Causal Discovery via Discrete Optimisation

James Cussens, Dept of Computer Science, University of Bristol

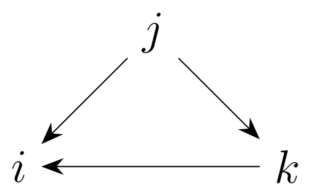
University of BRISTOL

1. Learning (causal?) directed acyclic graphs (DAGs)

Data | Learning algorithm | DAG

i	j	k
0	1	1
1	0	0
0	1	1
0	1	0
...	...	...
0	1	0

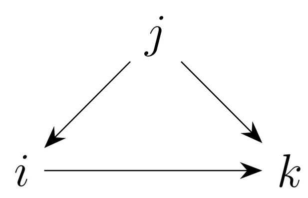
$\Rightarrow$



Data | Knowledge | Learning algorithm | DAG

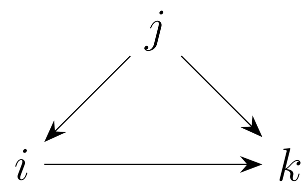
i	j	k
0	1	1
1	0	0
0	1	1
0	1	0
...	...	...
0	1	0

k cannot be an ancestor of i $\Rightarrow$



2. Encoding DAGs as vectors

This DAG

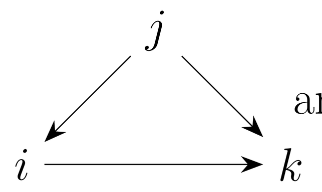


is this vector in $\mathbb{R}^{12}$:

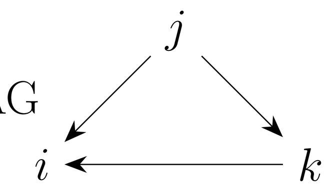
$x_{i\leftarrow\{\}}$	$x_{i\leftarrow\{j\}}$	$x_{i\leftarrow\{k\}}$	$x_{i\leftarrow\{j,k\}}$	$x_{j\leftarrow\{\}}$	$x_{j\leftarrow\{i\}}$	$x_{j\leftarrow\{k\}}$	$x_{j\leftarrow\{i,k\}}$	$x_{k\leftarrow\{\}}$	$x_{k\leftarrow\{i\}}$	$x_{k\leftarrow\{j\}}$	$x_{k\leftarrow\{i,j\}}$
0	1	0	0	1	0	0	0	0	0	0	1

- Why this encoding? Because many objective functions ('scores') for DAGs are sums of *local scores* which are determined by the choice of *parents* for each vertex.

This DAG



and this DAG



are *Markov equivalent*

- That means they represent the same (empty) set of *conditional independence relations.*
- Their Markov equivalence class has a vector representation using Studený's *characteristic imsets* which is:

$x_{\{i,j\}}$	$x_{\{i,k\}}$	$x_{\{j,k\}}$	$x_{\{i,j,k\}}$
1	1	1	1

3. Integer linear programming model for DAG learning

$$\textsc{Max} \sum_{\substack{i\in N\\ J\subseteq N\setminus\{i\}}} c_{i\leftarrow J}x_{i\leftarrow J}$$

SUBJECT TO:

$$\text{(is a directed graph)} \quad \sum_{J\subseteq N\setminus\{i\}} x_{i\leftarrow J} = 1 \qquad i\in N$$

$$\text{(is acyclic)} \quad \sum_{i\in C}\sum_{\substack{J\subseteq N\setminus\{i\}\\ J\cap C\neq\emptyset}} x_{i\leftarrow J} \leq |C|-1 \qquad C\subseteq N, |C|\geq 2$$

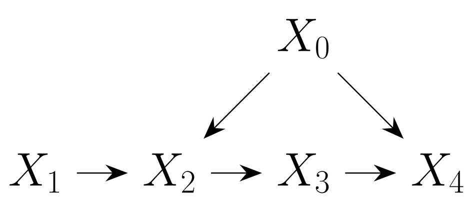
$$x_{i\leftarrow J}\in\{0,1\}, \qquad i\in N, J\subseteq N\setminus\{i\}$$

4. Solving the ILP: problems and solutions

- The ILP model has too many constraints!
  - Add only necessary (and strong) ones during solving: *cutting planes* [2]
- The ILP model has too many variables!
  - Add only necessary ones during solving: *pricing algorithm* (work in progress)
- The (exponentially many) acyclicity cutting planes are *facet-defining inequalities* (maximally tight linear inequalities) of the convex hull of DAGs [3, 4].
- Studený showed that every *connected matroid* defines a facet of this convex hull.[7]
- Facet-defining inequalities lead to tight linear relaxations and thus faster solving.

5. Detecting latent variables

- It is typically unrealistic to assume *causal sufficiency*, i.e. that all variables are observed.
- Suppose that X_0 is unobserved in the following DAG:



- The only conditional independence relations that hold between the observed variables are: $X_1 \perp X_3|X_2$ (X_1 is independent of X_3 given X_2), and $X_1 \perp X_4|X_3$.
- $X_1 \perp X_3|\{X_2,X_4\}$ and $X_1 \perp X_4|\{X_2,X_3\}$ **do not** hold, for example.
- (This model also satisfies the so-called *Verma constraint* because the quantity $q(X_4|X_3) \equiv \sum_{X_2} p(X_2|X_1)p(X_4|X_1,X_2,X_3)$ does not depend on X_1.) [5]
- Typically, latent variable models are 'discovered' via conditional independence tests on data (e.g. the *Fast Causal Inference (FCI)* algorithm).
- Optimisation based approaches are harder than in the fully observed case, but not impossible!

6. DAG learning algorithms

- DAG learning algorithms fall mainly into two camps: *score-based*—search for a DAG which maximises some score, or *constraint-based*—infer conditional independence constraints from data and construct a DAG which meets those constraints.
- There are also MCMC (model averaging) methods.
- DAG learning is NP-complete [1], so most score-based approaches are *heuristic*, but some are *exact*: they can return a guaranteed global optimum (but may be much slower or entirely useless on bigger problems).
- The ILP approach is exact and anytime (and it's not too hard to add additional constraints).
- Very many search algorithms have been applied to (score-based) DAG learning, including: dynamic programming, A^*, weighted constraint programming, hill-climbing, taboo search, various genetic algorithms, nonconvex continuous optimisation (with thresholding), weighted MAX-SAT,...
- The `benchpress` system [6] has been created to facilitate comparison of all these algorithms (and you can slot in new algorithms fairly easily).

References

[1] David Maxwell Chickering. Learning Bayesian networks is NP-complete. In *Learning from Data: Artificial Intelligence and Statistics V*. Springer, 1996.

[2] James Cussens. Bayesian network learning with cutting planes. In *Proc. UAI-2011*, pages 153–160, 2011.

[3] James Cussens, David Haws, and Milan Studený. Polyhedral aspects of score equivalence in Bayesian network structure learning. *Mathematical Programming*, 2016.

[4] James Cussens, Matti Järvisalo, Janne H. Korhonen, and Mark Bartlett. Bayesian network structure learning with integer programming: Polytopes, facets, and complexity. *Journal of Artificial Intelligence Research*, 58:185–229, 2017.

[5] Robin J. Evans. Margins of discrete Bayesian networks. *The Annals of Statistics*, 46(6A):2623 – 2656, 2018.

[6] Felix L. Rios, Giusi Moffa, and Jack Kuipers. Benchpress: a scalable and platform-independent workflow for benchmarking structure learning algorithms for graphical models. arXiv: 2107.03863, 2021.

[7] Milan Studený. How matroids occur in the context of learning Bayesian network structure. In *Proc. UAI-2015*, pages 832–841, 2015.