Lecture 19:

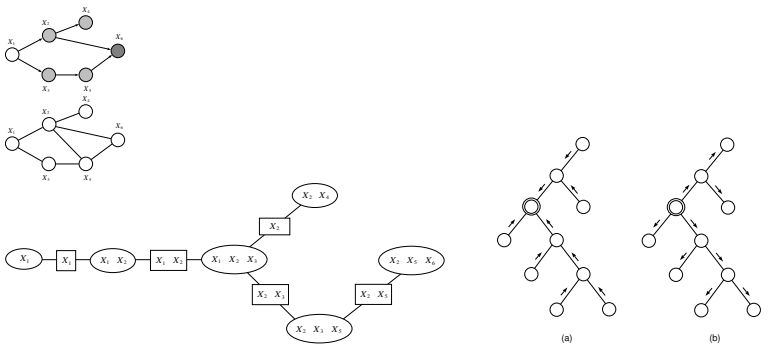
Junction Trees 2

Sam Roweis

March 17, 2003

The Main Setup

- Three main steps:
  1. Pre-processing (compiling) the original graphical model to prepare for inference: building the clique junction tree.
  2. Conditioning on the evidence.
  3. Marginalizing out the non-query nodes efficiently.



Junction Tree Algorithms

- We want to be able to condition on some "evidence" $\mathbf{x}_E$ (observed nodes) and compute the posterior probabilities of some "query" nodes $\mathbf{x}_F$ while marginalizing out "nuisance" nodes $\mathbf{x}_R$.
- We will focus on sets of nodes corresponding to *maximal cliques* of the original undirected or moralized graphical model.
- Q: Why maximal cliques?
  A: Because they are the sufficient statistics for learning with EM if the joint distribution factors according to our graphical model semantics.
- Junction tree algorithms are "tree-like" data structures and message passing systems to do efficient inference even when the underlying graph is not a tree.

Pre-Processing

The pre-processing step has several parts:

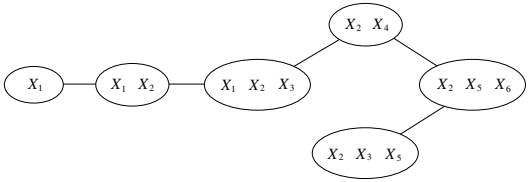
1. If the original graph is directed, *moralize* it. (easy)
2. Add edges until the graph is *triangulated*. (HARD!)
3. Identify the maximal cliques. (easy if triangulated).
4. Arrange the cliques into a *junction tree*. (today)
5. Associate a potential function with each clique in this hypergraph, and define the joint probability as the product of these potentials:

$$p(\mathbf{X}) = \frac{1}{Z} \prod_{\text{cliques } c} \psi_c(\mathbf{x}_c) \qquad Z = \sum_{\mathbf{X}} \prod_{\text{cliques } c} \psi_c(\mathbf{x}_c)$$

   very similar to setup for undirected models, except that the cliques here are maximal, whereas in undirected graphs they might not be.
6. Initialize the clique potentials based on the parameters of the underlying (directed or undirected) graphical model.

Reminder: Junction Tree Property

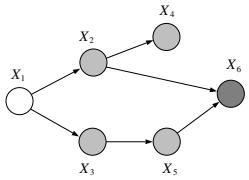
- Maximal cliques (post triangulation) are arranged into a clique tree.
  Q: Which tree should we pick for correct calculations?
  A: One in which local information transfer is sufficient to update all necessary quantities consistently.
- KEY IDEA: not all clique trees are equal! If, for every variable, all occurances of the variable appear in a connected subtree, then the clique tree has the *junction tree* property, and this is very desirable.
- Formally: for every pair of cliques V, W all cliques on the unique path b/w V and W contain $V \cap W$.

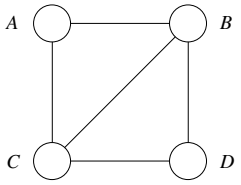


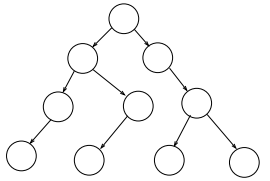
e.g. this clique tree does not have the j-tree property (x_3)

Triagulated $\Leftrightarrow$ ($\exists$ Junction Tree)

- Which graphs have junction trees?
- It turns out that having a junction tree is equivalent to another, well know graph property: *triangulation*.
- Triangulated $\equiv$ no undirected cycles of length > 3 containing only edges between successor nodes.
- Examples:

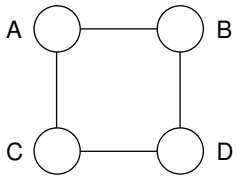


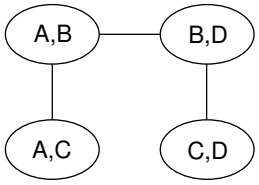




Not All Graphs Have Junction Trees

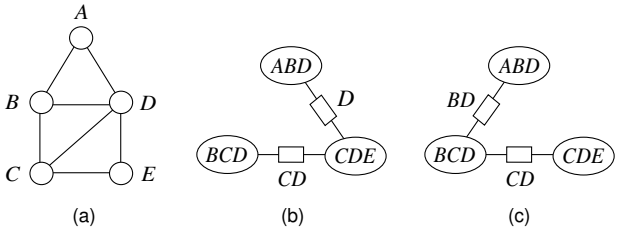
- Junction-tree inference algorithms will be based on maintaining local pariwise consistency between cliques in the clique tree.
  If the same variables appear in disconnected cliques their representations will not be guaranteed to be correct.
- Consider the graph below, with 4 maximal cliques.
- No clique tree on these cliques has the junction tree-property.
- The elimination algorithm, with any ordering, would have created at least one extra link.





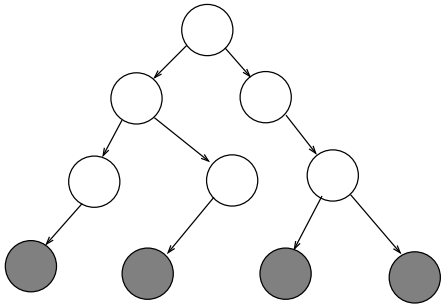
Constructing the Junction Tree: MWST

- Even for triangulated graphs, not every clique tree has the junction tree property.



- A junction tree is a *maximum weight spanning tree* where the weights are the *number of variables shared* by two cliques.
- We can construct a junction tree using Kruskal's or Prim's algorithms: at each step add an edge that has the most number of variables in common, unless this would create a cycle.

Example: Trees with Hidden Nodes



- No moralization or triagulation necessary.
- Cliques are pairs of nodes.
- Potentials ψ_C initialized to parent-conditionals.
- Evidence will flow from leaves to root and then marginals will be corrected on the way back down.

Conditioning (introducing evidence)

- Conditioning is like "generalized indexing" or "slicing".
- Product of potentials after conditioning must be $\propto p(\mathbf{x}_F|\mathbf{x}_E)$.
- "Evidence potential" approach: initialize any cliques containing evidence nodes using $\psi_C(x_C) = \psi_C(x_C)\delta(x_{EC}, \bar{\mathbf{x}}_{EC})$.
- This is correct, but gives us lots of zeros in our clique potentials and thus wastes storage and computation (when we sum).
- Another approach: keep only cliques and potentials that reference non-evidence nodes, using the correct "slice" of the original potentials to initialize them:

$$p(\mathbf{x}_F, \mathbf{x}_R|\mathbf{x}_E) \propto \prod_{C \in F,R} \psi_C(\mathbf{x}_{C \cap F,R}, \bar{\mathbf{x}}_{C \cap E})$$

- This works, except that the product over these reduced potentials will no longer be normalized.

Manipulations on Clique Potentials

- Initialization: the clique potentials must be initialized so that they represent the correct joint distribution.
  We use the potentials or conditionals from the underlying graph, assigning ambiguous potentials to one and only one clique.
- Conditioning on evidence: we must modify the initial potentials to indicate the values for quantites have been observed.
- Marginalizing: efficiently sum (integrate) over non-evidence and non-query nodes to return a conditional marginal over query nodes given observed evidence. We will do this by passing messages on the clique tree. Messages will have form of potentials.
- After conditioning and marginalization, the potential remaining on any clique node C will be equal to the marginal probability $p(x_C, x_E)$, which can be converted to the "answer" $p(x_C|x_E)$ via local normalization.

Reduced Potential Set Method

- Assume $\mathbf{x}_F$ empty for now: evidence $\mathbf{x}_E$; non-evidnce $\mathbf{x}_R$.
- For every clique potential, consider the subset of its variables on which we are conditioning and take a "slice" of the potential function by fixing those variables to their observed values.
- Thus, each clique becomes a function of the variables in the intersection of its original variables and the non-evidence variables.
- Now take the product of these "reduced" clique potentials and we get an unnormalized quantity proportional to $p(\mathbf{x}_R, \bar{\mathbf{x}}_E)$:

$$p(\mathbf{x}_R, \bar{\mathbf{x}}_E) = \frac{1}{Z} \prod_{C \in R} \psi_C(\mathbf{x}_{C \cap R}, \bar{\mathbf{x}}_{C \cap E})$$

- The normalizer $\bar{Z}$ for just these sliced potentials is now the evidence probability $p(\bar{\mathbf{x}}_E)$. The conditional we want is $p(\mathbf{x}_R|\bar{\mathbf{x}}_E) = p(\mathbf{x}_R, \bar{\mathbf{x}}_E)/p(\bar{\mathbf{x}}_E) = p(\mathbf{x}_R, \bar{\mathbf{x}}_E)/\bar{Z}$

Example: two node binary model

$$
\begin{aligned}
p(A,B) &= p(A)p(B|A) \\
p(A=1) &= 0.8 \\
p(B=1|A=1) &= 0.7 \\
p(B=1|A=0) &= 0.4 \\
\psi(A,B) &= \begin{bmatrix} .12 & .08 \\ .24 & .56 \end{bmatrix} \qquad Z = 1
\end{aligned}
$$

Now, if we observe $B = 1$:

$$
\begin{aligned}
\bar{\psi}(A) &= [.08; .56] \\
\bar{Z} &= .64 = p(B=1) \\
p(A|B=1) &= \frac{1}{\bar{Z}}\bar{\psi}(A) = [.125; .875]
\end{aligned}
$$

Next time: marginalization

- Now we know how to initialze potentials and condition on evidence so that the product of potentials is always equal to the joint probability or the conditional probability given evidence.
- Next step: efficiently sum over $\mathbf{x}_R$ to get conditional marginal $p(\mathbf{x}_F|\mathbf{x}_E)$. Done by passing messages on the clique tree that modify the potentials at each node until they are correct.
- After conditioning and marginalization, the potential remaining on any clique node C will be equal to the marginal probability $p(x_C, x_E)$, which can be converted to the "answer" $p(x_C|x_E)$ via local normalization.
- Generally we will only compute single clique marginals on the clique tree since these "single nodes" (cliques) are sufficient statistics for learning. But it is possible to have multiple cliques as query and compute pairwise (or higher) clique marginals.