Lecture 17:

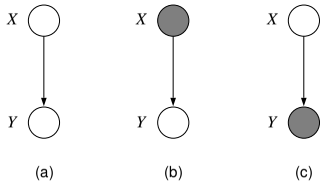
Junction Trees

Sam Roweis

March 13, 2002

Simple Case: Bayes Rule

- For simple models, we can derive the inference formulas by hand using Bayes rule (e.g. responsbility in mixture models).



$$a) \quad p(x,y) = p(x)p(y|x)$$
$$b) \quad p(y|x)$$
$$c) \quad p(x|y) = \frac{p(x)p(y|x)}{\sum_x p(x)p(y|x)}$$

This is called "reversing the arrow".

- In general, the calculation we want to do is:

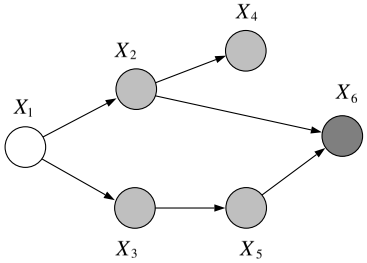
$$p(\mathbf{x}_F|\mathbf{x}_E) = \frac{\sum_{\mathbf{x}_H} p(\mathbf{x}_E, \mathbf{x}_F, \mathbf{x}_H)}{\sum_{\mathbf{x}_E, \mathbf{x}_H} p(\mathbf{x}_E, \mathbf{x}_F, \mathbf{x}_H)}$$

- Can we do these sums efficiently? Can we avoid repeating unecessary work each time we do inference?
  Yes, if we exploit the factorization of the joint distribution.

Reminder: Probabilistic Inference

- Partition the random variables in a domain $\mathbf{X}$ into three disjoint subsets, $\mathbf{x}_E$,$\mathbf{x}_F$,$\mathbf{x}_H$. The general *probabilistic inference* problem is to compute the posterior $p(\mathbf{x}_F|\mathbf{x}_E)$ over *query nodes* $\mathbf{x}_F$.
- This involves *conditioning* on *evidence nodes* $\mathbf{x}_E$ and *integrating out marginal nodes* $\mathbf{x}_H$.
- If the joint distribution is represented as a huge table, this is trivial: just select the appropriate indicies in the columns corresponding to $\mathbf{x}_E$ based on the values, sum over the columns corresponding to $\mathbf{x}_H$, and renormalize the resulting table over $\mathbf{x}_F$.
- If the joint is a known continuous function this can often be done as well, using calculus. (e.g. Gaussian: eliminate rows/cols corresponding to $\mathbf{x}_H$; apply conditioning formulas for $p(\mathbf{x}_F|\mathbf{x}_E)$).
- But what if the joint distribution is represented by a directed or undirected graphical model?

Example



The key is to factor and then apply the distributive law.

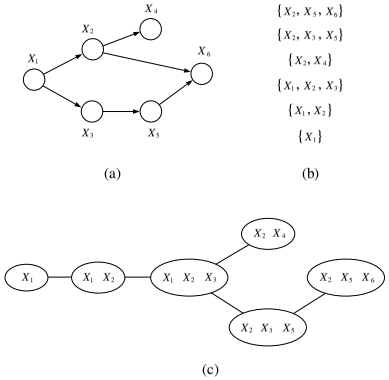
$$\begin{aligned}
p(\mathbf{x}_1, \bar{\mathbf{x}}_6) &= \sum_{\mathbf{x}_2}\sum_{\mathbf{x}_3}\sum_{\mathbf{x}_4}\sum_{\mathbf{x}_5} p(\mathbf{x}_1)p(\mathbf{x}_2|\mathbf{x}_1)p(\mathbf{x}_3|\mathbf{x}_1)p(\mathbf{x}_4|\mathbf{x}_2)p(\mathbf{x}_5|\mathbf{x}_3)p(\bar{\mathbf{x}}_6|\mathbf{x}_2,\mathbf{x}_3) \\
&= p(\mathbf{x}_1)\sum_{\mathbf{x}_2} p(\mathbf{x}_2|\mathbf{x}_1)\sum_{\mathbf{x}_3} p(\mathbf{x}_3|\mathbf{x}_1)\sum_{\mathbf{x}_4} p(\mathbf{x}_4|\mathbf{x}_2)\sum_{\mathbf{x}_5} p(\mathbf{x}_5|\mathbf{x}_3)p(\bar{\mathbf{x}}_6|\mathbf{x}_2,\mathbf{x}_3) \\
&= p(\mathbf{x}_1)\sum_{\mathbf{x}_2} p(\mathbf{x}_2|\mathbf{x}_1)\sum_{\mathbf{x}_3} p(\mathbf{x}_3|\mathbf{x}_1)\Phi_5(\mathbf{x}_2,\mathbf{x}_3)\sum_{\mathbf{x}_4} p(\mathbf{x}_4|\mathbf{x}_2) \\
&= p(\mathbf{x}_1)\sum_{\mathbf{x}_2} p(\mathbf{x}_2|\mathbf{x}_1)\Phi_4(\mathbf{x}_2)\sum_{\mathbf{x}_3} p(\mathbf{x}_3|\mathbf{x}_1)\Phi_5(\mathbf{x}_2,\mathbf{x}_3) \\
&= p(\mathbf{x}_1)\sum_{\mathbf{x}_2} p(\mathbf{x}_2|\mathbf{x}_1)\Phi_4(\mathbf{x}_2)\Phi_3(\mathbf{x}_1,\mathbf{x}_2) \\
&= p(\mathbf{x}_1)\Phi_2(\mathbf{x}_1)
\end{aligned}$$

Reminder: Elimination

- For a single node posterior (i.e. $\mathbf{x}_F$ is a single node), there was a simple, efficient algorithm based on eliminating nodes.
- Notation: $\bar{\mathbf{x}}_i$ is the value of evidence node $\mathbf{x}_i$.
- The algorithm, called *elimination*, required a *node ordering* to be given, which told it which order to do the summations in.

Elimination Cliques

- During a run, Elimination removes nodes by summing them out and then couples their remaining neighbours.
- Sets of nodes that get coupled at any point during the run are called *elimination cliques*.



Multiple Queries

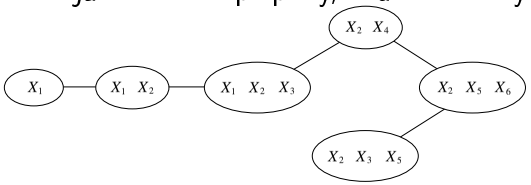
- The Elimination algorithm we described was query based: given the single node marginal to compute (the last item in the ordering), it efficiently summed out or conditioned on all other variables.
- But what if we want to do multiple inferences?
  We do not want to duplicate work unecessarily.
  We want to reuse calculations as best as possible.
  e.g. in HMM inference we had an efficient recursion for computing *all* the posteriors $p(x_t|Y)$ at once.
- This needs:
  1) A plan for which intermediate factors to compute in what order.
  2) Some storage for these intermediate factors.

Elimination Clique Trees

- As we sum out or condition on variables, we need to pass information between these storage sites (elimination cliques).
- Elimination essentialy creates a tree of cliques during each run based on the elimination ordering.
- Information flows only along one path: from the clique created by eliminating the first variable in the order to the last.
- Our goal is to create a more flexible "hypergraph" connecting the cliques so that information can flow in many ways.
- What is one step up from a chain? A tree!
  In fact, we will be creating elimination clique trees.
  Information will flow in both directions across the links.

Junction Tree Property

- Imagine arranging the elimination cliques into a tree structure.
  Q: Which structure should we pick for efficient calculations?
  A: One in which local information transfer is sufficient to update all necessary quantities.
- 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.



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

The Setup

Here is the setup for our general inference algorithm so far:

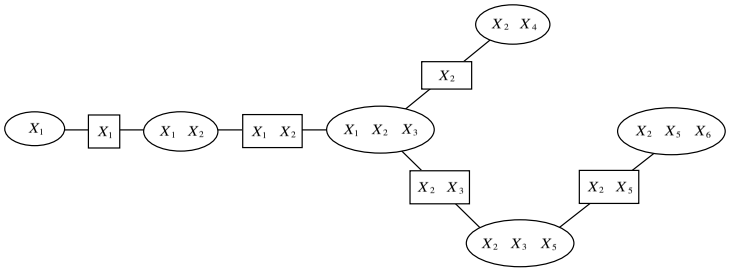
1. Start with a graph $G(V, E)$ and identify its maximal cliques C. (For certain graphs this is possible.) These correspond to the elimination cliques we had previously.
2. Arrange the cliques into a tree that has the *junction tree* property.
3. Associate a potential function with each clique in this 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.
4. Initialize these potentials so that they represent the correct model. Then try to condition and marginalize to perform inference.

Separator Sets

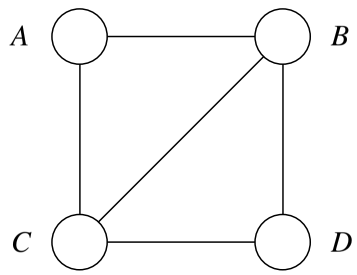
- We can explicitly show what information will be communicated from clique to clique in a clique tree by taking the intersection of variables in adjacent cliques.
- These intersections are called *separator sets* and are themselves cliques (fully connected in the underlying graph) although of course they are no longer maximal.



Initializing Potentials

- How do we initialize the potentials of the clique tree (hypergraph) based on the original graphical model so that the distribution is correctly represented?
- Easiest case: original graphical model is undirected, and its cliques correspond to the maximal cliques.
  Then we just assign potentials in a one-to-one fashion.
- Next easiest case: original model is undirected, but with non-maximal cliques.
  1. Assign any potential completely contained within only a single maximal clique to that clique.
  2. Assign potentials that are contained by more than one clique arbitrarily to exactly one of them.
     Multiple ways to do this.

Initializing from Undirected Graphs



ψ_{AB}
ψ_{AC}
ψ_{BC}
ψ_{BD}
ψ_{CD}

- Two choices:

$\psi_{ABC} = \psi_{AB}\psi_{AC}$
$\psi_{BCD} = \psi_{BC}\psi_{BD}\psi_{CD}$

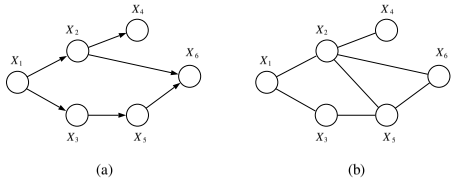
OR

$\psi_{ABC} = \psi_{AB}\psi_{AC}\psi_{BC}$
$\psi_{BCD} = \psi_{BD}\psi_{CD}$

- Each underlying undirected graph potential gets assigned to *one and only one* clique potential in the junction tree.

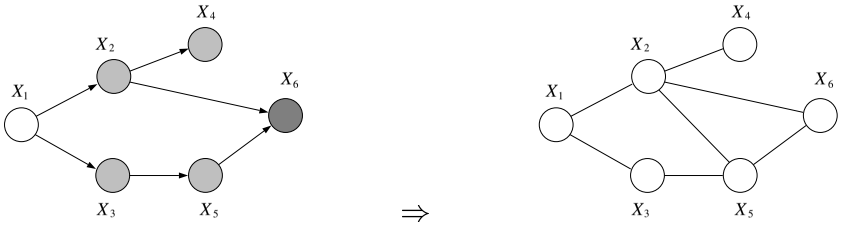
Moralization

- Now the conditionals are guaranteed to be functions on cliques and we proceed as above (plus these will be automatically normalized).
- Caution: moralization *adds* edges to the graph and converts it to an undirected model. So it changes the semantics.
- But all original dependencies are still possible. We just may have lost some conditional independencies.
- Moralization is essential: since conditioning couples parents in directed models ("explaining away") we need a mechanism for recording this when we do inference.



Initializing from Directed Graphs: Moralization

- How do we initialize from an underlying directed graph?
- Same idea, but with a slight complication: for directed graphs, the parents may not be explicitly connected, but they are involved in the same "potential" function $p(\mathbf{x}_i|\mathbf{x}_{\pi_i})$,
  (i.e. parental conditionals can have sets of arguments that are not contained in *any* maximal clique if the parents are not connected.)
- To handle this we *moralize* the graph:
  link parents and drop directions of edges.



The Plan

Summary so far:

1. Identify maximal cliques of underlying graph. (possible)
2. Arrange cliques into junction tree.
3. Associate potentials with each clique.
4. Initialize (moralizing first if necessary) by assigning potentials or conditionals from the underlying graphical model to clique potentials in the junction tree.
5. Coming up next: introducing evidence (conditioning).
6. And then: marginalizing.

The plan from here onwards:

- We will break the general inference problem into two parts:
  1. Condition on what we observe (evidence).
  2. Then marginalize out anything we need to (sum).