Learning Bayesian Network Structure[1]

David Barber

University College London

[1] These slides accompany the book *Bayesian Reasoning and Machine Learning*. The book and demos can be downloaded from www.cs.ucl.ac.uk/staff/D.Barber/brml. Feedback and corrections are also available on the site. Feel free to adapt these slides for your own purposes, but please include a link the above website.

Structure Learning

Lack of a priori independence knowledge

We assume we have a dataset, but don't know the independence assumptions we should make.

No missing data

For simplicity, we assume that the dataset is complete (there are no missing observations).

(almost) Complete ignorance

One could also consider the case of knowing some conditional independence assumptions, but not all. For simplicity, we assume that none are known.

Difficulty

Number of DAGs on $N > 1$ nodes is at least

$$\prod_{n=1}^{N-1} 2^n = 2^{N(N-1)/2}$$

and less than $N!2^{N(N-1)/2}$ (the $N!$ comes from the node ordering, but this will over-count). The exact number is bigger than 10^{18} for $N = 10$.

PC algorithm: Learning the skeleton

Removing links

- Start with a complete skeleton G
- Test all pairs $x \perp\!\!\!\perp y$? If an x and y pair are deemed independent then the link $x - y$ is removed from G.
- In the next round, for the remaining graph, one examines each $x - y$ link and conditions on a single neighbour z of x. If $x \perp\!\!\!\perp y | z$ then remove the link $x - y$.
- At each subsequent round the number of neighbours in the conditioning set is increased by one and all $x \perp\!\!\!\perp y | \mathcal{Z}$ are tested.

Storing conditions of independence

Whenever a (conditional) independence is found $x \perp\!\!\!\perp y | \mathcal{Z}$, then these conditioning variables are stored in a set $\mathcal{S}_{x,y} = \mathcal{Z}$ (this could be the empty set).

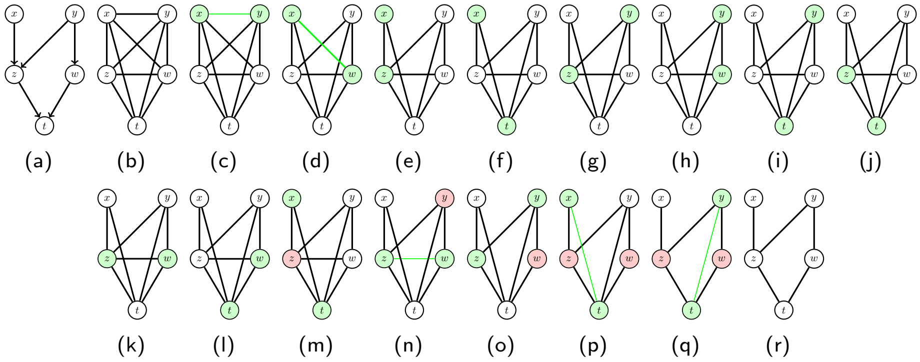
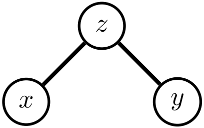
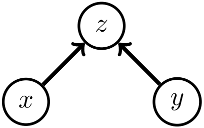


Figure : **(a)**: The BN from which data is assumed generated and against which conditional independence tests will be performed. **(b)**: The initial skeleton is fully connected. **(c-l)**: In the first round ($i = 0$) all the pairwise mutual informations $x \perp\!\!\!\perp y | \emptyset$ are checked, and the link between x and y removed if deemed independent (green line). **(m-o)**: $i = 1$. We now look at connected subsets on the three variables x, y, z of the remaining graph, removing the link $x - y$ if $x \perp\!\!\!\perp y | z$ is true. Not all steps are shown. **(p,q)**: $i = 2$. We now examine all $x \perp\!\!\!\perp y | \{a, b\}$. The algorithm terminates after this round (when i gets incremented to 3) since there are no nodes with 3 or more neighbours. **(r)**: Final skeleton. During this process the sets $S_{x,y} = \emptyset, S_{x,w} = \emptyset, S_{z,w} = y, S_{x,t} = \{z, w\}, S_{y,t} = \{z, w\}$ were found.

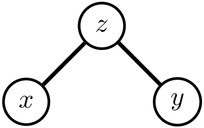
Skeleton Orienting



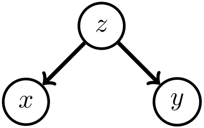
$x \perp\!\!\!\perp y | \emptyset \Rightarrow$



If x is (unconditionally) independent of y, it must be that z is a collider since otherwise marginalising over z would introduce a dependence between x and y.



$x \perp\!\!\!\perp y | z \Rightarrow$



If x is independent of y conditioned on z, z must not be a collider. Any other orientation is appropriate.

Skeleton Orienting

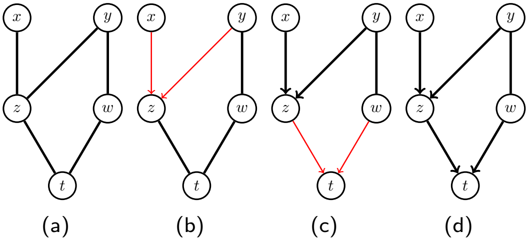


Figure : Skeleton orientation algorithm. **(a)**: The skeleton along with $S_{x,y} = \emptyset, S_{x,w} = \emptyset, S_{z,w} = y, S_{x,t} = \{z, w\}, S_{y,t} = \{z, w\}$. **(b)**: $z \notin S_{x,y}$, so form collider. **(c)**: $t \notin S_{z,w}$, so form collider. **(d)**: Final partially oriented DAG. The remaining edge may be oriented as desired, without violating the DAG condition.

Assessing Empirical Independence

Given a dataset of observations, how can we decide if two variables x and y are independent?

Mutual Information

We can form the empirical distributions $p(x)$ and $p(y)$ and $p(x, y)$. Define the mutual information

$$MI \equiv \mathrm{KL}(p(x, y)|p(x)p(y)) \geq 0$$

If $MI = 0$ then x and y are independent.

Problem

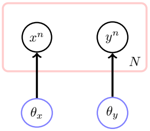
Since we formed $p(x)$ and $p(y)$ based on a set of observations, it's likely that, even for data sampled from a distribution for which $x \perp\!\!\!\perp y$, then the MI will not be zero. The classical approach is to use a hypothesis test, assuming that MI is chi-square distributed. This doesn't work well for small numbers of observations.

Bayesian Empirical Independence test

- A Bayesian approach to testing for independence can be made by comparing the likelihood of the data under the independence hypothesis, versus the likelihood under the dependent hypothesis.
- Which model has the higher likelihood will inform us about independence.
- Need to use a Bayesian approach to avoid overfitting (otherwise the dependence model will always win).

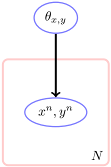
Independence

$$p(\mathcal{X}, \mathcal{Y}|\mathcal{H}_{indep})$$



Dependence

$$p(\mathcal{X}, \mathcal{Y}|\mathcal{H}_{dep})$$



Independence Hypothesis

$$p(x, y, \theta | \mathcal{H}_{indep}) = p(x|\theta_x) p(y|\theta_y) p(\theta_x) p(\theta_y)$$

Parameter prior

Convenient to use a Dirichlet prior $\mathrm{Dirichlet}\,(\theta|u)$ on the parameters θ, assuming also local as well as global parameter independence.

Model Likelihood

For a set of assumed i.i.d. data $(\mathcal{X}, \mathcal{Y}) = (x^n, y^n)\,, n = 1, \ldots, N$, the likelihood is then given by integrating over the parameters θ:

$$p(\mathcal{X}, \mathcal{Y} | \mathcal{H}_{indep}) = \int_\theta p(\theta | \mathcal{H}_{indep}) \prod_n p(x^n, y^n | \theta, \mathcal{H}_{indep})$$

Independence Hypothesis

Thanks to conjugacy, this is straightforward and gives the expression

$$p(\mathcal{X}, \mathcal{Y}|\mathcal{H}_{indep}) = \frac{Z(u_x + \sharp\,(x))}{Z(u_x)} \frac{Z(u_y + \sharp\,(y))}{Z(u_y)}$$

- u_x is a hyperparameter matrix of pseudo counts for each state of x.
- $\sharp\,(x)$ is the number of times state x appears in the data.
- $Z(v)$ is the normalisation constant of a Dirichlet distribution with vector parameter v.

Dependence Hypothesis

For the dependent hypothesis we have

$$p(x, y, \theta|\mathcal{H}_{dep}) = p(x, y|\theta_{x,y})p(\theta_{x,y})$$

Parameter Prior

Again we assumed a Dirichlet prior, so that the integration is straightforward.

Model Likelihood

The likelihood is then

$$p(\mathcal{X}, \mathcal{Y}|\mathcal{H}_{dep}) = \frac{Z(u_{x,y} + \sharp\,(x, y))}{Z(u_{x,y})}$$

Comparing Hypotheses

Assuming each hypothesis is equally likely, for a Bayes' Factor

$$\frac{p(\mathcal{X}, \mathcal{Y}|\mathcal{H}_{indep})}{p(\mathcal{X}, \mathcal{Y}|\mathcal{H}_{dep})}$$

greater than 1, we assume that independence holds, otherwise we assume the variables are conditionally dependent.

Bayes versus MI chi-square test

`demoCondindepEmp.m` suggests that the Bayesian hypothesis test tends to outperform the conditional mutual information approach, particularly in the small sample size case.

Conditional Independence

Straightforward to generalise to testing conditional independence (see the book).

Network Scoring

Local versus global methods

The PC algorithm is local in the sense that links are added based on the evidence for a link on the basis of local data. In a global method, a link is added based on how well that resulting distribution fits the data.

A probabilistic approach

- In a probabilistic context, given a model structure M, we wish to compute $p(M|\mathcal{D}) \propto p(\mathcal{D}|M)p(M)$.
- We have to first 'fit' each model with parameters θ, $p(\mathcal{V}|\theta, M)$ to the data $\mathcal{D}$. If we do this using Maximum Likelihood alone, with no constraints on θ, we will favour that model M with the most complex structure.
- This can be remedied by using the Bayesian technique

$$p(\mathcal{D}|M) = \int_\theta p(\mathcal{D}|\theta, M)p(\theta|M)$$

Dirichlet Parameter Priors

- For a discrete state network and Dirichlet priors, we have $p(\mathcal{D}|M)$ given explicitly by the Bayesian Dirichlet score.
- The score decomposes into terms involving each family of v:

$$p(\mathcal{D}|M) = \prod_v \prod_n p(v^n|\mathrm{pa}\left(v^n\right), M) = \prod_v \prod_j \frac{Z(\mathbf{u}'(v;j))}{Z(\mathbf{u}(v;j))}$$

  where the hyperparameter prior term is updated by the observed counts,

$$u'_i(v;j) \equiv u_i(v;j) + \sharp\left(v = i, \mathrm{pa}\left(v\right) = j\right)$$

- Searching over structures M is a computationally demanding. However, since the log-score decomposes into terms involving each family of v, we can easily compare two networks differing in a single arc. Search heuristics based on local addition/removal/reversal of links that increase the score are popular.

Dirichlet Hyperparameter setting

Flat prior

The simplest setting for the hyperparameters is set them all to unity.

BDeu setting

Another setting is the 'uninformative prior'

$$u_i(v;j) = \frac{\alpha}{\dim v \dim \mathrm{pa}\,(v)}$$

where $\dim x$ is the number of states of the variable(s) x for an 'equivalent sample size' parameter α.
Using Network scoring with this Dirichlet prior gives the 'BDeu' score function.

Network Scoring Example

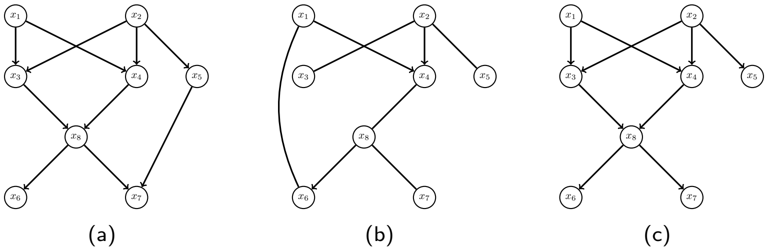
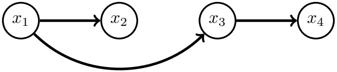


Figure : Learning the structure of a Bayesian network. **(a)**: The correct structure in which all variables are binary. The ancestral order is $x_2, x_1, x_5, x_4, x_3, x_8, x_7, x_6$. The dataset is formed from 800 samples from this network. **(b)**: The learned structure based on the PC algorithm using the Bayesian empirical conditional independence test. Undirected edges may be oriented arbitrarily (provided the graph remains acyclic). **(c)**: The learned structure based on the Bayes Dirichlet network scoring method (assuming we know the ancestral order and have maximally two parents).

Chow Liu Trees

A Chow-Liu Tree is a tree with at most one parent:



The variables may be indexed such that $1 \leq i \leq D$.

Fitting a Chow-Liu Tree

Task:

Given a multivariate distribution $p(\mathbf{x})$ we wish to approximate this with a Chow-Liu tree $q(\mathbf{x})$.

Parametrising the tree

We assume a labelling of the variables $1 \leq i \leq D$, for which the DAG single parent constraint means

$$q(x) = \prod_{i=1}^{D} q(x_i|x_{pa(i)}), \qquad pa(i) < i, \quad \text{or } pa(i) = \emptyset$$

where $pa(i)$ is the single parent index of node i. To find the best q in this constrained class, we may minimise

$$\mathrm{KL}(p|q) = \langle \log p(x) \rangle_{p(x)} - \sum_{i=1}^{D} \left\langle \log q(x_i|x_{pa(i)}) \right\rangle_{p(x_i, x_{pa(i)})}$$

Fitting a Chow-Liu Tree

By adding $\left\langle \log p(x_i|x_{pa(i)}) \right\rangle_{p(x_i, x_{pa(i)})}$ and ignoring constants

$$\mathrm{KL}(p|q) = -\sum_{i=1}^{D} \left\langle \left\langle \log q(x_i|x_{pa(i)}) \right\rangle_{p(x_i|x_{pa(i)})} - \left\langle \log p(x_i|x_{pa(i)}) \right\rangle_{p(x_i|x_{pa(i)})} \right\rangle_{p(x_{pa(i)})}$$

The optimal setting is therefore

$$q(x_i|x_{pa(i)}) = p(x_i|x_{pa(i)})$$

Using and $\log p(x_i|x_{pa(i)}) = \log p(x_i, x_{pa(i)}) - \log p(x_{pa(i)})$ we obtain

$$\mathrm{KL}(p|q) = \text{const.} - \sum_{i=1}^{D} \left\langle \log p(x_i, x_{pa(i)}) \right\rangle_{p(x_i, x_{pa(i)})} + \sum_{i=1}^{D} \left\langle \log p(x_{pa(i)}) \right\rangle_{p(x_{pa(i)})}$$

Fitting a Chow-Liu Tree

Still need to find the optimal parental structure $pa(i)$. If we add and subtract an entropy term we can write

$$\mathrm{KL}(p|q) = -\sum_{i=1}^{D} \left\langle \log p(x_i, x_{pa(i)}) \right\rangle_{p(x_i, x_{pa(i)})} + \sum_{i=1}^{D} \left\langle \log p(x_{pa(i)}) \right\rangle_{p(x_{pa(i)})} + \sum_{i=1}^{D} \langle \log p(x_i) \rangle_{p(x_i)} - \sum_{i=1}^{D} \langle \log p(x_i) \rangle_{p(x_i)} + \text{const.}$$

Using the definition of mutual information, we have

$$\mathrm{KL}(p|q) = -\sum_{i=1}^{D} \mathrm{MI}\big(x_i; x_{pa(i)}\big) - \sum_{i=1}^{D} \langle \log p(x_i) \rangle_{p(x_i)} + \text{const.}$$

Finding the parental indices $pa(i)$, is equivalent to maximising

$$\sum_{i=1}^{D} \mathrm{MI}\big(x_i; x_{pa(i)}\big)$$

under the constraint that $pa(i) \leq i$.

Fitting a Chow-Liu Tree

- Since we also need to choose the optimal initial labelling of the variables as well, the problem is equivalent to computing all the pairwise mutual informations

$$w_{ij} = \mathrm{MI}(x_i; x_j)$$

- We then find a maximal spanning tree for the graph with edge weights w.
- Once found, we need to identify a directed tree with at most one parent. This is achieved by choosing an arbitrary node and then orienting edges consistently away from this node.

Maximum likelihood Chow-Liu trees

If $p(x)$ is the empirical distribution

$$p(x) = \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}\left[x = x^n\right]$$

then

$$\mathrm{KL}(p|q) = \text{const.} - \frac{1}{N} \sum_{n} \log q(x^n)$$

Hence the approximation q that minimises the Kullback-Leibler divergence between the empirical distribution and p is equivalent to that which maximises the likelihood of the data. This means that if we use the mutual information found from the empirical distribution, with

$$p(x_i = \mathsf{a}, x_j = \mathsf{b}) \propto \sharp\left(x_i = \mathsf{a}, x_j = \mathsf{b}\right)$$

then the Chow-Liu tree produced corresponds to the Maximum Likelihood solution amongst all single-parent trees.