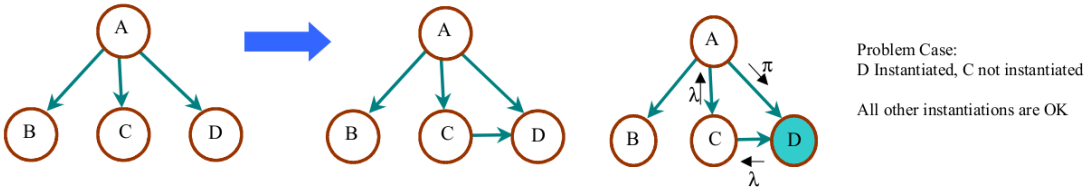
Lecture 8: Approximate Inference

This set of notes is out of date. Please see the corresponding slides for the latest material. I will revise this handout shortly

Pearls probability propagation algorithm is fast and intuitively appealing since it tells us about the causal structure upon which the inference was made. However it is limited to singly connected networks. Although for some cases of instantiations propagation is possible, termination cannot be guaranteed in the general case. If we adopt a naive network, or use the spanning tree algorithm to create a network we are almost certainly adopting an approximate inference approach. This is because there will be data dependences which cannot be added to the network without creating a loop.

Tree Augmented Network

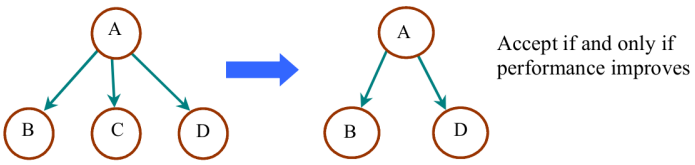
Given a naive Bayesian network we can try to make it exact by adding arcs between the child nodes. If C and D are dependent given A we can change the network structure to include the arc C→D or D→C. If we do not know which way to orient the causal direction, we make an arbitrary choice. If we choose C→D we then have the problem that propagation is not possible if D is instantiated and C and A are not. Similar problems occur if we add the complete set of arcs with significant dependency to the spanning tree. There are algorithms that can calculate probabilities for these cases, and we can always calculate them from data, however all such algorithms are n-p hard and become computationally infeasible for relatively small numbers of variables and states.



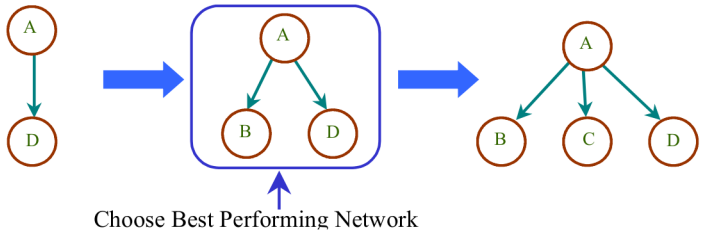
The fact that spanning trees and naive networks are usually only approximate solutions means that in many cases a naive Bayes network will out perform a spanning tree despite the fact that the latter should be a better representation of the dependencies. Indeed the spanning tree and the naive network may not even be the best approximate structure to choose. Thus different refinement algorithms have been investigated to address this problem.

Selective Naive Bayesian Network

The idea here is to use only a subset of the variables. This can be done by deletion followed by testing (Sucar 1993)

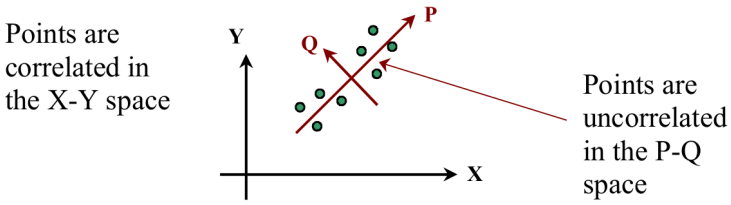


An alternative strategy is to add variables until no performance improvement is found (Langley and Sage (1994)).

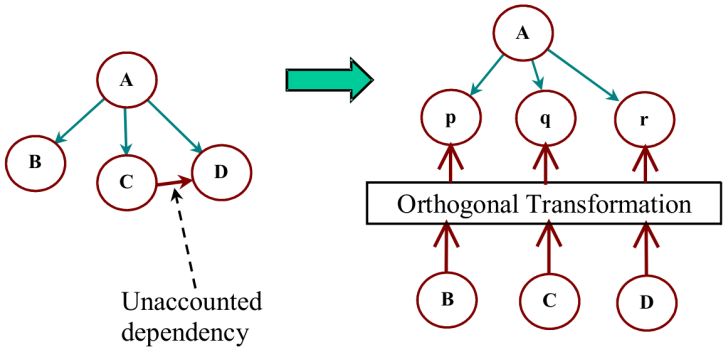


Orthogonal transformation of variables

There is a well known transformation called principal component analysis which transforms one set of variables into another where the correlations between variables are minimised. We will look at this in more detail later in the course. The basic transform is simply an axis change, but arranged in a multi dimensional space. In two dimensions it can be visualised simply:



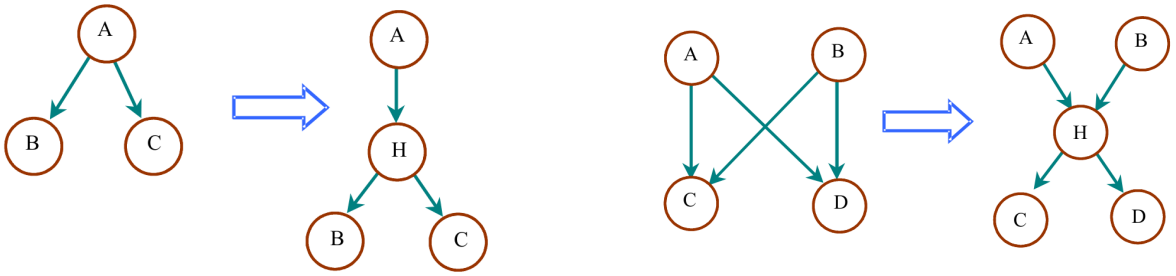
This idea can be applied to remove correlation from the data.



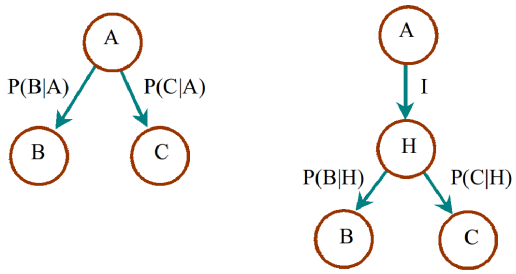
In practice the details become a little complex, and will not be discussed here.

Hidden Nodes

A third technique is adding a hidden nodes between the parent and a pair of child nodes which are not conditionally independent given the parent. The technique can be thought of as finding a common cause for the relationship between the two children. It should be appropriate where there is some mutual information linking the children, but not an obvious or strong dependency. Adding a hidden node can always be done since it does not cause a singly connected network to become multiply connected. Indeed, adding a hidden node could turn a multiply connected network into a singly connected one, as we saw in a previous example.



It will be seen that a network to which a hidden node has been added can always represent the same information as the network without the hidden node. In the network below suppose that we choose H to have the same number of states as A, and we set $P(B|H) = P(B|A)$, $P(C|H) = P(C|A)$ and $P(H|A) = I$ (the identity matrix). The two networks will yield identical results, thus we can expect at least the same performance from a network after adding a hidden node. In practice this may not always occur because, as we will see, we need to use an optimisation process to find the conditional probabilities



Sometimes we do have information on the hidden node added. This was the case for example when we added the eyes node to the cat network in lecture 2. However, in general, we don't have any data or other knowledge on the hidden node. This leaves us with two questions:

1. How many states should the hidden node have?
2. How can the conditional probabilities be found without data?

The number of states can be determined experimentally. Since the hidden node is, in effect, acting as a common cause, we expect that the number of states will be comparable to the number of states of the nodes it is separating. In fact as a good starting point we could set the number of states to the largest of the parent and children, and then prune out states that have very low probabilities in the final link matrices.

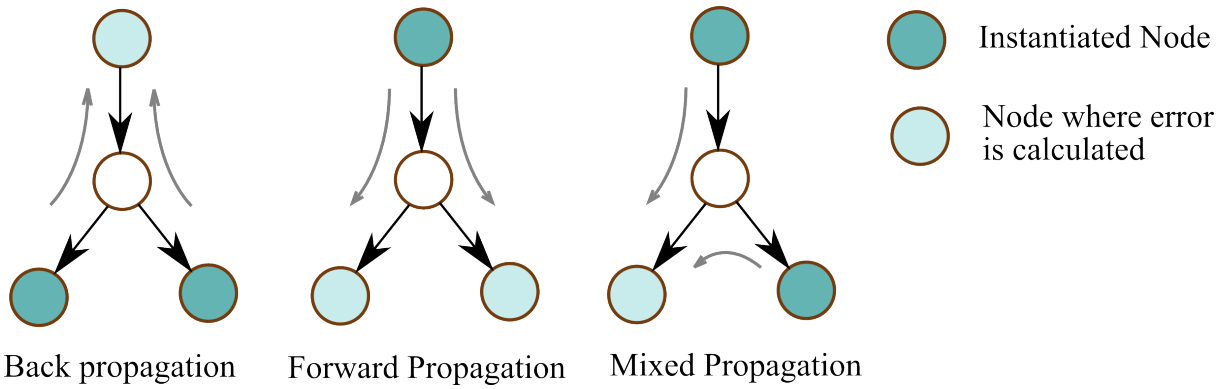
The conditional probabilities can be found by an optimisation technique using the networks predictive performance as a criterion for success. The idea is to go through the data set of values for A, B and C and for each point instantiate say B and C and calculate the value of A. An error function could be the Euclidean distance between the actual value of A and the calculated value. The optimisation is set up to minimise this error. In summary:

1. Given estimates of: $P(H|A)$, $P(B|H)$, $P(C|H)$ and data points (a_i, b_j, c_k)
2. Use b_j, c_k to compute $P(A)$, calculate an error measure: $E = (P(A) - P(A, a = aj))^2$
3. Minimise E over the data set by adjusting the elements of $P(H|A)$, $P(B|H)$, $P(C|H)$
   - For each conditional probability $P(c_j|h_k)$ find an analytical expression for $\partial E/\partial P(c_j|h_k)$
   - Then in each epoch update the conditional probabilities using:

$$P^{n+1}(c_j|h_k) = P^n(c_j|h_k) - \partial E/\partial P(c_j|h_k)$$

This process is called back propagation and is equivalent to the procedure used to train hidden layers in neural networks. In contrast to other optimisation processes it has a further complication that the conditional probabilities must obey the laws of probability. Thus special actions need to be taken if the optimisation process makes a value negative. At the end of each epoch the probabilities must be normalised so that the columns sum to 1.

Back propagation is not the only possibility for setting up and minimising an error function. In practice we can instantiate any subset of the variables and calculate an error function at the remainder. The following are possibilities.



Estimating a hidden node is an optimisation process, and as a consequence runs the risk of being trapped in a secondary minimum. Therefore it is usually desirable to include some form of simulating annealing and to use a fairly slow step size. Alternating the different propagation methods can benefit the process since they do not necessarily adjust the parameters in the same way.

Modifying networks using hidden nodes

The decision to modify a network can be made on the basis of how well it fits the data. We did discuss measures of model accuracy, but these are not sufficient for this purpose. In particular, we need to know which parts of the network should be modified. For example, for each pair of nodes without an arc we could see what residual mutual information remains between them. Alternatively, for children we could test their conditional independence given the parents.

We have previously noted that the naive network performs as well as more complex singly connected structures. Hence one effective methodology for classifiers is to start with a naive network and incrementally remove conditional dependencies between children.