Probabilistic Modelling and Reasoning
The Junction Tree Algorithm

David Barber
dbarber@anc.ed.ac.uk
course page : http://anc.ed.ac.uk/∼dbarber/pmr/pmr.html

Parts of this chapter are taken from *Expert Systems and Probabilistic Network Models* by E. Castillo, J. Gutierrez, and A. Hadi (Springer, 1997), and also *An introduction to bayesian networks* by F.V.Jensen (Springer 1996). Both are excellent introductions to the field.

1 The Junction Tree Algorithm

One may have had the nagging suspicion during the discussion of the algorithms relating to DAGs, that what we are really exploiting is the underlying graphical structure, whether this be directed or not. Does it really matter if the graph is directed. For example, in bucket elimination, we are essentially removing the directedness of the graph and defining (undirected) functions in their place. What this suggests is that the complexity of the calculations on directed graphs can be transformed into an undirected graph, possibly of greater connectivity than the directed graph from which it was derived.

Indeed, there is an algorithm that does this. A graph, directed or undirected, is transformed into an undirected graph on which the relevant computations can be performed. This is called the junction tree algorithm.

Cluster Potential Representation of a Graph

Consider a directed chain

$$p(U) = p(a|b)\, p(b|c)\, p(c|d)\, p(d)$$

where U, the "universe" represents all the variables in the graph. The

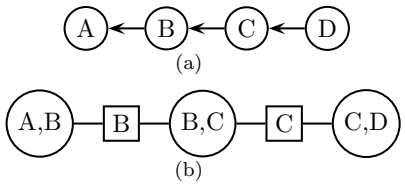


Figure 1: (a) A belief network. (b) A cluster graph representation of the network. The cluster potentials are defined on the round/oval nodes, and the separator potentials are defined on the square nodes, which share common variables with their neighbours.

cluster graph distribution is defined as the product of all the cluster potentials, divided by the product of the separator potentials. In fig(1), for the cluster graph to represent the BN we need

$$p(U) = \frac{\Psi(a,b)\,\Psi(b,c)\,\Psi(c,d)}{\Psi(b)\,\Psi(c)} = p(a|b)\, p(b|c)\, p(c|d)\, p(d)$$

One such assignment of the cluster and separator potentials to satisfy this would be $\Psi(a,b) = p(a|b)$, $\Psi(b,c) = p(b|c)$, $\Psi(c,d) = p(c|d)\, p(d)$, and $\Psi(b) = 1$, $\Psi(c) = 1$. Note that here we have defined the potentials to be functions of the (cluster) node of variables, and there is no restriction about symmetry of these potential functions (in contrast to the example of an undirected graph given in a previous chapter, where the potentials where functions of the links and not of the nodes).

For every cluster representation, we claim that there exists another cluster representation for which the clusters contain the marginals of the distribution. Consider the following, which follows simply from the definition of conditional probability:

$$p(U) = \frac{\Psi^*(a,b)\,\Psi^*(b,c)\,\Psi^*(c,d)}{\Psi^*(b)\,\Psi^*(c)} = \frac{p(a,b)\, p(b,c)\, p(c,d)}{p(b)p(c)}$$

Where the cluster and separator potentials are set to $\Psi^*(a,b) = p(a,b)$, $\Psi^*(b,c) = p(b,c)$, $\Psi(c,d) = p(c,d)$, and $\Psi^*(b) = p(b)$, $\Psi^*(c) = p(c)$. It turns out that every singly-connnected graph can

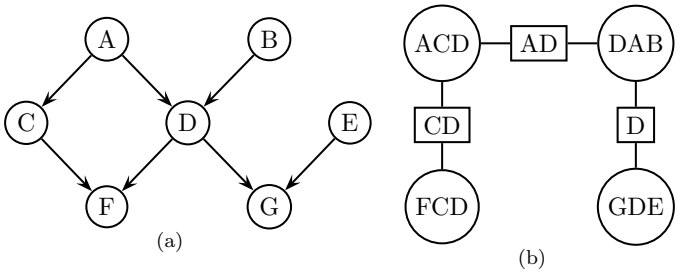


Figure 2: (a) A loopy Belief Network. (b) A cluster graph representation of the network. A valid assignment of the cluster potentials is $\Psi(f,c,d) = p(f|c,d)$, $\Psi(d,a,b) = p(d|a,b)p(a)p(b)$, $\Psi(g,d,e) = p(g|d,e)p(e)$, $\Psi(a,c,d) = p(c|a)$. All the separator potentials are set to 1. Note that whilst it would appear that it is unnecessary to make the cluster ACD dependent on d, this is nevertheless chosen, since the cluster graph then satisfies the running intersection property, namely that if a variable appears in two different clusters, it also appears on all clusters inbetween.

always be represented as a product of the the clique marginals divided by the product of the separator marginals – this is a useful and widely applied result in the development of exact and approximate inference algorithms.

What we would like to do for a general distribution is to define a potential representation of the graph such that, coupled with a suitable algorithm to modify these potentials, the effect will be, as above, that the marginals of individual or groups (in fact cliques) can be read off directly from the modified potentials. This is the idea of the junction tree algorithm.

Marginalization

Let $\Psi(v)$ be a potential on the set of variables v, and let w be a subset of v. A potential over w can be constructed by marginalization, by which we mean

$$\Psi(w) = \sum_{v \setminus w} \Psi(v)$$

1.1 Cluster Trees

Cluster A cluster is a set which contains one or more variables. A cluster is represented by a (cluster) node.

Separator The separator (set) is the intersection of two adjacent nodes. For example, in fig(2), the separator of the nodes containing DAB and FCD is D.

Cluster Tree A cluster tree over U is a tree of clusters of variables from U. The nodes are subsets of U and the union of all nodes is U.

Associated with each node and separator is a potential over the variables contained in the node or separator.

Note that we define potentials also on the separator sets. The reason for this will become clear later since it will help us construct an invariant representation of a graphical model.

A cluster tree corresponding to a BN is constructed in the following way:

- Form a family of nodes such that for each variable a with parents $\mathrm{pa}(a)$ there is at least one node v such that $\mathrm{pa}(a) \cup \{a\} \in v$.

- Organize the nodes as a tree with separators
- Initialise all node and separator potentials to unity.
- For each variable a choose exactly one node v containing $\text{pa}(a) \cup \{a\} \in v$ and multiply the potential on v by $p(a|\text{pa}(a))$.

It is clear from this construction that the product of all the potentials is just the product of all the conditional probabilities in the BN, so that this is indeed a potential representation of the BN.

For example, in fig(2) we depict a cluster tree representation of the BN in fig(2a)

In this case, the cluster potentials are

$$\Psi(d,a,b) = p(d|a,b)\,p(a)p(b), \quad \Psi(f,c,d) = p(f|c,d), \quad \Psi(g,d,e) = p(g|d,e)p(e)$$

The separator potentials $\Psi_1(d)$ and $\Psi_2(d)$ are set to unity.

Absorption in cluster trees

The idea behind absorption is that we wish the potentials to be modified in such a manner that the potential resulting from marginalization to their separator s from either v or w gives the same potential $\Psi(s)$. If this is the case, then we say that the link is consistent.

Consistent link Consider a potential representation with neighbouring clusters v and w, sharing the variables s in common. If our aim is that the JTA modifies the potentials such that the marginal of the distribution $p(w)$ is given by the (modified) potential $\Psi(w)$, then[1]

$$p(s) = \sum_{w \setminus s} \Psi(w)$$

Similarly,

$$p(s) = \sum_{v \setminus s} \Psi(v)$$

This then requires

$$\sum_{w \setminus s} \Psi(w) = p(s) = \sum_{v \setminus s} \Psi(v),$$

We identify the (modified) separator potential $\Psi(s) = p(s)$.

Imagine that by some process, we have managed to achiee consistency, but that now some new evidence changes $\Psi(v)$ to $\Psi^*(v)$ (this is achieved by clamping one of the variables in v to a particular state). In order that the link remains consistent, we need to change $\Psi(w)$ and $\Psi(s)$ in order to satisfy

$$\sum_{w \setminus s} \Psi^*(w) = \Psi^*(s) = \sum_{v \setminus s} \Psi^*(v).$$

Absorption Let v and w be neighbours in a cluster tree, let s be their separator, and let $\Psi^*(v)$, $\Psi(w)$ and $\Psi(s)$ be their potentials. Absorption replaces the tables $\Psi^*(s)$ and $\Psi^*(w)$ with

$$\Psi^*(s) = \sum_{v \setminus s} \Psi(v)$$

$$\Psi^*(w) = \Psi(w)\,\frac{\Psi^*(s)}{\Psi(s)}$$

[1] Note that, in the beginning, the assignment of the cluster potentials does not satisfy the consistency requirement. The aim is to find an algorithm that modifies them so that ultimately consistency is achieved.

The idea behind this definition is that, under the update of the table for v, the table for the separator s and neighbour w are updated such that the link remains consistent. To see this consider

$$\sum_{w\backslash s} \Psi^*(w) = \sum_{w\backslash s} \Psi(w) \frac{\Psi^*(s)}{\Psi(s)} = \frac{\Psi^*(s)}{\Psi^*(s)} \sum_{w\backslash s} \Psi(w) = \frac{\Psi^*(s)}{\Psi(s)} \Psi(s) = \Psi^*(s) = \sum_{v\backslash s} \Psi^*(v)$$

Note that if $\Psi(s)$ can be zero, then we need also $\Psi(w)$ to be zero when $\Psi(s)$ is zero for this procedure to be well defined. In this case, the potential takes the value unity at that point. (This requirement is on $\Psi(w)$ and not $\Psi^*(s)$ since we are considering whether or not it is possible to transmit the information through the current state of the link). We say that a link is supportive if it allows absorption in both directions (that is $\Psi(v)$ and $\Psi(w)$ will both be zero when $\Psi(s)$ is zero). Note that supportiveness is preserved under absorption.

Invariance of Cluster Tree under Absorption

Let T be a supportive cluster tree. Then the product of all cluster potentials divided by the product of all separator potentials is invariant under absorption.

Proof:

When w absorbs v though the separator s only the potentials of w and s are changed. It is enough therefore to show that the fraction of the w and s tables is unchanged. We have

$$\frac{\Psi^*(w)}{\Psi^*(s)} = \frac{\Psi(w)\frac{\Psi^*(s)}{\Psi(s)}}{\Psi^*(s)} = \frac{\Psi(w)}{\Psi(s)}$$

So that if we start with a BN over U, construct a corresponding cluster tree T, and then perform a series of absorptions, then T remains a representation of $p(U)$ and is given by the product of all cluster potentials divided by the product of all separator potentials.

Message Passing in cluster trees

In the above, we think of absorption as passing a message from v to w via the separator s. In general, however, s will be connected to more than one cluster v. This motivates the following message passing scheme (c.f belief propagation).

Message Passing

A node v can send exactly one message to a neighbour w, and it may only be sent when v has received a message from each of its other neighbours.

Sufficiency of message passing

Let T be a supportive cluster tree, and suppose that messages are passed according to the message passing scheme. Then

After a message has been passed in both directions along each link:

- there will be no further changes in the potentials (the algorithm converges).
- Each link in the tree T is consistent.

1.2 Junction Trees

Let T be a cluster tree over U and let a be a variable in U and suppose that a is an element of the nodes v and w. If T is consistent, we would expect that $\sum_{v\backslash a} \Psi(v) = \sum_{w\backslash a} \Psi(w)$. Certainly this is true it v and w are neighbours, but otherwise there is no guarantee.

Global Consistency

We say that a consistent cluster tree is globally consistent if for any nodes v and w with intersection I we have

$$\sum_{v\backslash I} \Psi(v) = \sum_{w\backslash I} \Psi(w)$$

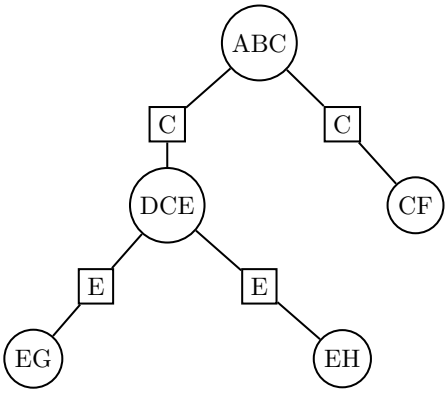


Figure 3: A junction tree

Junction Tree
Running Intersection

A cluster tree is a junction tree if, for each pair of nodes, v and w, all nodes on the path between v and w contain the intersection $v \cap w$. This is also called the running intersection property.

Local=Global consistency

From this definition, it is clear that, in a consistent junction tree, the local consistency will be passed on to any neighbours. That is, A consistent junction tree is globally consistent.

Marginals

Let T be a consistent junction tree over U, and let $\Psi(U)$ be the product of all potentials divided by the product of all separator potentials. Let v be a node with potential $\Psi(v)$. Then

$$\Psi(v) = \sum_{U \setminus v} \Psi(U)$$

To gain some intuition about the meaning of this theorem, consider the junction tree in fig(3). After a full round of message passing on this tree, each link is consistent, and the product of the potentials divided by the product of the separator potentials is just the BN itself. Imagine that we are interested in calculating the marginal for the node ABC. That requires summing over all the other variables, D, E, F, G, H. If we consider summing over H then, because the link is consistent,

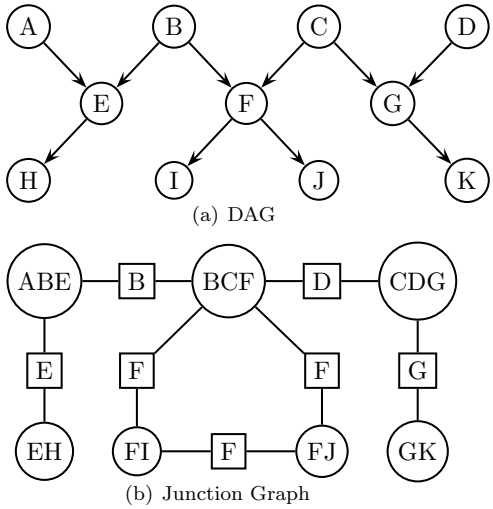
$$\sum_h \Psi(eh) = \Psi(e)$$

so that the ratio $\sum_h \frac{\Psi(eh)}{\Psi(e)}$ is unity, so that the effect of summing over node H is that the link between EH and DCE can be removed, along with the separator. The same happens for the link between node EG and DCE, and also for CF to ABC. The only nodes remaining are now DCE and ABC and their separator C, which have so far been unaffected by the summations. We still need to sum out over D and E. Again, because the link is consistent,

$$\sum_{de} \Psi(d, c, e) = \Psi(c)$$

so that the ratio $\sum_{de} \frac{\Psi(d,c,e)}{\Psi(c)} = 1$. The result of the summation of all variables not in ABC therefore produces unity for the cliques and their separators, and the summed potential representation reduces simply to the potential $\Psi(a, b, c)$ which is the marginal $p(a, b, c)$. It is clear that a similar effect will happen for other nodes. Formally, one can prove this using induction.

Because the BN is given by the product of node potentials divided by the product of separator potentials, then this means that the marginal for the node (group or clique of variables) is given by the potential in the globally consistent junction tree. That is, Let $p(U)$ be a BN, and



(a) DAG

(b) Junction Graph

Figure 4: (a) A singly connected graph and (b) its junction graph. By removing any of the links in (b) with separator F you get a junction tree.

let T be a corresponding junction tree. After a full round of message passing in T, we have for each node v and each separator s that

$$\Psi(v) = \sum_{U \setminus v} p(U) = p(v) \quad \text{and} \quad \Psi(s) = p(s)$$

We can then obtain the marginals for individual variables by simple brute force summation over the other variables in that potential. In the case that the number of variables in each node is small, this will not give rise to any computational difficulties. However, since the complexity is exponential in the clique size of the Junction Tree, it is prudent to construct the Junction Tree to have minimal clique sizes. Although, for a general graph, this is itself computationally difficult, there exist efficient heuristics for this task.

1.3 Constructing Junction Trees

Singly Connected DAGs

For each variable a with parents, form the cluster $\text{pa}(a) \cup a$. Between any two clusters with a non-empty intersection add a link with the intersection as the separator. The resulting graph is called a junction graph. All separators consist of a single variable, and if the junction graph contains cycles, then all separators on the cycle contain the same variable. Therefore any of the links can be removed to break the cycle, and by removing links until you have a tree, you get a junction tree.

Consider the graph in fig(4)a. Following the above procedure, we get the junction graph fig(4)b. By breaking the cycle any where in the loop $BCF--F--FI--F--FJ--F--BCF$, we obtain a junction tree.

Moral Graph We know that when we construct a cluster tree corresponding to a DAG, then for all variables a there must be a cluster v containing $\text{pa}(a) \cup a$. We can illustrate this on a graph by having a link between any pair of variables which must appear in the same cluster. This means that we take the DAG, add a link between any pair of variables with a common child, and drop the direction of the original links. The resulting graph is the moral graph. From the moral graph you can find the clusters, namely the cliques in the graph.

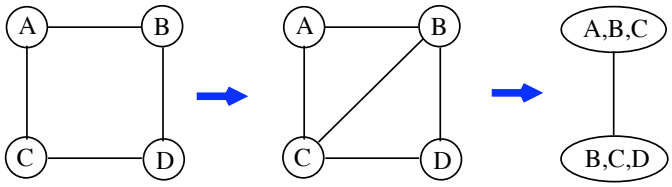


Figure 5: If we were to form a clique graph from the graph on the left, this would not satisfy the running intersection property, namely that if a node appears in two cliques, it appears everywhere on the path between the cliques. By introducing the extra link (middle picture), this forms larger cliques, of size three. The resulting clique graph (right) does satisfy the running intersection property. Hence it is clear that loops of length four or more certainly require the addition of such chordal links to ensure the running intersection property in the resulting clique graph. It turns out that adding a chord in for all loops of length four or more is sufficient to ensure the running intersection property for any resulting clique graph.

Coping with Cycles

The previous section showed how to construct a JT for a singly-connected graph. If we attempt to do this for a multiply connected (loopy) graph, we find that the above procedure generally does not work since the resulting graph will not necessarily satisfy the running intersection property. The idea is to grow larger clusters, such that these the resulting graph does satisfy the running intersection property. Clearly, a trivial solution would be to include all the variables in the graph in one cluster, and this will complete our requirements. However, of course, this does not help in finding an efficient algorithm for computing marginals. What we need is a sufficient approach that will guarantee that we can always form a junction tree from the resulting junction graph. This operation is called triangulation, and it generally increases the minimum clique size, sometimes substantially. The intuition for why triangulation is needed can be gained from considering a simple four variable loop as in fig(5), where the act of including a link makes the resulting graph satisfy the running intersection property. Formally, we need define the triangularisation operations as follows:

Chord This is a link joining two non-consecutive vertices of a loop.

Triangulation An undirected graph is triangulated if every cycle/loop of length 4 or more has a chord.

The importance of this definition derives from the following theorem.

Triangulated= ∃ Junction Tree An undirected graph is triangulated if and only if its junction graph has a junction tree.

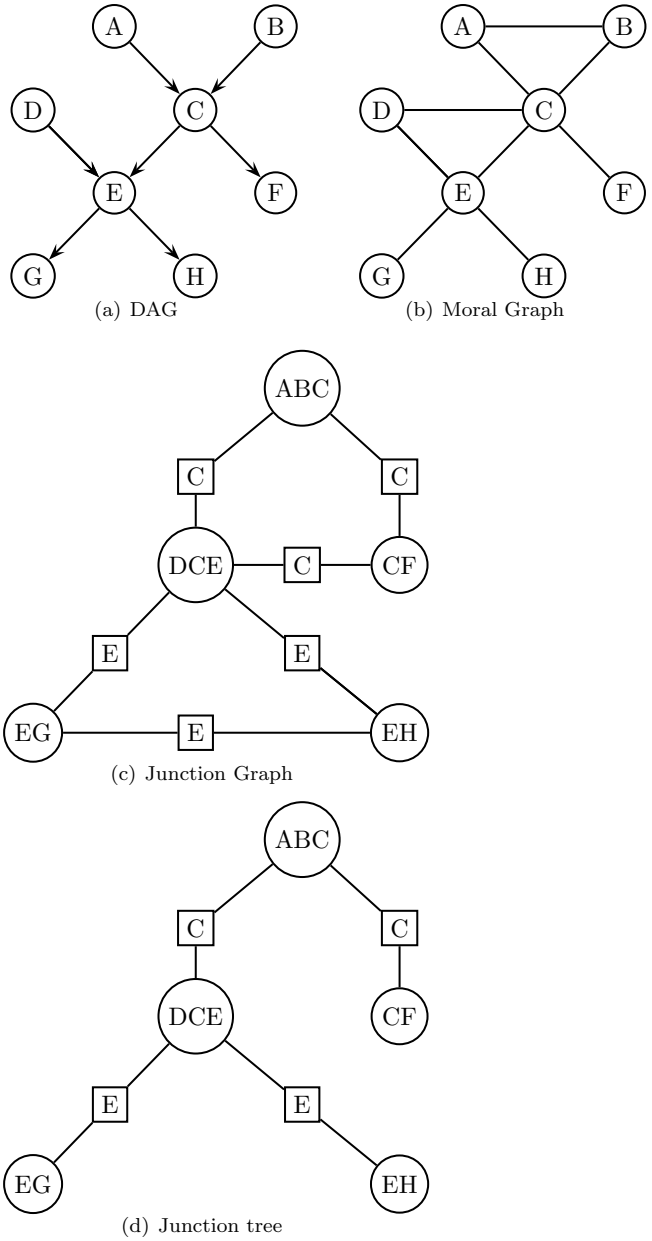
This therefore defines a method for constructing a junction tree.

- Moralise the graph. (That is, add links between parents of a common child).
- Triangulate the graph. (That is, for every loop of length four or more, ensure that the loop has a chord.)
- Assign the potentials accordingly.

The Junction Tree Algorithm

From the above, we essentially have all we need.

We need to do the following steps : moralisation and triangulation. We are then guaranteed to be able to form a Junction Tree. Assign

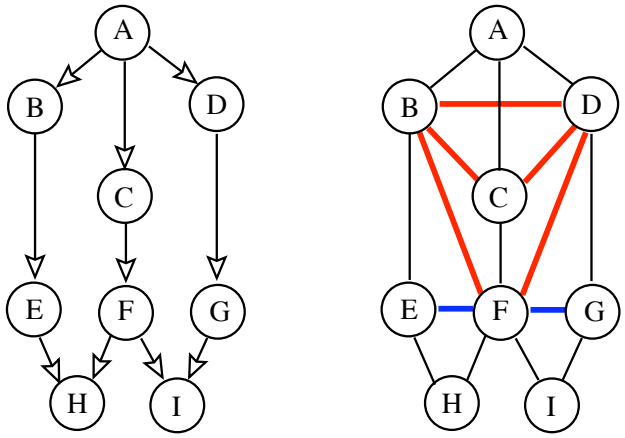


(a) DAG (b) Moral Graph

(c) Junction Graph

(d) Junction tree

Figure 6: Construction of a junction tree for a singly connected DAG.



(a) Original Graph

(b) Moralised and Triangulated

Figure 7: Example of the JTA. In (a) is the original loopy graph. (b) The moralisation links are between nodes E and F and between nodes F and G. The other additional links come from triangularisation. The clique size of the resulting clique tree (not shown) is four.

the potentials to the cliques on the Junction Tree and assign the separator potentials on the JT to unity. Then carry out the absorption procedure until updates have been passed along both directions of every link on the JT. Then the clique marginals can be read off from the JT. An example is given in fig(7).

There are some interesting points about the JTA. It provides an upper bound on the computation required to calculate marginals in the graph. This means that there may indeed exist more efficient algorithms in particular cases. There are, in general, many different ways to carry out the triangularisation step. Ideally, we would like to find a triangularised graph which has minimal clique size. However, it can be shown to be a hard-computation problem (NP-hard) to find the most efficient triangularisation. In practice, the triangularisation algorithms used are somewhat heuristic, and chosen to provide reasonable, but clearly not optimal, performance.

The precise algorithms for carry out triangulation and forming a JT are not detailed here, and the interested reader should consult the Jensen book for details.

The junction tree algorithm can be applied also to undirected graphical models since they naturally have an undirected graph representation. In that case, there is no need to carry out the intermediate moralisation stage.

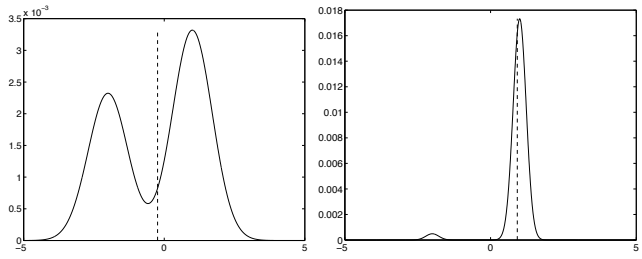
1.4 Finding the Most Likely State

Previously, we have mainly concentrated on finding marginal probabilities, $p(x_i)$. However, another natural question to ask is what is the most likely state of the distribution? There is a simple trick which will enable us to convert the JTA to enable us to answer this.

In general, a probability distribution may be written as

$$p = \frac{1}{Z} \prod_c \phi(x_c)$$

where $\phi(x_c)$ is the potential for cluster c. Consider a modified distribution in which we wish to re-weight the states, making the higher



(a) $p(x)$ (b) $p^*(x)$

Figure 8: $p*(x) \propto (p(x)^{10}$. In both figures the vertical dashed line indicates (on the x-axis the mean value for x. Note how p^* becomes much more peaked around its most probable valuem, and how the mean value in p^* shifts to be close to the most likely value. In the limit $p^*(x) \propto (p(x))^\beta$,$\beta \to \infty$, then the mean of the distribution p^* tends to the most-likely value.

probability states exponentially more likely than lower probability states. This can be achieved by defining

$$p^* = \frac{1}{Z^\beta} \prod_{x_c} \phi^\beta(x_c)$$

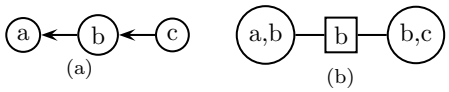
where β is a very large positive quantity. This makes the distribution p^* very peaked around the most-likely value of p, see fig(8).

In the JTA, we need to carry out summations over states. However, in the limit $\beta \to \infty$ it is clear that only the most-likely state will contribute, and hence that the summation operation can be replaced by a maximisation operation in the definition of absorption. The algorithm thus proceeds as normal, replacing the summations with maximisations, until the final stage, whereby from the table one reads off $\underset{x_c}{\text{argmax}}\, \phi(x_c)$ for the variables in the modified final potential on cluster c to find the most likely state.

A simple example of the JTA

Consider running the JTA on the simple graph

$$p(a, b, c) = p(a|b)p(b|c)p(c)$$



(a) (b)

Figure 9: (a) A belief network. (b) JTA for the network.

There are three questions we are interested in (i) What is p(b)? (ii) What is $p(b|a = 1, c = 1)$ (iii) What is the likelihood of the evidence $p(a = 1, c = 1)$

For this simple graph, the moralisation and triangularisation steps are trivial, and the JTA is given immediately by fig(9b). A valid assignment is $\Psi(a, b) = p(a|b)$, $\Psi(b) = 1$, $\Psi(b, c) = p(b|c)p(c)$.

First let's absorb from (a, b) through the separator b to (b, c):

Finding a marginal $p(b)$

First we just run the JTA as usual.

- The new separator is given by $\Psi^*(b) = \sum_a \Psi(a,b) = \sum_a p(a|b) = 1$.
- The new potential on (b,c) is given by $\Psi^*(b,c) = \frac{\Psi(b,c)\Psi^*(b)}{\Psi(b)} = \frac{p(b|c)p(b)\times 1}{1}$.
- The new separator is given by $\Psi^{**}(b) = \sum_c \Psi^*(b,c) = \sum_c p(b|c)p(c)$.
- The new potential on (a,b) is given by $\Psi^*(a,b) = \frac{\Psi(a,b)\Psi^{**}(b)}{\Psi^*(b)} = \frac{p(a|b)\sum_c p(b|c)p(c)}{1}$. This is therefore indeed equal to the marginal since $\sum_c p(a,b,c) = p(a,b)$.

Also, the new separator $\Psi^{**}(b)$ contains the marginal $p(b)$ since $\Psi^{**}(b) = \sum_c p(b|c)p(c) = \sum_c p(b,c) = p(b)$.

Finding a conditional marginal $p(b|a=1,c=1)$

First we clamp the evidential variables in their states. Then we claim that the effect of running the JTA is to produce on the cliques, the joint marginals $p(a=1,b,c=1)$, $p(a=1,b,c=1)$ and $p(a=1,b,c=1)$ for the final potentials on the two cliques and their separator. We demonstrate this below:

- In general, the new separator is given by $\Psi^*(b) = \sum_a \Psi(a,b) = \sum_a p(a|b) = 1$. However, since a is clamped in state $a=1$, then the summation is not carried out over a, and we have instead $\Psi^*(b) = p(a=1|b)$.
- The new potential on the (b,c) clique is given by $\Psi^*(b,c) = \frac{\Psi(b,c)\Psi^*(b)}{\Psi(b)} = \frac{p(b|c=1)p(c=1)p(a=1|b)}{1}$.
- The new separator is normally given by $\Psi^{**}(b) = \sum_c \Psi^*(b,c) = \sum_c p(b|c)p(c)$. However, since c is clamped in state 1, we have instead $\Psi^{**}(b) = p(b|c=1)p(c=1)p(a=1|b)$
- The new potential on (a,b) is given by $\Psi^*(a,b) = \frac{\Psi(a,b)\Psi^{**}(b)}{\Psi^*(b)} = \frac{p(a=1|b)p(b|c=1)p(c=1)p(a=1|b)}{p(a=1|b)} = p(a=1|b)p(b|c=1)p(c=1)$.

Hence, here in this special case, all the cliques contain the joint distribution $p(a=1,b,c=1)$.

In general, the effect of clamping a set of variables V in their evidential states, and running the JTA is that, for a clique i which contains the set of non-evidential variables H^i, the potentials after the end of the JTA contains the marginal $p(H^i, V)$.

Then calculating the conditional marginal $p(b|a=1,c=1)$ is a simple matter since $p(b|a=1,c=1) \propto p(a=1,b,c=1)$, where the proportionality is determined by the normalisation constraint.

Finding the likelihood $p(a=1,c=1)$

By the above procedure, the effect of clamping the variables in their evidential states and running the JTA produces the joint marginals, such as $\Psi^*(a,b) = p(a=1,b,c=1)$. Then calculating the likelihood is easy since we just sum out over the non-evidential variables of any converged potential : $p(a=1,c=1) = \sum_b \Psi^*(a,b) = \sum_b p(a=1,b,c=1)$.

Whilst we have demonstrated these results only on such a simple graph, the same story holds in the general case. Hence calculating conditional marginals and likelihoods can be obtained in exactly the same way. The main thing to remember is that clamping the variables in evidential states means that the *joint* distribution on the non-evidential variables in a clique with all the evidental variables clamped

in their evidential states is what is found a the end of the JTA. From this conditionals and the likelihood are straightforward to calculate.