iPMR

Recap

- What did we learn from the last lecture?

Summary

- Examples of Factor Graphs.
- Computing Conditional Independence from factor graphs.
- Motivating Inference in Factor Graphs

Inference in Factor Graphs

- Consider marginalisation and conditioning operations on a tree.
- Conditioning
  - Look at all neighbours. Replace factors at all neighbours to be conditional factors. This is called *absorbing.*
- Marginalising
  - Find all the factors containing the node to be marginalised. Replace all these factors with one big factor produced by marginalising over those factors only.
  - All other factors stay the same.
- This is the basis of the elimination algorithm.

Sum-Products

- Sum distribution in sum-products

$$
\begin{aligned}
P(x_1, x_2, x_4, x_5) &= \sum_{x_3} \frac{1}{Z} \psi_1(x_1, x_4)\psi_2(x_1, x_5)\psi_3(x_2, x_3)\psi_4(x_3, x_5) \\
&= \frac{1}{Z} \psi_1(x_1, x_4)\psi_2(x_1, x_5) \sum_{x_3} \psi_3(x_2, x_3)\psi_4(x_3, x_5) \\
&= \frac{1}{Z} \psi_1(x_1, x_4)\psi_2(x_1, x_5)\psi_*(x_2, x_5)
\end{aligned}
$$

where

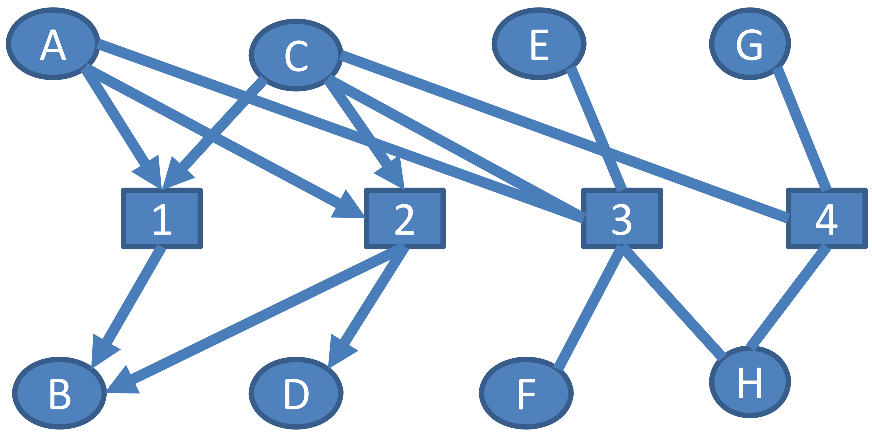
$$\psi_*(x_2, x_5) = \sum_{x_3} \psi_3(x_2, x_3)\psi_4(x_3, x_5)$$

- Order matters. Do cheap eliminations first.

Undirected Factor Graphs

- Focus on undirected factor graphs
- Any directed factor graph can be converted to undirected by removing arrows from edges.
- Lose some conditional dependence encoding, but still valid.

Elimination in General Factor Graphs

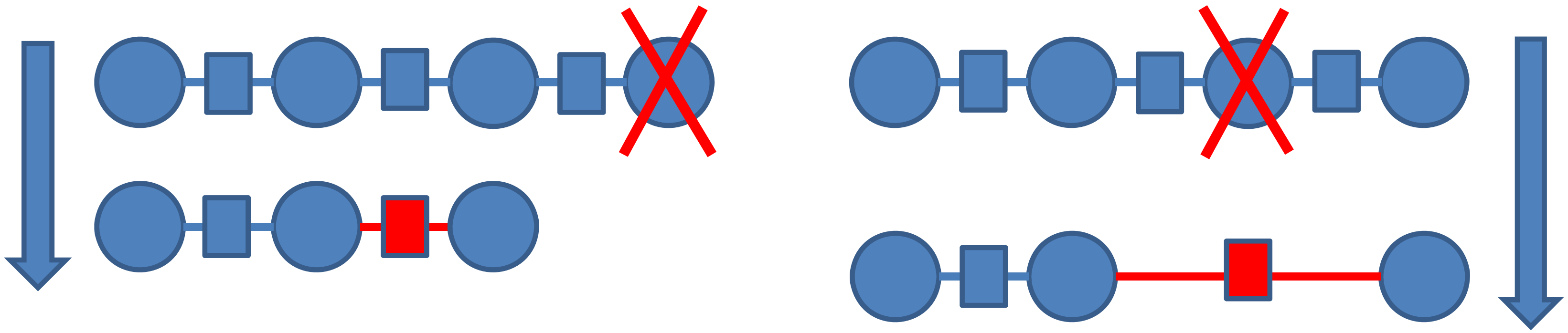


$$P(C|A,H) = \frac{P(C,A,H)}{\sum_C P(C,A,H)} = \frac{\sum_{B,D,E,F,G} P(A,B,C,D,E,F,G,H)}{\sum_C P(C,A,H)} \propto \sum_{B,D,E,F,G} \Phi_1(A,B,C)\Phi_2(A,B,C,D)\Phi_3(A,C,E,F,H)\Phi_4(C,G,H)$$

$$P(C|A,H) \propto \sum_{B,D,E,F,G} \Phi_1(A,B,C)\Phi_2(A,B,C,D)\Phi_3(A,C,E,F,H)\Phi_4(C,G,H)$$

Elimination Algorithm in Chains

- Consider Chains
  - If we eliminate from the ends of the chain, then it is cheap: results in a factor over one variable.
  - If we eliminate from the middle of the chain then it is cheap: results in a new link in the chain.



Elimination in Trees

- Consider any tree-structured factor graphs.
- Suppose we want the marginal distribution at one node. (Conditioned nodes have been absorbed.)
- Any node of an undirected tree can be viewed as the root. Make this the node you care about.
- Use elimination from *leaves* of the tree.
    - Just like the chain
    - Each step produces a subtree with at most two node factors.
    - Eventually just left with one node: the root.
    - Have one factor: the marginal distribution for this node.

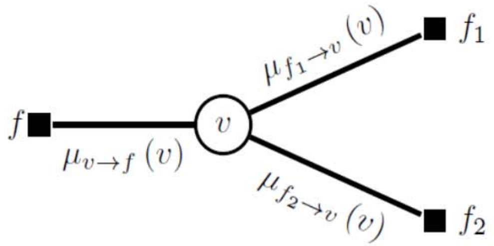
Message Passing

- We have seen that if we pass elimination messages up and down the tree, we can compute any marginal.
- On a factor graph this results in some simple message passing rules.
- Label variable nodes in factor graph by *v*: (**notation switch**)
- Turns out we can compute all the single marginals all at once using this message passing.

Variable to factor message

$$\mu_{v\to f}(v) = \prod_{f_i \sim v \setminus f} \mu_{f_i \to v}(v)$$

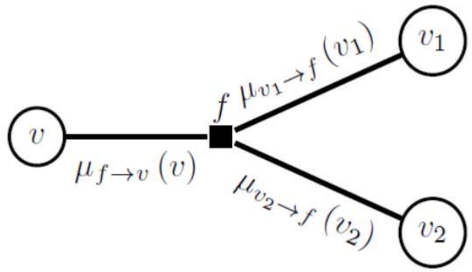
Messages from extremal variables are set to 1



Factor to variable message

$$\mu_{f\to v}(v) = \sum_{\{v_i\}} f(v, \{v_i\}) \prod_{v_i \sim f \setminus v} \mu_{v_i \to f}(v_i)$$

Messages from extremal factors are set to the factor



Marginal

$p(v) \propto \prod_{f_i \sim v} \mu_{f_i \to v}(v)$

Figure: David Barber

Break

Not Tree Structured?

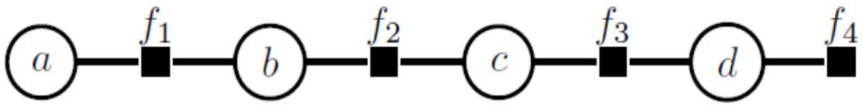
- Message Passing works for tree structured networks.
- What if it is not tree structured?
  - Well then the sizes of the factors created by the elimination process can grow. But elimination still works – it can just be costly.
  - We will see later we can consider a cluster graph.
  - We will see later we can just do approximate inference.

What about joint distributions?

- Computing single marginals is fine, but we might want to say something about joint distributions.
- Computing/working with joint distributions over many variables can be hard.
  - There are combinatorial many options.
  - Computing the normalisation is costly.
- However we can compute the highest posterior probability state.
  - Max product algorithm instead of sum product algorithm.
  - Max distributed just like the sum did in the elimination algorithm.

Max Product

$p(a,b,c,d) \propto f_1(a,b)\, f_2(b,c)\, f_3(c,d)\, f_4(d)$ $\quad a,b,c,d$ binary variables



$$\max_{a,b,c,d} p(a,b,c,d) = \max_{a,b,c,d} f_1(a,b)\, f_2(b,c)\, f_3(c,d)\, f_4(d)$$

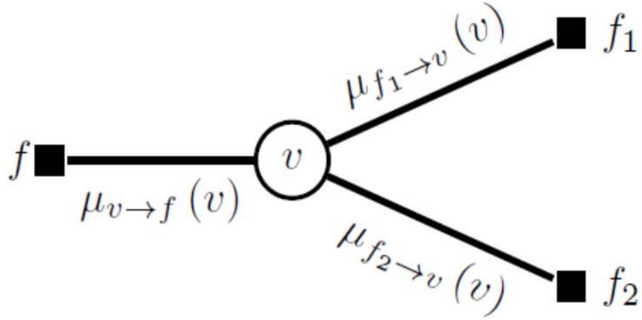
$$= \max_a \underbrace{\max_b f_1(a,b) \underbrace{\max_c f_2(b,c) \underbrace{\max_d f_3(c,d)\, f_4(d)}_{\mu_{d\to c}(c)}}_{\mu_{c\to b}(b)}}_{\mu_{b\to a}(a)}$$

Figure: David Barber

Max Product

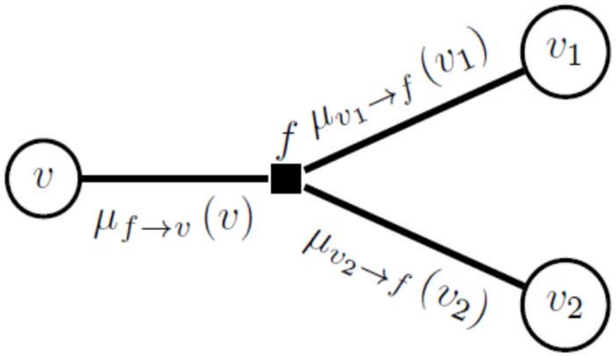
Variable to factor message

$$\mu_{v \to f}(v) = \prod_{f_i \sim v \setminus f} \mu_{f_i \to v}(v)$$



Factor to variable message

$$\mu_{f \to v}(v) = \max_{\{v_i\}} f(v, \{v_i\}) \prod_{v_i \sim f \setminus v} \mu_{v_i \to f}(v_i)$$



Most probable state (of joint)

$$v^* = \underset{v}{\operatorname{argmax}} \prod_{f_i \sim v} \mu_{f_i \to v}(v)$$

Figure: David Barber

Graphs are Important

- Hopefully you can see now why the graphs are important.
  - The graph determines how the messages are passed.
  - The actual functional form of the factors in the distribution determine what the messages are.

Not a tree?

- What if it is not a tree?
- Actually works for tree decompositions too:
  - Find all the variable sets that are overlaps between factors (we'll call them separator sets, or just separators). Label each separator.
  - Replace the variables nodes with separator nodes in the graph.
  - Can you build a tree with the separators, rather than the variables?
  - For every path in the tree: does each variable only occur on adjacent separators along the path (running intersection property)?
  - Then we can do message passing in this tree decomposition too, at a cost related to the number of states in the variable sets. We'll try to see why…
- What if I can't build a tree decomposition?
  - Then make the factors bigger, until you can build a tree decomposition.
- How?
- Junction Tree Algorithm. Chapter 6 of Barber.
  - This is something to work through yourself using that book
- But if I do this my variable sets at too big and inference is too expensive.
- Ah well. Perhaps you should just pretend it is a tree and pass messages anyway: loopy belief propagation.

Approximate Inference

- In many cases our graphs are not suitable for the exact inference process described to be computationally feasible
- Can resort to approximate inference:
  - Sampling
  - Loopy message passing:
    - Loopy belief propagation.
    - Variational message passing.
    - Expectation Propagation.
- More later…

Our Journey

Graphical Models	Decision Theory	Learning Probabilistic Models	Mixture and Factor Models	Markov Models	Approximate Inference

- Lecture 2&3: Introduce Factor Graphs
  - Distributions → Factor Graphs
  - Content v Form
    - Structure of distributions
  - Conditional Independence in Factor Graphs
- Lecture 4: Inference in Factor Graphs.
- Next Lecture: Other forms of Graphical Models.