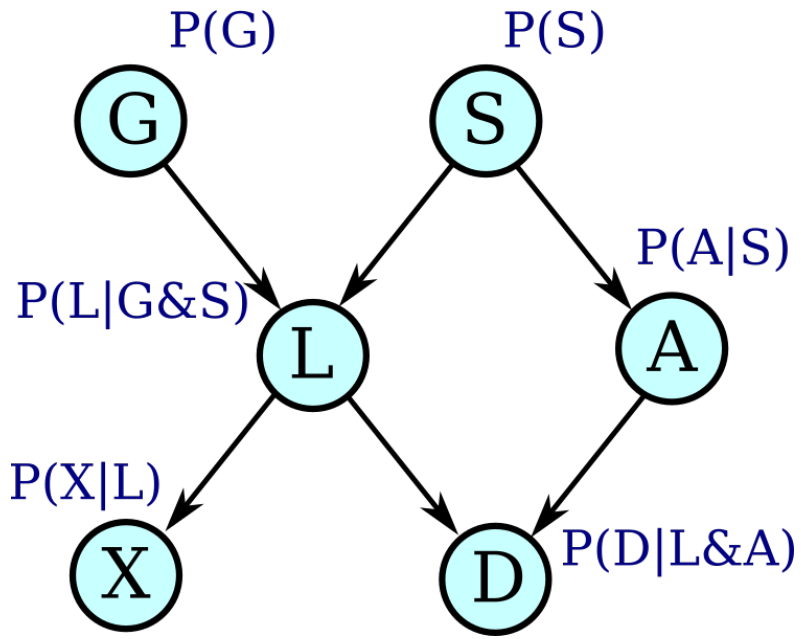
Lecture 3

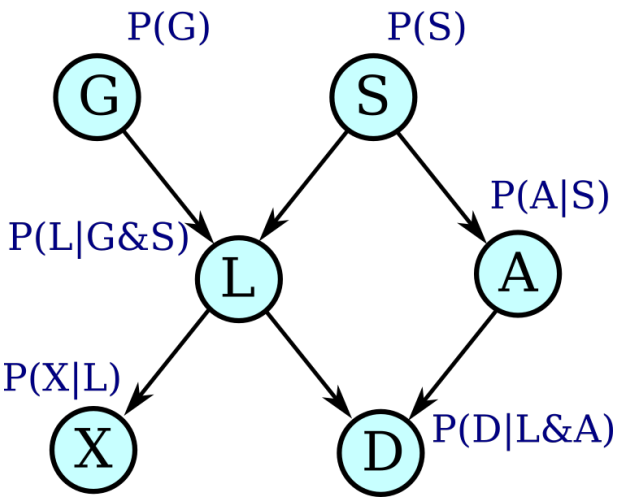
Cause and Independence

The story so far

We have developed a fast and effective algorithm for probability propagation in belief networks, but it only works for singly connected networks. Now we have discovered from our distinguished medical expert that our diagnostic network has a loop in in.



So what is the problem with loops?



When we calculate the joint π message from $L\&A$ to D we assume that L and A are independent.

Unfortunately they are not independent, so the computation of $P'(D)$ will be wrong.

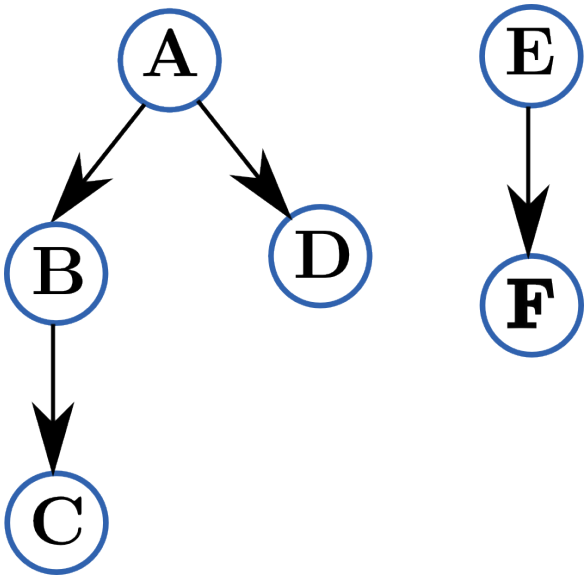
So what does it mean to be independent?

Independence

Independence is a clearly defined notion with well defined measures.

Given two variables X and Y, if changing X does not change Y and *vice versa* then X and Y are independent.

In a network if there is no path between two variables they are independent.



Nodes E and F are independent of nodes A, B, C and D.

Arcs and Independence

It is possible for two variables to be connected by a path in a network and still be independent. This is because a conditional probability matrix can express no dependency. Consider

$$P(B|A) = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

For any λ evidence on B say $[b_1, b_2]$ we have that:

$$\lambda_B(A) = [b_1, b_2] \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix} = [(b_1 + b_2)/2, (b_1 + b_2)/2]$$

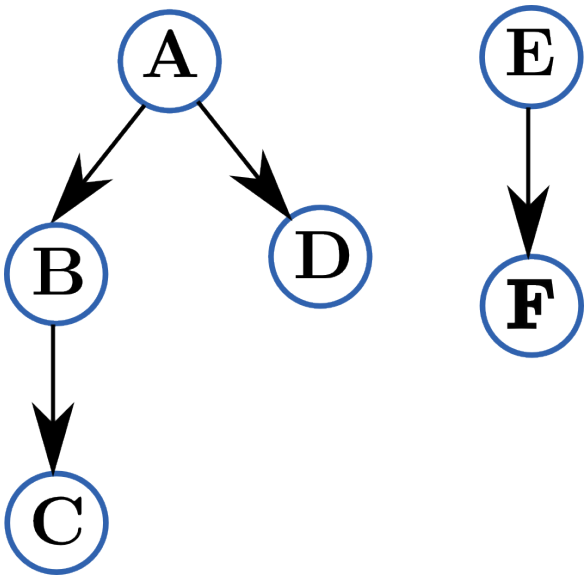
In other words the λ message contains no evidence and therefore A does not change when B changes.

Arcs and Independence

- So in theory, the absence of an arc in a network is more significant than the presence of an arc.
- However in practice we should avoid having arcs expressing very low dependency in networks.
- The maximum weighted spanning tree algorithm (MWST) can be adapted to help with this problem. Instead of continuing to include arcs until we create a connected graph we can terminate the process when the dependency becomes too low to be significant.

Dependency Separation (d-separation)

Any two variables X and Y which are not directly connected by an arc are conditionally independent if there is a set of nodes Z such that knowledge of Z makes X and Y independent. The set Z is said to d-separate X and Y.



Both nodes A and B d-separate C and D.

Cut-set conditioning

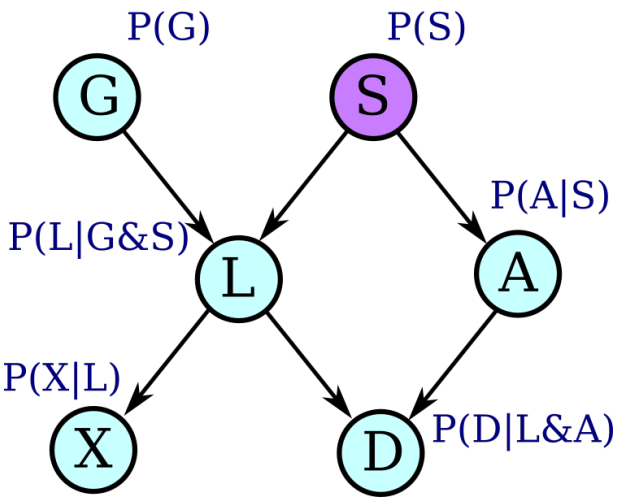
D-Separation gives us a possible way forward to achieving accurate belief propagation in the case of loops.

> If we can identify a cut-set of nodes that D-separate a loopy graph into singly connected subgraphs, and we instantiate those nodes we can propagate probabilities accurately.

If we don't have data for the cut-set nodes we can calculate probabilities for each joint state of the cut set and combine them with whatever prior information we have

Cut-set conditioning

So now we are back in business.



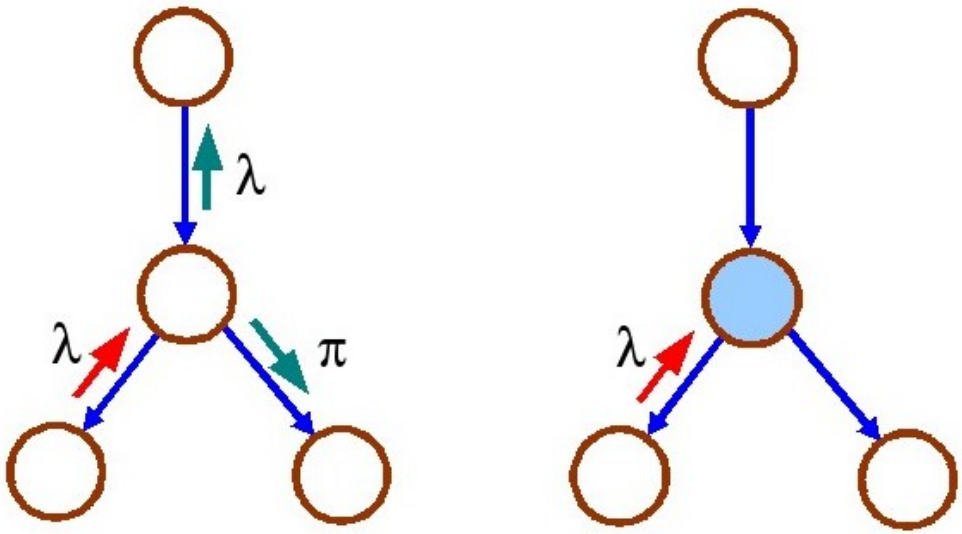
Node S is a good choice of cutset for this graph.

We can instantiate it to each of its four states in turn, and propagate probabilities for each state.

We can get an exact result by combining the four probabilities weighted by $P(S)$ - the prior probability of S

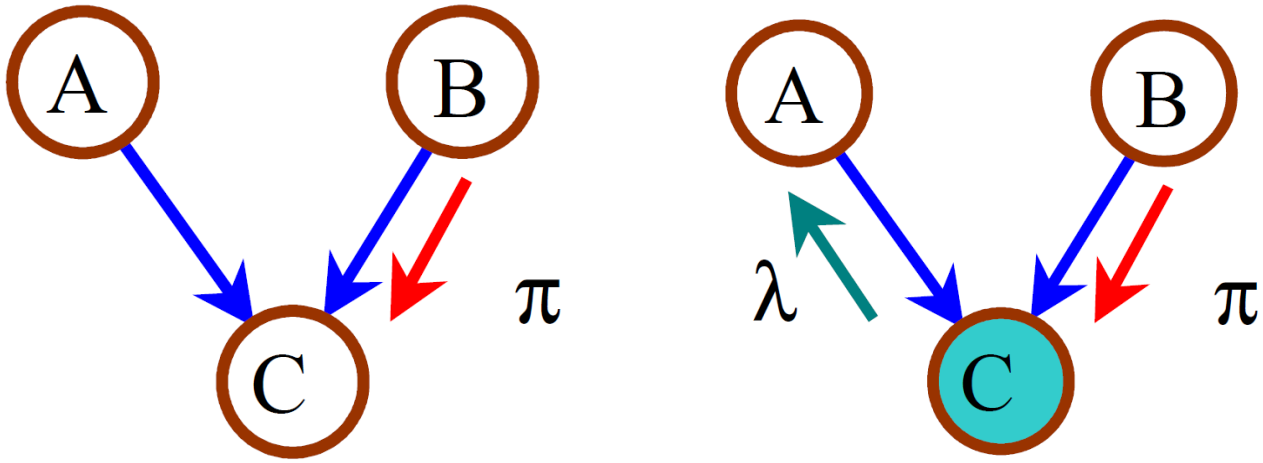
Blocked Paths

If a node is instantiated it will block some message passing. Instantiating the centre node below prevents any messages passing through it.



Converging Paths

However, as we have already seen, converging paths are blocked when there is no λ evidence for the child node, but unblocked if the child has λ evidence or is instantiated.



Lets look again at why this happens.

Converging Paths

To understand the behaviour of converging paths we look at the equation for the λ message in our previous example when there are two or more parents.

$$\boldsymbol{\lambda_L(G\&S) = \lambda(L)P(L|G\&S)}$$

If node L is not instantiated or has not received any λ evidence from its descendants then it will contain no $\lambda(L)$ evidence: $\lambda(L) = (1,1,1)$. So:

$$\boldsymbol{\lambda_L(G\&S) = (1,1,1)P(L|G\&S) = (1,1,1,1,1,1)}$$

This is because we are summing the columns of $\mathbf{P(L|G\&S)}$ which are probability distributions.

Converging Paths

If we now condition the joint message to get an individual λ message for G we use:

$$\lambda_L(g_i) = \sum_j (\pi_L(s_j) \times \lambda_L(g_i \& s_j))$$

Substituting $\lambda_L(G\&S) = (1,1,1,1,1,1)$ we get:

$$\lambda_L(g_i) = \sum_j \pi_L(s_j)$$

The sum is independent of i and hence the value of $\lambda_L(g_i)$ is the same for all values of i, that is to say there is no λ evidence sent to G.

Similarly no λ evidence is sent to S.

Causal Directions

This result tells us something important about cause.

- Bayesian networks have a causal direction associated with the arcs. The arrow points from cause to effect.
- The notion of cause in a Bayesian network comes from the idea of conditional probability:

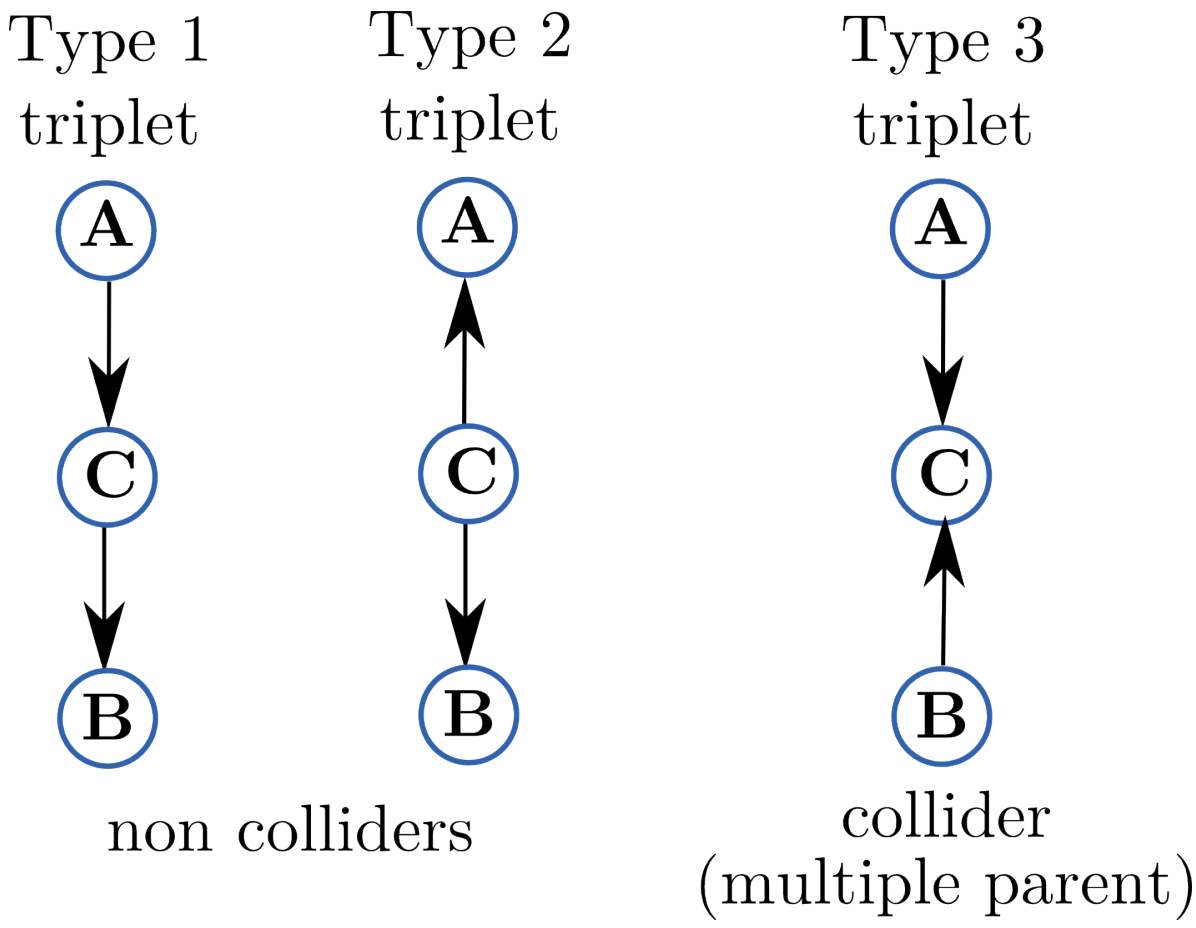
$$P(A\&B) = P(A)P(B|A)$$

- If $P(B|A) = P(B)$ then A and B are independent, if not we think of A causing (or influencing) B, since, given A, B's probability distribution has changed
- However there is nothing in the underlying mathematical theory that helps us to determine causal directions. Instead we usually turn to the semantics of the application to do this.

Causal Directions

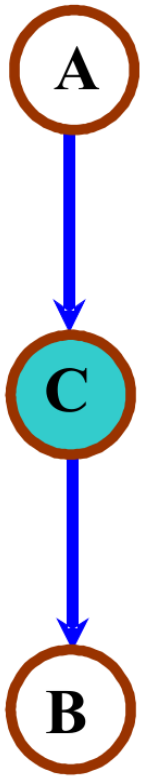
- So far we have taken our causal directions from knowledge about the root variables in our network.
- Knowledge of this kind can come from:
  - Our understanding of the semantics (expert advice)
  - Temporal sequences of events (Granger Causality)
- In certain circumstances, it is also possible to determine cause by examining variable independence.

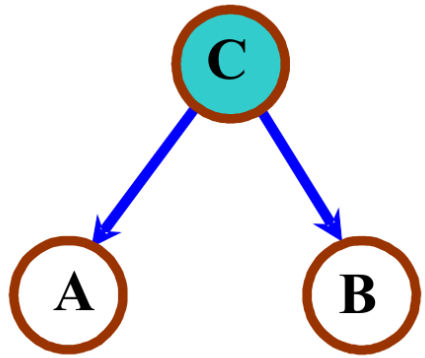
Possible configurations of connected triplets



Non-colliders

For the non colliders A and B are conditionally independent given C.

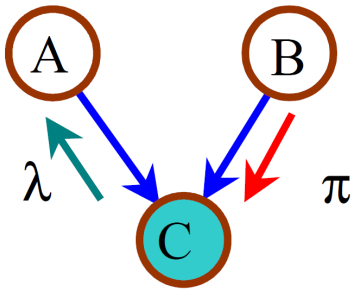




If C is instantiated no messages pass from A to B.

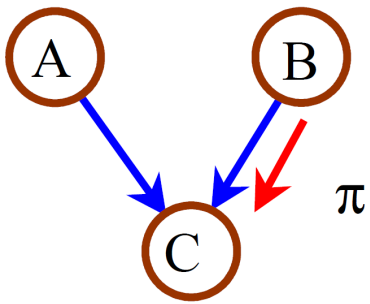
Colliders

However for a collider (multiple parent), the nodes *A* and *B* are only independent if there is no information on *C*.



If *C* is instantiated then changing *B* changes *A* and vice-versa.

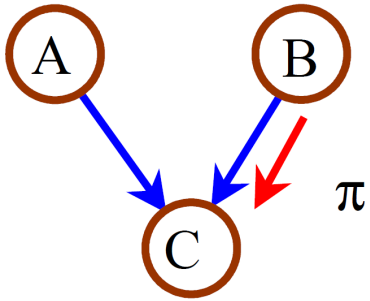
Converging paths are blocked when there is no λ evidence.



If *C* does not have λ evidence then changing *B* will not change *A* and vice-versa.

Marginal Independence

Converging paths are blocked when there is no λ evidence.

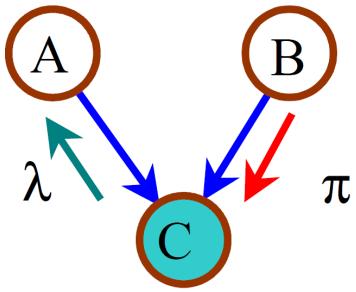


$$\lambda(C) = [1, 1, 1 \cdots 1]$$

Given a data set for the triplet $A - C - B$ we can measure the dependence between A and B using all the data. (ie with no information on C).

If this is low we may suspect that the configuration is a multiple parent.

Conditional Independence



C is instantiated:

$$\lambda(C) = [0, 0, 1 \cdots 0]$$

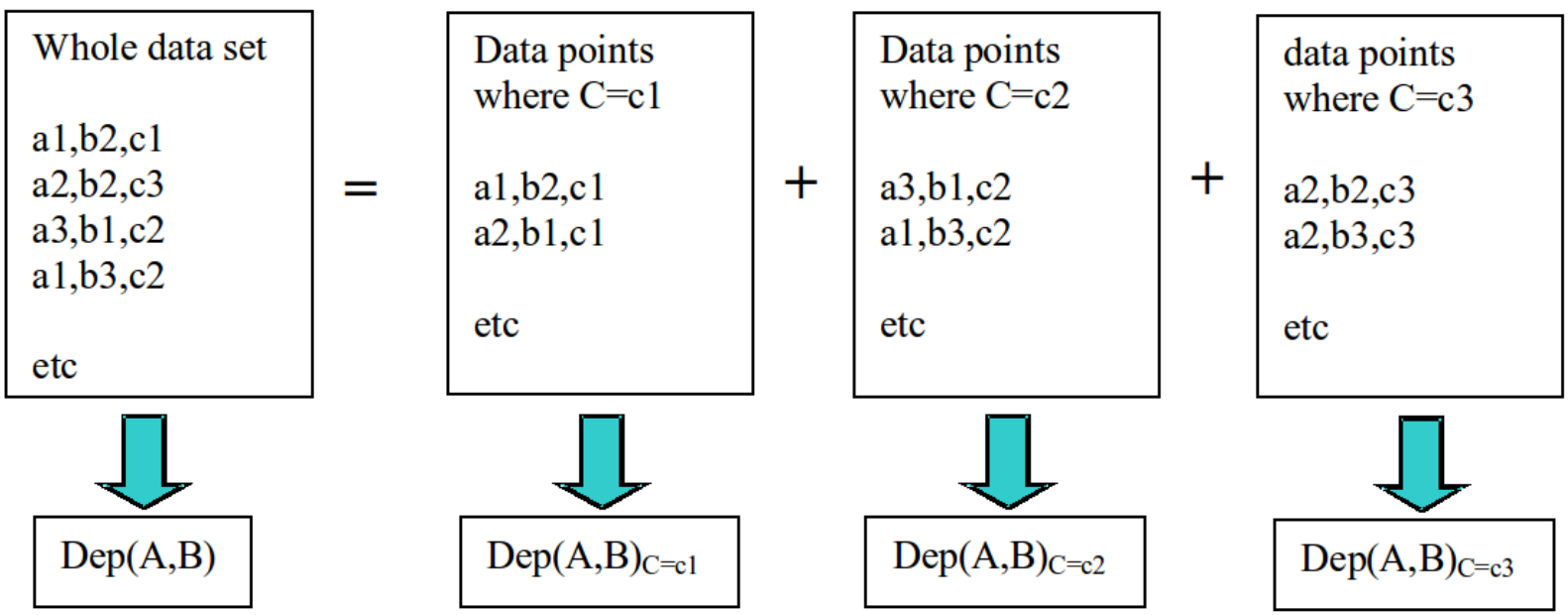
Alternatively we can partition our data according to the states of C, and then compute a set of dependency values (one for each state of C).

If any one of these is high we may again suspect that the configuration is a multiple parent.

Practical Computation

Given a triplet, partition the data according to the states of the middle node C, and calculate the dependency of the other variables A, B for each set.

if $Dep(A,B)$ small, and some $Dep(A,B)_{C=c_j}$ is large then the configuration is likely to be a multiple parent.



Computation from the joint probability matrix

Rather than compute the dependencies from the data we can marginalise the joint probability of a triplet:

$$P(A\&B) = \sum_C P(A\&B\&C)$$

For any joint state $[a_i, b_j]$ we sum all the matrix entries $[a_i, b_j, c_k]$.

We use $P(A\&B)$ to calculate the $Dep(A,B)$ which is the dependence of A and B with no information about C and is called the marginal independence.

Mutual Entropy

Since we are comparing probability distributions over discrete variables the best way to find the dependency is to use the Kullback-Liebler divergence.

$$Dep(A,B) = \sum_{A\times B} P(a_i \& b_j) log_2((P(a_i \& b_j)/(P(a_i)P(b_j)))$$

As previously noted:

- It is zero when two variables are completely independent
- It is positive and increasing with dependency when applied to probability distributions
- It is independent of the actual value of the probability

Computation from the joint probability matrix

Similarly we can use the joint probability matrix to calculate the conditional probabilities:

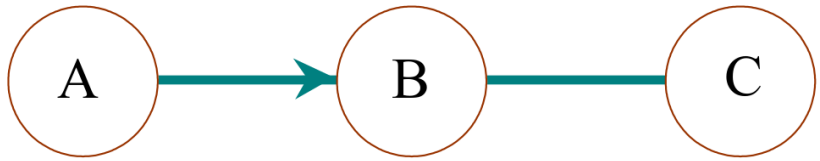
$$P(A\&B|C) = P(A\&B\&C)/P(C)$$

If we think of $P(A\&B\&C)$ as having a column for each state of C and a row for each joint state of A and B, then we normalise each column into a probability distribution.

We then compute a separate value of $Dep(A,B)$ for each column. Adding these together gives us a measure of conditional dependence. If this is close to zero the configuration is not a collider.

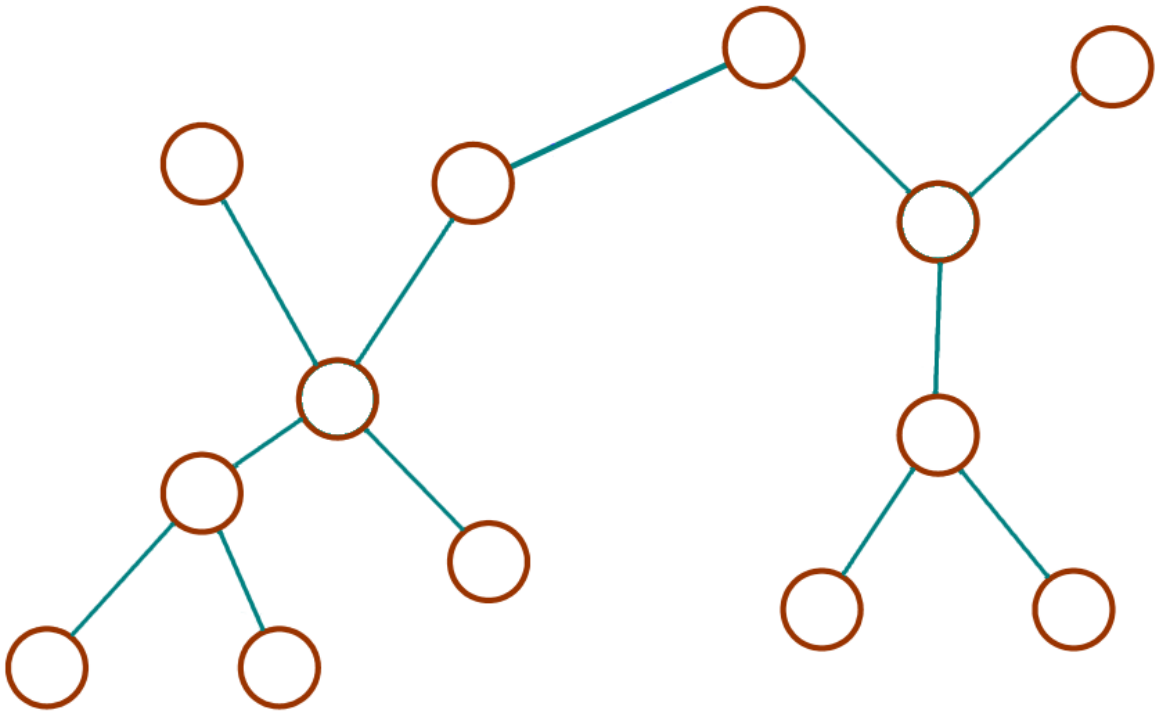
Algorithm for determining causal directions

- compute the maximally weighted spanning tree network
- for each connected triplet in the spanning tree
  - compute the joint probability of the triplet
  - compute the marginal dependence and conditional dependence
  - if the marginal dependence is low and the conditional dependence is high put in causal directions corresponding to a collider (multiple parent).
- propagate the causal arrows as far as possible. eg: given:



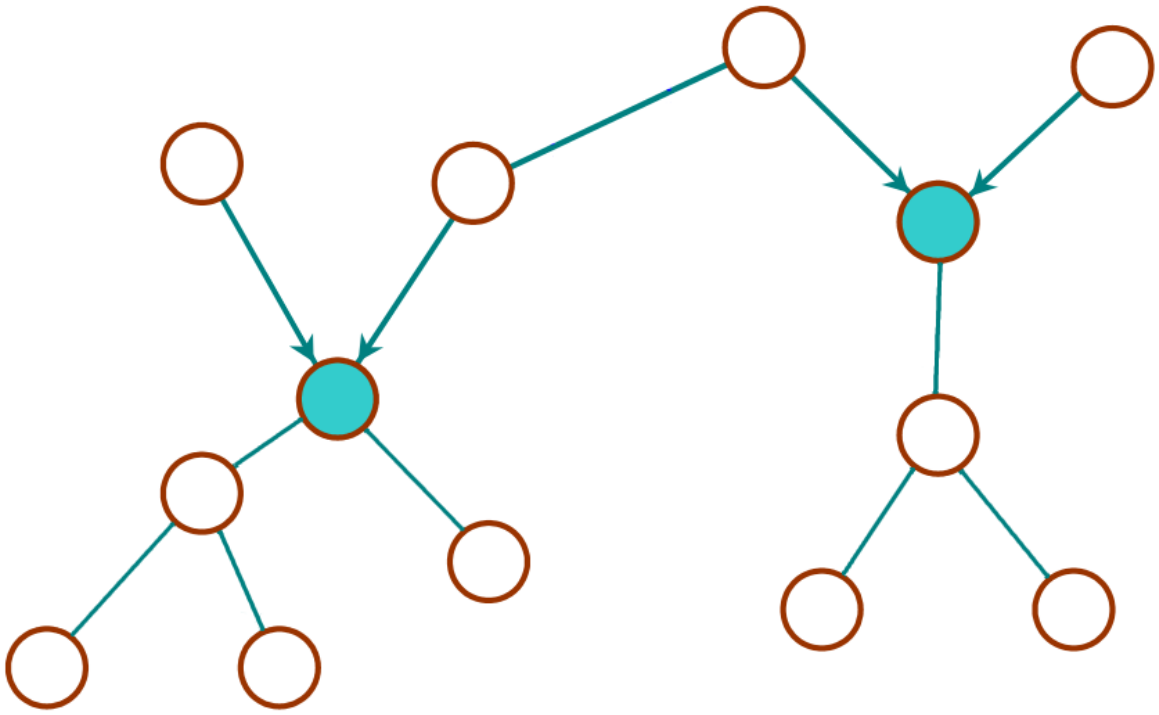
If A and C are independent given B, B is the parent of C and *vice versa*.

Example of finding causal directions



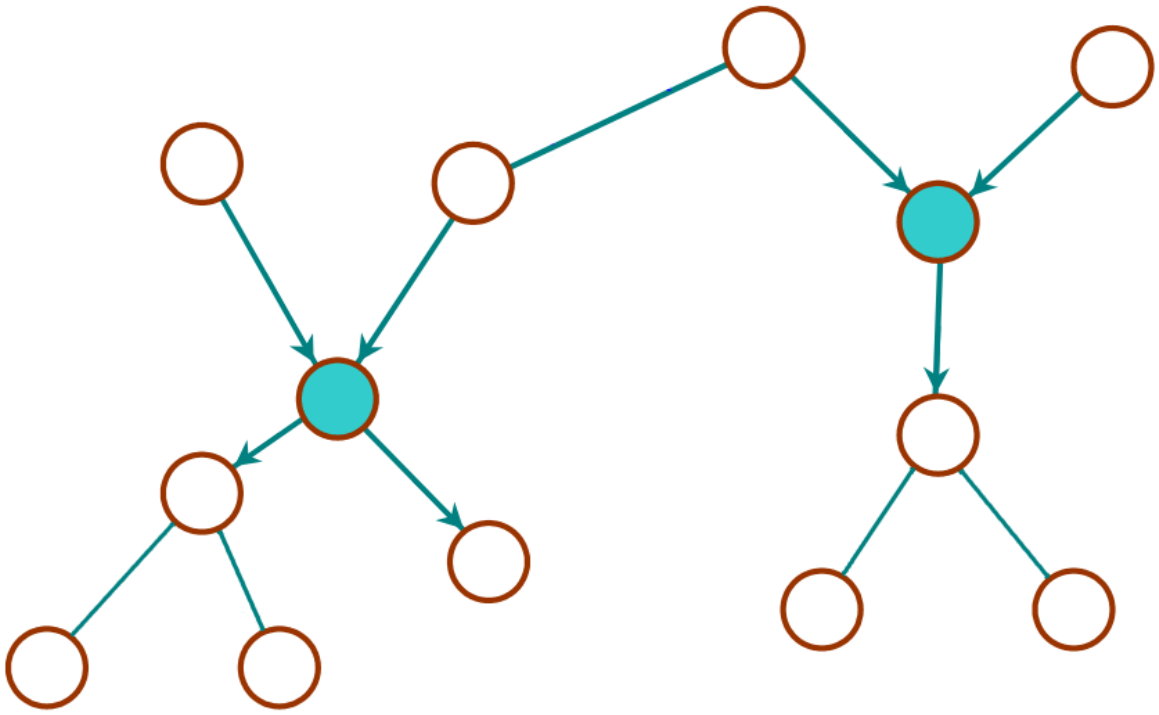
Start by finding the spanning tree

Example of finding causal directions



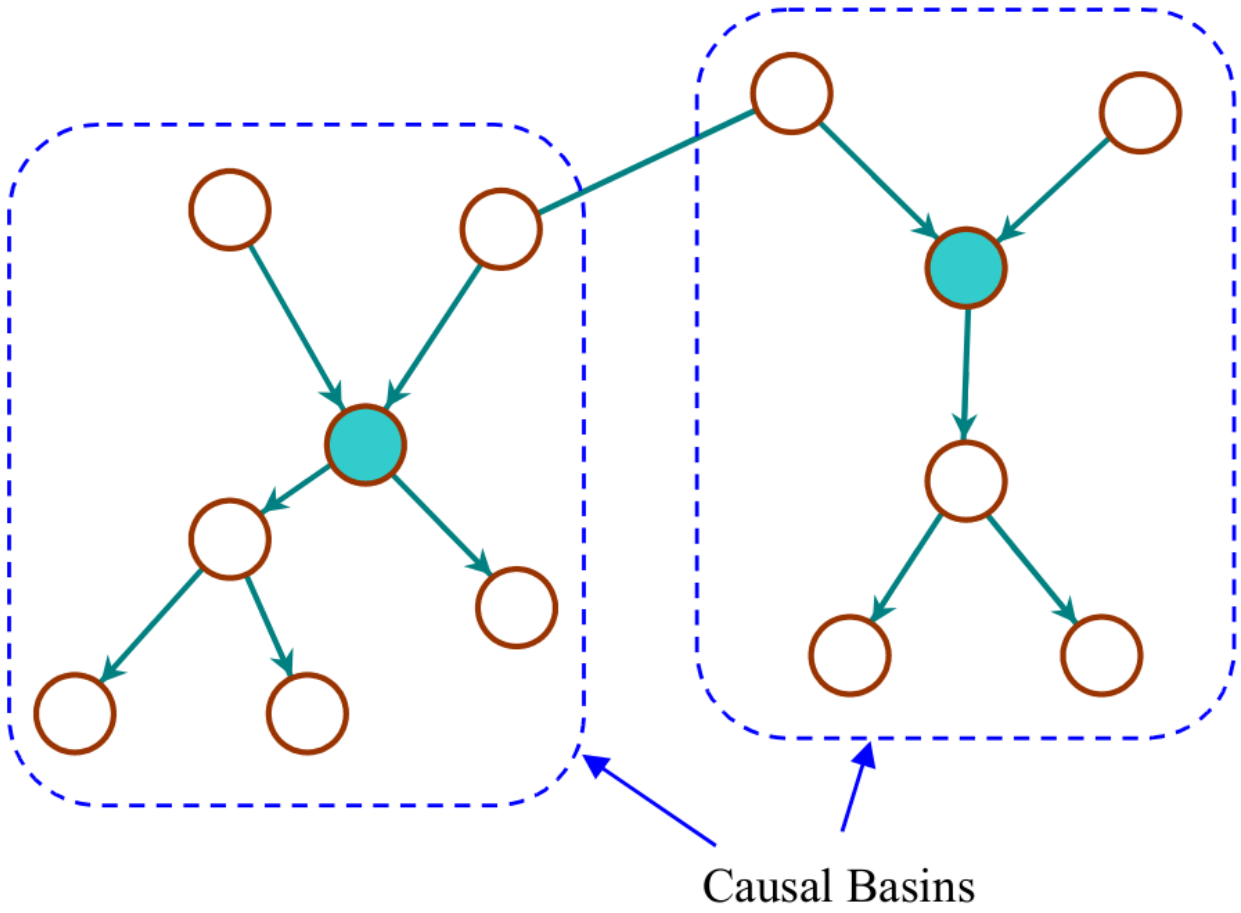
Find all multiple parents using marginal independence

Example of finding causal directions



Propagate arrows where possible

Example of finding causal directions



Continue to the leaf nodes

Problems in identifying causal directions

- Our dependency measures are unlikely to give us a decisive result (independent or dependent). We need heuristic thresholds in practice.
- We may find few (or no) cases of multiple parents.
- If there is an unknown (or unaccounted) source of dependency between two variables then the method will give incorrect results.
- We need to compute joint probabilities of triples of variables, hence there may computational problems.

Learning Cause

- Well, it may not be the most effective algorithm ever constructed, but at least we don't need to listen to that medical expert any more!
- And time is running out, so before she asks me any more embarassing questions, I'll finish with a brief overview on some of the methods that are available for propagating probabilities accurately in causal networks.
- Overall there is a trade off between accuracy and computation speed which gets more pronounced as the number of variables in the network gets bigger.

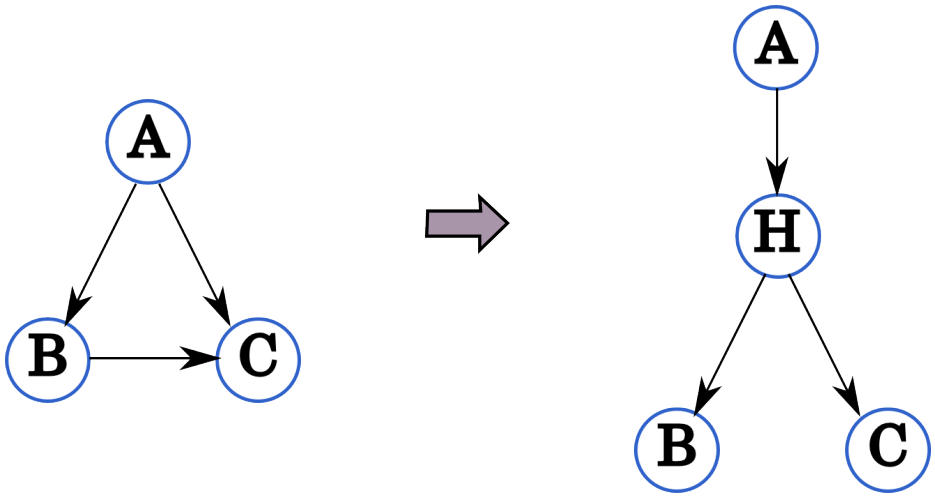
Loopy Belief Propagation

- For very big networks we may need to put up with some inaccuracies and just propagate the evidence for a set number of epochs, or until we satisfy some measure of convergence.
  - This was the approach we took for the random Markov field segmentation algorithm.

- Alternatively we can propagate using the loop and look for ways of correcting the result.
  - This is a theoretically interesting idea, but to date no one has found a way of making a correction when loops are overlapping.
  - There are also practical difficulties in determining when a correction can be applied or not.

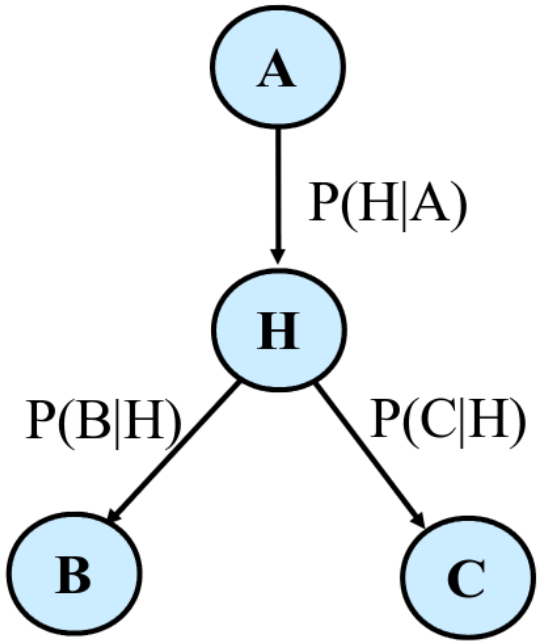
Hidden Nodes (aka Latent Variables)

This is a practical method of taking loops out of a causal network, but it does require statistical learning for which lots of data is needed.



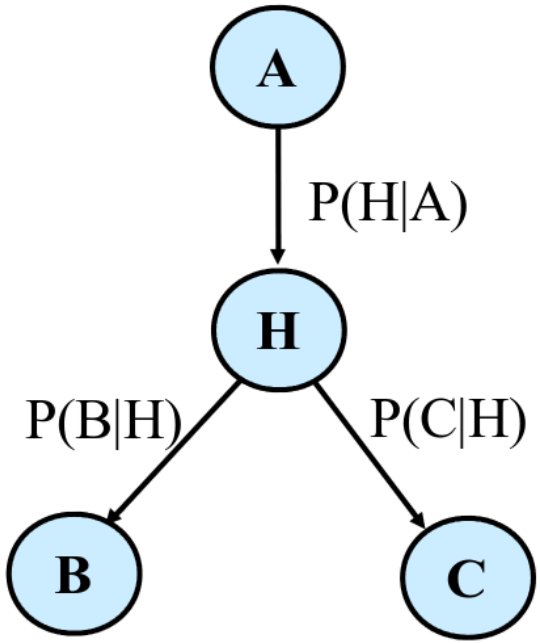
The process is to remove the least dependent link of a multiple parent.

Calculating the Conditional Probabilities



1. Given estimates of: $P(H|A), P(B|H), P(C|H)$ and a set of data points $[a_i, b_j, c_k]$
2. Use each b_j, c_k to compute $P'(A)$ from the network, calculate and accumulate an error:
$$E = (P'(A) - P(a_i))^2$$
3. Minimise E over the data set by adjusting the elements of $P(H|A), P(B|H), P(C|H)$

Calculating the Conditional Probabilities



For each conditional probability $P(c_j|h_k)$ we need to find a value for:

$$\frac{\partial E}{\partial P(c_j|h_k)}$$

Then in each epoch we update the conditional probabilities using:

$$P(c_j|h_k) \Rightarrow P(c_j|h_k) - \mu \frac{\partial E}{\partial P(c_j|h_k)}$$

Gradients may be calculated analytically or numerically. A closed form equation for the gradients was developed by Chee Keong Kwoh.

Gradient Descent and Probabilities

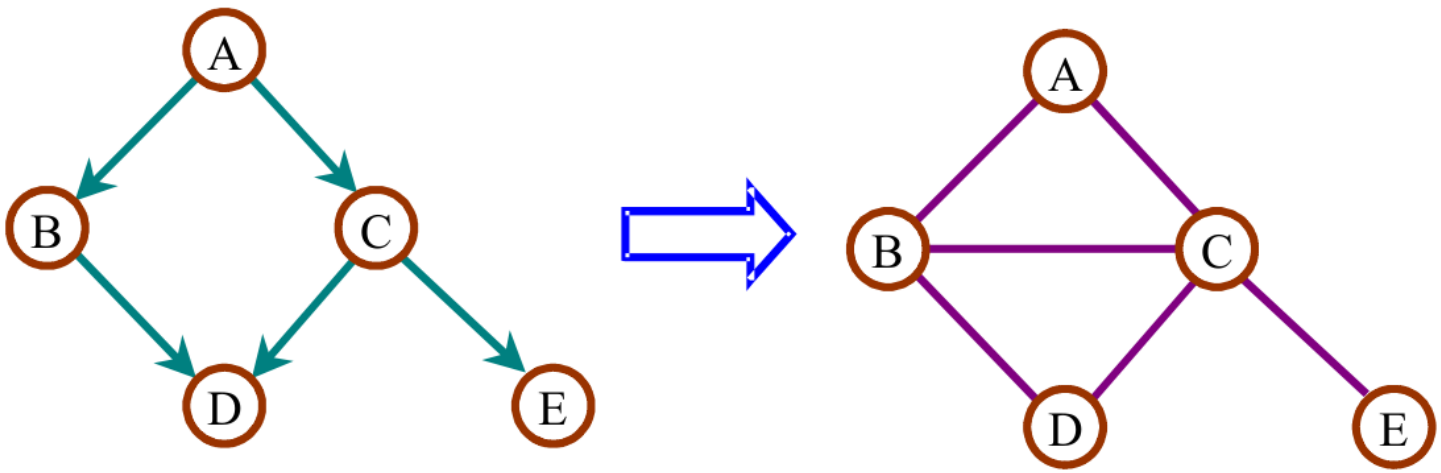
Gradient descent has problems when applied to probability distributions. After one cycle of updating:

- Distributions will no longer sum to 1
- Individual probability values may be greater than 1 or less than 0

The conditional probability matrices must be normalised so that the columns sum to 1. This may compromise finding an optimal solution.

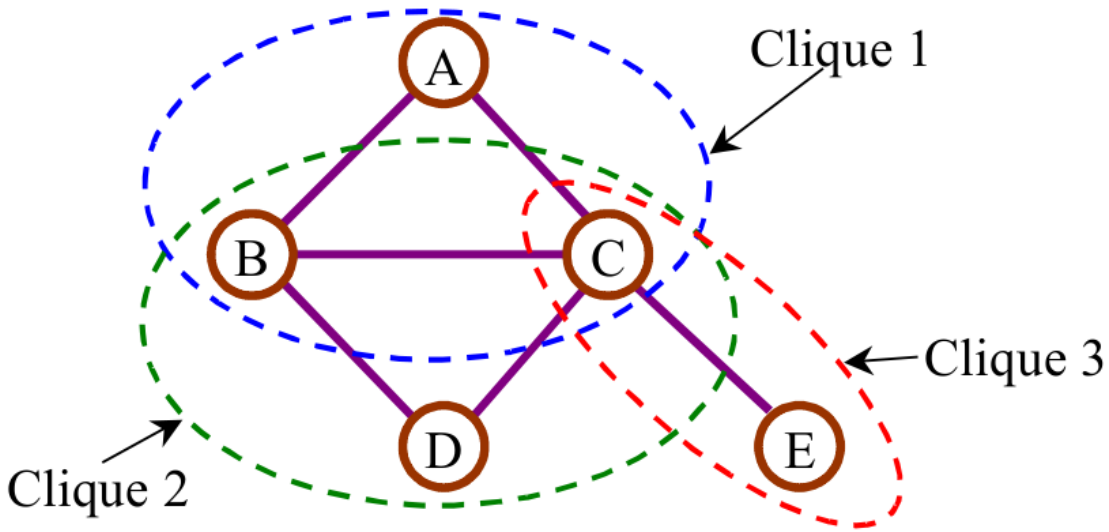
Join Trees

The join tree algorithm converts a causal network into a singly connected Markov random field. It identifies subsets of interdependent nodes and propagates probabilities between them. First a dependency graph is created:



Join Trees

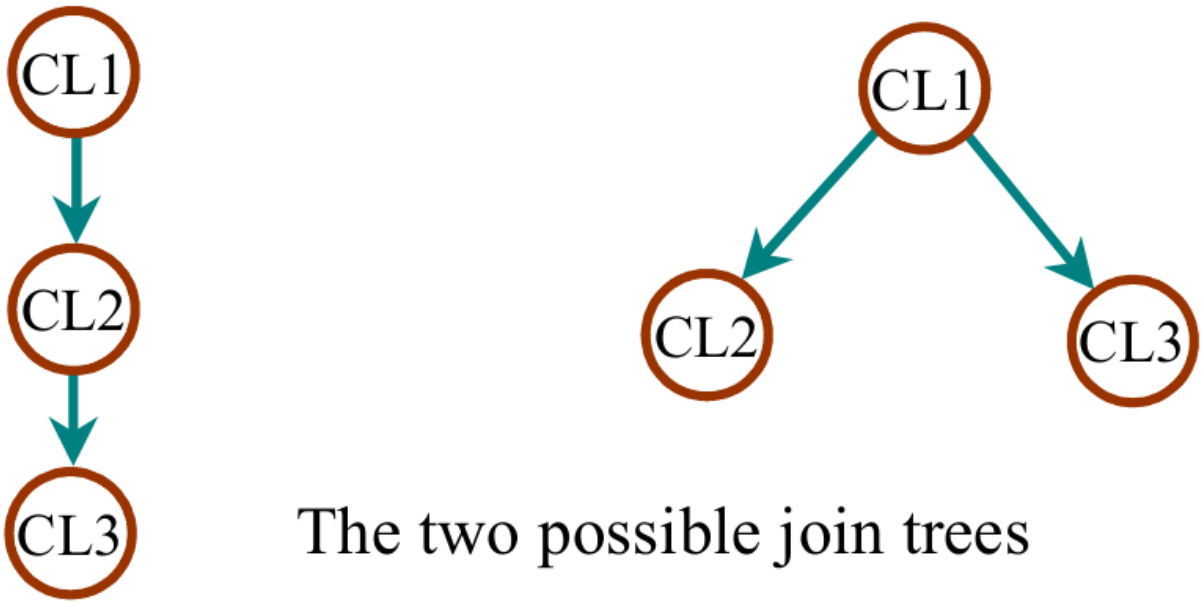
Next all maximal fully connected subsets (cliques) of the nodes are found:



These cliques are the nodes of the Join tree which is an undirected Random Markov field.

Join Trees

Propagation can be carried out between the joined variables in the cliques



The two possible join trees

Join Trees

The join tree algorithm allows us to have the best of both worlds.

- We can use causal models to incorporate our knowledge in the data model or to investigate causality in data sets that we want to model.
- We can use join trees for exact belief propagation (without the need to see that join tree).

Unfortunately there is no free lunch. The join tree algorithm is n-p hard. The more dependency there is in the data the slower they are.

Structure and Parameter Learning

- Networks combine both structure and parameters.
- We can express our knowledge (if any) about the data by choosing a network structure.
- Alternatively we can learn our network structure from data.
- We then optimise the performance by adjusting the parameters (link matrices).

Other machine learning formalisms

- Neural Networks
  - Neural networks are a class of inference systems which offer just parameter learning.
  - Generally it is very difficult to embed knowledge into a neural net, or infer a structure once the learning phase is complete.
- Rule based systems (Logic)
  - Traditional rule based inference systems have just structure (sometimes with a rudimentary parameter mechanism).
  - They do offer structure modification through methods such as rule induction.
  - However, they are difficult to optimise using large data sets.

Inferring Cause

- Neural networks are most applicable when we have no causal knowledge, but correspondingly we cannot extract causal information from them.
- Rule based systems are most applicable when we have good causal knowledge as they can represent it well.
- Bayesian networks can incorporate known causal relations, but also have the potential to learn cause from data.

Learning Cause

- The notion of learning cause from data caused considerable interest in data mining applications - for example genetics.
- Microarray data can measure simultaneous activities of genes normal and cancerous cells. We could potentially learn causal networks from these using the methods described above.
- However to date the large number of variables, small number of data sets and high experimental error has meant the technique has not met with huge success.
- However it remains an intriguing idea as data volumes and accuracy expands.

Network Inference

Well that really is the end!

I hope you have enjoyed these lectures.