NEMs are then essentially limited to small graphs or heuristic methods of dividing the nodes into smaller groups and trying to combine them into a single NEM at the end.

To make inference of moderate sized NEMs feasible we want to undo these issues and map the problem to something we can tackle with our order MCMC methods (Kuipers and Moffa, 2017; Kuipers et al., 2018). We look at two directions:

1. Scoring the order for a given attachment vector
2. Finding the highest scoring DAG in an order with marginalised attachments.

The first direction we have looked into, but the attachment leads to local optima and is not so promising. The notes in the next section give some details of the idea. For this project we look in the second direction outlined in the section afterwards.

3. Approach 1: Gibbs sampling

First we turn the sampling into a Gibbs scheme:

- For a given DAG, sample the attachment vector $\boldsymbol{\sigma}$ proportionally to their scores
- For a given attachment vector $\boldsymbol{\sigma}$ sample the DAG using an order type scheme

In the first step, each effect node can be placed independently so we would have a complexity of up to mnN. For the second step we would need to define the inference scheme.

3.1 Building score tables

First we treat the parents of each node separately, so we are not yet enforcing acyclicity or transitive closure. Each effect gene is connected to exactly one signalling gene. Let's build the score table when the E-genes are connected to signalling gene 1:

If we have a single knockdown per experiment, we can treat the parents of gene 1 independently so that the only affect the E-genes if the knockdown is a direct parent of the signalling gene. We compute the table

	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8
S_1	0	0	$A+B$	A	B	B	A	0
S_2	$+0$	$+0$	$+0$	$+0$	$+0$	$+0$	$+0$	$+0$
S_3	$+B$	$+B$	$-A$	$-A$	$+B$	$+B$	$-A$	$+B$
S_4	$+0$	$+0$	$+0$	$+0$	$+0$	$+0$	$+0$	$+0$
S_5	$+0$	$+0$	$+0$	$+0$	$+0$	$+0$	$+0$	$+0$

(5)

The first row is the (relative) log score of having just S-gene 1 as a parent for each E-gene, where we combine the two knockout experiments into a single score. The other rows are the increase in score by having the row label as an **additional** parent. For example for E-gene 1, there are 16 possible combinations of parents, with scores

parents	$\emptyset$	S_2	S_3	S_4	S_5	S_1, S_2	S_1, S_3	S_1, S_4	...
score	0	0	B	0	0	0	B	0	...

(6)

Now if we only have one gene attached, say E-gene 1, we can easily compute the total scores of all possibilities. For this we exponentiate the first column, and add 1 to all rows apart from the first and take the product:

$$\mathrm{e}^0(1+\mathrm{e}^0)(1+\mathrm{e}^B)(1+\mathrm{e}^0)(1+\mathrm{e}^0) \tag{7}$$

Expanding the product gives us the 16 terms above with their relative likelihoods. If we want to sample the parent set given that E-gene 1 is attached then we can sample one of the two choices in each bracket independently.

If several genes are attached, then their scores are correlated. Say we have E-genes 1, 2 and 5 attached to S-gene 1 then we would simply add those columns before exponentiating and adding 1 and taking the product.

Given the attachments, we can then easily sample the parents for each S-gene. Given the parents for each S-gene we select the appropriate combination of rows for each table

and then sample among the tables for each S-gene to sample the attachments. Sampling attachments takes $O(mn)$, while building all the score tables in the end takes $O(mn^2N)$ where N is the number of knockdown experiments. We also need to include a score vector for the case when the E-genes are disconnected (we make a dummy node with expected status of 0 for all knockdowns).

3.2 Order based sampling

So far we have had no restriction on the parent sets, so the graphs can be cyclic. To remove this, we can work again on the space of orders.

For a given order, then we simply remove rows of the table with parents outside the permissible set. For the score of each order we take the product over the rows (after selecting the columns, adding and exponentiating them, with the 1 for all but the base row).

For the transposition move, we only need to update one row of each affected table, which should take $O(m/n)$ on average. The global move should then take $O(m)$.

We can perform several iterations on the space of orders, sample an order and then the attachment vector, and then start a new order chain. Based on the complexity, we can run order $O(n^2)$ steps (balanced between transpositions and global swaps, or even our new single node move) for each order. Scoring the starting order in the first place should take $O(mn)$, which is updated as above, and then sampling the attachment vector is also $O(mn)$ so each Gibbs sampling step has this complexity.

3.3 Transitive closure

NEMs should however be transitively closed. For a given attachment vector we can sample a DAG using the order scheme (in a biased way) and then compute the transitive closure. By comparing the likelihood of the transitively closed DAG to the original one, we can perform importance sampling. Obviously going from DAGs to their transitive closure introduces another bias, which we can do little about, but probably we need not worry overly about the bias of the order representation. To 'remove' the bias, one step would be to start standard structure moves from a sample from the importance sampled transitive closures.

3.4 Double knockdowns

If we have double knockdown experiments, then the parents can no longer be seen as independent. Say we changed the K_3 knockdown to $K_{2,3}$ then we would have the terms

$$(1 + \mathrm{e}^B)(1 + \mathrm{e}^B) \tag{8}$$

for the parts where S_2 and S_3 are possible parents of S_1 for the attached E-gene 1. However, we could no longer expand and would need to treat the four possibilities explicitly

$$(1 + \mathrm{e}^B + \mathrm{e}^B + \mathrm{e}^B) \tag{9}$$

These sort of correlations would also occur across the E-genes attached, and we would need to code them carefully!

4. Approach 2: Order-based maximisation

For a particular DAG, for each E-gene we add the log-scores of the appropriate rows in the score table for each node, but then add in **real** space the scores of the different nodes. If s_i is the score of the E-gene for node i, then we need $\log \sum_{i=1}^{n} \exp(s_i)$. Often we refer to this as log-add (for numerical accuracy we also first subtract the maximum s_i and add it back after the exponentiation etc). For a given order, we don't know which DAG so to find the highest scoring DAG per order we add some auxillary variables.

First, for each node and score table, we extract the rows compatible with the order. Say in the example above, nodes 3 and 5 are permissible parents so we keep those rows:

	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8
S_1	0	0	$A+B$	A	B	B	A	0
S_3	$+B$	$+B$	$-A$	$-A$	$+B$	$+B$	$-A$	$+B$
S_5	$+0$	$+0$	$+0$	$+0$	$+0$	$+0$	$+0$	$+0$

(10)

Now, akin to equation (7) we exponentiate and combine columnwise, but this time we take a weighted average between 1 and the exponentiated scores in the lower rows:

	E_1	E_2	E_3	$\dots$
S_1	e^0	e^0	e^{A+B}	$\dots$
S_3	$\left(1-\rho+\rho\mathrm{e}^{B}\right)$	$\left(1-\rho+\rho\mathrm{e}^{B}\right)$	$\left(1-\rho+\rho\mathrm{e}^{-A}\right)$	$\dots$
S_5	$\left(1-\pi+\pi\mathrm{e}^{0}\right)$	$\left(1-\pi+\pi\mathrm{e}^{0}\right)$	$\left(1-\pi+\pi\mathrm{e}^{0}\right)$	$\dots$

(11)

where the coefficients ρ and π are the weights. If ρ is 0 we get the score when node 3 is not included as a parent, while for $\rho = 1$ we have the score when node 3 is a parent. Similarly for π and node 5 (though here it makes no difference to the score).

Next we take a product column-wise and take logs to store the scores in log space again.

$$\begin{aligned} s = [&\log\left(1-\rho+\rho\mathrm{e}^{B}\right)+\log\left(1-\pi+\pi\mathrm{e}^{0}\right), \\ &\log\left(1-\rho+\rho\mathrm{e}^{B}\right)+\log\left(1-\pi+\pi\mathrm{e}^{0}\right), \\ &A+B+\log\left(1-\rho+\rho\mathrm{e}^{-A}\right)+\log\left(1-\pi+\pi\mathrm{e}^{0}\right), \\ &\dots] \end{aligned} \tag{12}$$

Next we compute a similar s vector over the E-genes for each node i, and log-add the vectors element-wise. The final vector is added normally (in log-space) to get the total log score, which however depends on all the auxilliary variables.

To write that more explicitly in terms of formulas,

$$s_j^i = S_{1,j}^{i,\prec} + \sum_{k=1}^{K^{i,\prec}} \log\left(1-\rho_k^i+\rho_k^i \mathrm{e}^{S_{k-1,j}^{i,\prec}}\right) \tag{13}$$

where $S^{i,\prec}$ is the table of log scores, filtered by rows compatible with the order $\prec$ and $K^{i,\prec}$ is the number of rows in the table excluding the first one. The total log score of the order is then

$$l = \sum_j \log \sum_i \mathrm{e}^{s_j^i} \tag{14}$$

To try to maximise this, we can look at the difference in log-likelihood when varying a single weight component

$$l(\tilde{\rho} - \tilde{\rho}_k^i, \rho_k^i) - l(\tilde{\rho}) = \sum_j \log\left(w_j^i \frac{1 - \rho_k^i + \rho_k^i \mathrm{e}^{S_{k-1,j}^{i,\prec}}}{1 - \tilde{\rho}_k^i + \tilde{\rho}_k^i \mathrm{e}^{S_{k-1,j}^{i,\prec}}} + 1 - w_j^i\right) \tag{15}$$

where $\tilde{\rho}$ are the values of the weights, for example from the previous optimisation step and the w_j^i are the relative weight of each cell to the different nodes.

$$w_j^i = \frac{\mathrm{e}^{s_j^i}}{\sum_i \mathrm{e}^{s_j^i}} \tag{16}$$

In equation (15), the denominator of the fraction is a constant, so we multiply the logs by this to arrive at

$$\sum_j \log\left(w_j^i \left[1 - \rho_k^i + \rho_k^i \mathrm{e}^{S_{k-1,j}^{i,\prec}}\right] + (1 - w_j^i)\left[1 - \tilde{\rho}_k^i + \tilde{\rho}_k^i \mathrm{e}^{S_{k-1,j}^{i,\prec}}\right]\right) \tag{17}$$

Let the vector A have elements $a_j = w_j^i \left(\mathrm{e}^{S_{k-1,j}^{i,\prec}} - 1\right)$ then we can rewrite as

$$\sum_j \log\left(\rho_k^i a_j - \tilde{\rho}_k^i a_j + 1 + \tilde{\rho}_k^i \left(\mathrm{e}^{S_{k-1,j}^{i,\prec}} - 1\right)\right) \tag{18}$$

setting $b_j = 1 - \tilde{\rho}_k^i a_j + \tilde{\rho}_k^i \left(\mathrm{e}^{S_{k-1,j}^{i,\prec}} - 1\right)$ we can divide by these constants and have

$$\sum_j \log\left(\rho_k^i \frac{a_j}{b_j} + 1\right) \tag{19}$$

which we optimise numerically.

4.1 Project

We wish to find the optimal values in $\{0, 1\}$ for all the auxiliary variables to maximise the log-score for a given order:

- Does numerical optimisation (starting at $\frac{1}{2}$ say) lead to reasonable optima? (Compared to result from full enumeration say)
- Is there a matching algorithm that can find the optimum in polynomial time?

Given a reasonable way to find a high scoring DAG in the order, and transitively closing it, we will adapt our order-based searches (Kuipers and Moffa, 2017; Kuipers et al., 2018) to find a good order and hence a good NEM:

- In simulation studies do we outperform current heuristic approaches?

References

J. Kuipers and G. Moffa. Partition MCMC for inference on acyclic digraphs. *Journal of the American Statistical Association*, 112:282–299, 2017.

J. Kuipers, P. Suter, and G. Moffa. Efficient sampling and structure learning of Bayesian networks. *arXiv:1803.07859*, 2018.

Florian Markowetz, Dennis Kostka, Olga G Troyanskaya, and Rainer Spang. Nested effects models for high-dimensional phenotyping screens. *Bioinformatics*, 23:i305–i312, 2007.

Achim Tresch and Florian Markowetz. Structure learning in nested effects models. *Statistical applications in genetics and molecular biology*, 7, 2008.