Data Mining
Bayesian Networks (2)

Ad Feelders

Universiteit Utrecht

Learning Bayesian Networks

1. Parameter learning: structure known/given; estimate the conditional probabilities from the data.
2. Structure learning: structure unknown; learn the networks structure and the conditional probabilities from the data.

Bayesian Network Factorisation

For a directed independence graph, the joint distribution factorises according to

$$\boxed{P(X) = \prod_{i=1}^{k} p(X_i \mid X_{pa(i)})}$$

So to specify the distribution we have to estimate the probabilities

$$p(X_i \mid X_{pa(i)}) \qquad\qquad i = 1, 2, \ldots, k$$

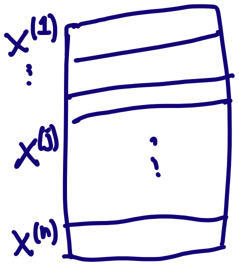
of the conditional distribution of each variable given its parents.

ML Estimation of BN

The joint probability for n independent observations is

$$P(X^{(1)}, \ldots, X^{(n)}) = \prod_{j=1}^{n} P(X^{(j)}) = \prod_{j=1}^{n} \prod_{i=1}^{k} p(X_i^{(j)} \mid X_{pa(i)}^{(j)}),$$

where $X^{(j)}$ denotes the j-th row in the data table.



The likelihood function is given by

$$L = \prod_{i=1}^{k} \prod_{x_i, x_{pa(i)}} p(x_i \mid x_{pa(i)})^{n(x_i, x_{pa(i)})}$$

where $n(x_i, x_{pa(i)})$ is a count of the number of rows with $X_i = x_i$, and $X_{pa(i)} = x_{pa(i)}$.

ML Estimation of BN

Taking the log of the likelihood function, we get

$$\mathcal{L} = \sum_{i=1}^{k} \sum_{x_i, x_{pa(i)}} n(x_i, x_{pa(i)}) \log p(x_i \mid x_{pa(i)})$$

- Maximize the log-likelihood function with respect to the unknown parameters $p(x_i \mid x_{pa(i)})$.
- This decomposes into a collection of independent multinomial estimation problems.
- Separate estimation problem for each X_i and configuration of $X_{pa(i)}$.

ML Estimation of BN

The maximum likelihood estimate of $p(x_i \mid x_{pa(i)})$ is given by:

$$\hat{p}(x_i \mid x_{pa(i)}) = \frac{n(x_i, x_{pa(i)})}{n(x_{pa(i)})},$$

where

- $n(x_i, x_{pa(i)})$ is the number of records in the data with $X_i = x_i$ and $X_{pa(i)} = x_{pa(i)}$, and
- $n(x_{pa(i)})$ is the number of records in the data with $X_{pa(i)} = x_{pa(i)}$.

In case X_i has no parents, this simplifies to

$$\hat{p}(x_i) = \frac{n(x_i)}{n}$$

The Log-Likelihood Score

Fill in the maximum likelihood estimates in the log-likelihood function to obtain the log-likelihood score:

$$\mathcal{L} = \sum_{i=1}^{k} \sum_{x_i, x_{pa(i)}} n(x_i, x_{pa(i)}) \log \frac{n(x_i, x_{pa(i)})}{n(x_{pa(i)})}$$

- The higher its value, the better the model fits the data.
- The saturated model (complete graph) always has the highest log-likelihood score.
- To avoid overfitting, we must penalize model complexity.

-Scoring Functions for Structure Learning

Scoring functions:

diff: size of panelty

$$\boxed{\text{AIC}(M) = \mathcal{L}^M - \dim(M)}$$

$$\boxed{\text{BIC}(M) = \mathcal{L}^M - \frac{\log n}{2}\dim(M)}$$

* higher panelty for extra parameter

when $n > 7$, $\frac{\log n}{2} > 1$

where $\mathcal{L}^M$ is the log-likelihood score of model M and $\dim(M)$ is the number of parameters of M.

BIC gives a higher penalty for model complexity ($n > 7$), so tends to lead to less complex models than AIC.

Note: earlier we defined $AIC(M) = 2(\mathcal{L}^{\text{sat}} - \mathcal{L}^M) + 2\dim(M)$. Dividing by -2 and ignoring the constant $\mathcal{L}^{\text{sat}}$ gives the current definition.

Structure Learning as an Optimization Problem

Given

- Training data.
- Scoring function (BIC or AIC).
- Space of possible models (all DAGs).

find the model that maximizes the score.

The number of labeled acyclic directed graphs on k nodes is given by the recurrence

$$a_k = \sum_{j=1}^{k} (-1)^{j-1} \binom{k}{j} 2^{j(k-j)} a_{k-j}$$

For example, $a_6 = 3,781,503$.

Heuristic Search

- Exhaustive search is not feasible.
- Local search: define which models are neighbors of a given model (typically: addition, removal, or reversal of an arc).
- Traverse search space looking for high-scoring models, e.g. by greedy hill-climbing.
- Most search algorithms do not require an a priori ordering of the variables!

Score Decomposes

The loglikelihood score

$$\mathcal{L} = \sum_{i=1}^{k} \sum_{x_i, x_{pa(i)}} n(x_i, x_{pa(i)}) \log \frac{n(x_i, x_{pa(i)})}{n(x_{pa(i)})}$$

must be computed many times for different models in structure learning.

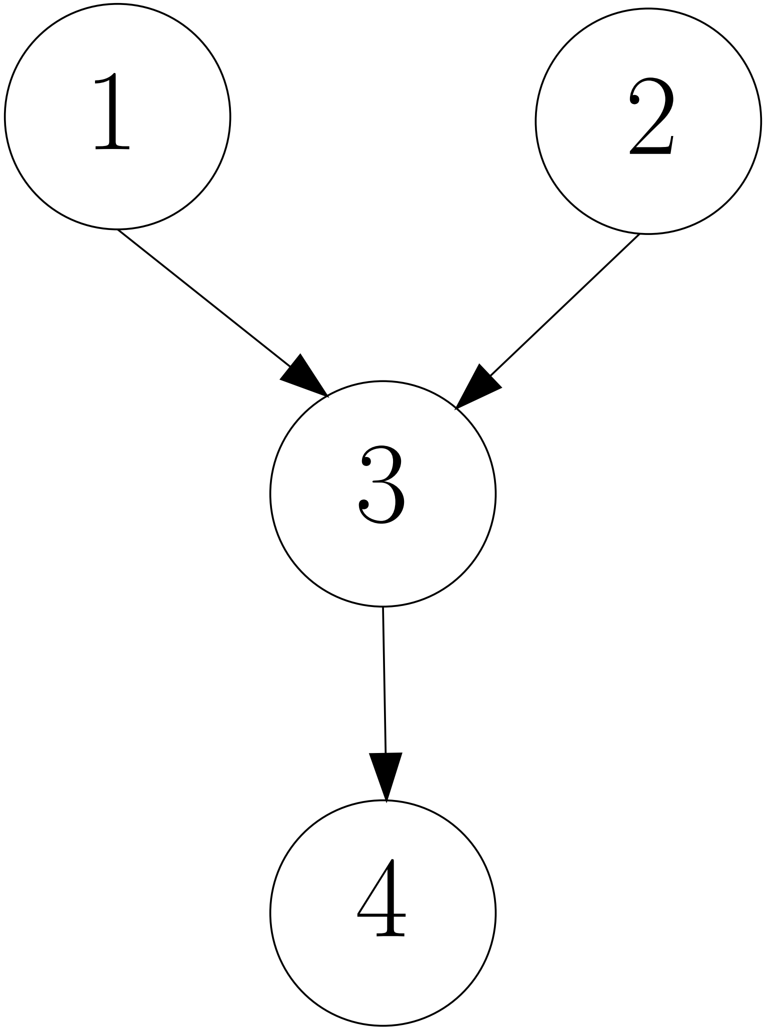
Luckily, it is a sum of terms, where each term contains the variables $\{i\} \cup \mathrm{pa}(i)$.

Hence, when making a change to the model, we only have to recompute the score for those variables whose parent set has changed!

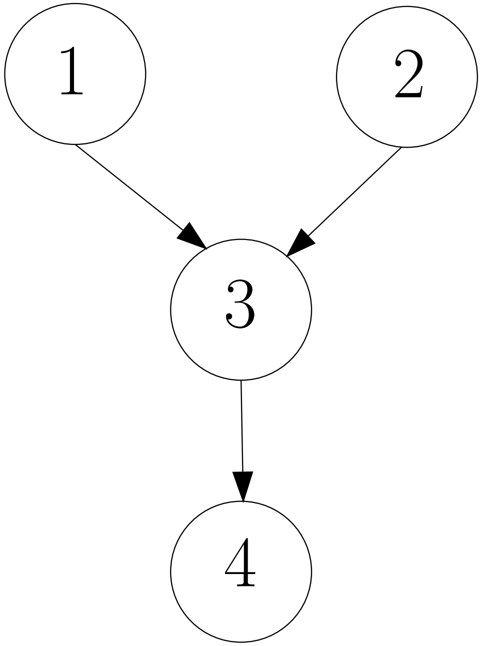
Example Data Set

obs	X_1	X_2	X_3	X_4
1	1	1	1	1
2	1	1	1	1
3	1	1	2	1
4	1	2	2	1
5	1	2	2	2
6	2	1	1	2
7	2	1	2	3
8	2	1	2	3
9	2	2	2	3
10	2	2	1	3

Score this model



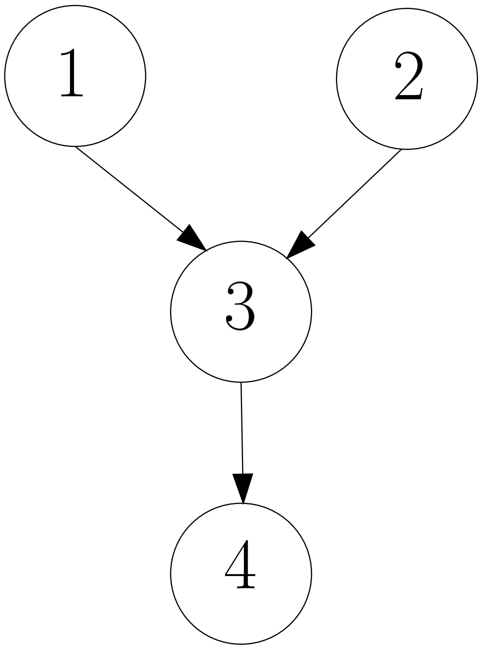
Relevant Data For Scoring Node 1



obs	X_1	X_2	X_3	X_4
1	1	1	1	1
2	1	1	1	1
3	1	1	2	1
4	1	2	2	1
5	1	2	2	2
6	2	1	1	2
7	2	1	2	3
8	2	1	2	3
9	2	2	2	3
10	2	2	1	3

Score node 1 $= 5 \log \frac{5}{10} + 5 \log \frac{5}{10}$

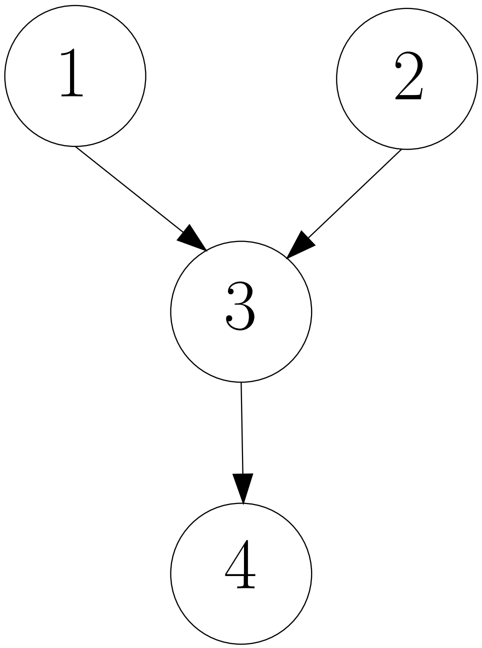
Relevant Data For Scoring Node 2



obs	X_1	X_2	X_3	X_4
1	1	1	1	1
2	1	1	1	1
3	1	1	2	1
4	1	2	2	1
5	1	2	2	2
6	2	1	1	2
7	2	1	2	3
8	2	1	2	3
9	2	2	2	3
10	2	2	1	3

Score node 2 $= 6 \log \frac{6}{10} + 4 \log \frac{4}{10}$

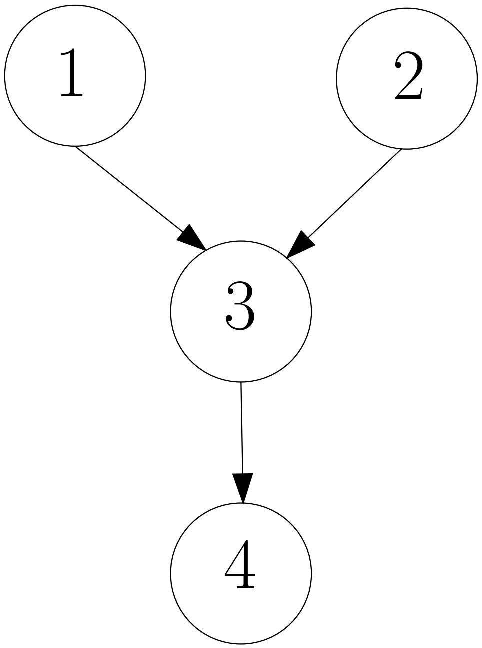
Relevant Data For Scoring Node 3



obs	X_1	X_2	X_3	X_4
1	1	1	1	1
2	1	1	1	1
3	1	1	2	1
4	1	2	2	1
5	1	2	2	2
6	2	1	1	2
7	2	1	2	3
8	2	1	2	3
9	2	2	2	3
10	2	2	1	3

Score node 3 $= 2 \log \frac{2}{3} + \log \frac{1}{3}$

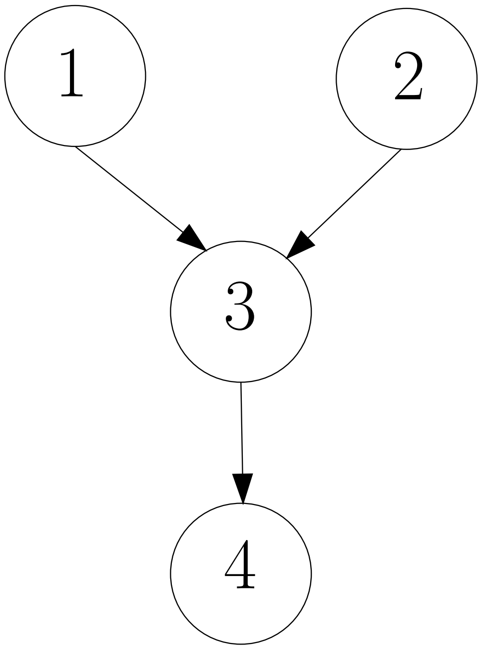
Relevant Data For Scoring Node 3



obs	X_1	X_2	X_3	X_4
1	1	1	1	1
2	1	1	1	1
3	1	1	2	1
4	1	2	2	1
5	1	2	2	2
6	2	1	1	2
7	2	1	2	3
8	2	1	2	3
9	2	2	2	3
10	2	2	1	3

Score node 3 $= 2\log\frac{2}{3} + \log\frac{1}{3} + 2\log\frac{2}{2}$

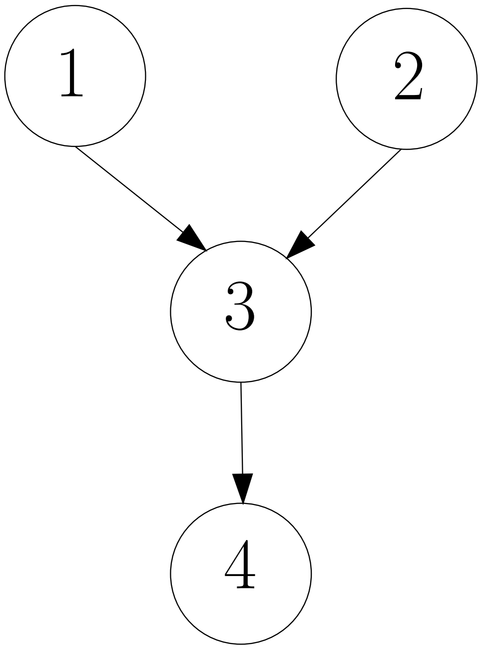
Relevant Data For Scoring Node 3



obs	X_1	X_2	X_3	X_4
1	1	1	1	1
2	1	1	1	1
3	1	1	2	1
4	1	2	2	1
5	1	2	2	2
6	2	1	1	2
7	2	1	2	3
8	2	1	2	3
9	2	2	2	3
10	2	2	1	3

Score node 3 $= 2\log\frac{2}{3} + \log\frac{1}{3} + 2\log\frac{2}{2} + \log\frac{1}{3} + 2\log\frac{2}{3}$

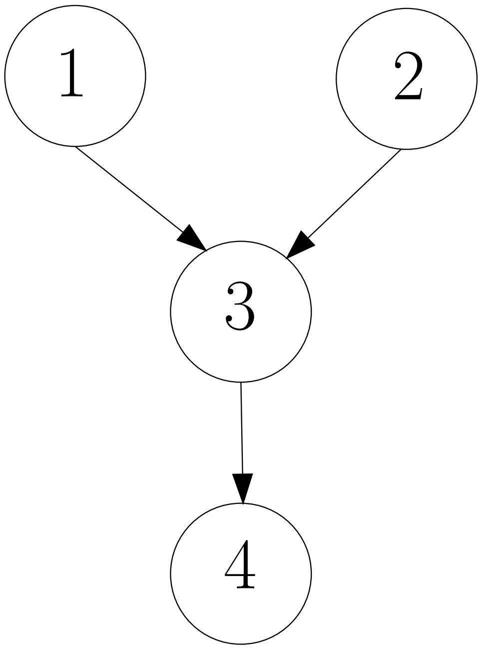
Relevant Data For Estimating Scoring Node 3



obs	X_1	X_2	X_3	X_4
1	1	1	1	1
2	1	1	1	1
3	1	1	2	1
4	1	2	2	1
5	1	2	2	2
6	2	1	1	2
7	2	1	2	3
8	2	1	2	3
9	2	2	2	3
10	2	2	1	3

Score node 3 $= 2\log\frac{2}{3} + \log\frac{1}{3} + 2\log\frac{2}{2} + \log\frac{1}{3} + 2\log\frac{2}{3} + \log\frac{1}{2} + \log\frac{1}{2}$

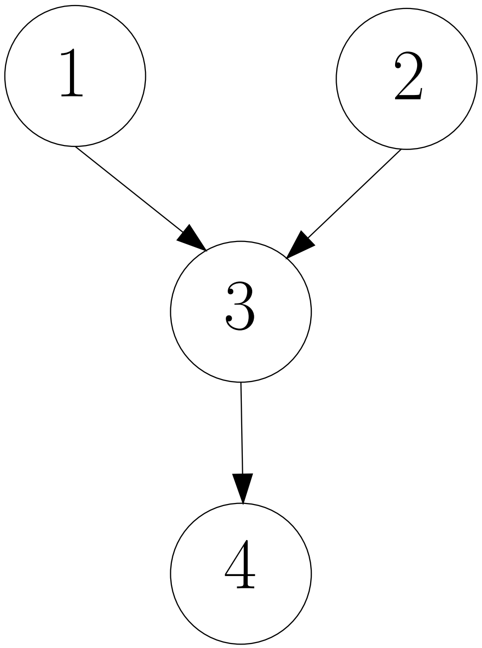
Relevant Data For Scoring Node 4



obs	X_1	X_2	X_3	X_4
1	1	1	1	1
2	1	1	1	1
3	1	1	2	1
4	1	2	2	1
5	1	2	2	2
6	2	1	1	2
7	2	1	2	3
8	2	1	2	3
9	2	2	2	3
10	2	2	1	3

Score node 4 $= 2\log\frac{2}{4} + \log\frac{1}{4} + \log\frac{1}{4}$

Relevant Data For Scoring Node 4



obs	X_1	X_2	X_3	X_4
1	1	1	1	1
2	1	1	1	1
3	1	1	2	1
4	1	2	2	1
5	1	2	2	2
6	2	1	1	2
7	2	1	2	3
8	2	1	2	3
9	2	2	2	3
10	2	2	1	3

Score node 4 $= 2\log\frac{2}{4} + \log\frac{1}{4} + \log\frac{1}{4} + 2\log\frac{2}{6} + \log\frac{1}{6} + 3\log\frac{3}{6}$

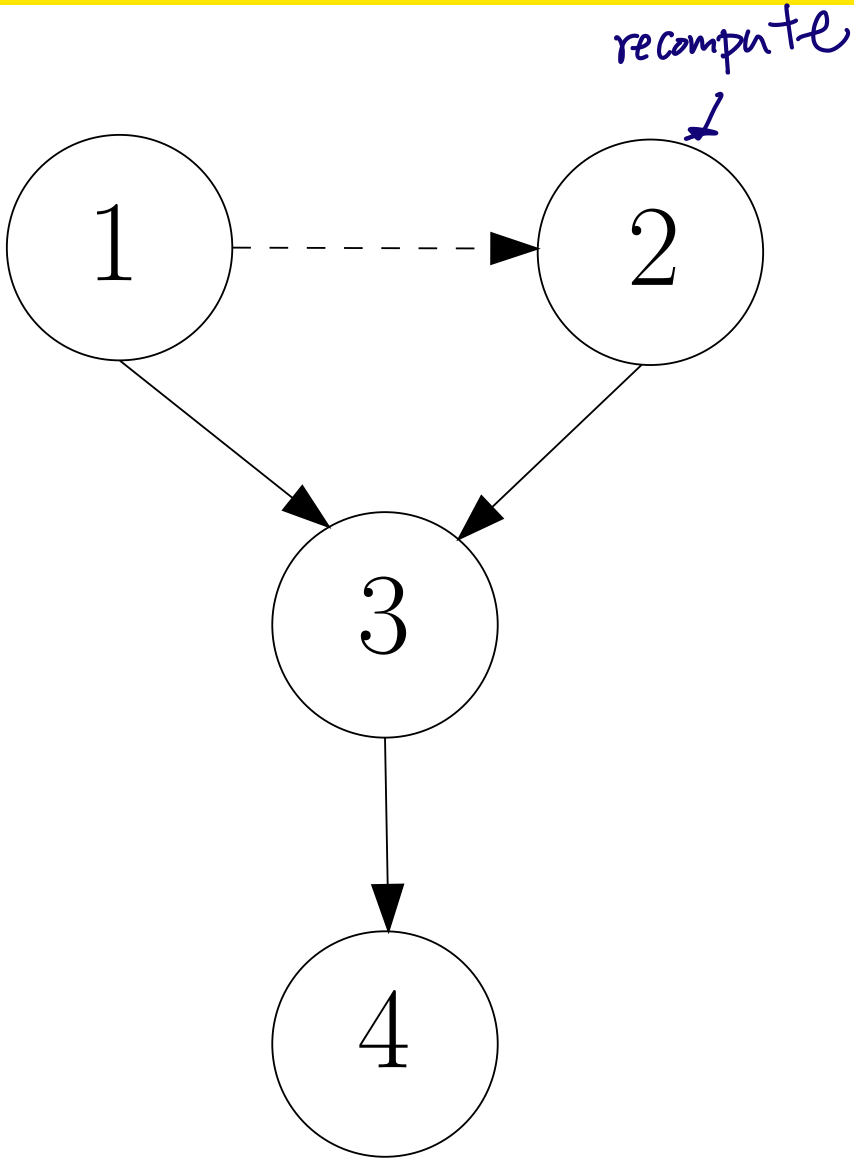
Total Score

Summing the loglikelihood score over all nodes, we get:

$$
\begin{aligned}
\mathcal{L} &= 5\log\frac{5}{10} + 5\log\frac{5}{10} && \text{(node 1)}\\
&+ 6\log\frac{6}{10} + 4\log\frac{4}{10} && \text{(node 2)}\\
&+ 2\log\frac{2}{3} + \log\frac{1}{3} + 2\log 1 + \log\frac{1}{3} + 2\log\frac{2}{3} + \log\frac{1}{2} + \log\frac{1}{2} && \text{(node 3)}\\
&+ 2\log\frac{2}{4} + \log\frac{1}{4} + \log\frac{1}{4} + 2\log\frac{2}{6} + \log\frac{1}{6} + 3\log\frac{3}{6} && \text{(node 4)}\\
&\approx \underline{-29.09}
\end{aligned}
$$

itself → no meaning, compare with others

Add an edge from X_1 to X_2

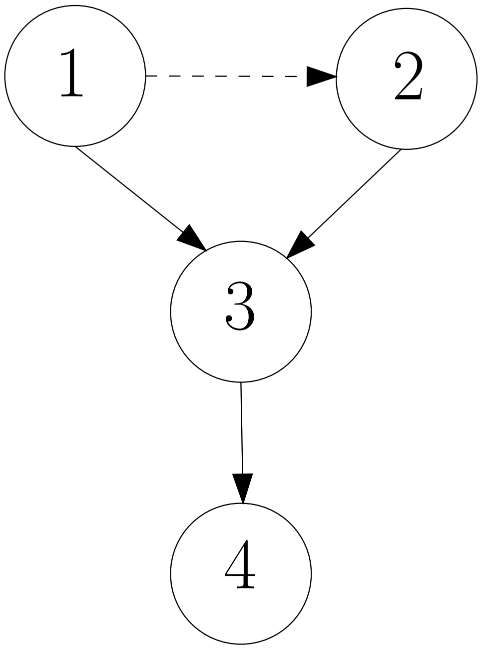


Score is Decomposable

$$
\begin{aligned}
\mathcal{L} &= 5\log\frac{5}{10} + 5\log\frac{5}{10} && \text{(node 1)}\\
&+ \boxed{6\log\tfrac{6}{10} + 4\log\tfrac{4}{10}} && \text{(node 2)}\\
&+ 2\log\frac{2}{3} + \log\frac{1}{3} + 2\log 1 + \log\frac{1}{3} + 2\log\frac{2}{3} + \log\frac{1}{2} + \log\frac{1}{2} && \text{(node 3)}\\
&+ 2\log\frac{2}{4} + \log\frac{1}{4} + \log\frac{1}{4} + 2\log\frac{2}{6} + \log\frac{1}{6} + 3\log\frac{3}{6} && \text{(node 4)}\\
&\approx -29.09
\end{aligned}
$$

- When we add an edge from X_1 to X_2, only the parent set of node 2 changes.
- Therefore, only the score of node 2 (the boxed part) has to be recomputed.

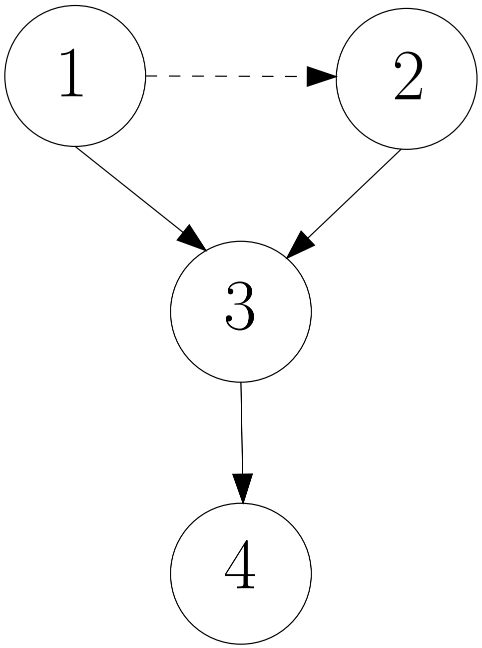
Relevant Data For Re-scoring Node 2



obs	X_1	X_2	X_3	X_4
1	1	1	1	1
2	1	1	1	1
3	1	1	2	1
4	1	2	2	1
5	1	2	2	2
6	2	1	1	2
7	2	1	2	3
8	2	1	2	3
9	2	2	2	3
10	2	2	1	3

New score node 2 $= 3 \log \frac{3}{5} + 2 \log \frac{2}{5}$

Relevant Data For Re-scoring Node 2



obs	X_1	X_2	X_3	X_4
1	1	1	1	1
2	1	1	1	1
3	1	1	2	1
4	1	2	2	1
5	1	2	2	2
6	2	1	1	2
7	2	1	2	3
8	2	1	2	3
9	2	2	2	3
10	2	2	1	3

New score node 2 $= 3\log\frac{3}{5} + 2\log\frac{2}{5} + 3\log\frac{3}{5} + 2\log\frac{2}{5}$

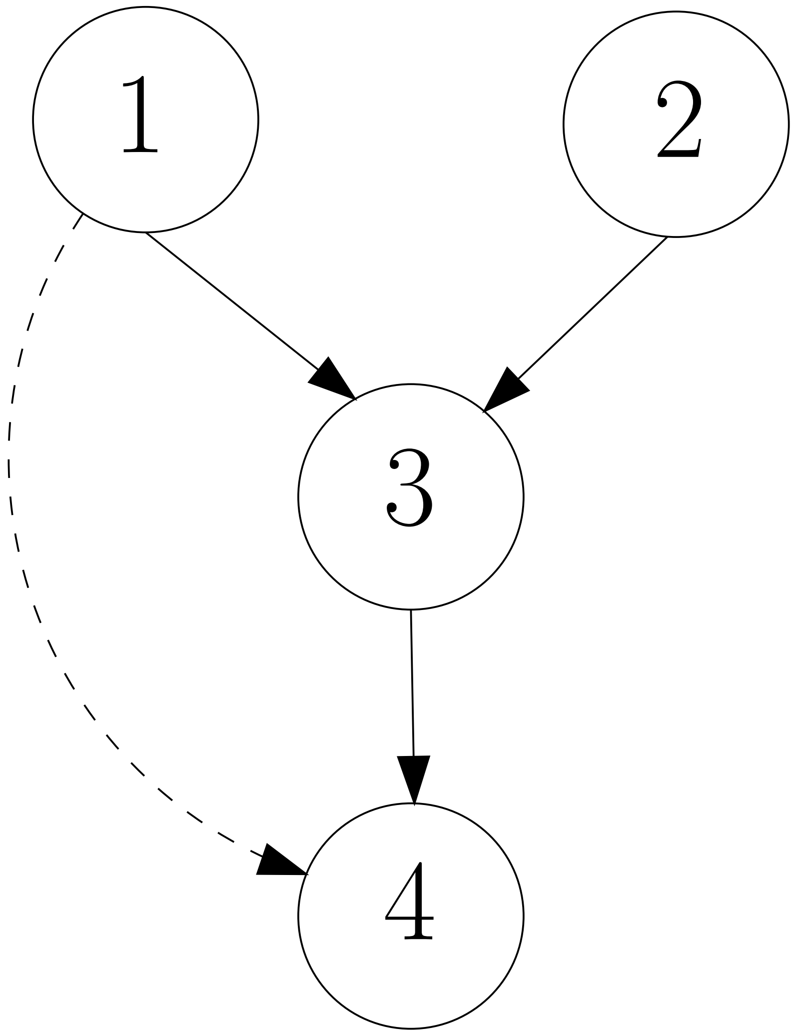
Score Decomposes

$$
\begin{aligned}
\mathcal{L} &= 5\log\frac{5}{10} + 5\log\frac{5}{10} && \text{(node 1)}\\
&+ \boxed{3\log\tfrac{3}{5} + 2\log\tfrac{2}{5} + 3\log\tfrac{3}{5} + 2\log\tfrac{2}{5}} && \text{(node 2)}\\
&+ 2\log\frac{2}{3} + \log\frac{1}{3} + 2\log 1 + \log\frac{1}{3} + 2\log\frac{2}{3} + \log\frac{1}{2} + \log\frac{1}{2} && \text{(node 3)}\\
&+ 2\log\frac{2}{4} + \log\frac{1}{4} + \log\frac{1}{4} + 2\log\frac{2}{6} + \log\frac{1}{6} + 3\log\frac{3}{6} && \text{(node 4)}\\
&\approx -29.09
\end{aligned}
$$

$\leftarrow$ same score $\because$ $X_1 \perp\!\!\!\perp X_2$ do not influence result

The boxed part is the new contribution of node 2 to the score.

Add an edge from X_1 to X_4

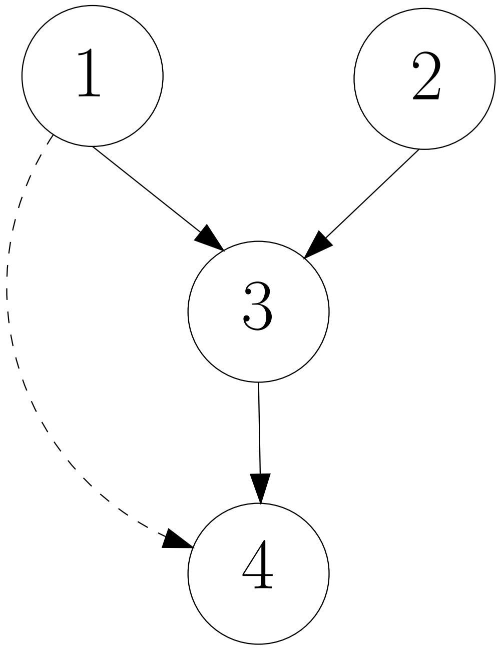


Score Decomposes

$$
\begin{aligned}
\mathcal{L} &= 5\log\frac{5}{10} + 5\log\frac{5}{10} && \text{(node 1)}\\
&+ 6\log\frac{6}{10} + 4\log\frac{4}{10} && \text{(node 2)}\\
&+ 2\log\frac{2}{3} + \log\frac{1}{3} + 2\log 1 + \log\frac{1}{3} + 2\log\frac{2}{3} + \log\frac{1}{2} + \log\frac{1}{2} && \text{(node 3)}\\
&+ \boxed{2\log\tfrac{2}{4} + \log\tfrac{1}{4} + \log\tfrac{1}{4} + 2\log\tfrac{2}{6} + \log\tfrac{1}{6} + 3\log\tfrac{3}{6}} && \text{(node 4)}\\
&\approx -29.09
\end{aligned}
$$

- When we add an edge from X_1 to X_4, only the parent set of node 4 changes.
- Therefore, only the score of node 4 (the boxed part) has to be recomputed.

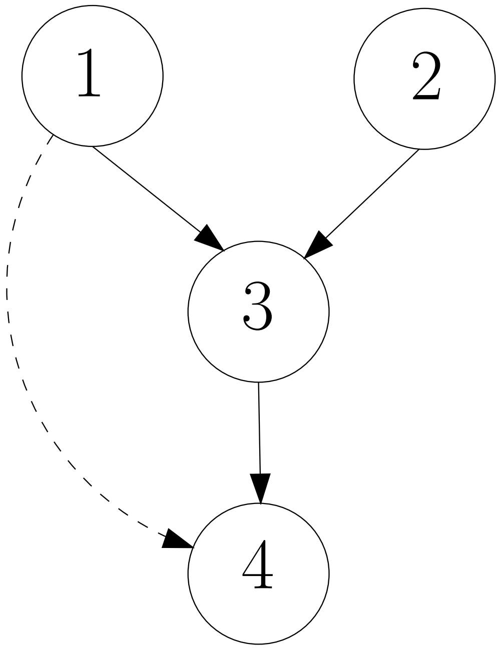
Relevant Data For Re-scoring Node 4



obs	X_1	X_2	X_3	X_4
1	1	1	1	1
2	1	1	1	1
3	1	1	2	1
4	1	2	2	1
5	1	2	2	2
6	2	1	1	2
7	2	1	2	3
8	2	1	2	3
9	2	2	2	3
10	2	2	1	3

New score node 4 $= 2 \log \frac{2}{2}$

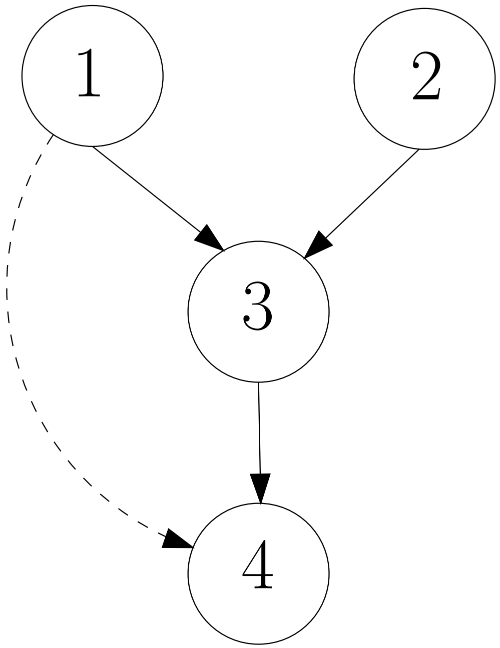
Relevant Data For Re-scoring Node 4



obs	X_1	X_2	X_3	X_4
1	1	1	1	1
2	1	1	1	1
3	1	1	2	1
4	1	2	2	1
5	1	2	2	2
6	2	1	1	2
7	2	1	2	3
8	2	1	2	3
9	2	2	2	3
10	2	2	1	3

New score node 4 $= 2\log\frac{2}{2} + 2\log\frac{2}{3} + \log\frac{1}{3}$

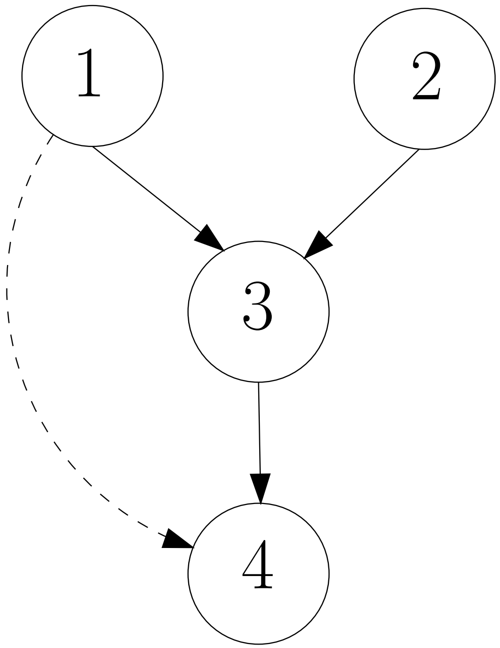
Relevant Data For Re-scoring Node 4



obs	X_1	X_2	X_3	X_4
1	1	1	1	1
2	1	1	1	1
3	1	1	2	1
4	1	2	2	1
5	1	2	2	2
6	2	1	1	2
7	2	1	2	3
8	2	1	2	3
9	2	2	2	3
10	2	2	1	3

New score node 4 $= 2\log\frac{2}{2} + 2\log\frac{2}{3} + \log\frac{1}{3} + \log\frac{1}{2} + \log\frac{1}{2}$

Relevant Data For Re-scoring Node 4



before

$2 \times 2 = 4$

add additional parent

$4 \times 2 = 8$

additional # of parameter: 4

obs	X_1	X_2	X_3	X_4
1	1	1	1	1
2	1	1	1	1
3	1	1	2	1
4	1	2	2	1
5	1	2	2	2
6	2	1	1	2
7	2	1	2	3
8	2	1	2	3
9	2	2	2	3
10	2	2	1	3

New score node 4 $= 2\log\frac{2}{2} + 2\log\frac{2}{3} + \log\frac{1}{3} + \log\frac{1}{2} + \log\frac{1}{2} + 3\log\frac{3}{3}$

Score Decomposes

$$
\begin{aligned}
\mathcal{L} &= 5\log\frac{5}{10} + 5\log\frac{5}{10} && \text{(node 1)}\\
&+ 6\log\frac{6}{10} + 4\log\frac{4}{10} && \text{(node 2)}\\
&+ 2\log\frac{2}{3} + \log\frac{1}{3} + 2\log 1 + \log\frac{1}{3} + 2\log\frac{2}{3} + \log\frac{1}{2} + \log\frac{1}{2} && \text{(node 3)}\\
&+ \boxed{2\log 1 + 2\log\tfrac{2}{3} + \log\tfrac{1}{3} + \log\tfrac{1}{2} + \log\tfrac{1}{2} + 3\log 1} && \text{(node 4)}\\
&\approx -22.16
\end{aligned}
$$

adding edge improve log likelihood score

The boxed part is the new contribution of node 4 to the score.

Counting Parameters

The number of parameters of a Bayesian network M is:

$$\boxed{\dim(M) = \sum_{i=1}^{k} \left\{ (d_i - 1) \prod_{j \in pa(i)} d_j \right\}}$$

where k is the number of variables in the network, and d_i is the number of possible values of X_i.

$\prod_{j \in pa(i)} d_j$ is the number of parent configurations for X_i.

If X_i has no parents, the number of parent configurations should be taken to be 1, so X_i contributes $d_i - 1$ parameters in that case.

A Simple Structure Learning Algorithm

Algorithm 1 BN Hill Climbing

```
G ← initial graph {E.g. "empty" graph}
max ← score(G)
repeat
    nb ← neighbours(G) {Don't create cycles!}
    for all G' ∈ nb do
        if score (G') > max then
            max ← score(G')
            G ← G'
        end if
    end for
until no change to G
return G
```

different directed edges : $k(k-1) = O(k^2)$

$O(n)$ work need to do with each neighbor

$O(nk^2)$

-Complexity: Naive

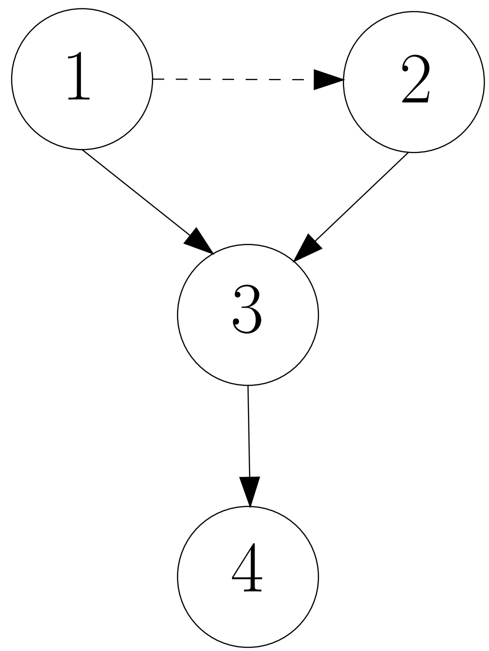
- A DAG with k nodes has $k(k-1)$ possible directed edges.
  - edge present: delete or reverse
  - edge absent: add

  So there are $O(k^2)$ neighbours that have to be scored.
- There are k components in the score and for each component we have to compute the counts which requires traversing the training data (n rows). So scoring a single neighbour takes $O(kn)$ time.
- The total complexity is $O(k^3 n)$ per search step.

Complexity: Exploiting Decomposability

- To score a neighbour we actually only have to score a single node (add, delete), or two nodes (reversal), since we only have to recompute the score for the nodes whose parent set changed by the local operation performed. So the complexity of scoring a neighbour is only $O(n)$ instead of $O(kn)$. We are down to $O(k^2n)$.
- optimization
  If we compute the *change* in score (Δ score) due to a local operation (add, delete, reverse), then we can reuse the Δ scores computed in previous iterations. We only need to recompute the Δ scores of operations that change the parent set of a single node, namely of the node whose parent set changed in the previous iteration. There are only $O(k)$ operations that change this parent set.
- Hence the total complexity per search step is down to $O(kn)$.

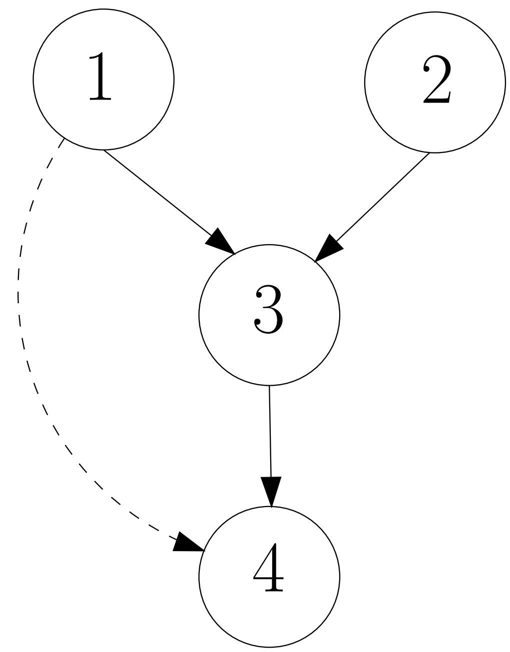
Delta Scores: Add an edge from X_1 to X_2



$$\Delta\mathrm{Score}(\mathrm{add}(X_1 \to X_2)) = \overbrace{(3\log\frac{3}{5} + 2\log\frac{2}{5} + 3\log\frac{3}{5} + 2\log\frac{2}{5})}^{\text{new score}}$$

$$-\underbrace{(6\log\frac{6}{10} + 4\log\frac{4}{10})}_{\text{old score}} = 0$$

Delta Scores: Add an edge from X_1 to X_4



$$
\begin{aligned}
\Delta\text{Score}(\text{add}(X_1 \rightarrow X_4)) &= (2\log 1 + 2\log\frac{2}{3} + \log\frac{1}{3} + \log\frac{1}{2} + \log\frac{1}{2} + 3\log 1) \\
&- (2\log\frac{2}{4} + \log\frac{1}{4} + \log\frac{1}{4} + 2\log\frac{2}{6} + \log\frac{1}{6} + 3\log\frac{3}{6}) \\
&\approx 6.93
\end{aligned}
$$

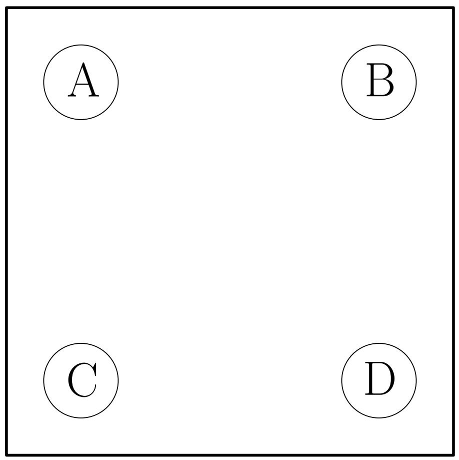
Delta Scores: Add an edge from X_1 to X_4

- Suppose we decide to add the arrow $X_1 \rightarrow X_4$.
- In the next iteration only the Δ scores of operations that change the parent set of X_4 have to be recomputed.
- For example, $\Delta\text{Score}(\text{add}(X_1 \rightarrow X_2))$ doesn't have to be recomputed because it is the same as in the previous iteration.

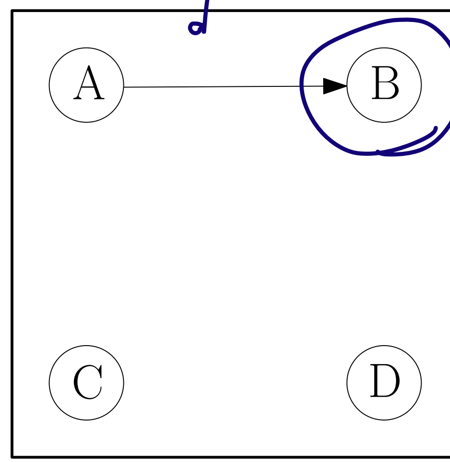
Delta Scores: Example

We start the search from the empty graph (mutual independence model).
Suppose we're only allowed to add edges.

best neighbor



biggest improvement



only node that parentset change

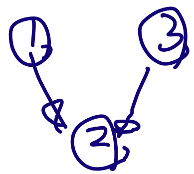
Iteration 1	
$\Delta(A \to B)$	$\Delta(B \to A)$
$\Delta(A \to C)$	$\Delta(B \to C)$
$\Delta(A \to D)$	$\Delta(B \to D)$
$\Delta(C \to A)$	$\Delta(D \to A)$
$\Delta(C \to B)$	$\Delta(D \to B)$
$\Delta(C \to D)$	$\Delta(D \to C)$

Iteration 2	
$\Delta(C \to B)$	$\Delta(D \to B)$

only recompute $\boxed{??} \to B$

No need to compute others, store here

Interpretation: warning!



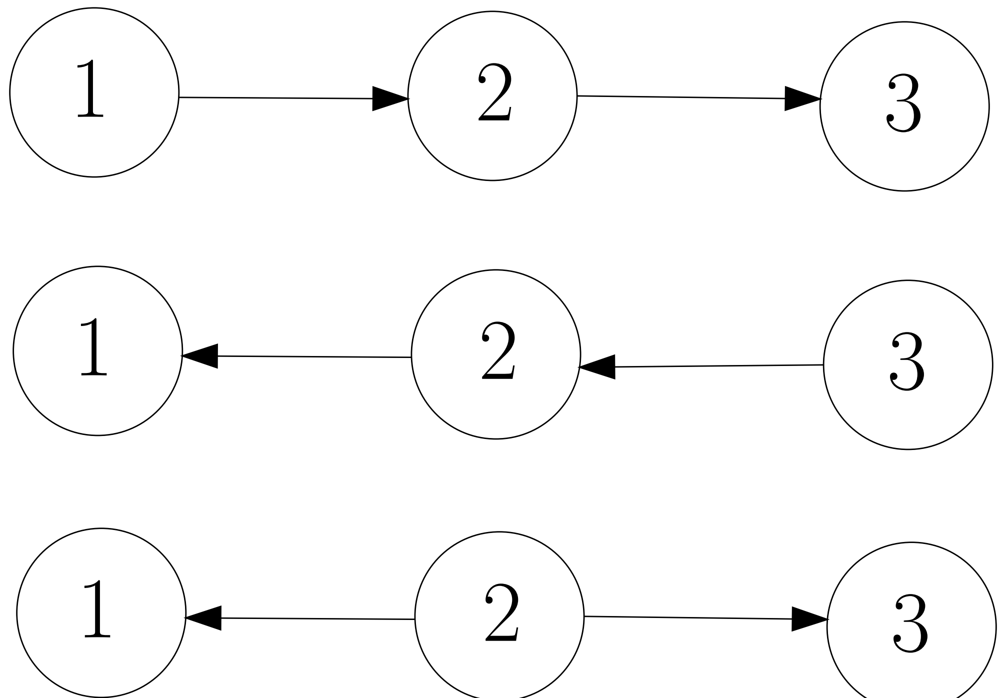
v-structure

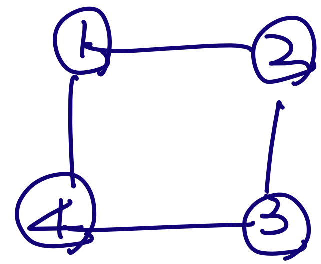
very different

$1 \perp\!\!\!\perp 3$

$1 \not\!\perp\!\!\!\perp 3 \mid 2$

diff. between direct ↔ undirect graph





no directed graph

No V-structure or cycle allowed in directed graph.

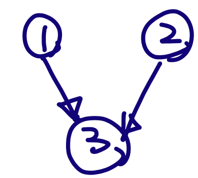
These models can not be distinguished from data alone.
They represent the same independencies!

AIC and *BIC* give equivalent networks the same score.

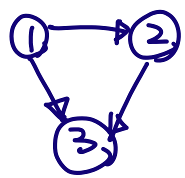
Markov Equivalence and Essential Graph

Two DAGs are Markov equivalent if and only if

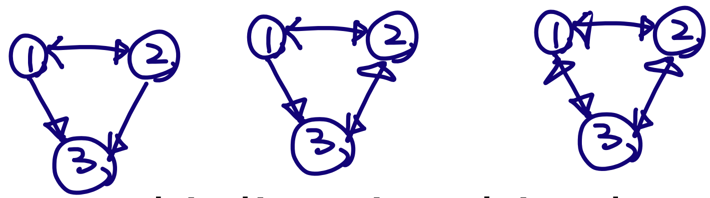
1. they have the same skeleton (same undirected graph when you drop the directions of all edges), and
2. they have the same immoralities (v-structures).



x break/create v-structures



Essential Graph:



For a given DAG, an edge becomes bi-directional in the essential graph if there is an equivalent DAG in which the direction of the edge is reversed.

Example Analysis

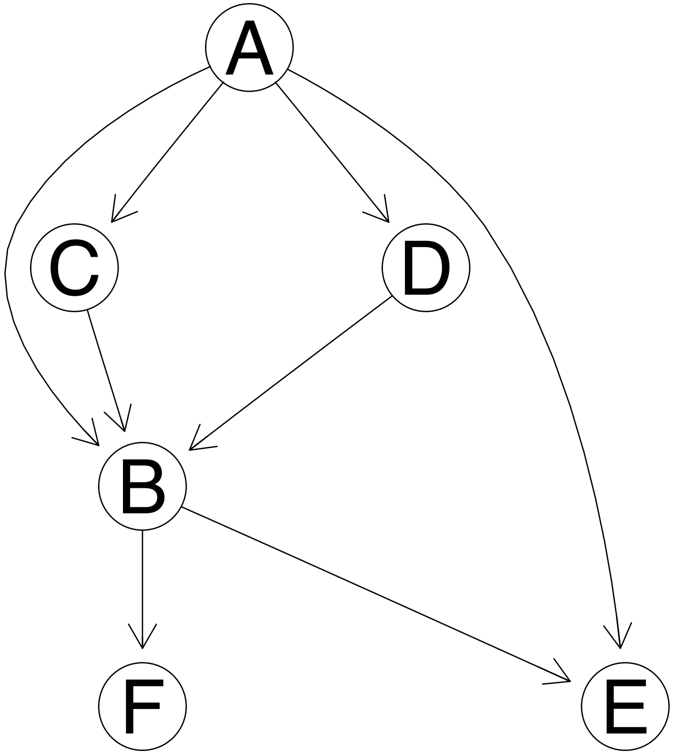
We analyze a data set concerning risk factors for coronary heart disease. For a sample of 1841 car-workers, the following information was recorded

Variable	Description
A	Does the person smoke?
B	Is the person's work strenuous mentally?
C	Is the person's work strenuous physically?
D	Systolic blood pressure $<$ 140mm?
E	Ratio of beta to alfa lipoproteins $<$ 3?
F	Is there a family history of coronary heart disease?

Example Analysis

For learning Bayesian networks, we use the *bnlearn* package in R.
Hill-climbing with the BIC score function (default), and starting from the empty graph (mutual independence model):

```
> coronary.hc <- hc(coronary)
> plot(coronary.hc)
```



The Search Process

```
> coronary.hc <- hc(coronary, debug=T)
----------------------------------------------------------------
* starting from the following network:
  model:
   [A][B][C][D][E][F]
```

← empty graph

but no B→A equal model

```
* current score: -7061.714
* caching score delta for arc A -> B (17.531166).
* caching score delta for arc A -> C (9.981480).
* caching score delta for arc A -> D (1.757126).
* caching score delta for arc A -> E (4.941129).
* caching score delta for arc A -> F (-3.224701).
* caching score delta for arc B -> C (264.272873).
* caching score delta for arc B -> D (2.313656).
* caching score delta for arc B -> E (21.030213).
* caching score delta for arc B -> F (2.303571).
* caching score delta for arc C -> D (-3.711314).
* caching score delta for arc C -> E (4.577177).
* caching score delta for arc C -> F (-3.673929).
* caching score delta for arc D -> E (2.645583).
* caching score delta for arc D -> F (-3.197133).
* caching score delta for arc E -> F (-2.257169).
```

← largest

The Search Process

- The initial model (the mutual independence model `[A][B][C][D][E][F]`) has a BIC score of `-7061.714`.
- The output gives the *change* in score between the current model and its neighbors.
- Why is the score of only 15 of the 30 neighbors computed? (e.g. `A -> B`, but not `B -> A`)?

The Search Process

- The initial model (the mutual independence model `[A][B][C][D][E][F]`) has a BIC score of `-7061.714`.
- The output gives the *change* in score between the current model and its neighbors.
- Why is the score of only 15 of the 30 neighbors computed? (e.g. `A -> B`, but not `B -> A`)?
- `A -> B` and `B -> A` are Markov equivalent, and therefore have the same score.
- Adding `B -> C` results in the largest increase in score so we move to that neighbor.

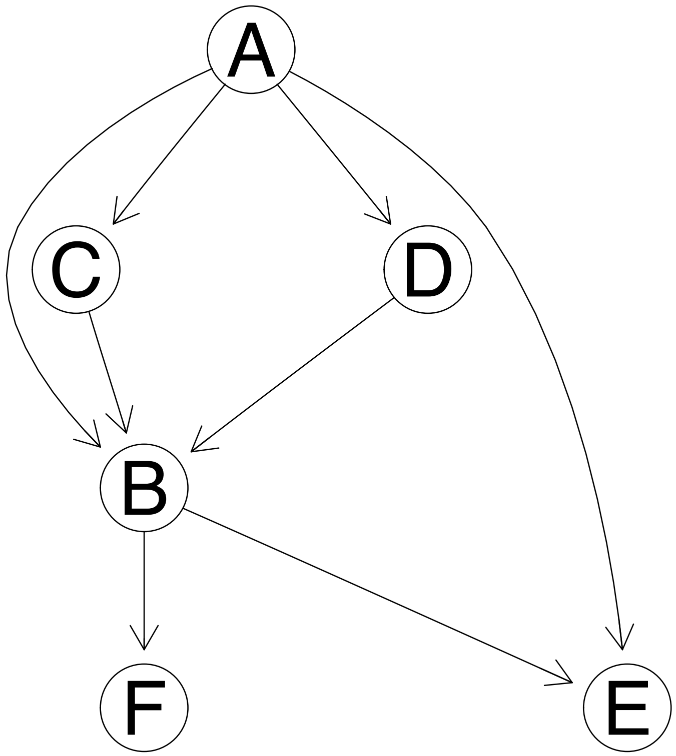
The Search Process

```
* best operation was: adding B -> C .
* current network is :

  model:
   [A][B][D][E][F][C|B]

* current score: -6797.441
* caching score delta for arc A -> C (9.975823).
* caching score delta for arc B -> C (-264.272873).
* caching score delta for arc D -> C (-1.472731).
* caching score delta for arc E -> C (-6.587044).
* caching score delta for arc F -> C (-6.059896).
```

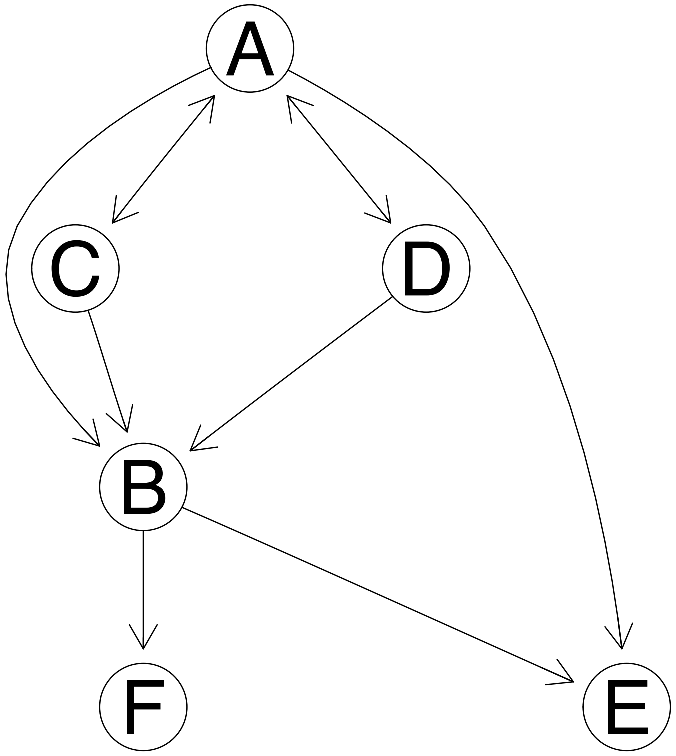
The Search Process

- We don't have to recompute the change in score caused by, for example, adding `A -> B`, because the parent set of `B` is the same as in the previous iteration.
- Therefore, adding `A -> B` now will cause the same score change as in the previous iteration.
- Only the parent set of `C` has changed in the previous iteration, so we just have to recompute the change in score for operations that change the parent set of `C`.
- Adding `B -> E` results in the largest increase in score so we move to that neighbor.
- The current model becomes: `[A][B][D][F][C|B][E|B]`.

Final Model and its Essential Graph



Final Model

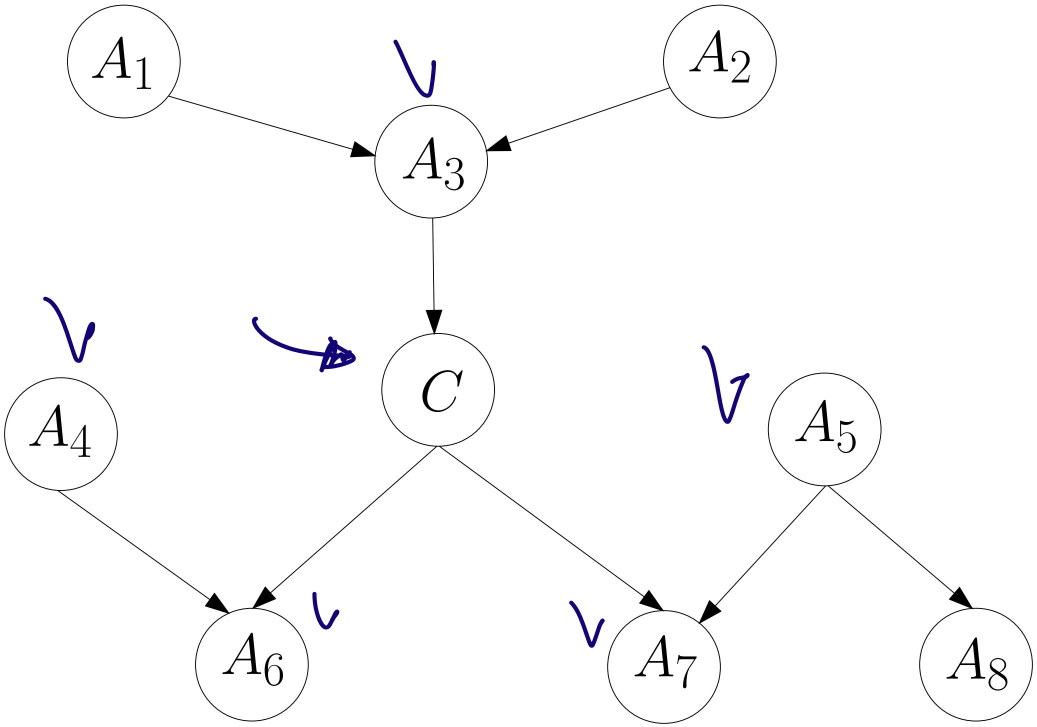


Essential Graph

Example of Model Use (prediction) B

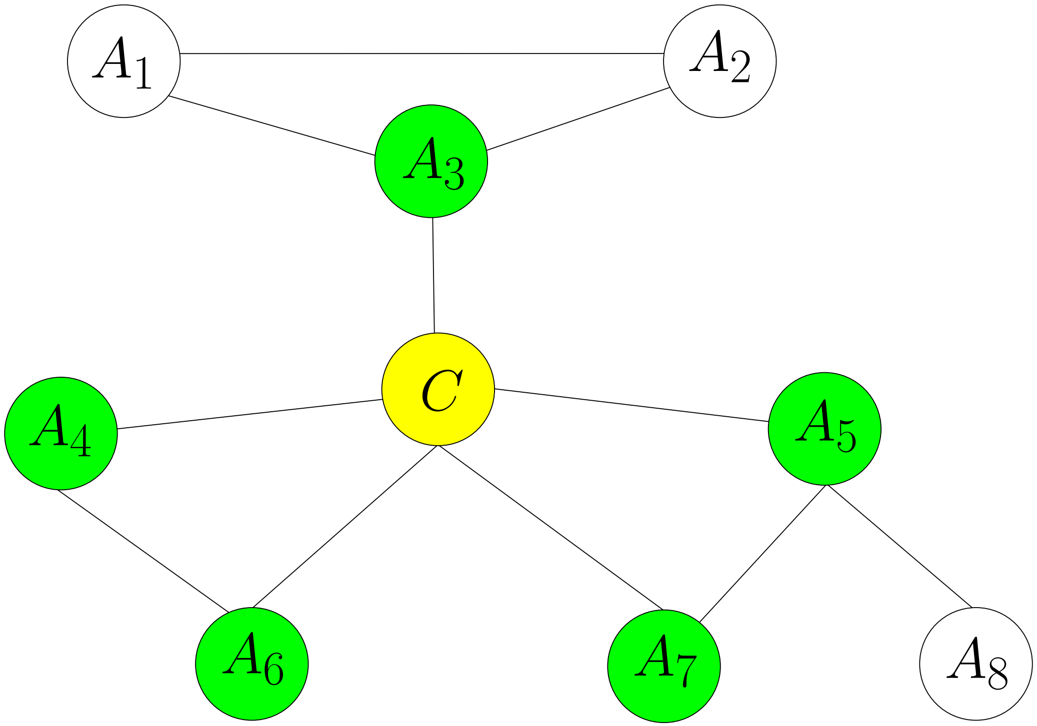
```
# estimate parameters for selected model structure
> coronary.hc.fit <- bn.fit(coronary.hc,coronary.dat,"mle")
# predict B from remaining variables
> coronary.hc.pred <- predict(coronary.hc.fit,node="B",
                               data=coronary.dat)
# make confusion matrix
> table(coronary.dat$B,coronary.hc.pred)
     coronary.hc.pred
       no yes
  no  944 186
  yes 208 503
> (944+503)/1841
[1] 0.7859859
> (944+186)/1841
[1] 0.6137968
```

Bayesian Networks as Classifiers



Markov Blanket: Parents, Children and Parents of Children.

Markov Blanket of C: Moral Graph



Connect all to C

Markov Blanket: Parents, Children and Parents of Children.

Local Markov property: $C \perp\!\!\!\perp \text{rest} \mid \text{boundary}(C)$

Right Heart Catheterization Data: Variable Description

1. cat1: primary disease category (9 different values)
2. death: did the patient die within 180 days after admission to ICU?
3. swang1: was right heart catheterization (Swan-Ganz catheter) performed within first 24 hours?
4. gender: male/female
5. race: black/white/other
6. ninsclas: type of medical insurance of patient (six different values)
7. income: income of patient, divided into 4 categories
8. ca: cancer status (yes/no/metastatic)
9. age: age of patient divided into 5 categories
10. meanbp1: mean blood pressure of patient divided into 2 categories

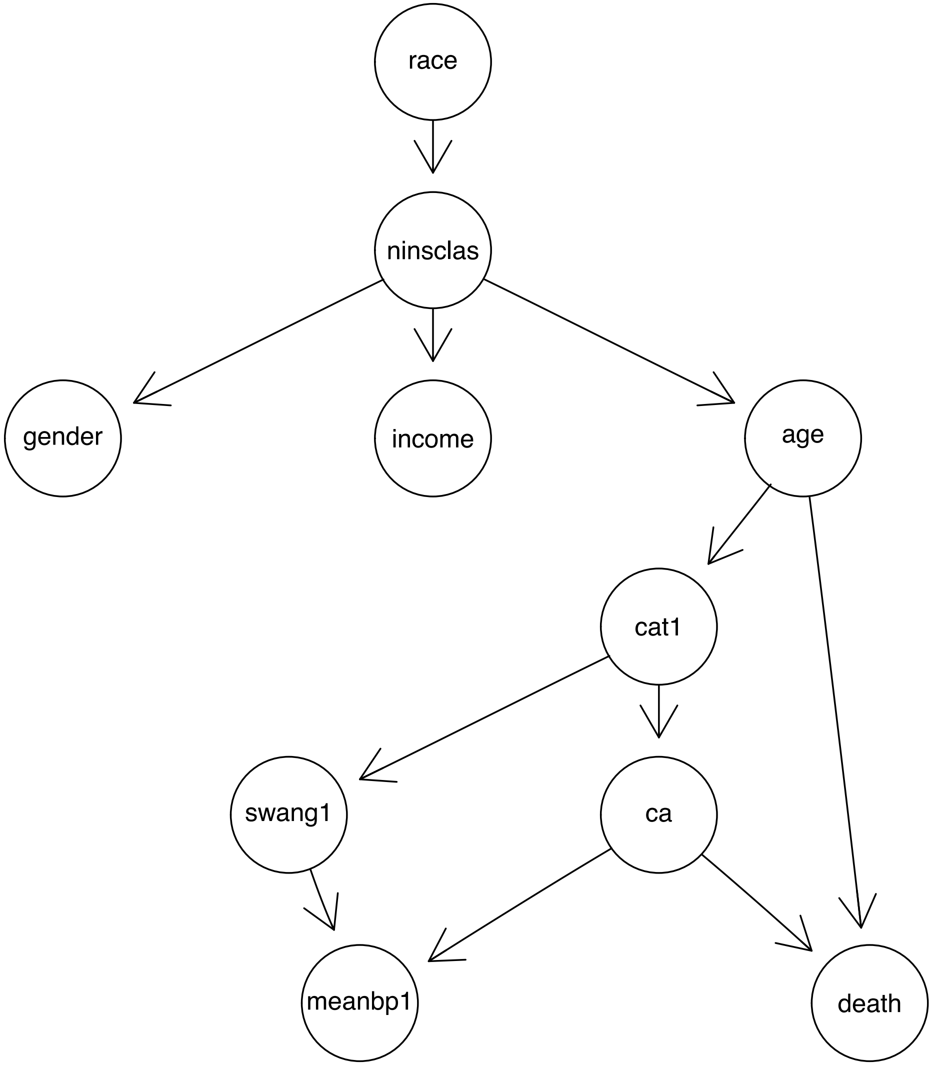
Right Heart Catheterization Data: Descriptive Statistics

```
> summary(rhc.dat)
                 cat1        death          swang1          gender          race
 ARF               :2490   No :2013   No RHC:3551   Female:2543   black: 920
 MOSF w/Sepsis     :1227   Yes:3722   RHC   :2184   Male  :3192   other: 355
 COPD              : 457                                          white:4460
 CHF               : 456
 Coma              : 436
 MOSF w/Malignancy: 399
 (Other)           : 270

                   ninsclas           income               ca              age
 Medicaid             : 647   $11-$25k   :1165   Metastatic: 384   (50,60] : 917
 Medicare             :1458   $25-$50k   : 893   No        :4379   (60,70] :1390
 Medicare & Medicaid: 374   > $50k      : 451   Yes       : 972   (70,80] :1337
 No insurance         : 322   Under $11k:3226                      (80,102]: 667
 Private              :1698                                        [18,50] :1424
 Private & Medicare :1236

     meanbp1
(85,259]:1975
[0,85]  :3760
```

Learning the Graph Structure

```
# load bayesian network library
> library(bnlearn)

# load library for graph vizualization
> library(Rgraphviz)

# use hill climbing with BIC scoring
# starting from empty graph
> rhc.bn <- hc(rhc.dat)

# plot the model structure
> plot(as(amat(rhc.bn),"graphNEL"))
```

Graph Structure for Right Heart Catheterization Data



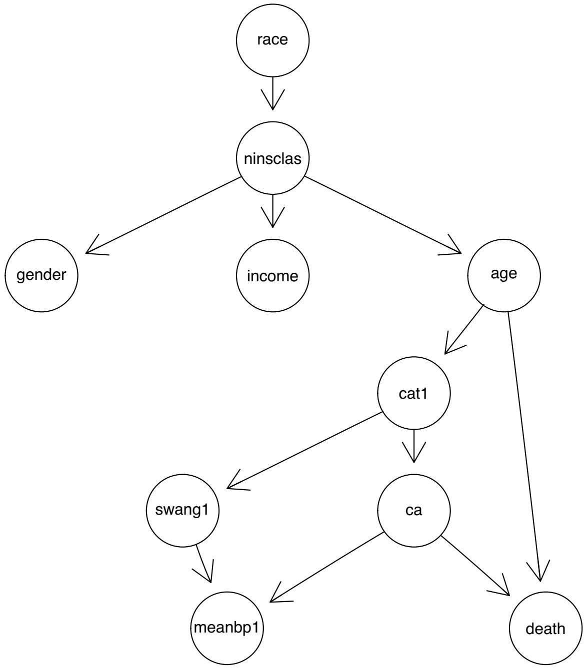
Performing Inference

```
# estimate the network parameters
> rhc.bn.fit <- bn.fit(rhc.bn,data=rhc.dat)

# perform sampling based inference
# probability of death for metastatic cancer and
# mean blood pressure > 85
> cpquery(rhc.bn.fit,event=death=="Yes",evidence=
    ca=="Metastatic" & meanbp1=="(85,259]",n=100000)
[1] 0.9033019

# probability of death for no cancer and
# mean blood pressure > 85
> cpquery(rhc.bn.fit,event=death=="Yes",evidence=
    ca=="No" & meanbp1=="(85,259]",n=100000)
[1] 0.6020206
```

Combining Data and Prior Knowledge

An expert studies the graph and argues that the edge from `swang1` to `meanbp1` is in the wrong direction, since the blood pressure influences the decision to apply right heart catheterization, not the other way around.

Can we turn the edge around without changing the "meaning" of the network, i.e. without changing the conditional independencies expressed by the graph?



Combining Data and Prior Knowledge

Common sense suggest that the variables can be divided into a number of ordered blocks, where arrows are not allowed to point from a variable in a higher numbered block to a variable in a lower numbered block.

As an example, consider the following block structure:

1. `race, gender`
2. `age, income`
3. `ninsclass`
4. `cat1, ca, meanbp1`
5. `swang1`
6. `death`

We can use the `blacklist` parameter to avoid edges pointing from higher numbered blocks to lower numbered blocks.

Learning with a Blacklist

```
# learn structure with blacklist
> rhc.bn.ord <- hc(rhc.dat,blacklist=blackL)

# has the score become much worse?
> score(rhc.bn.ord,rhc.dat)
[1] -54059.03
> score(rhc.bn,rhc.dat)
[1] -53749.15

# has inferences changed much?
> rhc.bn.ord.fit <- bn.fit(rhc.bn.ord,data=rhc.dat)

> cpquery(rhc.bn.ord.fit,event=death=="Yes",evidence=
    ca=="Metastatic" & meanbp1=="(85,259]",n=100000)
[1] 0.9039467
> cpquery(rhc.bn.ord.fit,event=death=="Yes",evidence=
    ca=="No" & meanbp1=="(85,259]",n=100000)
[1] 0.610249
```

The Blacklist

The blacklist simply enumerates all the forbidden edges:

```
> blackL
          X1       X2
1       cat1   gender
2       cat1     race
3       cat1 ninsclas
4       cat1   income
5       cat1      age
6      death     cat1
7      death   swang1
etc.
```

Graph Structure Learned with Blacklist