iPMR

Our Journey

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

- Informally Introduce Belief Networks
- Formalise
  - Graph Theory
  - Probabilistic Graphical Models
    - Belief Networks (Bayesian Networks)
    - Markov Networks
    - Factor Graphs
  - Inference in Graphical Models

Recap

- We introduced Belief Networks
- We introduced Factor Graphs, and Markov Networks.
- We related different networks
- We discussed conditional independence encoding
- We stated the problem of inference
- We illustrated the elimination algorithm

Factor Graphs

- Belief Networks and Markov Networks can be represented as Factor Graphs without any loss of conditional independence information (why?), so we focus on Factor Graphs.

$$P(\mathbf{x}) = \frac{1}{Z}\prod_{i=1}^{N_f}\psi_i(\chi_i)$$

- In inference we wish to compute some conditional distribution. Write $\mathbf{x} = (\mathbf{v}^T, \mathbf{h}^T)^T$. Then we wish to find each $P(h_i|\mathbf{v})$.
- As before we can do conditioning trivially. We just fix all the values of **v** in the factors, to the right values. For example if

$$\chi_1(.) = \chi_1(h_1, h_2, h_4, v_1, v_3)$$

- then we just set the values $v1$ and $v3$ in the factor to the conditioned values.
- We consider only tree structured networks at this stage. For factor graphs this means there are no cycles.

Marginalisation

- Conditioning is trivial but marginalisation is not.
- Consider the tree structured factor graph

$$P(\mathbf{h}) = \frac{1}{Z}\prod_{i=1}^{N_f}\psi_i(\chi_i)$$

where we have absorbed the conditioned information into the factors and only view it as a graph over the latent (hidden) variables.

  - As it is tree structured, the graph over the latent nodes is tree structured. Hence we can choose a root, and label the tree ordering: Hence for each hidden node *i* we can define *Pa*(*i*).
  - Consider now the belief network for that ordering.
  - If we knew all the conditional probabilities, we can get all the marginal distributions for that belief network trivially. Start at the top and marginalise down.
  - So if we can match our factor graph to a belief network (on just the hidden nodes) we have solved the problem.
  - To do this we have to get conditional probabilities that match the conditional probabilities on the original factor graph.
  - How?

Trees

- We already know how. Elimination (Sum-product Algorithm).
- If a node is a leaf node, it only turns up in one factor, so we can read off its conditional distribution immediately:

$$P(h_{\text{leaf}}|h_{Pa}) = \frac{\psi_*(h_{\text{leaf}}, h_{Pa})}{\sum_{h_{\text{leaf}}} \psi_*(h_{\text{leaf}}, h_{Pa})}$$

where ψ_* is the only factor containing the leaf node and h_{Pa} is the unique parent of that leaf node.

- If the node is not a leaf node we can make it one by elimination.
- So all we need to do is do elimination from leaf to root of the tree, keeping track of the conditional distributions, and then do marginalisation from root to leaf in the resulting belief network.
- In fact the marginalisation from root to leaf is also a (trivial) form of elimination, so it turns out these two processes are symmetric.
- This is not surprising, as the root of an (undirected) tree is arbitrary.
- Hence we can write down the full inference process as elimination messages.

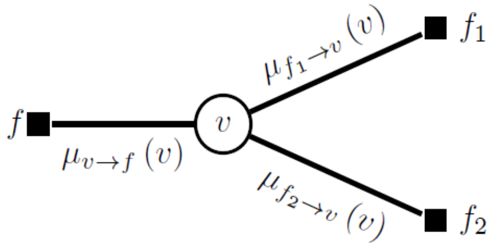
Message Passing

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

Variable to factor message

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

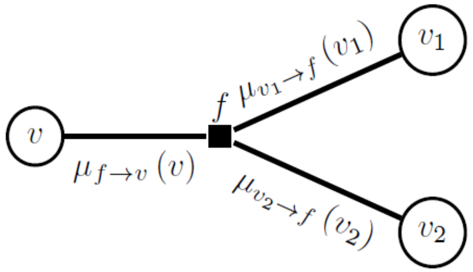
Messages from extremal variables are set to 1



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

Stop Point

- Questions?

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.
  - 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. In short
  - Start with Markov Network.
  - Triangulate (ensure all loops length>4 have a chord).
  - Find all maximal cliques of triangulated graph, and define factors for those.
- 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.

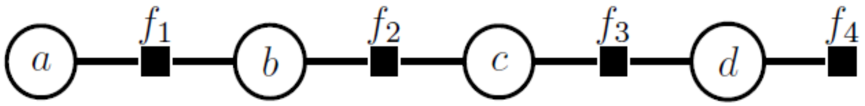
Notes on Junction Trees

What about joint distributions.

- Computing marginals is fine, but computing 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



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

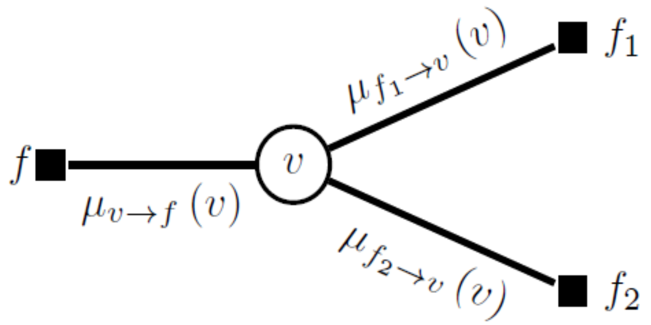
$$= \max_a \underbrace{\max_b f_1(a,b) \underbrace{\max_c f_2(b,c) \underbrace{\max_d f_3(c,d) f_4(d)}_{\mu_{d\to c}(c)}}_{\mu_{c\to b}(b)}}_{\mu_{b\to a}(a)}$$

Figure: David Barber

Max Product

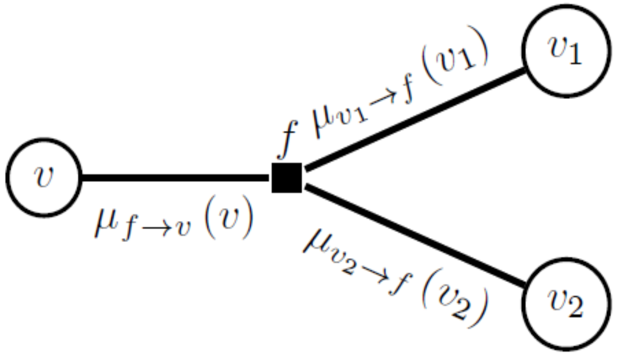
Variable to factor message

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



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

Summary

- Graphical Models
- Sum product rule and message passing for inference in tree structured networks.
- Trees over separators, with running intersection.
- Loopy belief propatation, and max product.
- Continue to read and work through Chapters 4 and 5 of Barber.
- Chapter 6 is the Junction Tree.
- The details of Chapter 6 will not be examinable. However a broad understanding of the Junction Tree as presented here will be.