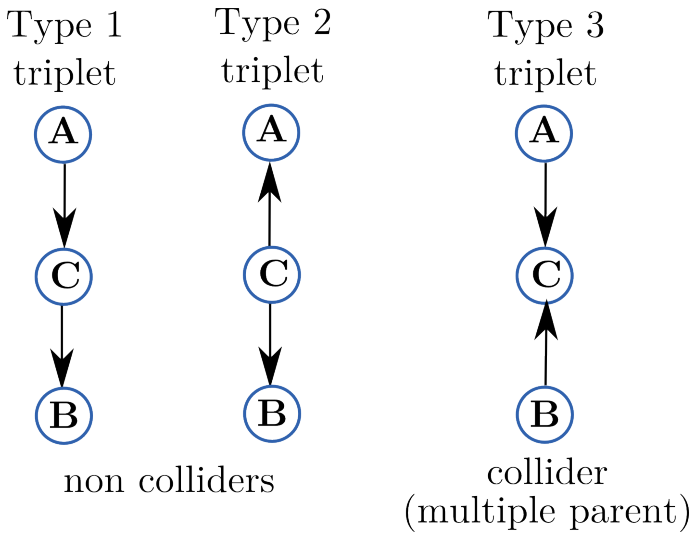
Lecture 6: Cause and Independence

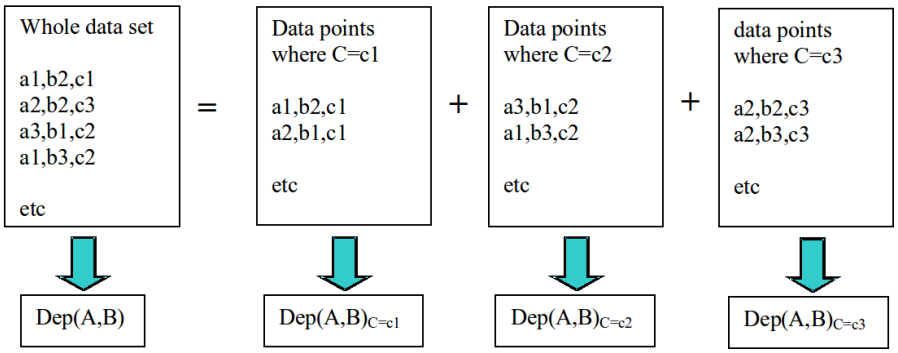
Inferring Cause from Data

When determining the causal directions in a spanning tree we assumed that we knew the root nodes. This information was provided from our knowledge of the problem, in particular the variables involved. However it is possible (at least in theory) to determine some of the causal directions in a network by purely statistical means.

We will begin by considering the simple case of a triplet which may be part of a spanning tree. We can enumerate all possible causal directions for the triplet, there are just four:



Now consider the effect that instantiating node C will have in each case. In configurations of type 1 and type 2, the nodes A and B are conditionally independent given C. Hence if we instantiate C no λ or π messages can pass between A and B. Any change to the evidence or posterior probability at A will not change the evidence for B and *vice versa* (providing, of course, that the network is singly connected). Thus for any given state of C we expect no dependency between A and B. We can test explicitly using the data to see whether this is the case. All we need to do is to partition the data according to the states of C and calculate the dependency for each partition.

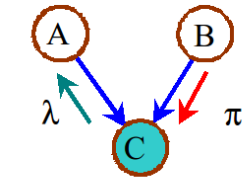


If all of the individual dependencies (ie $Dep(A,B)_{C=ci}$) are low then we can infer that A and B are conditionally independent given C and hence the tuplet is of type 1 or type 2. However, if one or more of them are large, and at the same time we find that the overall dependency $Dep(A,B)$ is small then there is strong evidence that A and B are parents of C. If we recall how the operating equations for the λ message in a multiple parent worked we can see why this should be so.

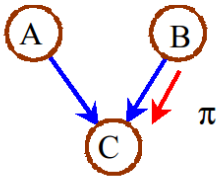
$$\lambda_C(a_i) = \sum_{j=1}^{m} \pi_C(b_j) \sum_{k=1}^{n} P(c_k|a_i \& b_j)\lambda(c_k)$$

Considering the case where $\lambda(C) = [1,1,1,...1]$ we can see that the second sum evaluates to 1 since we are simply summing one of the columns of the link matrix. Thus for each state a_k of A the λ message entry $\lambda_C(a_k)$

will be the same, namely $\sum_i \pi_C(b_i)$, and hence no evidence is passed to A. This is not the case if there is λ evidence for C or if C is instantiated.



λ(C) = { 0,1,0, . . 0}



λ(C) = { 1,1,1, . . 1}

This is the idea behing the *Marginal Independence* method first proposed by Pearl and Rebane. It is so called because, rather than compute the dependencies from the data, we can compute them directly from the joint probability matrix $P(A, B, C)$. If we marginalise this over C, that is for any joint state $a_i \& b_j$ we sum the entries for each state of C, we can form the joint probability matrix $P(A\&B)$:

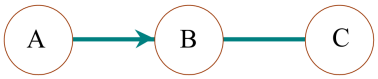
$$P(A\&B) = \sum_C P(A\&B\&C)$$

and we use this to compute $Dep(A\&B)$. If this is low we have marginal independence.

We can also calculate the conditional probability matrix from the joint probability matrix. If we think of $P(A\&B\&C)$ as having a column for each value 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. If all these are low we have conditional independence. For determining cause, we might simply add up these conditional dependencies to compare with the marginal dependence.

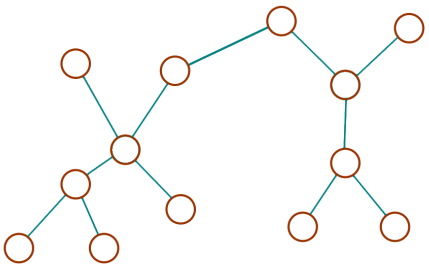
We use the results of the dependency calculations in the following algorithm for determining causal directions:

- compute the maximally weighted spanning tree
- 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:

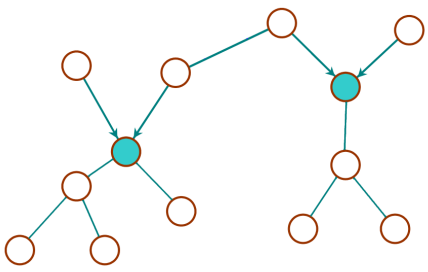


If A and C are independent given B, B is the parent of C and *vice versa*.

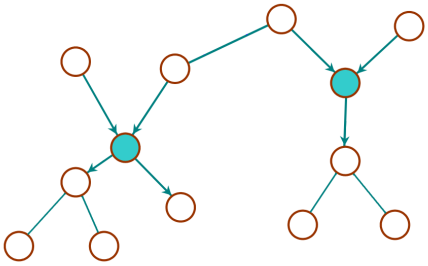
An example of this algorithm is shown in the following diagrams.



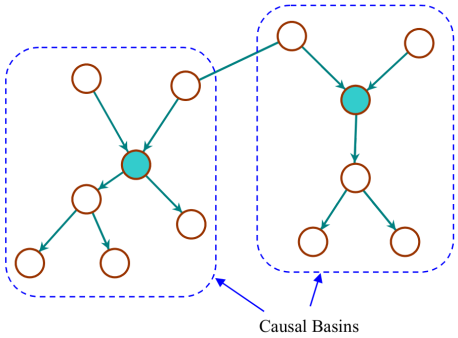
Start by finding the spanning tree



Find all multiple parents using marginal independence



Propagate arrows where possible



Continue to the leaf nodes

Problems in determining cause

Mutual entropy is a continuous function, it tells us only the degree to which variables are dependent. Thus we need thresholds to decide whether a node has multiple parents. The same problem is inherent in the spanning tree algorithm. Furthermore, we may find a few (or no) cases of multiple parents, resulting in there being small causal basins or no cause information at all. Lastly, the algorithm requires us to examine every triplet in the network. Hence it will scale very badly for large networks.

Structure and Parameter Learning

One of the good features of Bayesian networks is that they combine both structure and parameters. We can express our knowledge about the data, in the form of cause and effect, by choosing a structure. We then optimise the performance by adjusting the parameters (link matrices).

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. 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.

We noted previously that the joint probability of the variables in a Bayesian Network is simply the product of the conditional probabilities and the priors of the root(s). If this result is to apply to the data from which the network was built, then the network must represent the dependency exactly. However, using a spanning tree algorithm this may not be the case. In particular, we may not be able to insert an arc between two nodes with significant dependency because it would form a loop, and make the network multiply connected. The effect of unaccounted dependencies is likely to be more pronounced as the number of variables in the network increases. For this reason there seems to be some advantage in keeping networks as small as possible, and this has engendered a number of algorithms.

Multi-Trees

An interesting way of reducing the size of Bayesian networks was proposed by Heckerman. Here the data set is partitioned according to the states of the root node(s). A separate tree is then found for each state of the root nodes. The method is illustrated by the following example.

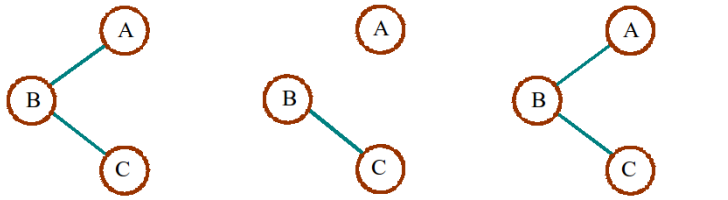
Given a data set with the root identified as D:

$$[a_1, b_1, c_1, d_1][a_2, b_1, c_1, d_1][a_2, b_1, c_2, d_1][a_2, b_2, c_1, d_1]$$
$$[a_1, b_2, c_2, d_2][a_2, b_1, c_1, d_2][a_2, b_2, c_2, d_2][a_1, b_1, c_1, d_2]$$
$$[a_1, b_1, c_1, d_3][a_2, b_2, c_2, d_3][a_2, b_2, c_1, d_3][a_2, b_2, c_2, d_3]$$

Since D has three states we split the data into 3 sets:

- Data set for $D = d_1$: $[a_1, b_1, c_1][a_2, b_1, c_1][a_2, b_1, c_2][a_2, b_2, c_1]$
- Data set for $D = d_2$: $[a_1, b_2, c_2][a_2, b_1, c_1][a_2, b_2, c_2][a_1, b_1, c_1]$
- Data set for $D = d_3$: $[a_1, b_1, c_1][a_2, b_2, c_2][a_2, b_2, c_1][a_2, b_2, c_2]$

and we use the spanning tree methodology to find three trees with 3 variables rather than one tree with 4 variables.



Tree found for D=d1 Tree found for D=d2 Tree found for D=d3

Looking at the data we see that the dependencies are not the same for the three reduced data sets. Thus we could expect that this method would give us a more accurate representation of the true causal structure than a single tree.

For a given data point: (ai,bj,ck) we now calculate the joint probability using each tree found:

- Evidence for d_1 is $P_{D=d1}(a_i \& b_j \& c_k)$
- Evidence for d_2 is $P_{D=d2}(a_i \& b_j \& c_k)$
- Evidence for d_3 is $P_{D=d3}(a_i \& b_j \& c_k)$

These quantities can be calculated using the fact that the joint probability is simply the product of the priors and conditional probabilities. For the case of the tree for $D = d_2$ this is simply $P(A)P(B)P(C|B)$ or $P(A)P(C)P(B|C)$ which is the same thing. The evidence can then be normalised to form a distribution over the states of D, and a decision made on the result.