Bayesian network structure learning using integer programming

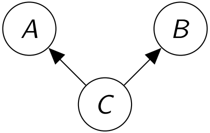
James Cussens

ISL & Interactive AI CDT seminar
University of Bristol, 2012-11-10

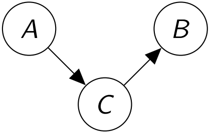
Outline

Conditional Independence in (causal?) Bayesian Networks

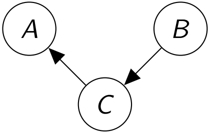
$$A \perp B | C,\; A \not\perp B$$



(a)

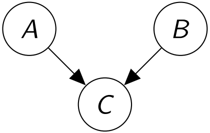


(b)



(c)

$$A \not\perp B | C,\; A \perp B$$



(d)

If the data indicates that $A \not\perp B | C$ and $A \perp B$ can we really infer that A and B are (probabilistic) causes of C?

Bayesian network structure learning (BNSL)

Constraint-based Do statistical tests to infer e.g. $A \not\perp B | C$ and $A \perp B$ and then find a DAG $\mathcal{G}$ that represents the inferred (conditional indpendence) constraints.

Score-based Choose a score function and search for $\arg\max_{\mathcal{G}} \mathrm{Score}(\mathcal{G})$.

Decomposable scores for BNs

- If a Score is *decomposable* then:

$$\mathrm{Score}(\mathcal{G}) = \sum_{u \in V} \mathrm{Score}_u(\mathcal{G}) \tag{1}$$

- where $\mathrm{Score}_u(\mathcal{G})$ only depends on the parents of u in $\mathcal{G}$.
- The score for variable u having parents W is denoted $c(W \to u)$ and is known as a *local score*.
- $c(W \to u)$ basically measures how well W predicts u (and typically incorporates a penalty for overfitting).
- Time for a little demo.

Encoding BN learning as an IP

- Create a (binary) *family variable* $I(W \to u)$ for each local score.
- $I(W \to u) = 1$ iff W are the parents of u in an optimal BN.

Then BN learning is the following IP:

Instantiate the $I(W \to u)$ to maximise:
$\sum_{u,W} c(W \to u) I(W \to u)$
subject to the $I(W \to u)$ representing a DAG.

Simple convexity constraints

Each variable has exactly one (possibly empty) parent set.

$$\forall u : \sum_{W} I(W \to u) = 1 \tag{2}$$

Jaakkola *et al*'s cluster constraint

- In each subset ('cluster') C of vertices, at least one variable has no parents in that subset:

$$\forall C \subseteq V : \sum_{u \in C} \sum_{W : W \cap C = \emptyset} I(W \rightarrow u) \geq 1 \qquad (3)$$

- Since there are exponentially many of these, they are added 'on the fly' as *cutting planes*.
- They are *facets* of the convex hull of DAGs.
- Studený showed that every connected matroid has an associated facet, (cluster constraints correspond to uniform matroids of rank one).

Branch and cut

```
1. Let x* be the LP solution.
2. If x* worse than incumbent then exit.
3. If there are valid inequalities
       not satisfied by x*
       add them and go to 1.
   Else if x* is integer-valued then
       the current problem is solved
   Else branch on a variable with
       non-integer value in x*
       to create two new sub-problems
       (propagate if possible)
```

Using integer programming

- Solvers do most of the 'heavy lifting'.
- Get anytime solving, multiple solutions, parallelisation for 'free'.
- Theory matters: Tight linear relaxations are crucial: need to know what the facets are!
- Easy to extend with additional constraints.

Exact estimation of multiple directed acyclic graphs

- Learn BNs $G^{(k)}$ for multiple related but non-identical units or ‘individuals’ $k \in \{1, 2, \ldots, K\}$.
- Improve robustness, reduce small sample bias.
- *Exchangeable learning*: Penalise structural difference of the BNs for any pair of individuals.
- *Non-exchangeable learning*: Penalise structural difference of the BNs only for *related* individuals. And learn which are related.
- Have used simulated and fMRI data.

Learning scenario

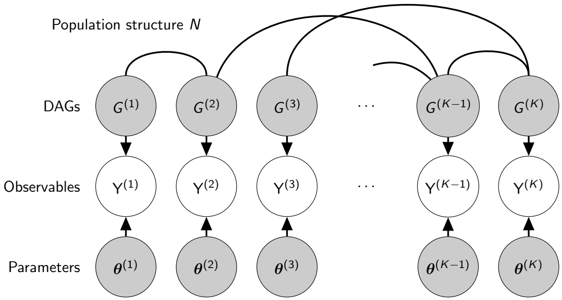


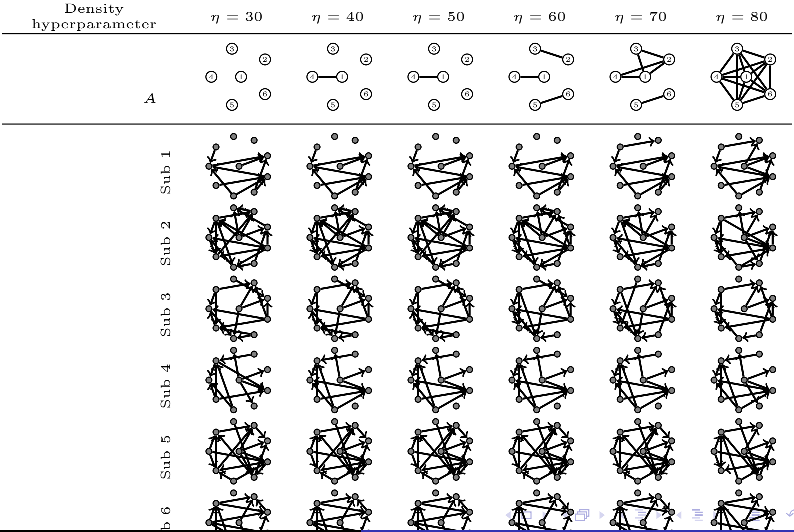
Figure: Multiple directed, acyclic graphical models (DAGs) with population structure encoded by an undirected network N. [Shaded nodes are unobserved. $G^{(1:K)}$ = data-generating graphs, $\boldsymbol{\theta}^{(1:K)}$ = data-generating parameters, $\mathrm{Y}^{(1:K)}$ = observation vectors.]

Non-exchangeable learning

The MAP estimate $(\hat{G}^{(1:K)}, \hat{N})$ is the solution of the integer linear program

$$
\begin{aligned}
(\hat{G}^{(1:K)}, \hat{N}) \quad := \quad & \underset{G^{(1:K)} \in \mathcal{G}^K, N \in \mathcal{N}}{\arg\max} \sum_{k=1}^{K} \sum_{i=1}^{P} \sum_{\pi \subseteq \{1:P\}} s^{(k)}(i,\pi)\Pi^{(k)}(i,\pi) \\
& -\lambda \sum_{i=1}^{P} \sum_{j=1}^{P} \sum_{k=1}^{K} \sum_{l=k+1}^{K} D^{(k,l)}(j,i) + \eta \sum_{k=1}^{K} \sum_{l=k+1}^{K} E^{(k,l)} \\
& \textit{subject to certain constraints}
\end{aligned}
$$

Non-exchangeable learning



Variable pricing

- There is an IP variable for every possible parent set of each BN variable.
- So we need to add IP variables 'on the fly', a process known as *variable pricing*.
- This is nothing other than adding a cutting plane to the dual of the linear relaxation.

Adding variables and constraints during solving

```
 time | cons|cols  |rows |cuts |
p 0.4s|2067 | 1923 |2064 |   0 |
  0.5s|2067 | 1984 |2064 |   0 |
  1.7s|2067 | 2043 |2064 |   0 |
  1.7s|2067 | 2083 |2110 |  46 |
..
L12.8s|2064 | 2212 |2136 | 411 |
```

Variable pricing as doubly-penalised regression

- The job of the variable pricer is to find a new IP variable which if created would allows a better solution to the current LP relaxation.
- The *reduced local score* of a variable $I(W \rightarrow u)$ is its normal local score *minus* an 'acyclicity penalty' (which is determined by the dual values of the constraints in which it will be injected).
- The reduced local score is: fit minus normal complexity penalty minus acyclicity penalty.
- Need to find new IP variables with positive reduced local score.
- For Gaussian BNs this is a mixed integer quadratic programming problem.

MIP in machine learning

- There are quite a few papers now where MIP solvers are applied to various machine learning problems.
- My approach to variable pricing is basically an extension to the MIP approach by Berstimas *et al*.
- Let's finish with a quick look at this work.