CSC412 – Probabilistic Learning & Reasoning Sam Roweis

Lecture 14:

Elimination Algorithm

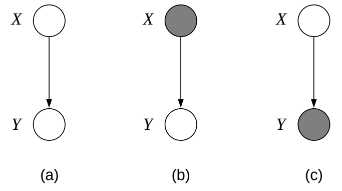
March 1, 2006

Probabilistic Inference

- Partition the random variables in a domain $\mathbf{X}$ into three disjoint subsets, $\mathbf{x}_E$,$\mathbf{x}_F$,$\mathbf{x}_R$. 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 (summing) out marginal nodes* $\mathbf{x}_R$.
- 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 $\mathbf{x}_R$, and renormalize the resulting table over $\mathbf{x}_F$.
- If the joint is a known continuous function this can sometimes be done analytically. (e.g. Gaussian: eliminate rows/cols corresponding to $\mathbf{x}_R$; apply conditioning formulas for $p(\mathbf{x}_F|\mathbf{x}_E)$).
- But what if the joint distribution over $\mathbf{X}$ is represented by a directed or undirected graphical model?

Simple Case: Bayes Rule

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



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

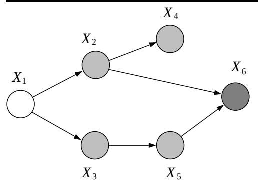
This is called "reversing the arrow".

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

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

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

Example



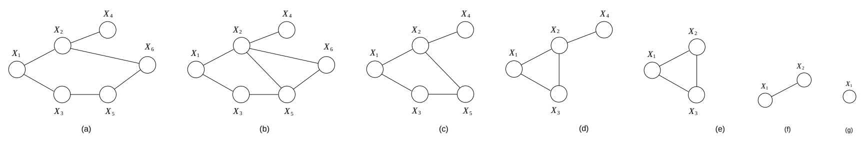
The key is to factor and then apply the distributive law.

$$\begin{aligned} p(\mathbf{x}_1|\bar{\mathbf{x}}_6) &= p(\mathbf{x}_1,\bar{\mathbf{x}}_6)/p(\bar{\mathbf{x}}_6) \\ &= p(\mathbf{x}_1,\bar{\mathbf{x}}_6)/\sum_{\mathbf{x}_1{}'} p(\mathbf{x}_1{}',\bar{\mathbf{x}}_6) \end{aligned}$$

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

Single Node Posteriors

- For a single node posterior (i.e. $\mathbf{x}_F$ is a single node), there is a simple, efficient algorithm based on eliminating nodes.
- Notation: $\bar{x}_i$ is the value of evidence node x_i.
- The algorithm, called *elimination*, requires a *node ordering* to be given, which tells it which order to do the summations in.
- In this ordering, the query node must appear last.
  Graphically, we'll remove a node from the graph once we sum it out.



Evidence Potentials

- Elimination also uses a bookeeping trick, called *evidential functions*:
$$g(\bar{x}_i) = \sum_{x_i} g(x_i)\delta(x_i, \bar{x}_i)$$
  where $\delta(x_i, \bar{x}_i)$ is 1 if $x_i = \bar{x}_i$ and 0 otherwise.
- This trick allows us to treat conditioning in the same way as we treat marginalization. So everything boils down to doing sums:
$$p(\mathbf{x}_F|\bar{\mathbf{x}}_E) = p(\mathbf{x}_F, \bar{\mathbf{x}}_E)/p(\bar{\mathbf{x}}_E)$$
$$p(\mathbf{x}_F, \bar{\mathbf{x}}_E) = \sum_{\mathbf{x}_R}\sum_{\mathbf{x}_E} p(\mathbf{x}_F, \mathbf{x}_E, \mathbf{x}_R)\delta(\mathbf{x}_E, \bar{\mathbf{x}}_E)$$
$$p(\bar{\mathbf{x}}_E) = \sum_{\mathbf{x}_R}\sum_{\mathbf{x}_E}\sum_{\mathbf{x}_F} p(\mathbf{x}_F, \mathbf{x}_E, \mathbf{x}_R)\delta(\mathbf{x}_E, \bar{\mathbf{x}}_E)$$
- We just pick an ordering and go for it...

Elimination Algorithm

```
ELIMINATE($\mathcal{G}, E, F$)
    INITIALIZE($\mathcal{G}, F$)
    EVIDENCE($E$)
    UPDATE($\mathcal{G}$)
    NORMALIZE($F$)

INITIALIZE($\mathcal{G}, F$)
    choose an ordering $I$ such that $F$ appears last
    for each node $X_i$ in $\mathcal{V}$
        place $p(x_i \,|\, x_{\pi_i})$ on the active list
    end

EVIDENCE($E$)
    for each $i$ in $E$
        place $\delta(x_i, \bar{x}_i)$ on the active list
    end

UPDATE($\mathcal{G}$)
    for each $i$ in $I$
        find all potentials from the active list that reference $x_i$ and remove them from the active list
        let $\phi_i(x_{T_i})$ denote the product of these potentials
        let $m_i(x_{S_i}) = \sum_{x_i} \phi_i(x_{T_i})$
        place $m_i(x_{S_i})$ on the active list
    end

NORMALIZE($F$)
    $p(x_F \,|\, \bar{x}_E) \leftarrow \phi_F(x_F) / \sum_{x_F} \phi_F(x_F)$
```

The ELIMINATE algorithm for probabilistic inference on directed graphs.

`Above,` T_i `denotes` i `plus all other nodes referenced by potentials on` i

Algorithm Details

- At each step we are trying to remove the current variable in the elimination ordering from the distribution.
  For marginal nodes this sums them out, for evidence nodes this conditions on their observed values using the evidential functions.
- Each step in UPDATE performs a sum over a product of potential functions. Potentials can be original functions $p(x_i|x_{\pi_i})$, evidential functions $\delta(x_i, \bar{x}_i)$ or intermediate potentials m_i.
- The algorithm terminates when we reach the query node, which always appears last in the ordering.
- We renormalize what we have left to get the final result:
$$p(\mathbf{x}_F|\mathbf{x}_E)$$
- For undirected models, everything is the same except the initialization phase uses the clique potentials instead of the parent-conditionals.

Marginalization without Evidence

- Marginalization of joint distributions represented by graphical models is a special case of probabilistic inference.
- To compute the marginal $p(x_i)$ of a single node, we set it to be the query node and set the evidence set to be empty.
- In directed models, we can ignore all nodes downstream from the query node, and marginalize only the part of the graph before it.
- If the node has no parents, we can read off its marginal directly.
- In undirected models, we need to do the full computation: compute $p(x_i)/Z$ using elmination and then normalize in the last step of elmination to get Z.
  (We can reuse Z later if we want to save work).

Efficiency Trick in Directed Elimination

- In directed models, we often know that a certain sum must evaluate to unity, since it is a conditional probability.
- For example, consider the term $\Phi_4(\mathbf{x}_2)$ in our six node example:

$$\Phi_4(\mathbf{x}_2) = \sum_{\mathbf{x}_4} p(\mathbf{x}_4|\mathbf{x}_2) \equiv 1$$

- We can't use this trick in undirected models, because there are no guarantees about what clique potentials sum to.

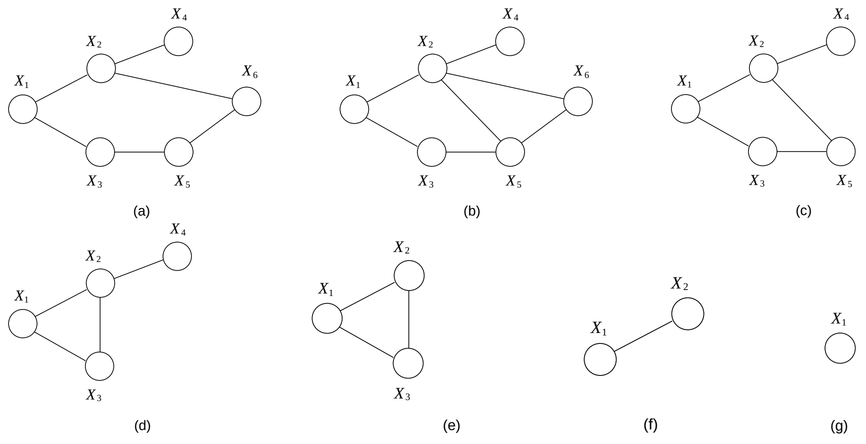
Node Elimination

- The algorithm we presented is really a way of eliminating nodes from a graph one by one. For undirected graphs:

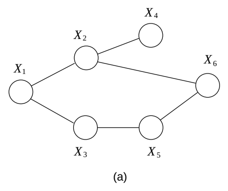
```
foreach node $x_i$ in ordering $I$:
    connect all the neighbours of $x_i$
    remove $x_i$ from the graph
end
```

- The removal operation requires summing out x_i (or conditioning on observed evidence for x_i).
- Summing out x_i leaves a function involving all its previous neighbours and thus they become connected by this step.
- The original graph, augmented by all the added edges is now a *triangulated* graph. (Reminder: triangulated means that every cycle of length >3 contains a chord, ie an edge not on the cycle but between two nodes in the cycle.)

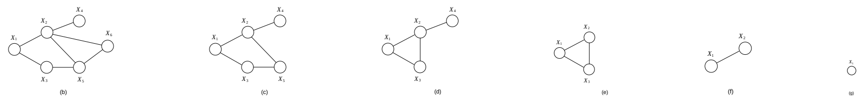
Example

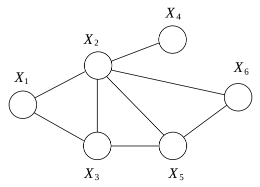


Added Edges == Triangulation



It is easy to check if a graph is triangulated in linear time.
It is easy to triangulate a non-triangulated graph.
But it is very hard to do so in a way that induces small clique sizes.





Moralization

- For directed graphs, the parents may not be explicitly connected, but they are involved in the same potential function $p(x_i|x_{\pi_i})$.
- Thus to think of ELIMINATION as a node removal algorithm, we first must connect all the parents of every node and the drop the directions on the links.
- This step is known as "Moralization".