EECS 491

Inference in Graphical Models

Recap of Inference in Bayesian networks

- For queries in Bayesian networks, we divide variables into three classes:
  - evidence variables: $e = \{e_1, \ldots, e_m\}$ what you know
  - query variables: $x = \{x_1, \ldots, x_n\}$ what you want to know
  - non-evidence variables: $y = \{y_1, \ldots, y_l\}$ what you don't care about
- The complete set of variables in the network is $\{e \cup x \cup y\}$.
- Inferences in Bayesian networks consist of computing p(x|e), the posterior probability of the query given the evidence:

$$p(x|e) = \frac{p(x,e)}{p(e)} = \alpha\, p(x,e) = \alpha \sum_y p(x,e,y)$$

- This computes the marginal distribution p(x,e) by summing the joint over all values of y.
- Recall that the joint distribution is defined by the product of the conditional pdfs:

$$p(z) = \prod_{i=1} P(z_i|\text{parents}(z_i))$$

where the product is taken over all variables in the network.

Variable elimination

- When we've pulled out all the redundant terms we get:

$$p(b|o) = \alpha\, p(b) \sum_d p(d|b) \sum_w p(w) p(o|w,b) \sum_c p(c|w)$$

- We can also note the last term sums to one. In fact, every variable that is not an ancestor of a query variable or evidence variable is *irrelevant* to the query, so we get

$$p(b|o) = \alpha\, p(b) \sum_d p(d|b) \sum_w p(w) p(o|w,b)$$

which contains far fewer terms: In general, *complexity is **linear** in the # of CPT entries*.

- This method is called *variable elimination*.
  - if # of parents is bounded, also linear in the number of nodes.
  - the expressions are evaluated in right-to-left order (bottom-up in the network)
  - intermediate results are stored
  - sums over each are done only for those expressions that depend on the variable

- Note: for multiply connected networks, variable elimination can have exponential complexity in the worst case.
- This is similar to CSPs, where the complexity was bounded by the hypertree width.

The joint pdf for a Markov net is a product of potential functions

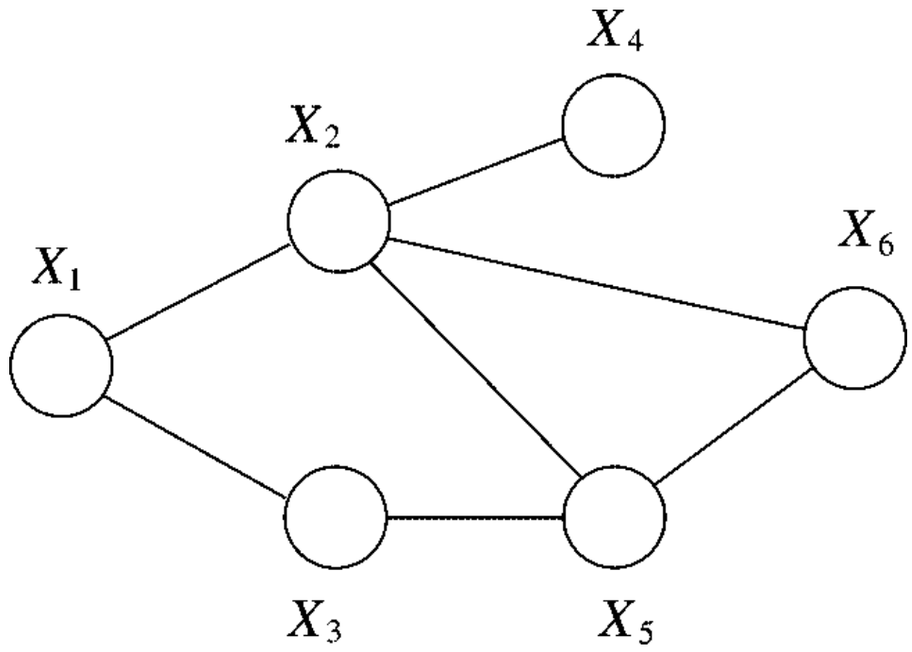
- The joint probability is a product of *clique* potential functions:

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \frac{1}{Z} \prod_{\text{cliques } c} \psi_c(\mathbf{x}_c)$$

- Where each $\psi_c(\mathbf{x}_c)$ is an arbitrary positive function of it's arguments.
- The set of cliques is the set of maximal complete subgraphs.
- Z is a normalization constant that defines a valid joint pdf, and is sometimes called the *partition function*.

$$Z = \sum_{\mathbf{x}} \prod_{\text{cliques } c} \psi_c(\mathbf{x}_c)$$

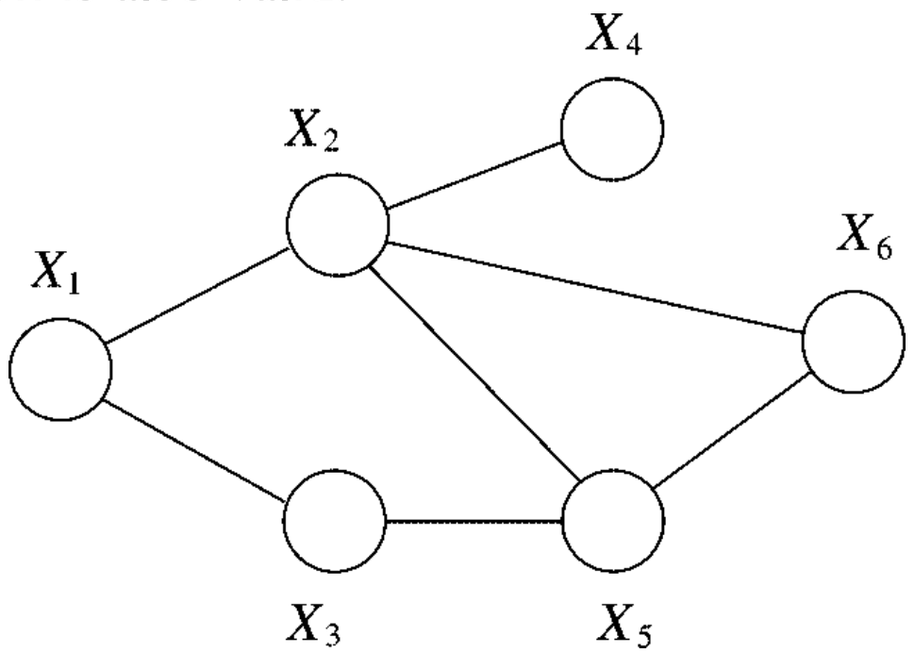
This is a greatly reduced representation.



$$p(x_{\mathcal{V}}) = \frac{1}{Z}\psi(x_1, x_2)\psi(x_1, x_3)\psi(x_2, x_4)\psi(x_3, x_5)\psi(x_2, x_5, x_6)$$

Multiple definitions are possible

- It is not necessary to restrict the definition to maximal cliques
- The following definition is also valid:



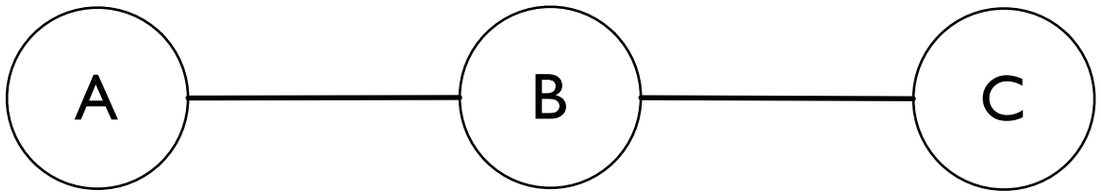
$$p(X) = \frac{1}{Z}\psi(x_1, x_2)\psi(x_1, x_3)\psi(x_2, x_4)\psi(x_3, x_5)\psi(x_2, x_5)\psi(x_2, x_6)\psi(x_5, x_6)$$

- Here, we have assumed a factorization in terms of 2D densities.
- Why can we do this?

This is equivalent to assuming that $\psi(x_2, x_5, x_6)$ factors.

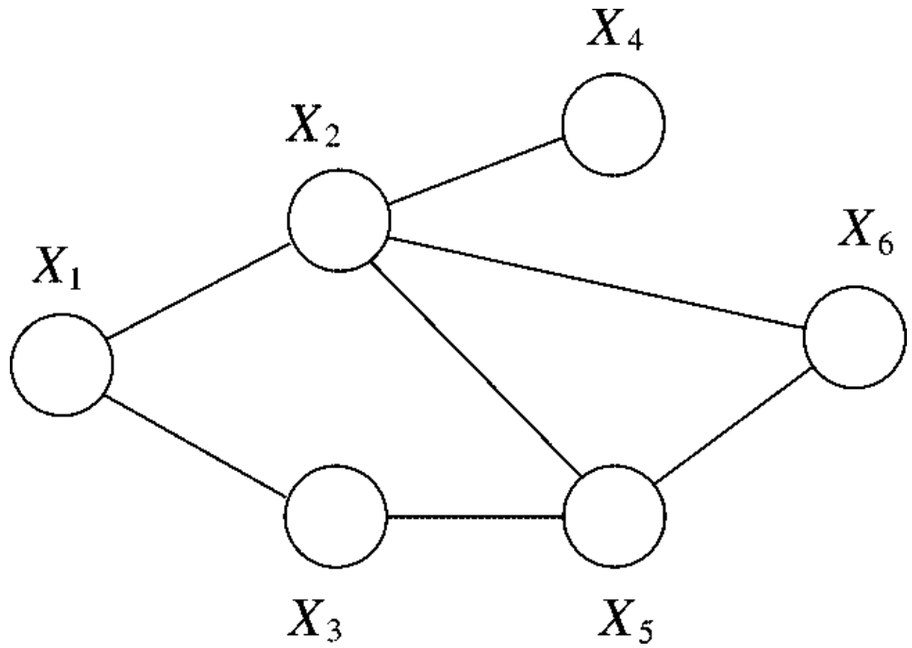
What are the clique potential functions?

- Consider the following model:



- The model specifies that $A \perp C \mid B$.
- The joint distribution can then be written as
  - $p(A,B,C) = p(B)\, p(A|B)\, p(C|B)$
- This can be written in two ways:
  - $p(A,B,C) = p(A,B)\, p(C|B) = \psi_1(A,B)\, \psi_2(B,C)$
  - $p(A,B,C) = p(A|B)\, p(B,C) = \psi_3(A,B)\, \psi_4(B,C)$
- This shows that the potential functions cannot both be marginals or both be conditionals.
- In general, the clique potential functions *do not* represent probability distributions. They are simply factors in the joint pdf.

Exact inference on undirected graphs by elimination



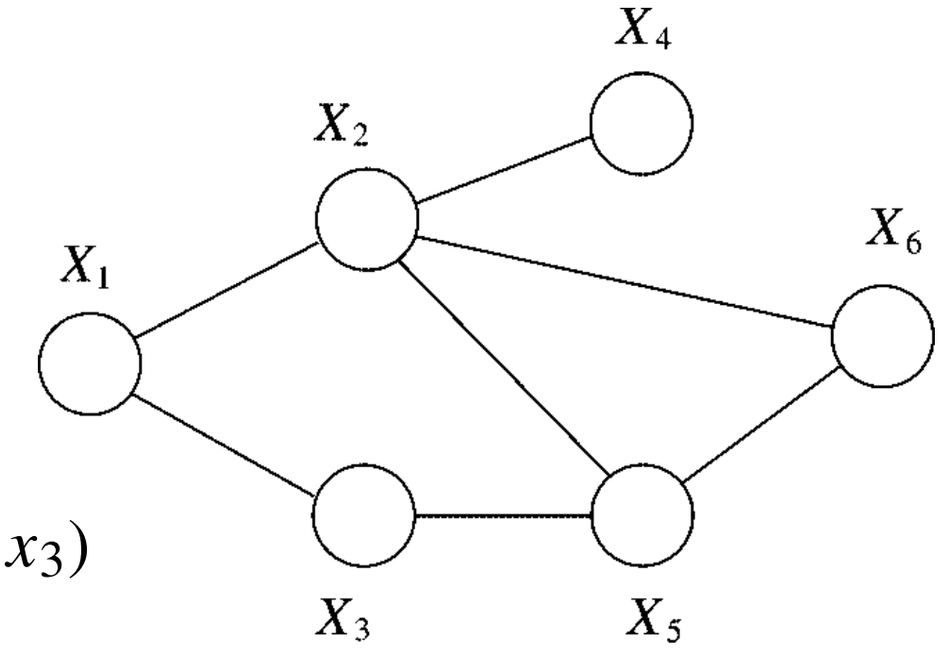
- How do we compute a marginal distribution, eg $p(x_1)$, on an undirected graph?
- Want to compute marginal, $p(x_1)$, sum over remaining vars:

$$p(x_1) = \sum_{x_2} \sum_{x_3} \sum_{x_4} \sum_{x_5} \sum_{x_6} \frac{1}{Z} \psi(x_1, x_2)\psi(x_1, x_3)$$
$$\cdot \psi(x_2, x_4)\psi(x_3, x_5)$$
$$\cdot \psi(x_2, x_5, x_6).$$

- Like before, the naive sum is r^6, but we can push the sums in the products to obtain:

$$p(x_1) = \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) \sum_{x_3} \psi(x_1, x_3) \sum_{x_4} \psi(x_2, x_4)$$
$$\cdot \sum_{x_5} \psi(x_3, x_5) \sum_{x_6} \psi(x_2, x_5, x_6)$$

The triangulation algorithm



- The summation could have been performed in a different order

$$p(x_1) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5}\sum_{x_6}\frac{1}{Z}\psi(x_1,x_2)\psi(x_1,x_3)$$
$$\cdot\,\psi(x_2,x_4)\psi(x_3,x_5)$$
$$\cdot\,\psi(x_2,x_5,x_6).$$

- Want an *elimination* order so that the worst-case summation is as small as possible.
- The triangulation algorithm:
  1. choose next node in elimination order
  2. link all remaining nodes that are neighbors of that node
  3. remove node from graph
- Largest clique that arises in sequence of graphs determines worst-case complexity.
- But: Choosing an elimination ordering that achieves treewidth is NP-hard, but good orderings can often be found for specific cases.

The triangulation algorithm

$$p(x_1) = \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) \sum_{x_3} \psi(x_1, x_3) \sum_{x_4} \psi(x_2, x_4)$$

$$\cdot \sum_{x_5} \psi(x_3, x_5) m_6(x_2, x_5)$$

$$= \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) \sum_{x_3} \psi(x_1, x_3) m_5(x_2, x_3)$$

$$\cdot \sum_{x_4} \psi(x_2, x_4)$$

$$= \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) m_4(x_2)$$

$$\cdot \sum_{x_3} \psi(x_1, x_3) m_5(x_2, x_3)$$

$$= \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) m_4(x_2) m_3(x_1, x_2)$$

$$= \frac{1}{Z} m_2(x_1),$$

- This yields considerably simpler complexity, because the largest sum, m_6, is order r^3.
- Worst case complexity is equal to the size of the largest clique.
- This is known as the *treewidth* of the graph.

Computing general marginal probabilities

- The elimination approach only computes a single marginal, e.g. p(x1).
- How do we compute general marginals without requiring multiple runs?
- In the special case where the graph is a tree:
  - cliques are pairs of nodes and singleton nodes
  - probability distribution is parameterized by

    $\psi(x_i,x_j)$: $(i,j) \in$ edges and $\psi(x_i)$: $i \in$ vertices.
  - to compute $p(x_f)$, where node f is taken to be the root of the tree and choose elimination order such that all children are eliminated first.
- In this case, the steps of elimination can be written in general form:

$$m_{ji}(x_i)$$

$$= \sum_{x_j} \left(\psi(x_j)\psi(x_i, x_j) \prod_{k \in \mathcal{N}(j)\setminus i} m_{kj}(x_j) \right)$$

- where $k \in \mathcal{N}(j)\setminus i$ specifies the neighbors of node j without node i.

A general form for the marginal

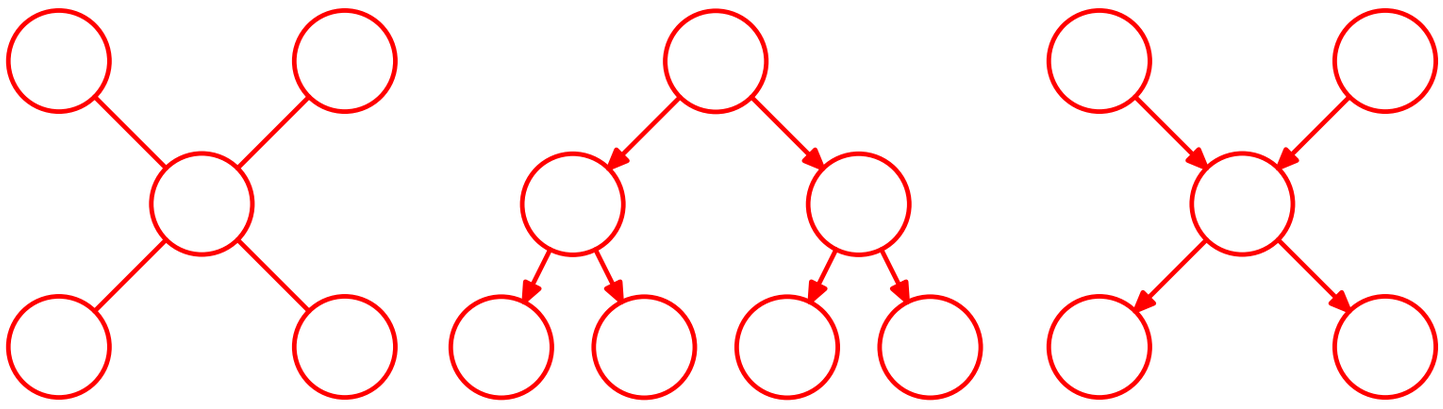
- The marginal for the root node f is then given by

$$p(x_f) \propto \psi(x_f) \prod_{e \in \mathcal{N}(f)} m_{ef}(x_f)$$

- This is only for the root node. We wanted all nodes. How?

Can we derive an inference algorithm for general graphs?

- Will show that efficient inference is possible in trees (*Belief Propagation*):



(a) undirected tree (b) directed tree (c) directed poly-tree

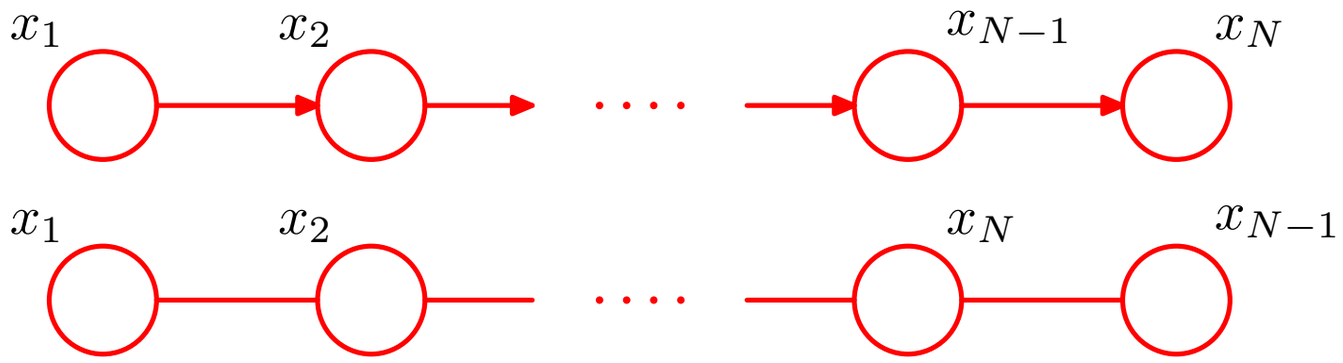
- Then we develop a more general method of efficient inference (*sum-product algorithm)*
- First define on chain graphs.
- To generalize, we need another type of graph: *factor graph*

Inference in Graphical Models

The lecture notes are based on Chapter 8 of Bishop, *Pattern Recognition and Machine Learning.*

This chapter is available online and on blackboard. Also see Barber Ch. 5.1-3 or Murphy Ch. 20.

Consider the special case of chain graphs



- The joint distribution for the directed graph is:

$$p(\mathbf{x}) = p(x_1)p(x_2|x_1)p(x_3|x_2)\cdots p(x_N|x_{N-1}).$$

- For the undirected graph, the joint distribution is:

$$p(\mathbf{x}) = \frac{1}{Z}\psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3)\cdots\psi_{N-1,N}(x_{N-1}, x_N).$$

- Note that in this case we can easily convert one to the other:

$$\begin{aligned}
\psi_{1,2}(x_1, x_2) &= p(x_1)p(x_2|x_1) \\
\psi_{2,3}(x_2, x_3) &= p(x_3|x_2) \\
&\vdots \\
\psi_{N-1,N}(x_{N-1}, x_N) &= p(x_N|x_{N-1})
\end{aligned}$$

Z=1

How do we do inference?

Computing marginals in chain graphs

- To find the marginal distribution, part way along the chain:

$$p(x_n) = \sum_{x_1} \cdots \sum_{x_{n-1}} \sum_{x_{n+1}} \cdots \sum_{x_N} p(\mathbf{x}).$$

- Note: this sums over all variables except x_n.
- Naïve summation scales exponentially with N, so use variable elimination.
- The potential $\psi_{N-1,N}(x_{N-1}, x_N)$ is the only term that depends on x_N
- So, we can perform the following summation to get a function of x_{N-1}

$$\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N)$$

- With this we can sum over x_{N-1} which involves only $\psi_{N-2,N-1}(x_{N-2}, x_{N-1})$
- and so on.
- Each summation removes a variable (or node)

Computing marginals in chain graphs

- We can regroup the potentials to express the marginals in the following way:

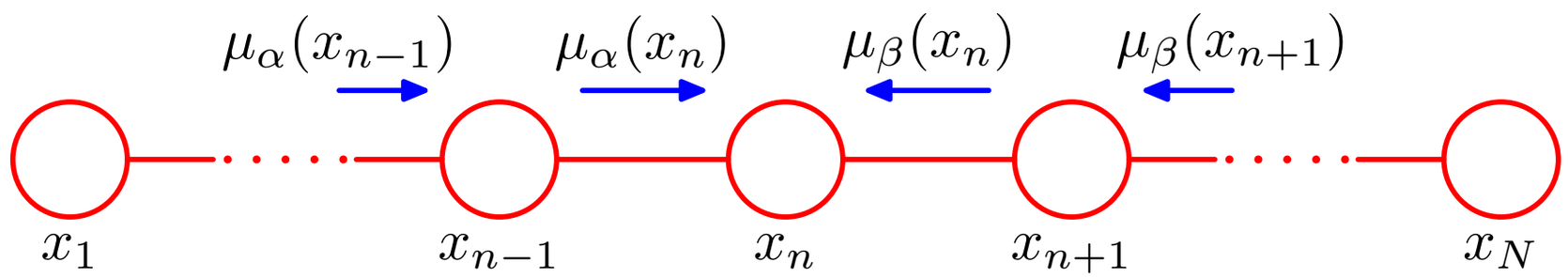
$$p(x_n) = \frac{1}{Z} \underbrace{\left[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \cdots \left[\sum_{x_2} \psi_{2,3}(x_2, x_3) \left[\sum_{x_1} \psi_{1,2}(x_1, x_2) \right] \right] \cdots \right]}_{\mu_\alpha(x_n)}$$

$$\underbrace{\left[\sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \cdots \left[\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N) \right] \cdots \right]}_{\mu_\beta(x_n)}. \tag{8.52}$$

- This is just using the distributive law ab + ac = a(b+c).
- The cost works out to be linear in the length of the chain $O(NK^2)$, where K is the number of states for each variable, e.g. 2 for binary nodes.
- We can interpret each node as using *local messages* to compute the marginals:

$$p(x_n) = \frac{1}{Z} \mu_\alpha(x_n) \mu_\beta(x_n).$$

Message passing in chain graphs

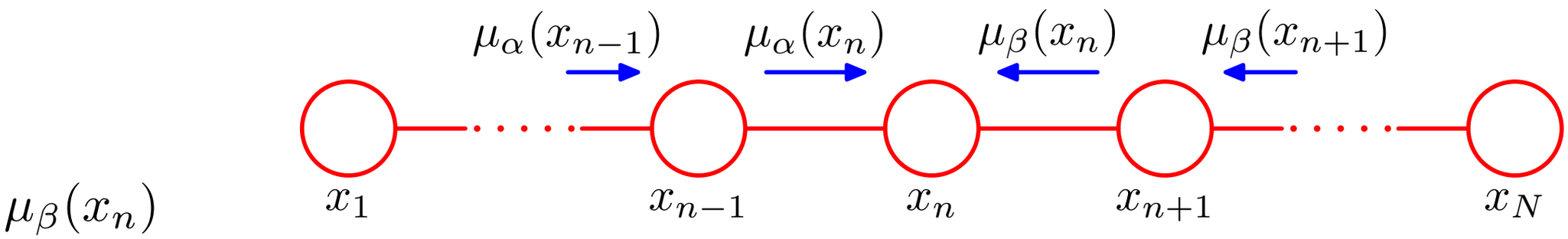


- $\mu_\alpha(x_n)$ is a message passed forwards along the chain from x_{n-1} to x_n
- $\mu_\beta(x_n)$ is a message passed backwards along the chain from x_{n+1} to x_n
- Note that each message is a set of K values, one for each variable state.
- The product:

$$p(x_n) = \frac{1}{Z}\mu_\alpha(x_n)\mu_\beta(x_n).$$

is a point-wise multiplication that gives the *marginal*, which also has K values.

Messages are defined recursively



- $\mu_\alpha(x_n)$ can be evaluated recursively:

$$\begin{aligned}\mu_\alpha(x_n) &= \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \left[\sum_{x_{n-2}} \cdots \right] \\ &= \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n)\mu_\alpha(x_{n-1}).\end{aligned}$$

- Therefore we sum over the first potential first to get the first message:

$$\mu_\alpha(x_2) = \sum_{x_1} \psi_{1,2}(x_1, x_2)$$

- Subsequent messages are then evaluated recursively until we have what we want.
- The other set of messages is computed similarly, starting from the last node:

$$\begin{aligned}\mu_\beta(x_n) &= \sum_{x_{n+1}} \psi_{n+1,n}(x_{n+1}, x_n) \left[\sum_{x_{n+2}} \cdots \right] \\ &= \sum_{x_{n+1}} \psi_{n+1,n}(x_{n+1}, x_n)\mu_\beta(x_{n+1}).\end{aligned}$$

What about the partition function (normalizing constant) Z?

- Recall for the undirected graphical model:

$$p(\mathbf{x}) = \frac{1}{Z}\prod_C \psi_C(\mathbf{x}_C)$$

- the normalizing constant is given by:

$$Z = \sum_{\mathbf{x}}\prod_C \psi_C(\mathbf{x}_C)$$

- Here, however, the cliques are just pairwise. As we have seen:

$$p(x_n) = \frac{1}{Z}\mu_\alpha(x_n)\mu_\beta(x_n).$$

- which implies

$$Z = \sum_{x_n} \mu_\alpha(x_n)\mu_\beta(x_n)$$

- So the evaluation of Z is O(K) once we have computed the messages.

Computing marginals for other nodes?

- Naïvely, we could do the above procedure for every node, but that would be wasteful.
- Simply store the messages, so they don't have to be recomputed. They're the same.
- Cost of computing all marginals is $O(NK^2)$.
  - *Only twice the cost of computing a single marginal.*

What if some nodes are observed?

- If x_n = a is given, this is equivalent to setting the joint distribution p($\underline{x}$) = a for x_n.
- This means all the values in the definition of p($\underline{x}$) are fixed to a.
- So, we can avoid the summation over x_n in the evaluation of the messages:

$$\begin{aligned}\mu_\alpha(x_n) &= \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \left[\sum_{x_{n-2}} \cdots \right] \\ &= \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \mu_\alpha(x_{n-1}).\end{aligned}$$

What about computing other quantities?

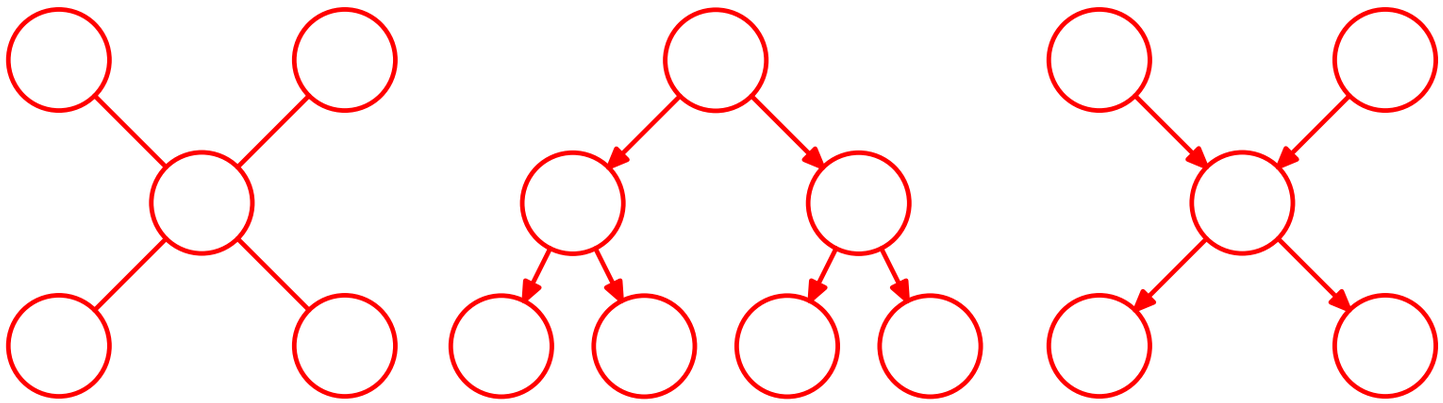
- For example, what if we want $p(x_{n-1}, x_n)$?
- We proceed as before, but we don't sum over x_{n-1}.
- Therefore we obtain:

$$p(x_{n-1}, x_n) = \frac{1}{Z}\mu_\alpha(x_{n-1})\psi_{n-1,n}(x_{n-1}, x_n)\mu_\beta(x_n).$$

- This means we can easily compute joint distributions over sets of variables once we've computed the messages for every node.

Can we extend this?

- Showed we can do exact inference on a chain graph in linear time.
- Will show that efficient inference is possible in trees:



(a) (b) (c)

undirected tree directed tree directed poly-tree

- To do this we will develop a more general method of efficient inference called: the *sum-product algorithm*
- First, we need to introduce another type of graph

Factor graphs

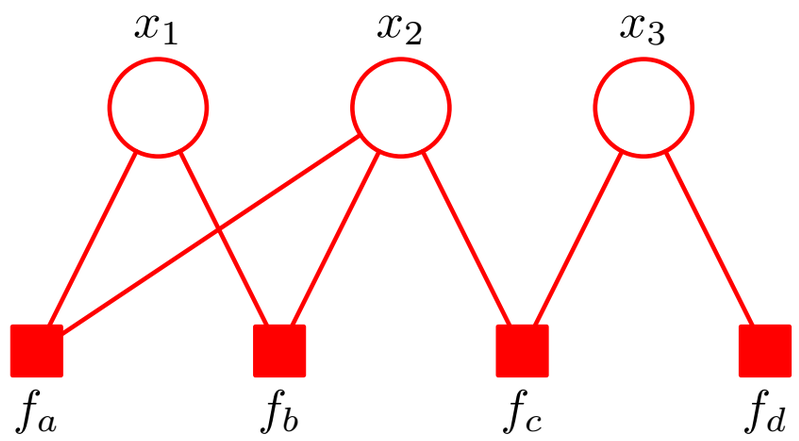
- Both directed and undirected graphs decompose the joint distribution in products.

- More generally, we can write a joint distribution as a product of factors:

$$p(\mathbf{x}) = \prod_s f_s(\mathbf{x}_s)$$

- where x_s denotes a subset of the variables.
- For:
  - directed graphs: fs = local conditional distribution
  - undirected graphs: fs = potential functions over maximal cliques
- Factor graphs keep all factors explicit.
  - a node for every variable
  - additional nodes for every factor

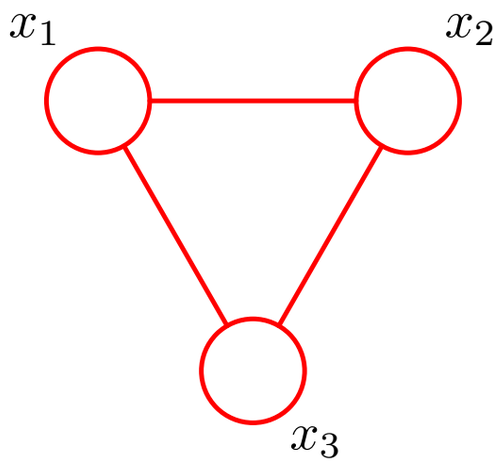
Factor graphs



- Variables are depicted by circles.
- Factors $f_s(x_s)$ are depicted by squares.
- The corresponding joint distribution for the above graph is

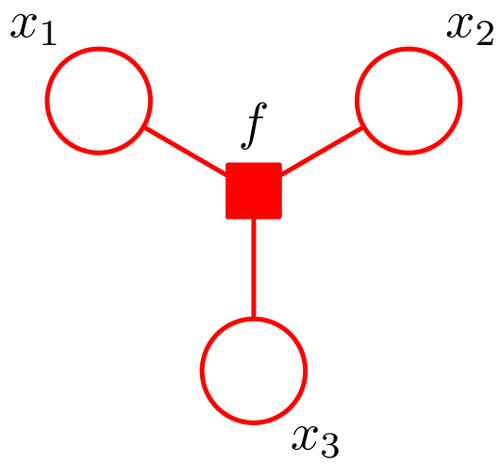
$$p(\mathbf{x}) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3).$$

- What is each joint distribution for the following graphs?



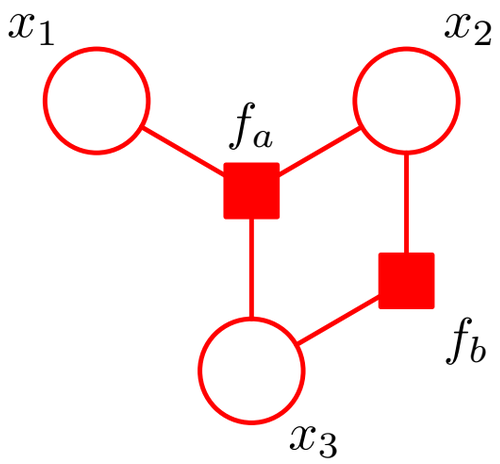
(a)

$$\psi(x_1, x_2, x_3)$$



(b)

$$f(x_1, x_2, x_3) = \psi(x_1, x_2, x_3)$$

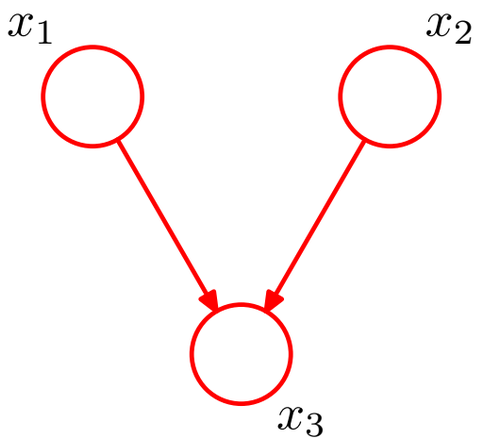


(c)

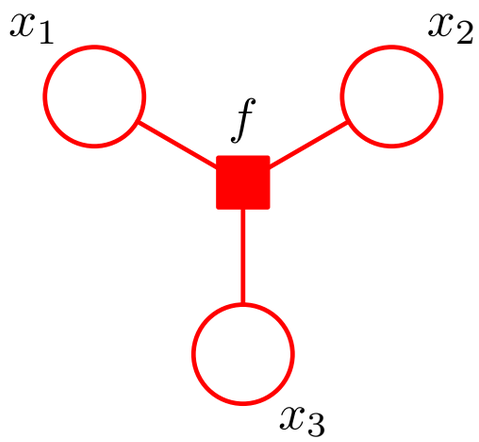
$$f_a(x_1, x_2, x_3) f_b(x_1, x_2) = \psi(x_1, x_2, x_3)$$

Factor graphs

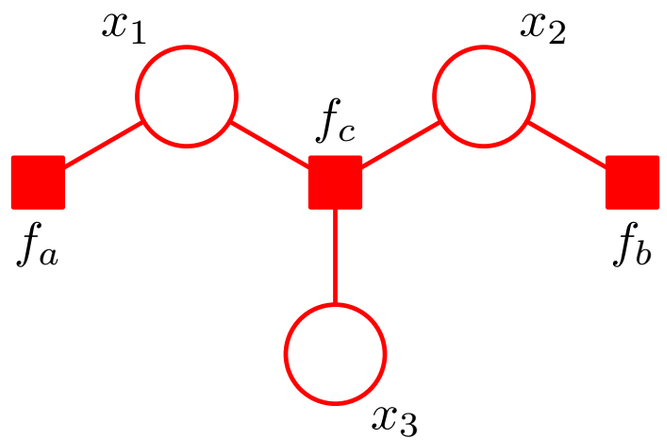
- We can also represent directed graphical models with factor graphs:



(a)



(b)



(c)

$$p(x_1)p(x_2)p(x_3|x_1,x_2)$$

$$f(x_1,x_2,x_3) = p(x_1)p(x_2)p(x_3|x_1,x_2)$$

$$f_a(x_1) = p(x_1),\ f_b(x_2) = p(x_2)$$
$$f_c(x_1,x_2,x_3) = p(x_3|x_1,x_2)$$

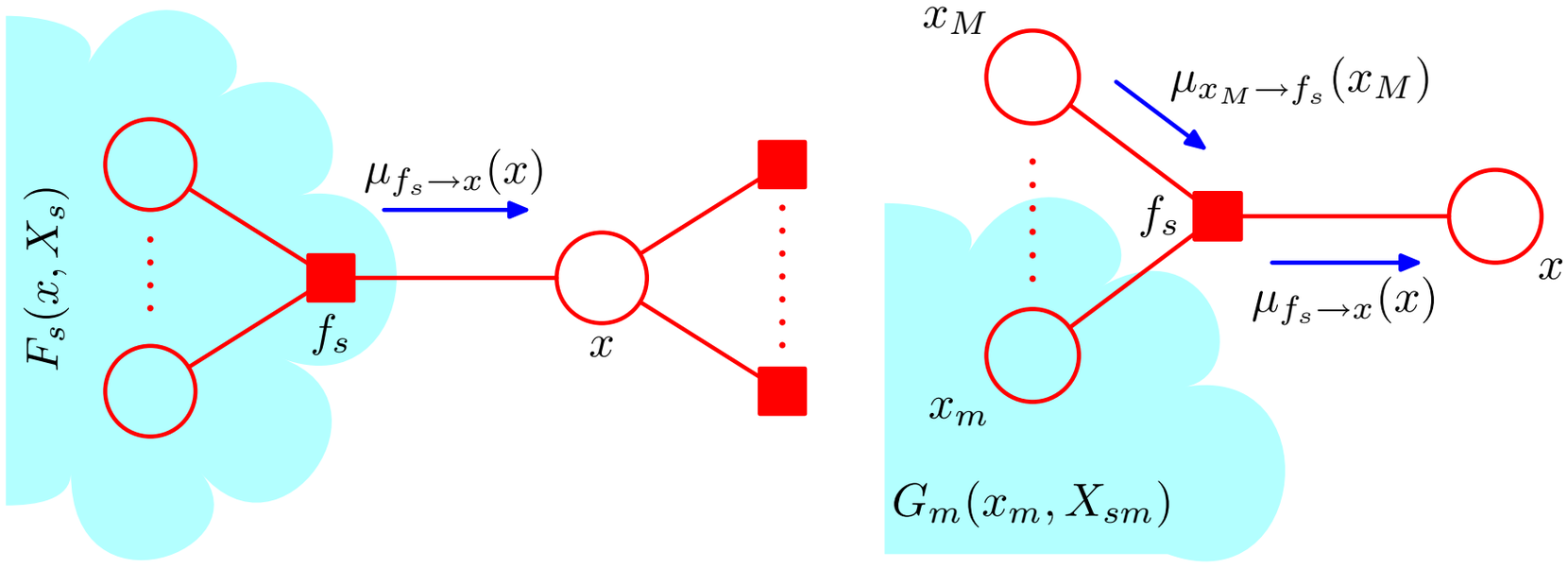
- factor graphs are *bipartite* because the use two distinct kinds of nodes:
  - variable nodes and factor nodes
  - factor nodes only connect to variable nodes and vice versa

The sum-product algorithm

- UG and DG in same framework (because both can be converted to a factor graph)
- Generalization of *belief propagation* algorithm.
- Efficient and exact inference algorithm for computing marginals
- Derivation is follows that on chain graphs
  - obtain a recursive function for *messages* required at each node
  - now there are two types of messages: one for each type of node
  - each message can be computed efficiently
  - base cases are the leaf nodes

Messages in the sum-product algorithm

- $\mu_{f_s \to x}(x)$ represents a message going from a factor node f_s to variable node x



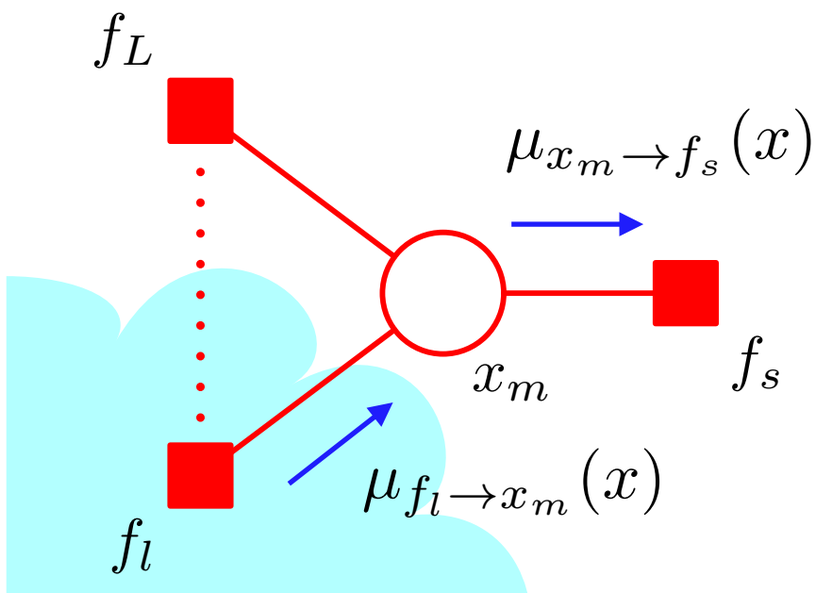
- Like before the equation is recursive (derivation omitted):

$$\mu_{f_s \to x}(x) = \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \ldots, x_M) \prod_{m \in \mathrm{ne}(f_s) \setminus x} \left[\sum_{X_{xm}} G_m(x_m, X_{sm}) \right]$$

$$= \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \ldots, x_M) \prod_{m \in \mathrm{ne}(f_s) \setminus x} \mu_{x_m \to f_s}(x_m) \qquad (8.66)$$

Messages in the sum-product algorithm

- $\mu_{x_m \to f_s}(x)$ represents a message going from variable node x_m to factor f_s



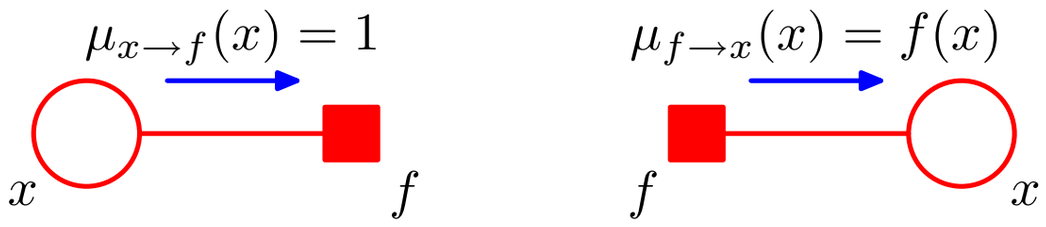
- Like before the equation is recursive (see notes and Bishop Ch.8 for derivation):

$$\mu_{x_m \to f_s}(x_m) = \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} \left[\sum_{X_{ml}} F_l(x_m, X_{ml}) \right]$$

$$= \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m)$$

Base cases: leaf nodes

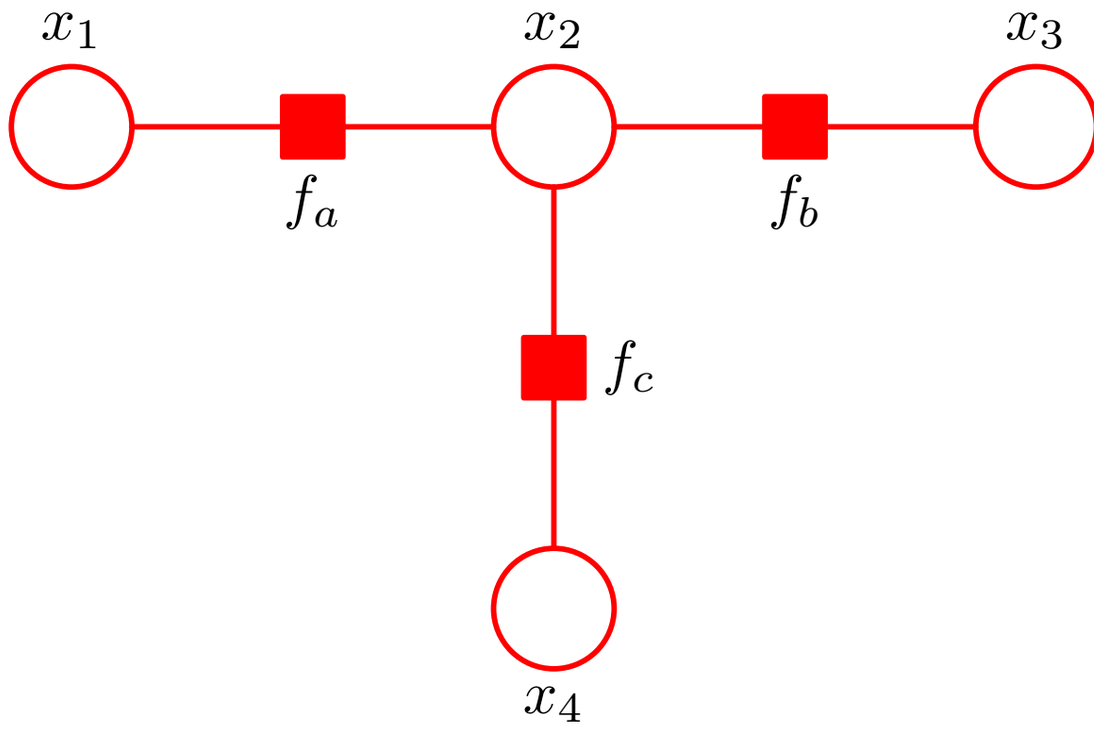
- There are two bases cases, one for each type of leaf node:



- The sum-product algorithm begins with messages sent from the leaf nodes.

An example

- Like in the chain example, the marginals are computed by recursively evaluating the messages.

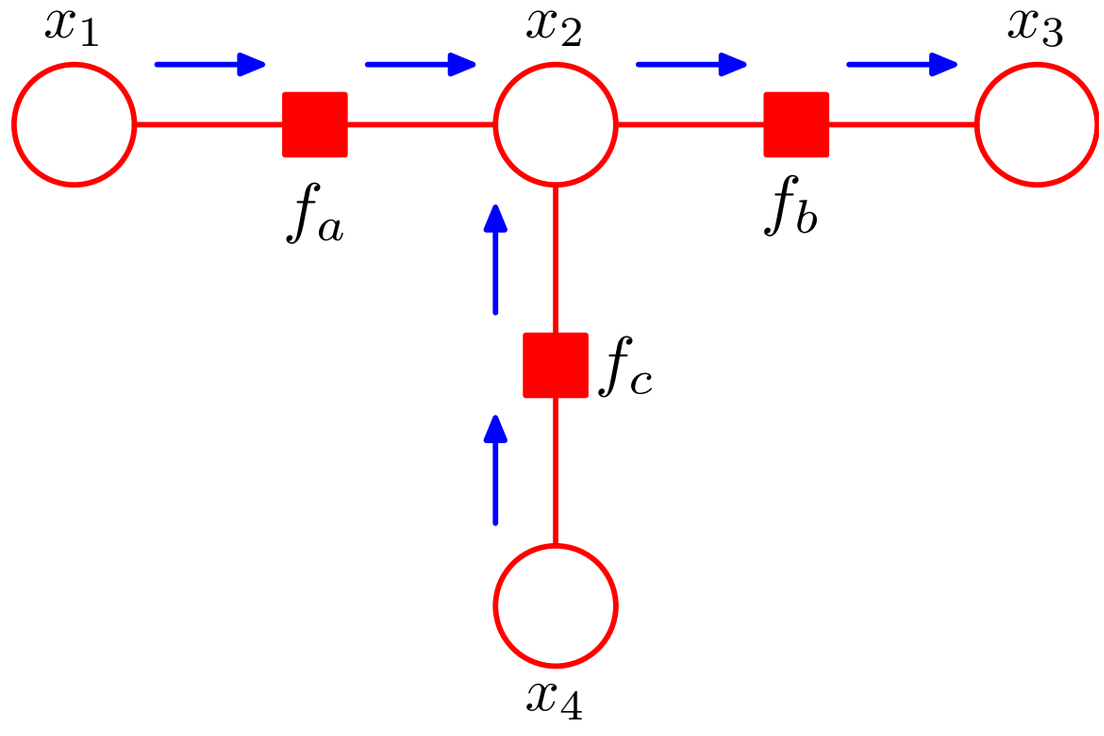


$$\widetilde{p}(\mathbf{x}) = f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4).$$

An example

- Compute first set of messages:

$$
\begin{aligned}
\mu_{x_1 \to f_a}(x_1) &= 1 \\
\mu_{f_a \to x_2}(x_2) &= \sum_{x_1} f_a(x_1, x_2) \\
\mu_{x_4 \to f_c}(x_4) &= 1 \\
\mu_{f_c \to x_2}(x_2) &= \sum_{x_4} f_c(x_2, x_4) \\
\mu_{x_2 \to f_b}(x_2) &= \mu_{f_a \to x_2}(x_2)\mu_{f_c \to x_2}(x_2) \\
\mu_{f_b \to x_3}(x_3) &= \sum_{x_2} f_b(x_2, x_3)\mu_{x_2 \to f_b}.
\end{aligned}
$$

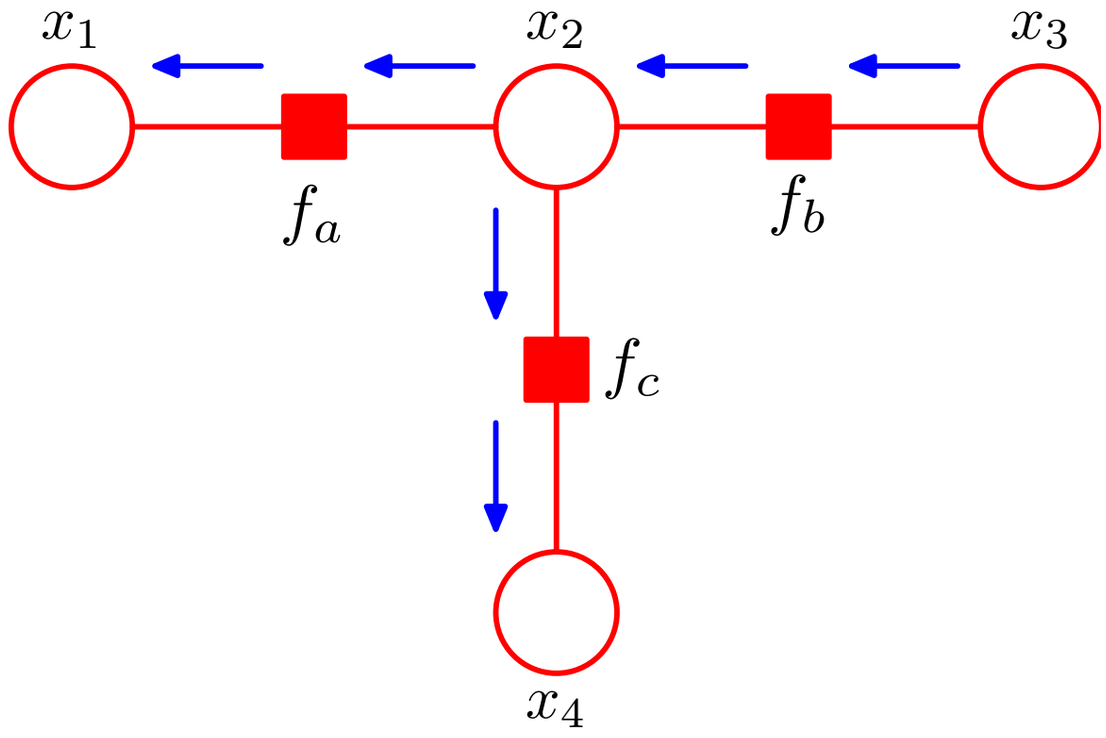


$$\widetilde{p}(\mathbf{x}) = f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4).$$

An example

- Compute second set of messages:

$$
\begin{aligned}
\mu_{x_3 \to f_b}(x_3) &= 1 \\
\mu_{f_b \to x_2}(x_2) &= \sum_{x_3} f_b(x_2, x_3) \\
\mu_{x_2 \to f_a}(x_2) &= \mu_{f_b \to x_2}(x_2)\mu_{f_c \to x_2}(x_2) \\
\mu_{f_a \to x_1}(x_1) &= \sum_{x_2} f_a(x_1, x_2)\mu_{x_2 \to f_a}(x_2) \\
\mu_{x_2 \to f_c}(x_2) &= \mu_{f_a \to x_2}(x_2)\mu_{f_b \to x_2}(x_2) \\
\mu_{f_c \to x_4}(x_4) &= \sum_{x_2} f_c(x_2, x_4)\mu_{x_2 \to f_c}(x_2).
\end{aligned}
$$

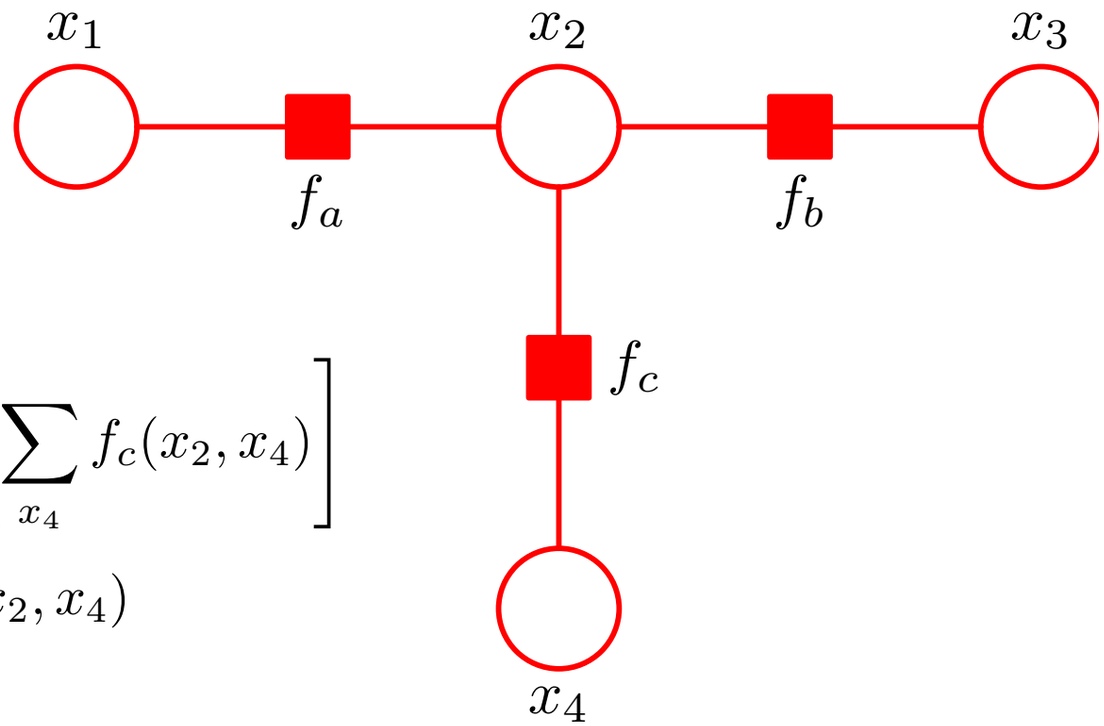


$$\widetilde{p}(\mathbf{x}) = f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4).$$

Verify result

- Compute a specific marginal:

$$
\begin{aligned}
\widetilde{p}(x_2) &= \mu_{f_a \to x_2}(x_2)\mu_{f_b \to x_2}(x_2)\mu_{f_c \to x_2}(x_2) \\
&= \left[\sum_{x_1} f_a(x_1, x_2)\right]\left[\sum_{x_3} f_b(x_2, x_3)\right]\left[\sum_{x_4} f_c(x_2, x_4)\right] \\
&= \sum_{x_1}\sum_{x_2}\sum_{x_4} f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4) \\
&= \sum_{x_1}\sum_{x_3}\sum_{x_4} \widetilde{p}(\mathbf{x})
\end{aligned}
$$



$$\widetilde{p}(\mathbf{x}) = f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4).$$