Graphical Models in Computer Vision

Andreas Geiger

Max Planck Institute for Intelligent Systems
Perceiving Systems

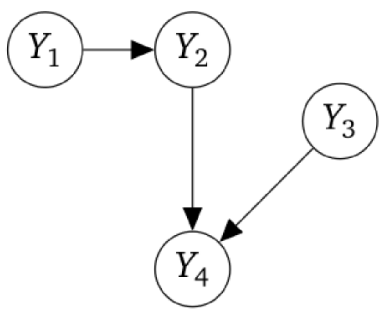
April 25, 2016

Syllabus

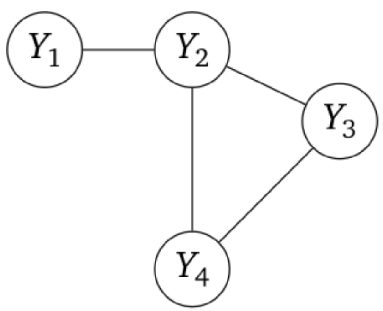
11.04.2016	Introduction
18.04.2016	Graphical Models 1
25.04.2016	Graphical Models 2 (Sand 6/7)
02.05.2016	Graphical Models 3
09.05.2016	Graphical Models 4
23.05.2016	Body Models 1
30.05.2016	Body Models 2
06.06.2016	Body Models 3
13.06.2016	Body Models 4
20.06.2016	Stereo
27.06.2016	Optical Flow
04.07.2016	Segmentation
11.07.2016	Object Detection 1
18.07.2016	Object Detection 2

Todays topic

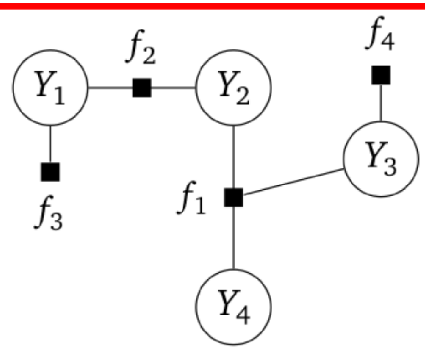
- Recap
  - Belief Networks
  - Markov Random Fields
- Graphical Models
  - Factor Graphs
  - Inference



(a) Bayesian Network

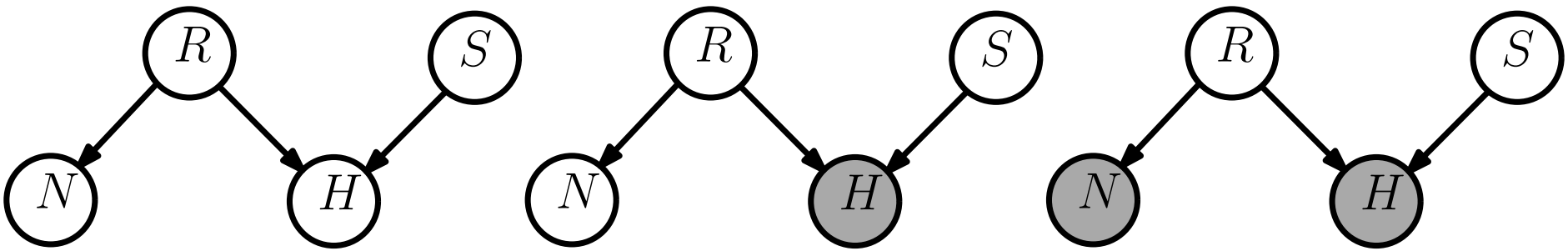


(b) Markov Random Field



(c) Factor Graph

This example as a Belief Network



- Rain, Sprinkler, Holmes wet lawn, Neighbours wet lawn
- This is called a directed graphical model or belief network
  - observing the wet grass
  - observing the neighbours wet grass

Belief Networks

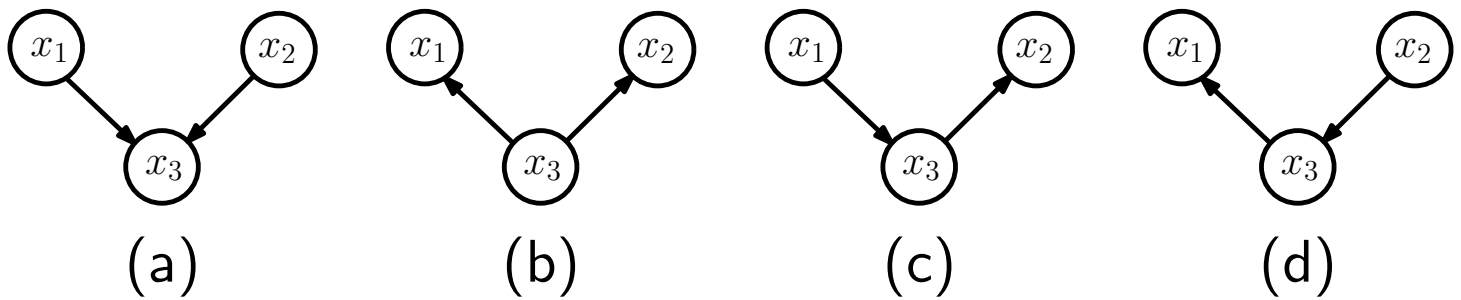
Belief network

A belief network is a distribution of the form

$$p(x_1, \ldots, x_D) = \prod_{i=1}^{D} p(x_i \mid pa(x_i)),$$

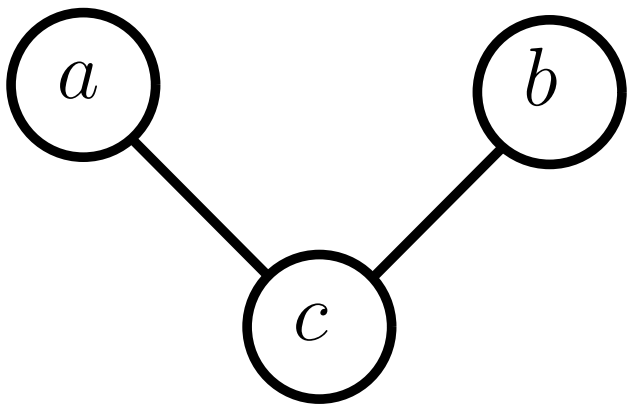
where $pa(x)$ denotes the parental variables of x

Markov Equivalence



- All have the same skeleton
- (b,c,d) have no immoralities
- (b,c,d) have the same set of conditional independence statements
- (a) has immorality (x_1, x_2) and is thus not equivalent

Example for a Markov Random Field (MRF)



$$p(a, b, c) = \frac{1}{Z}\phi_{ac}(a, c)\phi_{bc}(b, c)$$

- Potential functions ϕ_{ab}, ϕ_{bc}

Markov Network

Markov Network

For a set of variables $\mathcal{X} = \{x_1, \ldots, x_D\}$ a Markov network is defined as a product of potentials over the maximal cliques $\mathcal{X}_c$ of the graph $\mathcal{G}$

$$p(x_1, \ldots, x_D) = \frac{1}{Z} \prod_{c=1}^{C} \phi_c(\mathcal{X}_c)$$

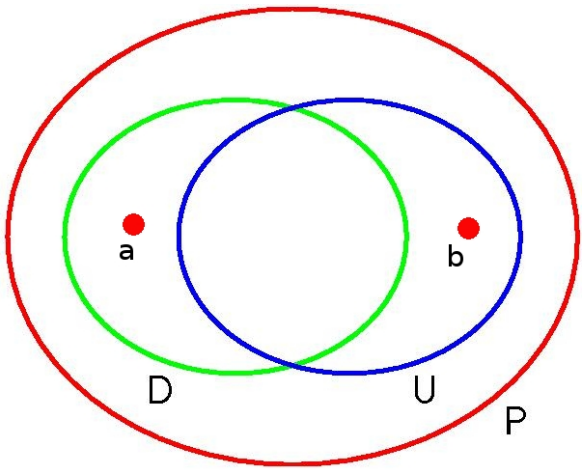
- Special case: cliques of size 2 – pairwise Markov network
- If all potentials are strictly positive this is called a Gibbs distribution

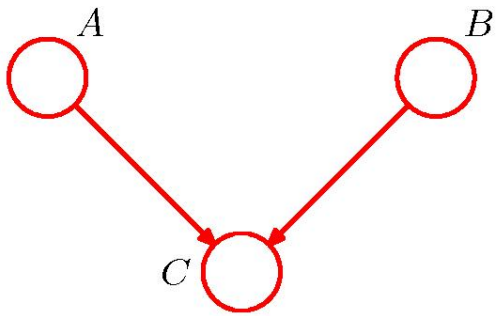
Directed vs. Undirected Models

Perfect Map

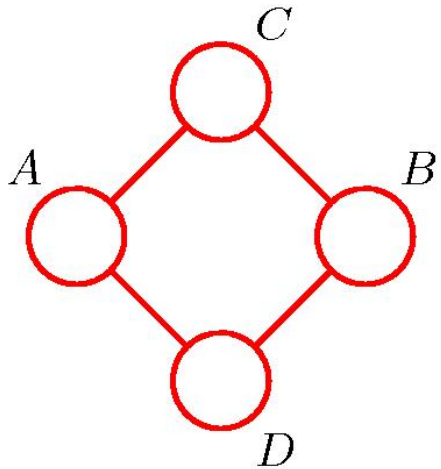
If every conditional independence property of the distribution is reflected in the graph, **and vice versa**, then the graph is said to be a perfect map for that distribution.

- A perfect map: Both I map and a D map of the distribution





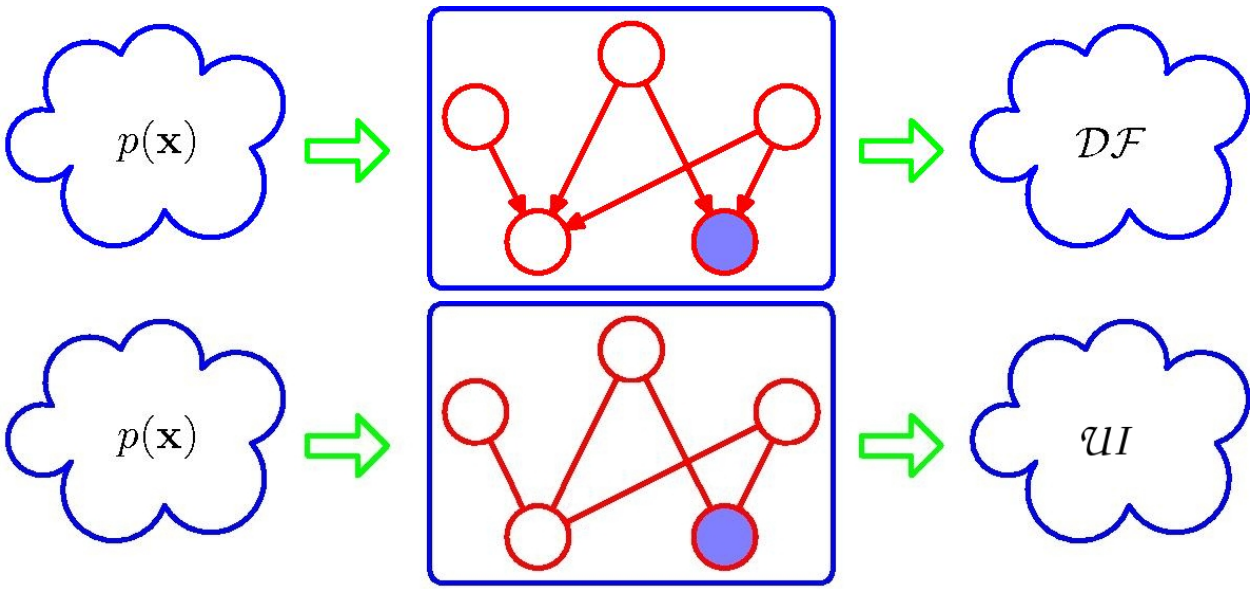
(a)



(b)

- (a) Conditional independence properties cannot be expressed using an undirected graph over the same three variables
- (b) Conditional independence properties cannot be expressed using a directed graph over the same four variables

Filter View of a Graphical Model



- One graph describes a whole family of probability distributions
- Extremes:
  - Fully connected, no constraints, all p pass
  - no connections, only product of marginals may pass

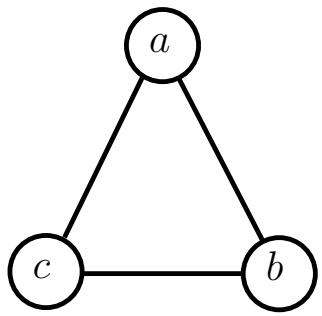
Factor Graphs

Relationship Potentials to Graphs

- Consider this factorization into potential functions:

$$p(a, b, c) = \frac{1}{Z}\phi(a, b)\phi(b, c)\phi(c, a)$$

- What is the corresponding Markov network (graphical model)?



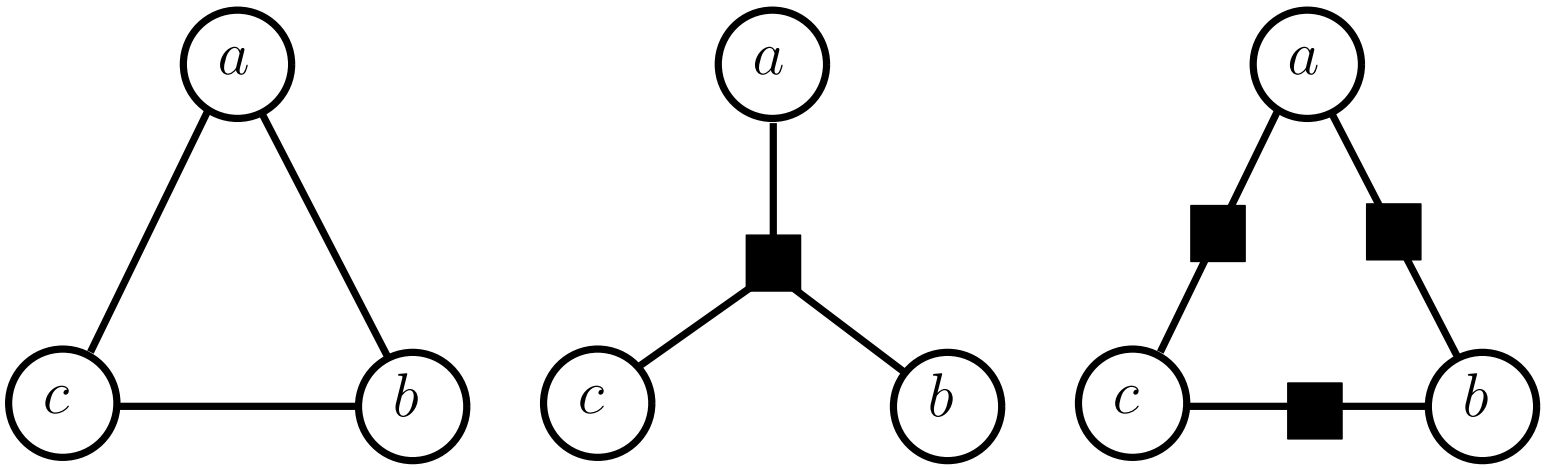
- and which other factorization is represented by this network?

$$p(a, b, c) = \frac{1}{Z}\phi(a, b, c)$$

- The factorization of the *potentials* is not specified by the graph

Relationship Potentials to Graphs

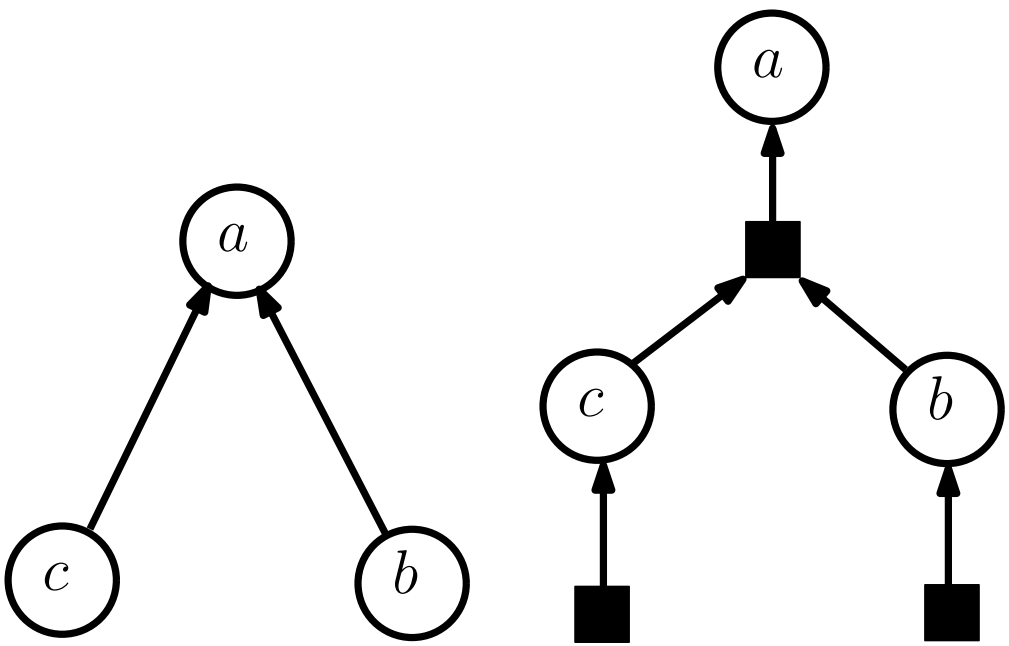
- Now consider we introduce an extra node (a square) for each factor



- Left: Markov Network
- Middle: Factor graph representation of $\phi(a, b, c)$
- Right: Factor graph representation of $\phi(a, b)\phi(b, c)\phi(c, a)$
- Different factor graphs can have the same Markov network (b,c)⇒(a)

Similarly for Directed Graphs

- A directed factor graph also retains the structure of the factorization for a belief network



- But we skip that arrow usually

Factor Graph Definition

Factor Graph

Given a function

$$f(x_1, \ldots, x_n) = \prod_i f_i(\mathcal{X}_i)$$

the factor graph (FG) has a node (represented by a square) for each factor $f_i(\mathcal{X}_i)$ and a variable node (represented by a circle) for each variable x_j.

We typically specify this factorization up to a normalization constant

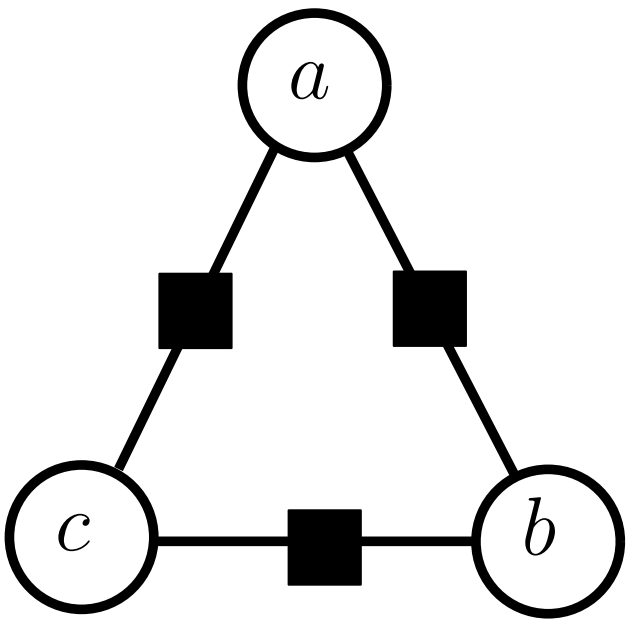
$$p(x_1, \ldots, x_n) = \frac{1}{Z} \prod_i f_i(\mathcal{X}_i)$$

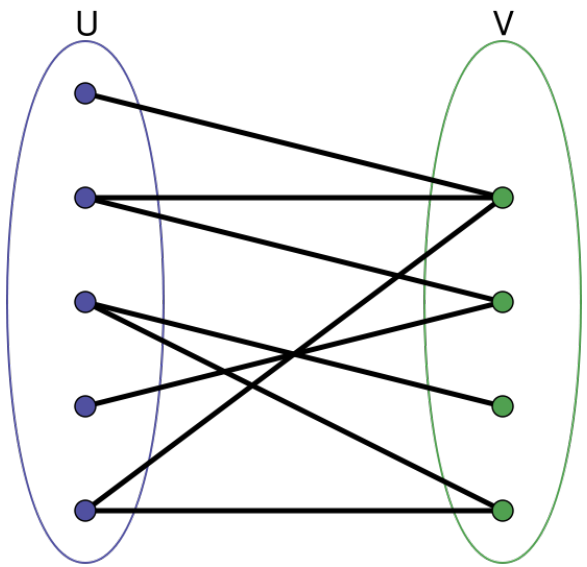
when representing a distribution $p(\cdot)$.

Bipartite Graphs

Bipartite Graphs

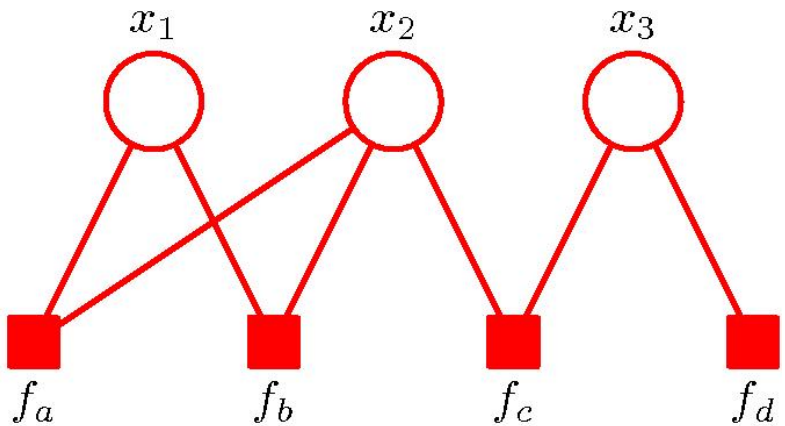
A bipartite graph is a graph whose vertices can be divided into two disjoint sets U and V such that every edge connects a vertex in U to one in V





Factor Graph: Example 1

- Question: which distribution ?



- Answer:

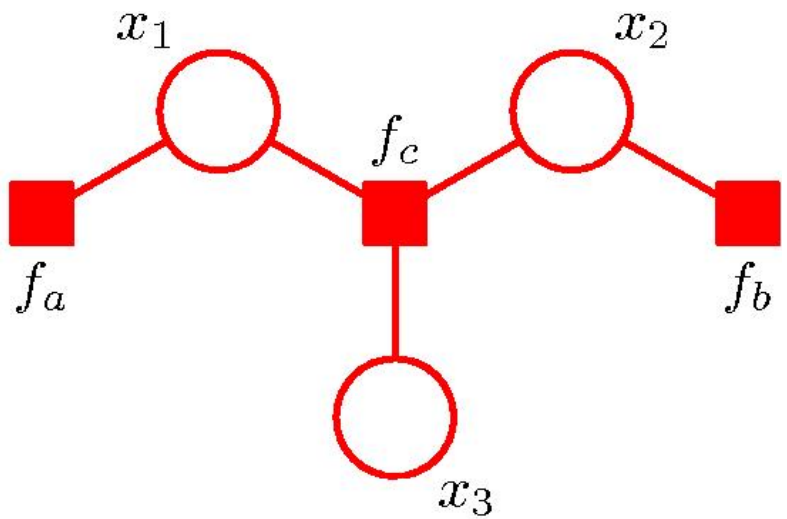
$$p(x) = \frac{1}{Z} f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3)$$

Factor Graph: Example 2

- Question: Which factor graph?

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

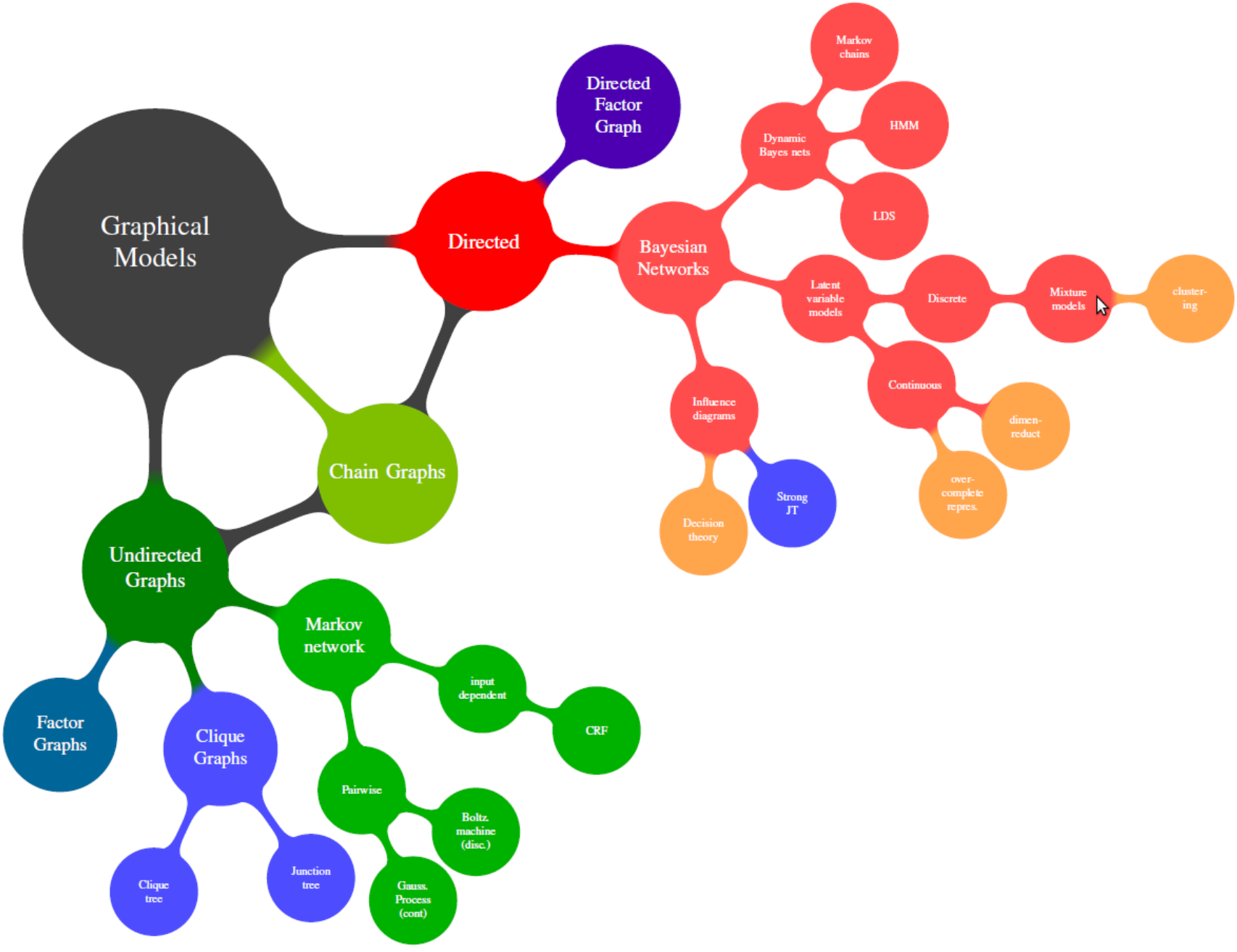
- Answer:



Summary

With graphical models we represent probability distributions graphically:

- **Belief networks**
  - Directed graphs
  - Useful tool to express causal dependencies
- **Markov networks**
  - Undirected graphs
  - Specification in terms of local cliques
  - Graph does not expose causal dependencies
- **Factor graphs**
  - Makes the factorization explicit
  - Useful representation from an "implementation viewpoint"


Graphical Models
Directed
Directed Factor Graph
Bayesian Networks
Markov chains
Dynamic Bayes nets
HMM
LDS
Latent variable models
Discrete
Mixture models
cluster-ing
Continuous
dimen-reduct
over-complete repres.
Influence diagrams
Decision theory
Strong JT
Chain Graphs
Undirected Graphs
Markov network
input dependent
CRF
Pairwise
Boltz. machine (disc.)
Gauss. Process (cont)
Factor Graphs
Clique Graphs
Clique tree
Junction tree

Inference

Inference

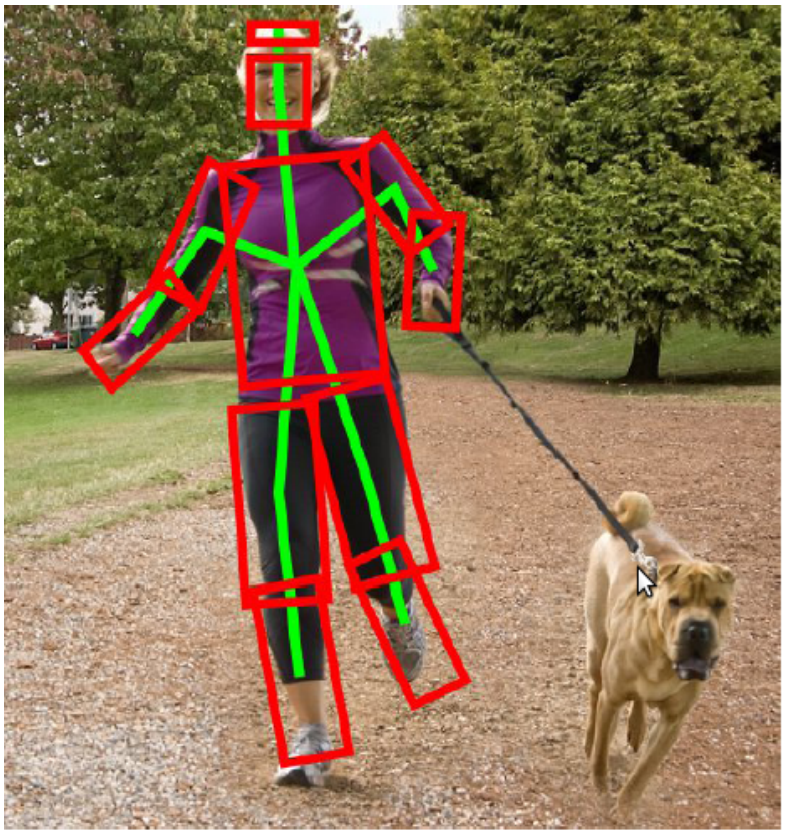
- Given distribution

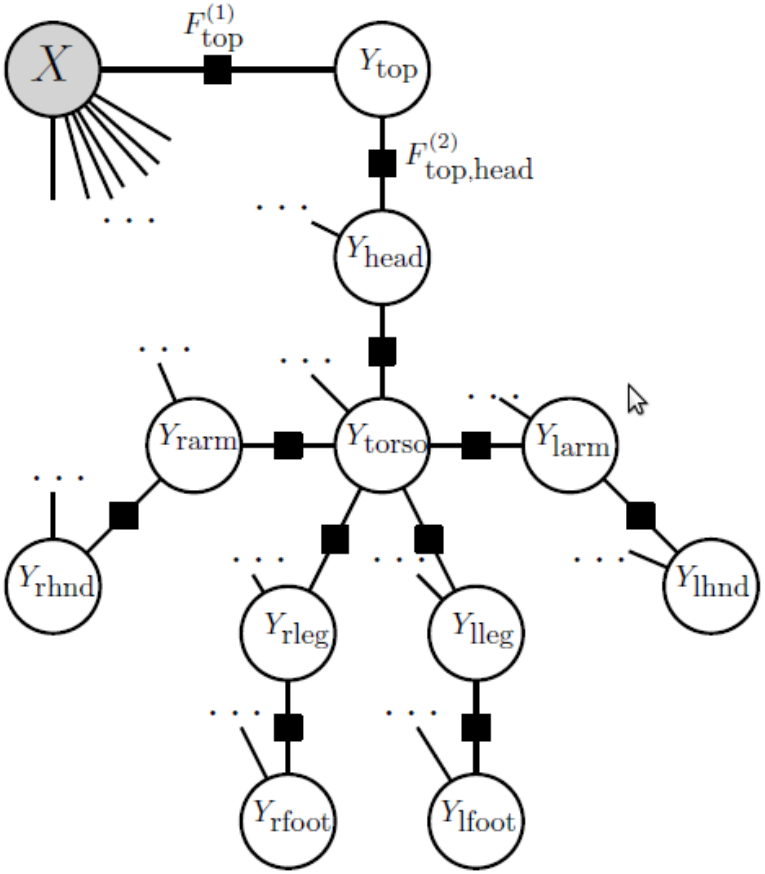
$$p(x_1, \ldots, x_n)$$

- Compute functions of the distribution
  - Marginal distributions
  - Most probable state
  - Mean
  - Conditionals
- **Today:** Inference in singly-connected graph (chains, trees)
- **Next week:** Inference in loopy graphs

Example: Pictorial Structures

- Find body parts

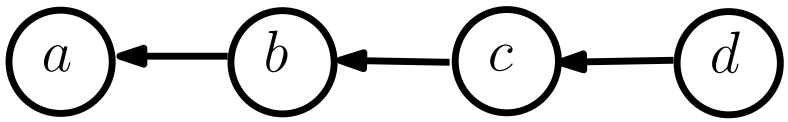




[Fischler & Elschlager, 1973] [Felsenzwalb & Huttenlocher, 2000]

Variable Elimination

- Consider the following Markov chain ($a, b, c, d \in \{0, 1\}$)

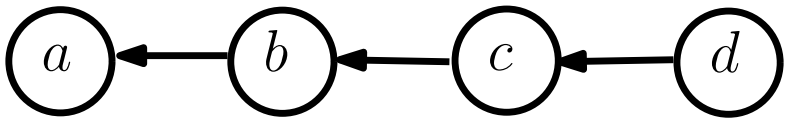


with distribution

$$p(a, b, c, d) = p(a \mid b)p(b \mid c)p(c \mid d)p(d)$$

- Task: compute the marginal $p(a)$
- How would you do it?

Variable Elimination



$$
\begin{aligned}
p(a) &= \sum_{b,c,d} p(a,b,c,d) \\
&= \sum_{b,c,d} p(a \mid b)p(b \mid c)p(c \mid d)p(d)
\end{aligned}
$$

- Naive: $2 \times 2 \times 2 = 8$ states to sum over
- Re-order summation:

$$
p(a) = \sum_{b,c} p(a \mid b)p(b \mid c) \underbrace{\sum_{d} p(c \mid d)p(d)}_{\gamma_d(c)}
$$

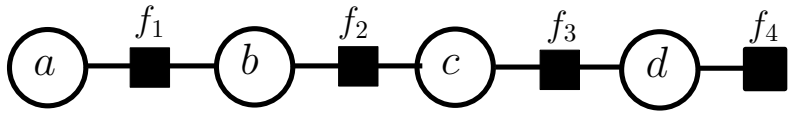
Variable Elimination

$$
\begin{aligned}
p(a) &= \sum_{b,c} p(a \mid b) p(b \mid c) \underbrace{\sum_{d} p(c \mid d) p(d)}_{\gamma_d(c)} \\
p(a) &= \sum_{b} p(a \mid b) \underbrace{\sum_{c} p(b \mid c) \gamma_d(c)}_{\gamma_c(b)} \\
p(a) &= \sum_{b} p(a \mid b) \gamma_c(b)
\end{aligned}
$$

- We need $2 + 2 + 2 = 6$ calculations! In general?
- For a chain of length n this scales linearly: $2\,(n-1)$ operations
- Naive approach: 2^{n-1} operations!

Inference in Factor Graphs

Variable Elimination on Chain Structured Factor Graphs



$$p(a,b,c,d) = \frac{1}{Z} f_1(a,b) f_2(b,c) f_3(c,d) f_4(d)$$

$$\begin{aligned} p(a,b,c) &= \sum_d p(a,b,c,d) \\ &= \frac{1}{Z} f_1(a,b) f_2(b,c) \underbrace{\sum_d f_3(c,d) f_4(d)}_{\mu_{d\to c}(c)} \end{aligned}$$

$$p(a,b) = \sum_c p(a,b,c) = \frac{1}{Z} f_1(a,b) \underbrace{\sum_c f_2(b,c) \mu_{d\to c}(c)}_{\mu_{c\to b}(b)}$$

Inference in Chain Structured Factor Graphs

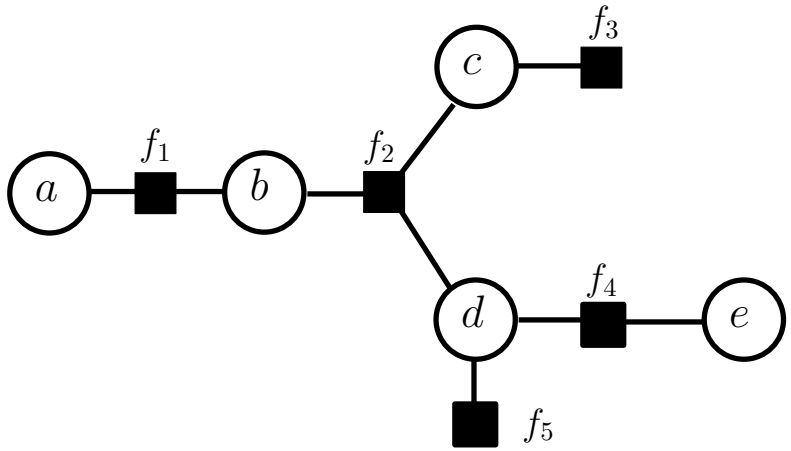
- Simply recurse further:

$$p(a) = \sum_b p(a, b) = \frac{1}{Z} \sum_b f_1(a, b)\, \mu_{c \to b}(b) = \frac{1}{Z} \mu_{b \to a}(a)$$

- How to get the marginal distribution?
- $\mu_{m \to n}(n)$ carries the information beyond m
- We did not need the factors yet
- But we will see that making a distinction is helpful

Inference in Tree Structured Factor Graphs

- Consider a branching graph:



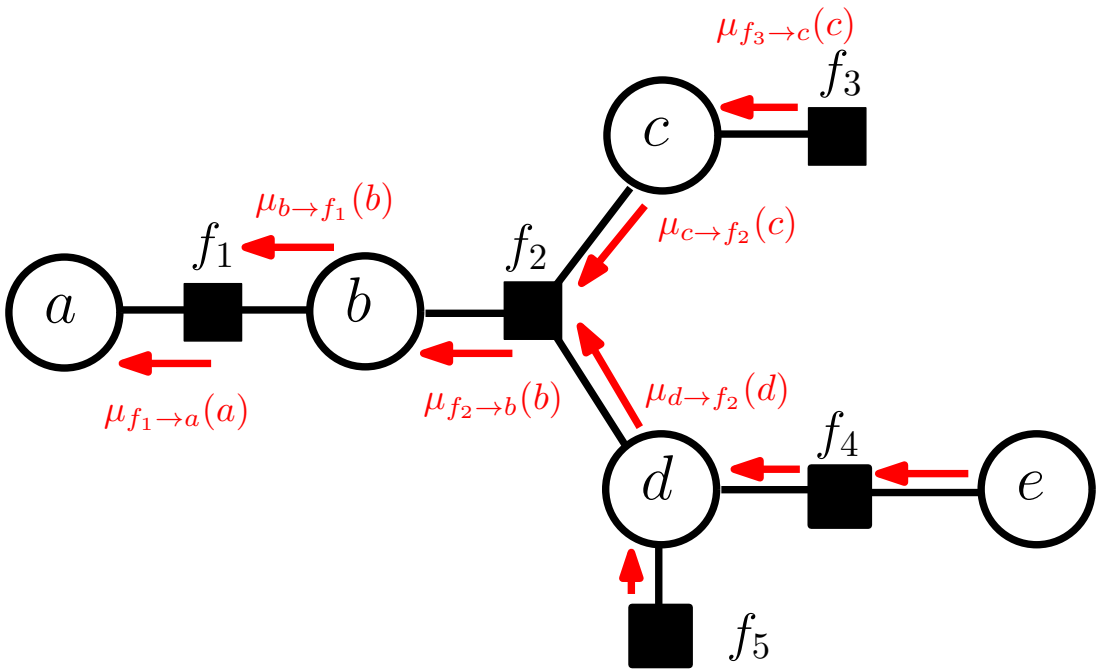
with factors

$$f_1(a,b)f_2(b,c,d)f_3(c)f_4(d,e)f_5(d)$$

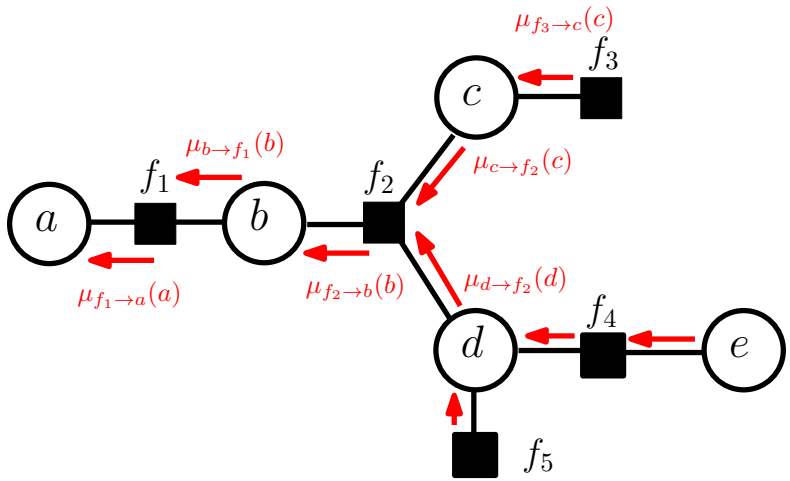
- How to find marginal $p(a,b)$?

Inference in Tree Structured Factor Graphs

- Idea: compute messages



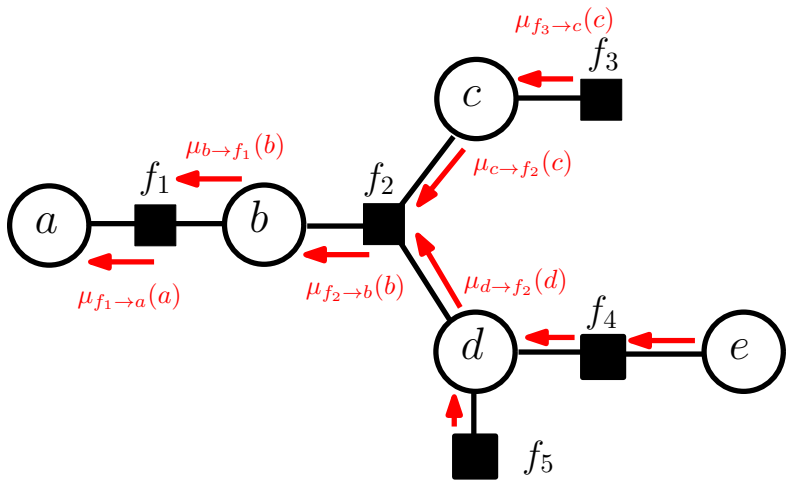
Inference in Tree Structured Factor Graphs



$$p(a,b) = f_1(a,b) \underbrace{\sum_{c,d,e} f_2(b,c,d) f_3(c) f_5(d) f_4(d,e)}_{\mu_{f_2\to b}(b)}$$

$$\mu_{f_2\to b}(b) = \sum_{c,d} f_2(b,c,d) f_3(c) f_5(d) \sum_{e} f_4(d,e)$$

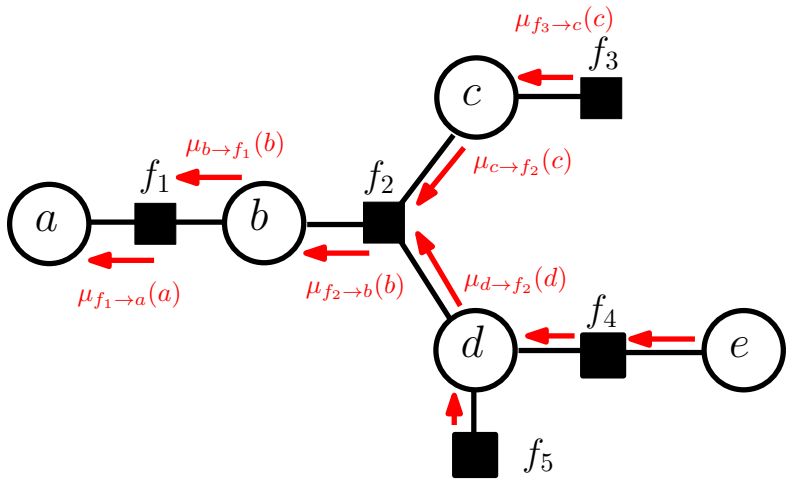
Inference in Tree Structured Factor Graphs



$$p(a,b) = f_1(a,b) \underbrace{\sum_{c,d,e} f_2(b,c,d) f_3(c) f_5(d) f_4(d,e)}_{\mu_{f_2\to b}(b)}$$

$$\mu_{f_2\to b}(b) = \sum_{c,d} f_2(b,c,d) \underbrace{f_3(c)}_{\mu_{c\to f_2}(c)} \underbrace{f_5(d) \sum_{e} f_4(d,e)}_{\mu_{d\to f_2}(d)}$$

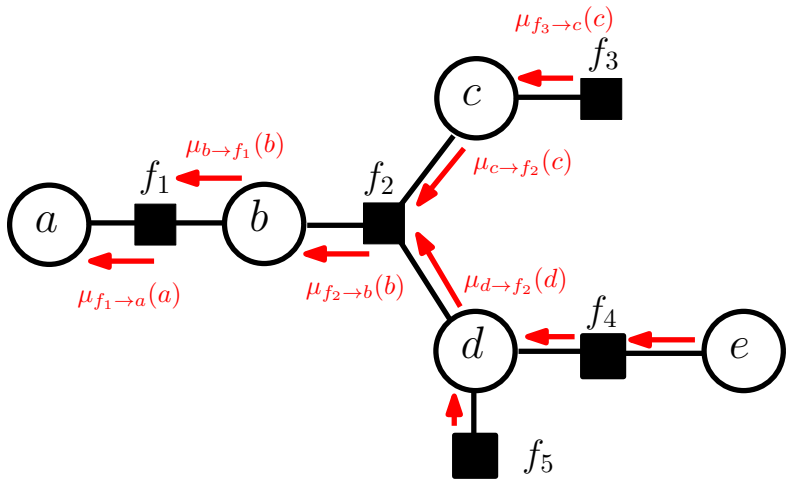
Inference in Tree Structured Factor Graphs



$$p(a,b) = f_1(a,b) \underbrace{\sum_{c,d,e} f_2(b,c,d) f_3(c) f_5(d) f_4(d,e)}_{\mu_{f_2\to b}(b)}$$

$$\mu_{f_2\to b}(b) = \sum_{c,d} f_2(b,c,d) \mu_{c\to f_2}(c) \mu_{d\to f_2}(d)$$

Factor-to-Variable Messages



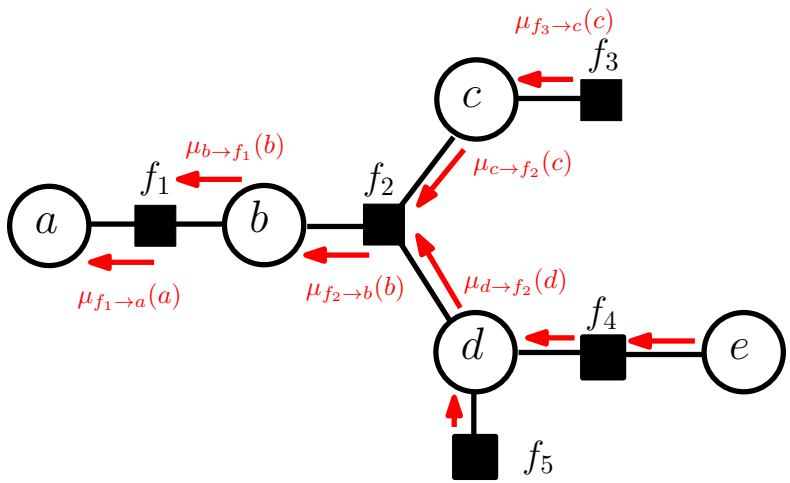
- Here (repeated from last slide):

$$\mu_{f_2 \to b}(b) = \sum_{c,d} f_2(b, c, d) \mu_{c \to f_2}(c) \mu_{d \to f_2}(d)$$

- More general:

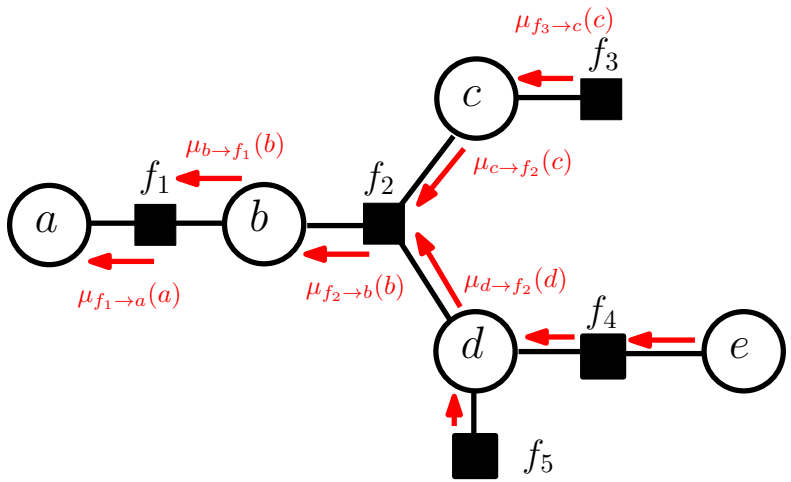
$$\mu_{f \to x}(x) = \sum_{y \in \mathcal{X}_f \setminus x} \phi_f(\mathcal{X}_f) \prod_{y \in \{\mathsf{ne}(f) \setminus x\}} \mu_{y \to f}(y)$$

Variable-to-Factor Messages



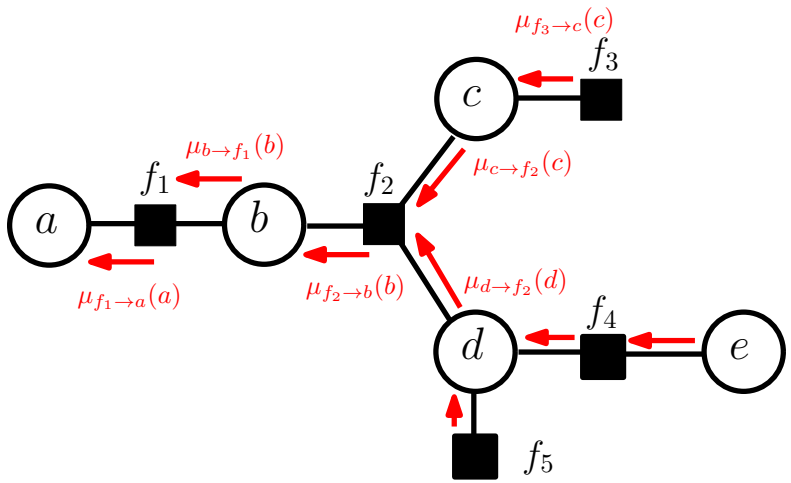
$$\mu_{d \to f_2}(d) = f_5(d) \sum_e f_4(d, e)$$

Variable-to-Factor Messages



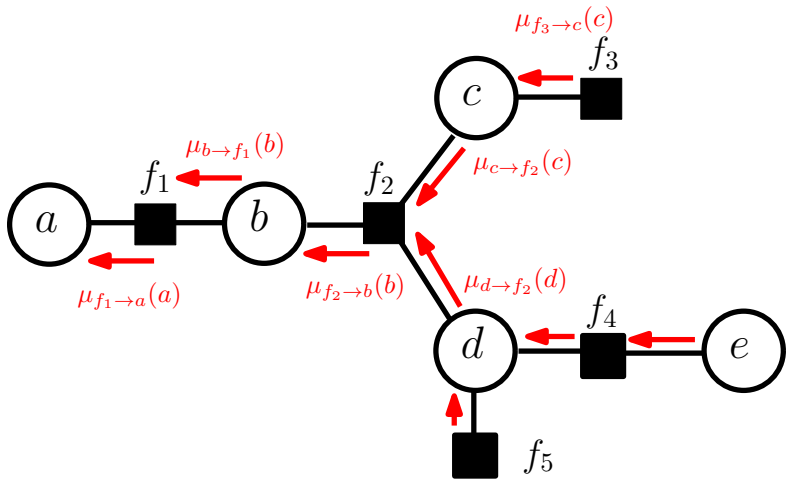
$$\mu_{d \to f_2}(d) = \underbrace{f_5(d)}_{\mu_{f_5 \to d}(d)} \underbrace{\sum_e f_4(d, e)}_{\mu_{f_4 \to d}(d)}$$

Variable-to-Factor Messages



$$\mu_{d \to f_2}(d) = \mu_{f_5 \to d}(d) \mu_{f_4 \to d}(d)$$

Variable-to-Factor Messages



- Here (repeated from last slide):

$$\mu_{d\to f_2}(d) = \mu_{f_5\to d}(d)\mu_{f_4\to d}(d)$$

- General:

$$\mu_{x\to f}(x) = \prod_{g\in\{\mathsf{ne}(x)\setminus f\}} \mu_{g\to x}(x)$$

Comments

- Many subscripts :)
- Once computed, messages can be re-used
- Important observation: All marginals $(p(c), p(d), p(c, d), ...)$ can be written as a function of messages
- We need an algorithm to compute all messages
- For marginal inference: Sum-product algorithm

Sum-Product Algorithm

Sum-Product Algorithm – Overview

Belief Propagation:

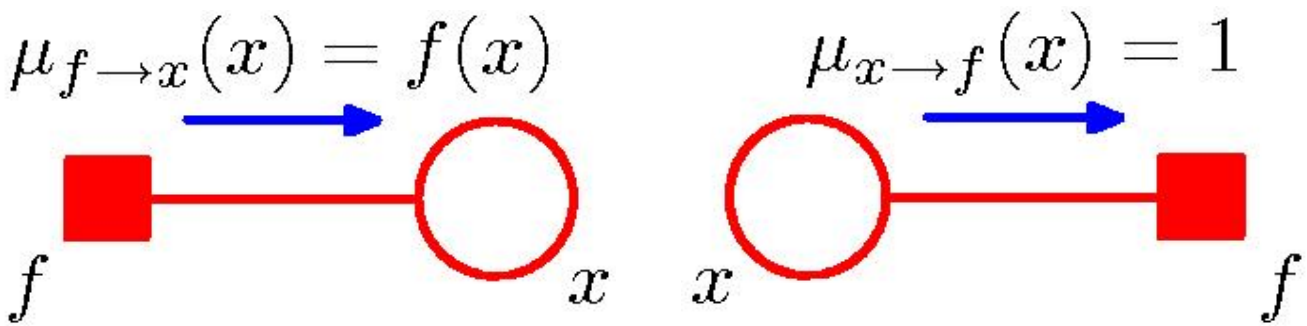
- Algorithm to compute all messages efficiently
- Assumes that the graph is singly-connected (chain, tree)

Algorithm:

1. Initialization
2. Variable to Factor message
3. Factor to Variable message
4. Repeat until all messages have been calculated
5. Calculate the desired marginals from the messages

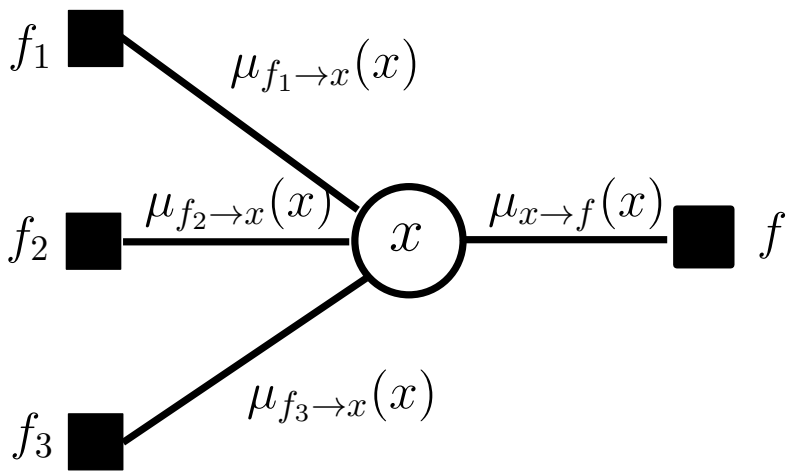
1. Initialization

- Messages from extremal node factors are initialized to the factor
- Messages from extremal variable nodes are set to unity



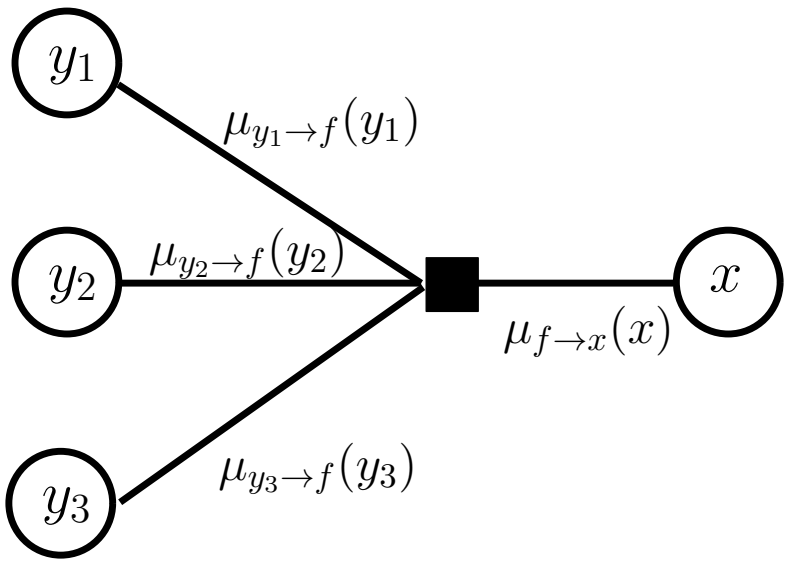
2. Variable-to-Factor Message

$$\mu_{x \to f}(x) = \prod_{g \in \{\mathsf{ne}(x) \setminus f\}} \mu_{g \to x}(x)$$



3. Factor-to-Variable Message

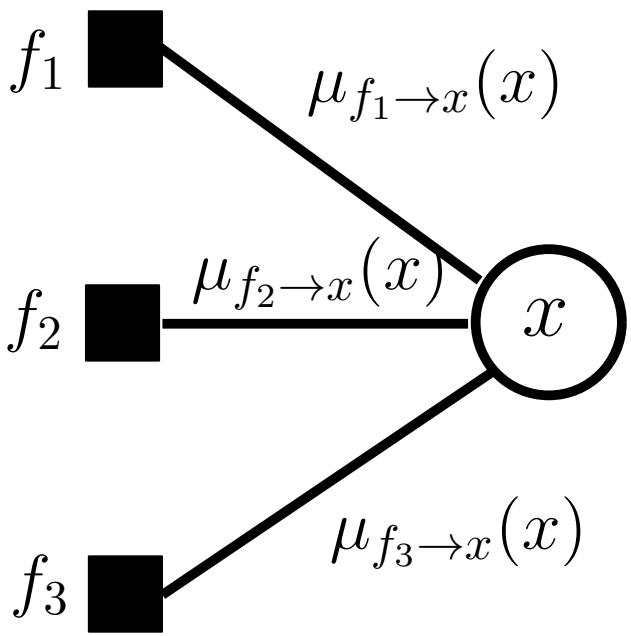
$$\mu_{f\to x}(x) = \sum_{y\in\mathcal{X}_f\setminus x} \phi_f(\mathcal{X}_f) \prod_{y\in\{\mathsf{ne}(f)\setminus x\}} \mu_{y\to f}(y)$$



- We sum over all states in the set of variables
- This explains the name for the algorithm (sum-product)
- Great, this is tractable now! Or not?

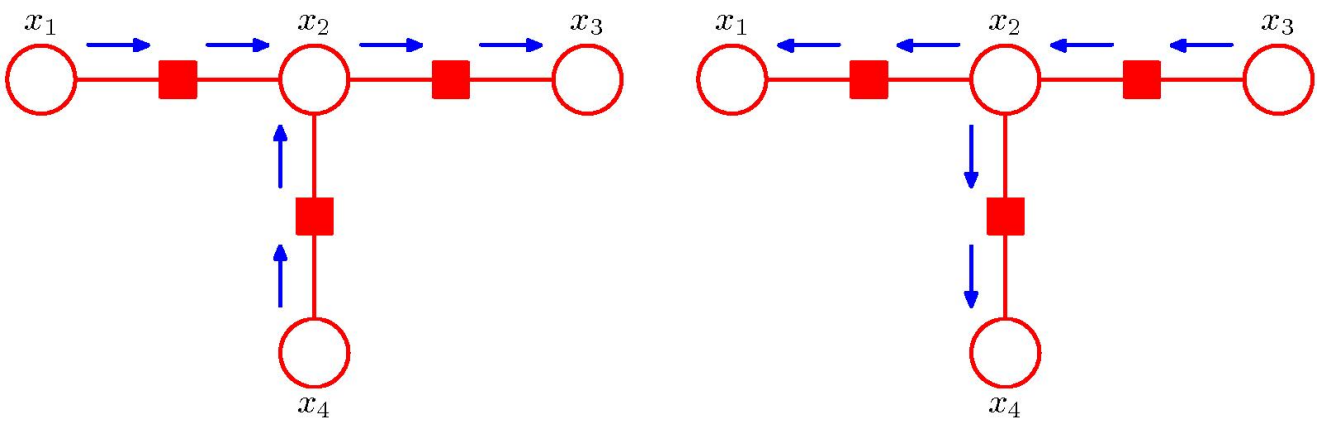
5. Calculate Marginals

$$p(x) \propto \prod_{f \in \mathsf{ne}(x)} \mu_{f \to x}(x)$$



Message Ordering

- Messages depend on previous computed messages
- Only extremal nodes/factors do not depend on other messages
- To compute all messages in the graph
  1. leaf-to-root: (pick root node, compute messages pointing towards root)
  2. root-to-leave: (compute messages pointing away from root)



Log-Messages

- In large graphs, messages may become very small/big
- Work with log-messages instead $\lambda = \log \mu$
- Variable-to-factor messages

$$\mu_{x \to f}(x) = \prod_{g \in \{\mathsf{ne}(x) \setminus f\}} \mu_{g \to x}(x)$$

then becomes

$$\lambda_{x \to f}(x) = \sum_{g \in \{\mathsf{ne}(x) \setminus f\}} \lambda_{g \to x}(x)$$

Log-Messages

- Work with log-messages instead $\lambda = \log \mu$
- Factor-to-variable messages

$$\mu_{f \to x}(x) = \sum_{y \in \mathcal{X}_f \setminus x} \Phi_f(\mathcal{X}_f) \prod_{y \in \{\mathsf{ne}(f) \setminus x\}} \mu_{y \to f}(y)$$

then become

$$\lambda_{f \to x}(x) = \log \left(\sum_{y \in \mathcal{X}_f \setminus x} \Phi(\mathcal{X}_f) \exp \left[\sum_{y \in \{\mathsf{ne}(f) \setminus x\}} \lambda_{y \to f}(y) \right] \right)$$

Trick

- Log-Factor-to-Variable Message:

$$\lambda_{f\to x}(x) = \log \sum_{y\in\mathcal{X}_f\setminus x} \Phi_f(\mathcal{X}_f) \exp \sum_{y\in\{\mathsf{ne}(f)\setminus x\}} \lambda_{y\to f}(y)$$

- Large numbers $\exp(\dots)$ lead to numerical instability
- Use the following equality:

$$\log \sum_i \exp(v_i) = \alpha + \log \sum_i \exp(v_i - \alpha)$$

with $\alpha = \max \lambda_{y\to f}(y)$

So far..

- So far marginals
- Now: finding the maximal state

Max-Product Algorithm

Finding the maximal state: Max-Product

- For a given distribution $p(x)$ find the most likely state:

$$x^* = \underset{x_1,\ldots,x_n}{\operatorname{argmax}}\, p(x_1, \ldots, x_n)$$

- Again use factorization structure to distribute maximisation to local computations
- Chain example:

$$p(x_1, x_2, x_3, x_4) = \phi(x_1, x_2)\phi(x_2, x_3)\phi(x_3, x_4)$$

Example: Chain

$$
\begin{aligned}
\max_x p(x) &= \max_{x_1,x_2,x_3,x_4} \phi(x_1,x_2)\phi(x_2,x_3)\phi(x_3,x_4) \\
&= \max_{x_1,x_2,x_3} \phi(x_1,x_2)\phi(x_2,x_3) \underbrace{\max_{x_4} \phi(x_3,x_4)}_{\gamma_{x_4}(x_3)} \\
&= \max_{x_1,x_2} \phi(x_1,x_2) \underbrace{\max_{x_3} \phi(x_2,x_3)\gamma_{x_4}(x_3)}_{\gamma_{x_3}(x_2)} \\
&= \max_{x_1} \underbrace{\max_{x_2} \phi(x_1,x_2)\gamma_{x_3}(x_2)}_{\gamma_{x_2}(x_1)} \\
&= \max_{x_1} \gamma_{x_2}(x_1)
\end{aligned}
$$

- Is this what we wanted to compute in the beginning?

Example: Chain

- Once messages are computed, find the optimal values:

$$
\begin{aligned}
x_1^* &= \operatorname*{argmax}_{x_1} \gamma_{x_2}(x_1) \\
x_2^* &= \operatorname*{argmax}_{x_2} \phi(x_1^*, x_2)\gamma_{x_3}(x_2) \\
x_3^* &= \operatorname*{argmax}_{x_3} \phi(x_2^*, x_3)\gamma_{x_4}(x_3) \\
x_4^* &= \operatorname*{argmax}_{x_4} \phi(x_3^*, x_4)
\end{aligned}
$$

- This is called backtracking (an application of dynamic programming)
- Also know as "Viterbi" algorithm

Be careful: not maximal marginal states!

- The most likely state

$$x^* = \underset{x_1,\ldots,x_n}{\operatorname{argmax}}\, p(x_1, \ldots, x_n)$$

  does not need to be the one for which the marginals are maximized!

- For all $i = 1, \ldots, n$

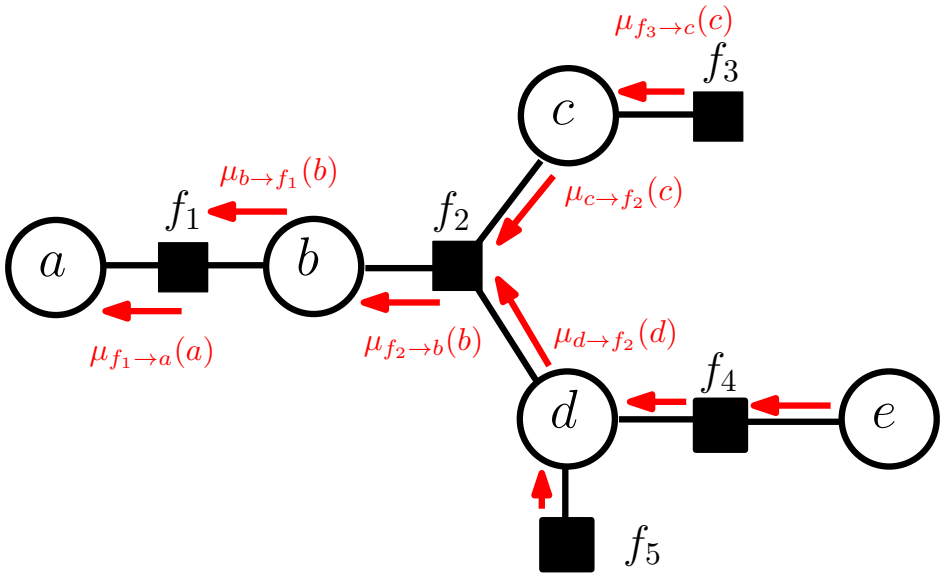
$$x_i^* = \underset{x_i}{\operatorname{argmax}}\, p(x_i)$$

- Example:

	$x = 0$	$x = 1$
$y = 0$	0.3	0.4
$y = 1$	0.3	0.0

Trees

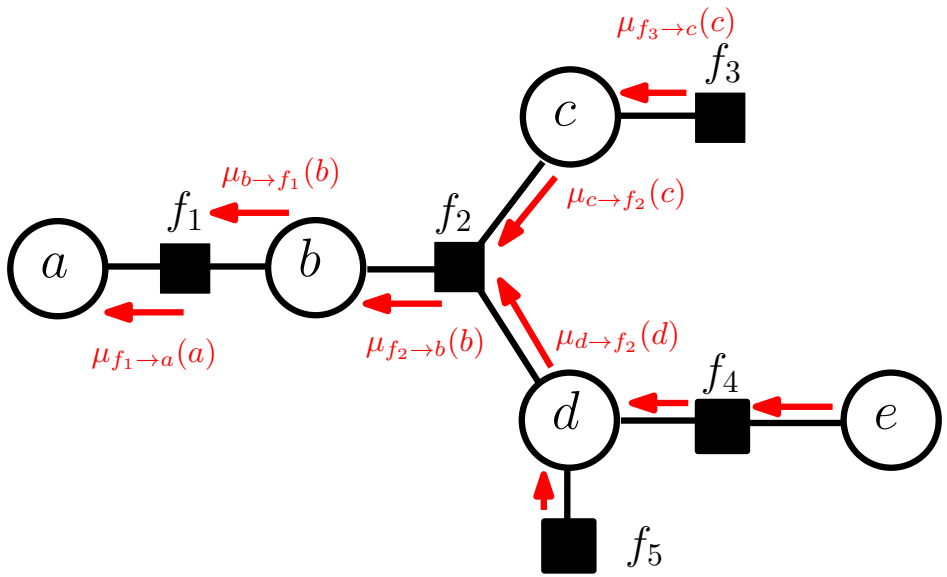
- Spot the messages:



$$\begin{aligned}
\max_{a,b,c,d,e} f(\cdot) &= \max_{a,b,c,d,e} f_1(a,b) f_2(b,c,d) f_3(c) f_4(d,e) f_5(d) \\
&= \max_a \max_b f_1(a,b) \max_{c,d} f_2(b,c,d) f_3(c) f_5(d) \max_e f_4(d,e)
\end{aligned}$$

Trees

- Spot the messages:

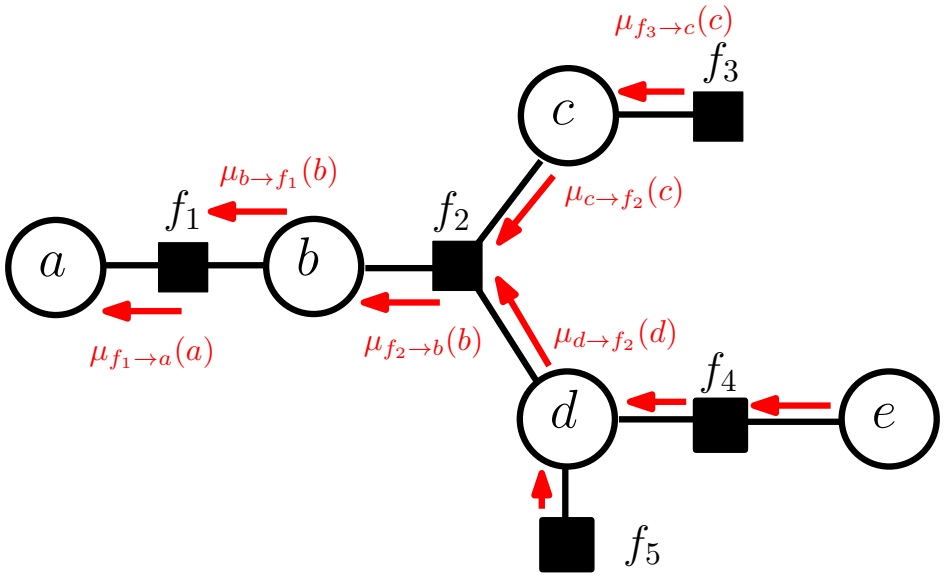


$$\max_{a,b,c,d,e} f(\cdot) = \max_{a,b,c,d,e} f_1(a,b) f_2(b,c,d) f_3(c) f_4(d,e) f_5(d)$$

$$= \max_a \max_b f_1(a,b) \max_{c,d} f_2(b,c,d) f_3(c) \underbrace{f_5(d)}_{\mu_{f_5 \to d}(d)} \underbrace{\max_e f_4(d,e)}_{\mu_{f_4 \to d}(d)}$$

Trees

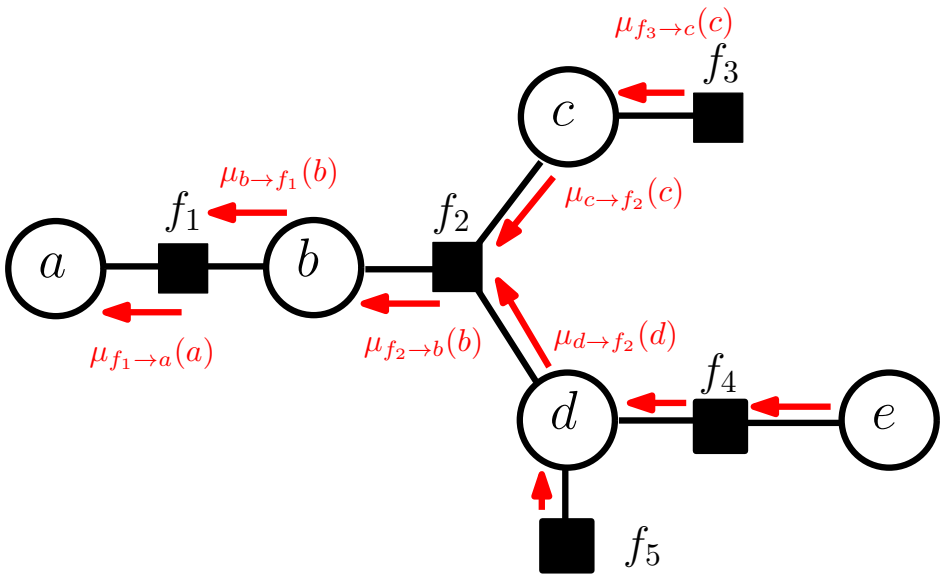
- Spot the messages:



$$\max_{a,b,c,d,e} f(\cdot) = \max_{a,b,c,d,e} f_1(a,b) f_2(b,c,d) f_3(c) f_4(d,e) f_5(d)$$
$$= \max_a \max_b f_1(a,b) \max_{c,d} f_2(b,c,d) f_3(c) \mu_{f_5 \to d}(d) \mu_{f_4 \to d}(d)$$

Trees

- Spot the messages:

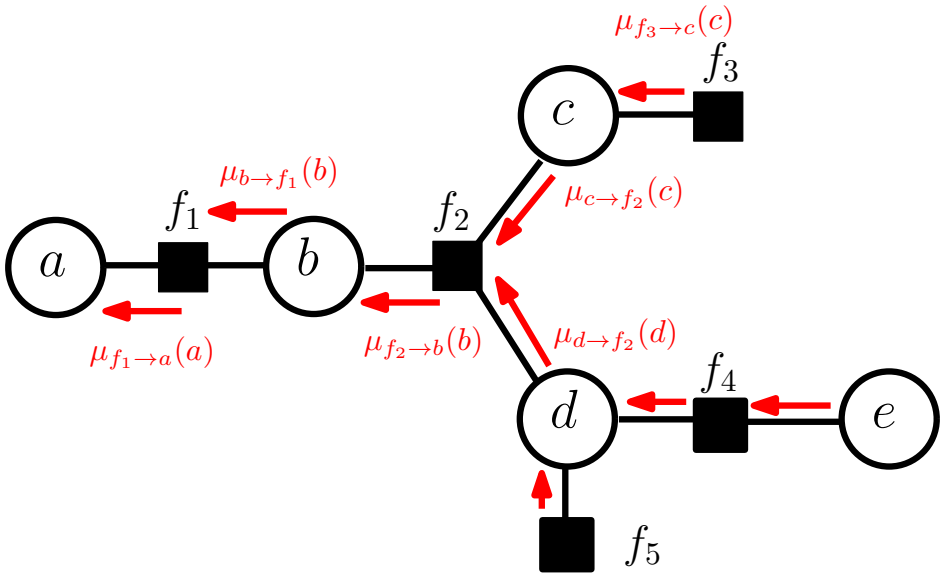


$$\max_{a,b,c,d,e} f(\cdot) = \max_{a,b,c,d,e} f_1(a,b) f_2(b,c,d) f_3(c) f_4(d,e) f_5(d)$$

$$= \max_a \max_b f_1(a,b) \max_{c,d} f_2(b,c,d) \underbrace{f_3(c)}_{\mu_{c\to f_2}(c)} \underbrace{\mu_{f_5\to d}(d)\mu_{f_4\to d}(d)}_{\mu_{d\to f_2}(d)}$$

Trees

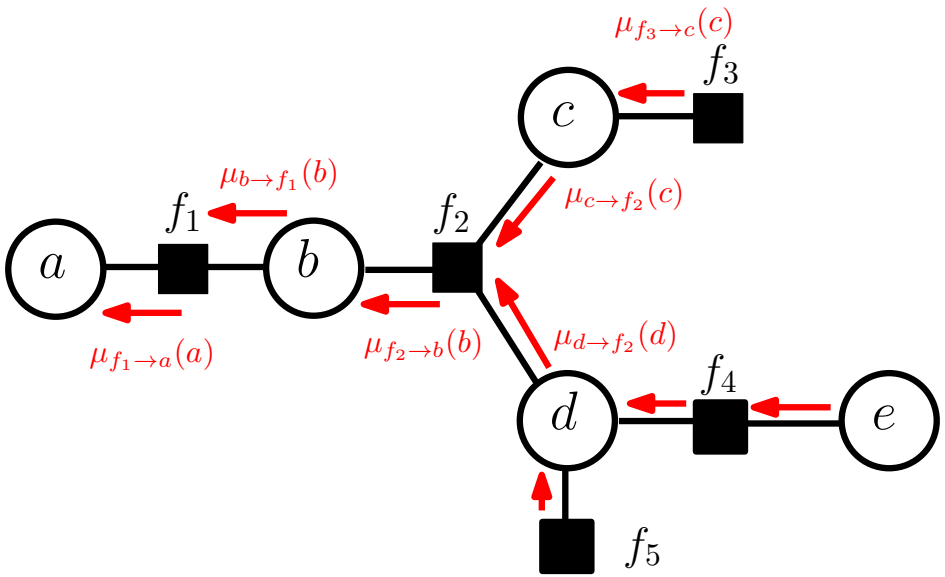
- Spot the messages:



$$\begin{aligned}
\max_{a,b,c,d,e} f(\cdot) &= \max_{a,b,c,d,e} f_1(a,b) f_2(b,c,d) f_3(c) f_4(d,e) f_5(d) \\
&= \max_a \max_b f_1(a,b) \max_{c,d} f_2(b,c,d) \mu_{c \to f_2}(c) \mu_{d \to f_2}(d)
\end{aligned}$$

Trees

- Spot the messages:

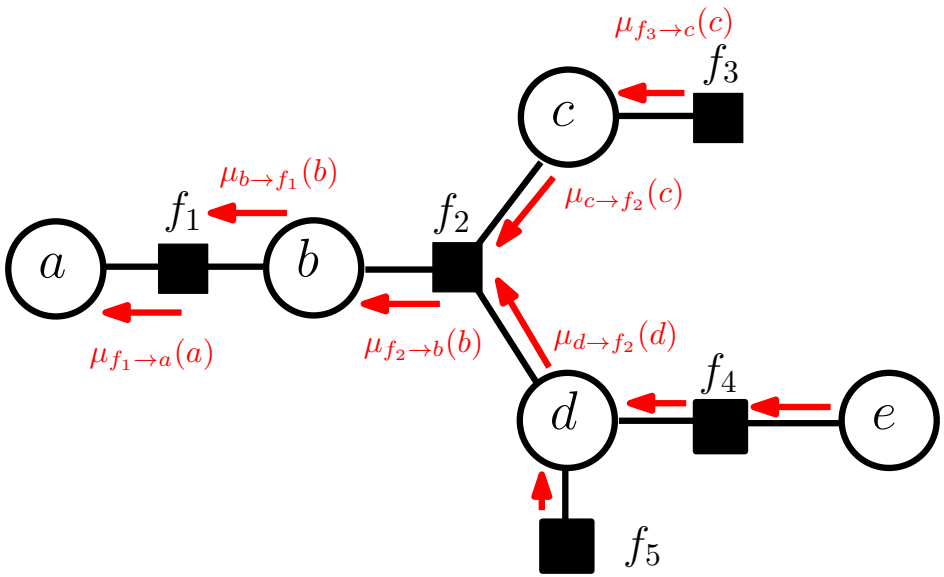


$$\max_{a,b,c,d,e} f(\cdot) = \max_{a,b,c,d,e} f_1(a,b)f_2(b,c,d)f_3(c)f_4(d,e)f_5(d)$$

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

Trees

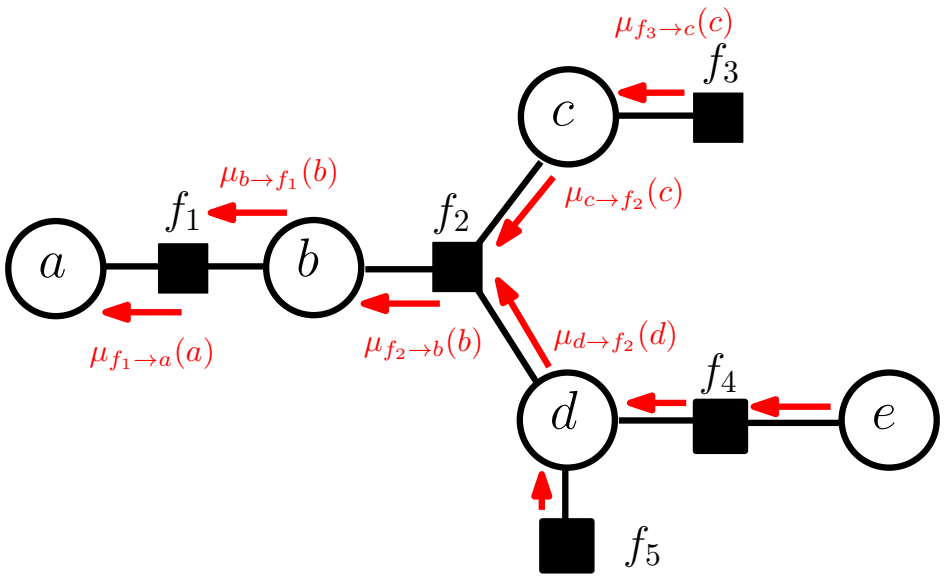
- Spot the messages:



$$\max_{a,b,c,d,e} f(\cdot) = \max_{a,b,c,d,e} f_1(a,b) f_2(b,c,d) f_3(c) f_4(d,e) f_5(d)$$
$$= \max_a \max_b f_1(a,b) \mu_{f_2 \to b}(b)$$

Trees

- Spot the messages:

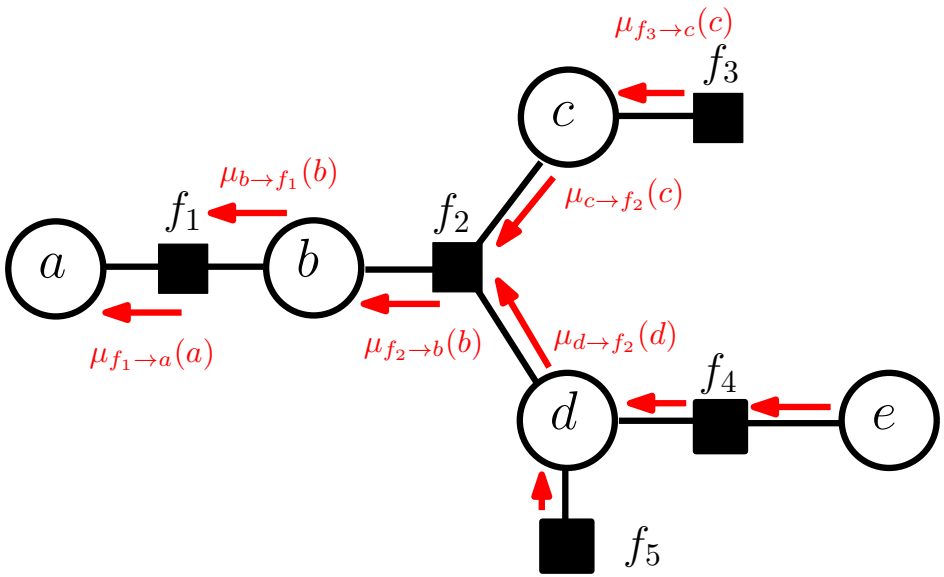


$$\max_{a,b,c,d,e} f(\cdot) = \max_{a,b,c,d,e} f_1(a,b) f_2(b,c,d) f_3(c) f_4(d,e) f_5(d)$$

$$= \max_a \max_b f_1(a,b) \underbrace{\mu_{f_2 \to b}(b)}_{\mu_{b \to f_1}(b)}$$

Trees

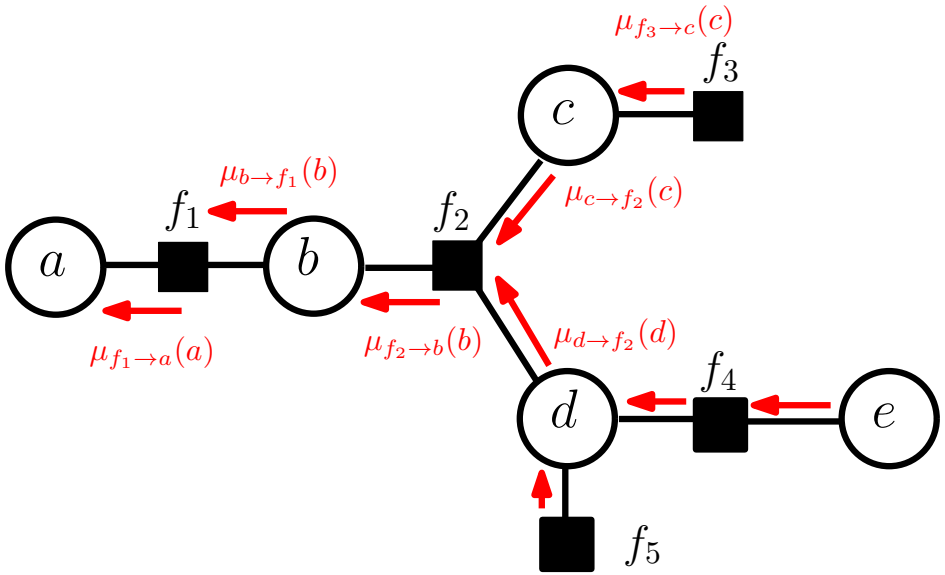
- Spot the messages:



$$\max_{a,b,c,d,e} f(\cdot) = \max_{a,b,c,d,e} f_1(a,b) f_2(b,c,d) f_3(c) f_4(d,e) f_5(d)$$
$$= \max_a \max_b f_1(a,b) \mu_{b \to f_1}(b)$$

Trees

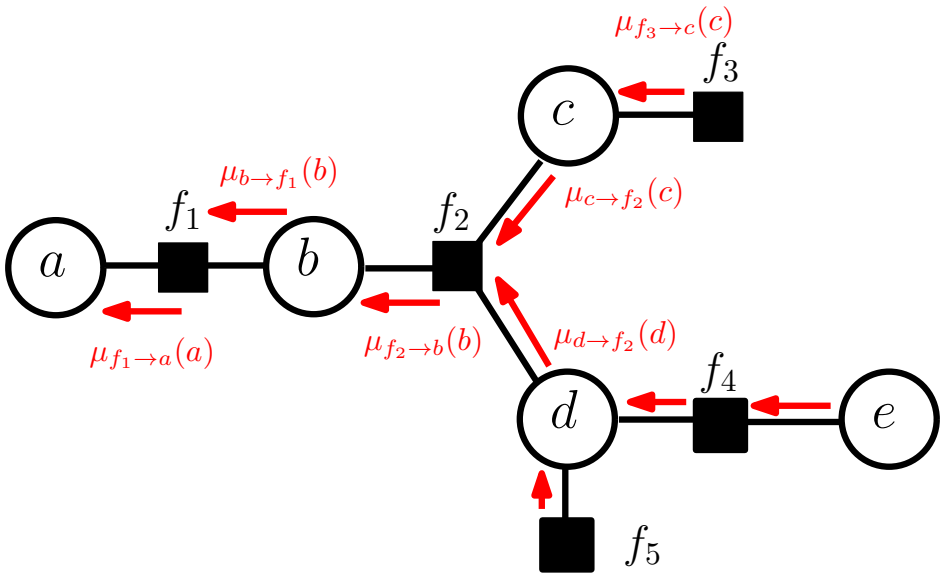
- Spot the messages:



$$\begin{aligned}\max_{a,b,c,d,e} f(\cdot) &= \max_{a,b,c,d,e} f_1(a,b) f_2(b,c,d) f_3(c) f_4(d,e) f_5(d) \\ &= \max_a \underbrace{\max_b f_1(a,b)\mu_{b \to f_1}(b)}_{\mu_{f_2 \to a}(a)}\end{aligned}$$

Trees

- Spot the messages:



$$\max_{a,b,c,d,e} f(\cdot) = \max_{a,b,c,d,e} f_1(a,b) f_2(b,c,d) f_3(c) f_4(d,e) f_5(d)$$
$$= \max_{a} \mu_{f_2 \to a}(a)$$

Max-Product Algorithm

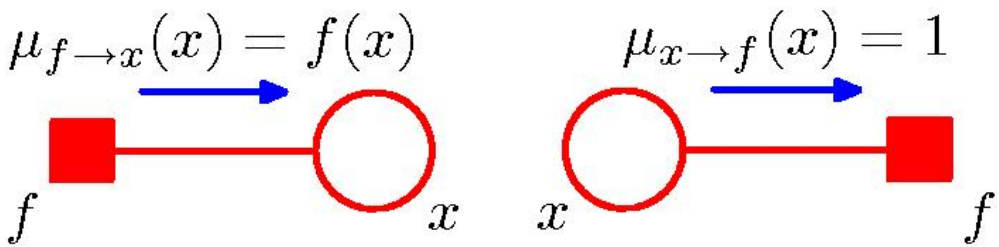
- So we need an algorithm to compute the messages
- Pick any variable as root

1. Initialisation (same as sum-product)
2. Variable to Factor message (same as sum-product)
3. Factor to Variable message
4. Repeat

- Then compute the maximal state

1. Initialisation

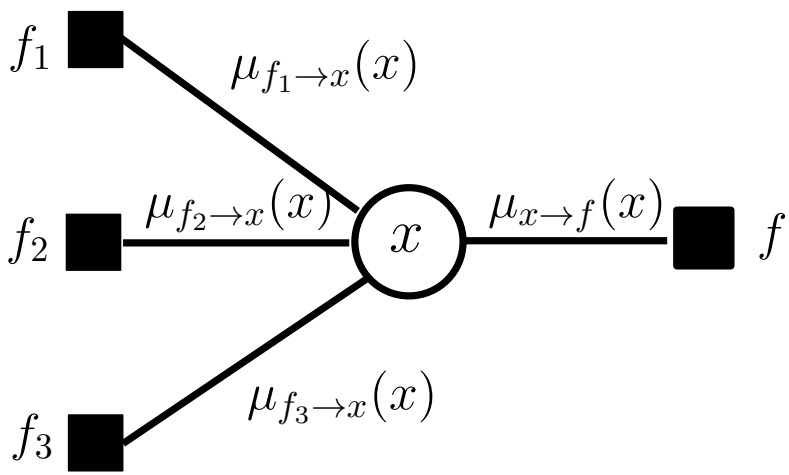
- Messages from extremal node factors are initialized to the factor
- Messages from extremal variable nodes are set to unity



- Same as for sum-product

2. Variable to Factor message

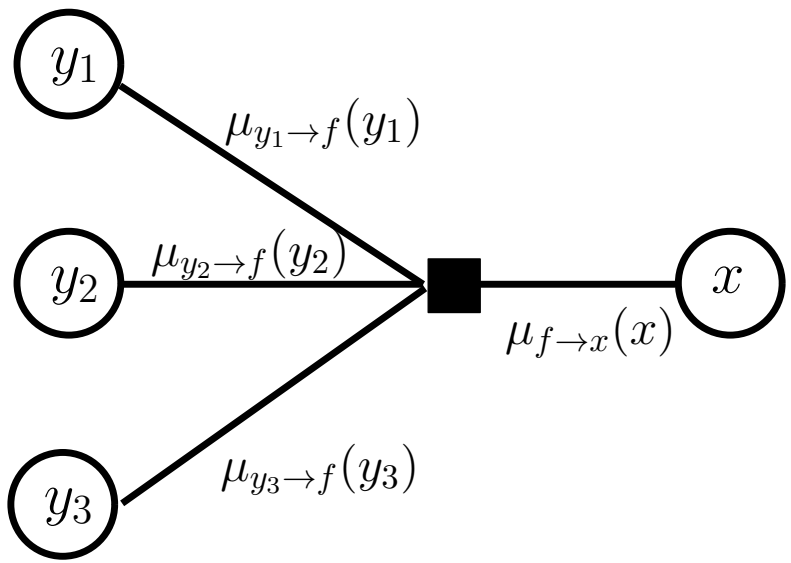
$$\mu_{x\to f}(x) = \prod_{g\in\{\mathsf{ne}(x)\setminus f\}} \mu_{g\to x}(x)$$



- Same as for sum-product

3. Factor to Variable message

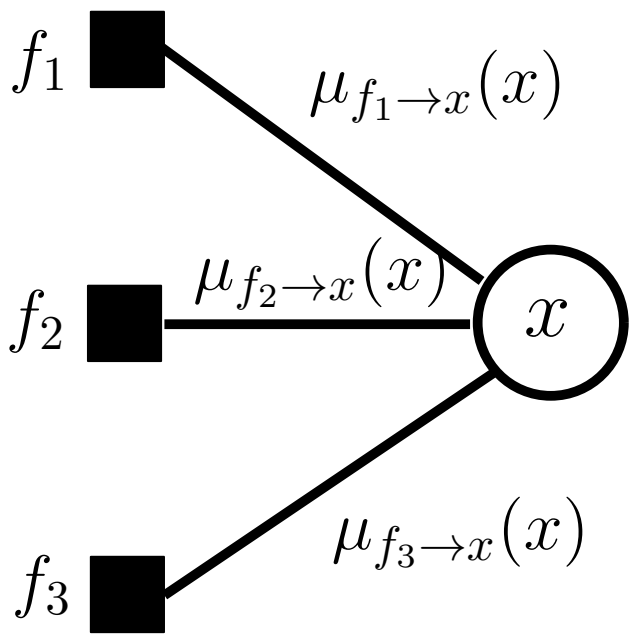
$$\mu_{f\to x}(x) = \max_{y\in\mathcal{X}_f\setminus x} \phi_f(\mathcal{X}_f) \prod_{y\in\{\mathsf{ne}(f)\setminus x\}} \mu_{y\to f}(y)$$



- Different message than in sum-product
- This is now a max-product!

Computing the Maximal State of a Variable

$$x^* = \operatorname*{argmax}_x \prod_{f \in \mathsf{ne}(x)} \mu_{f \to x}(x)$$



Comments

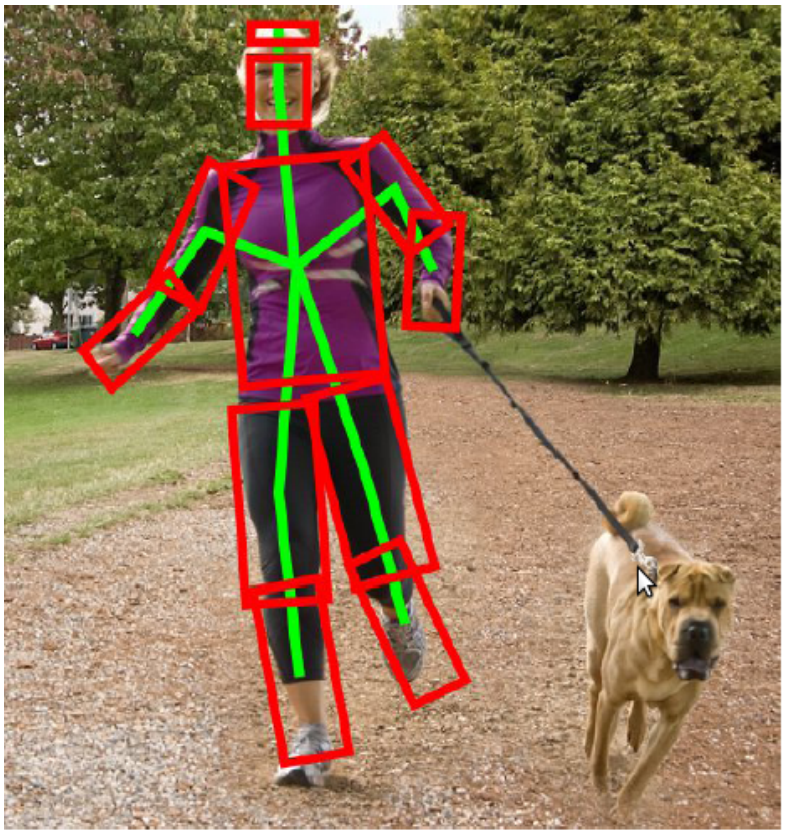
- Products of small probabilities may lead to numerical instabilities
- What can we do?
- Take the logarithm

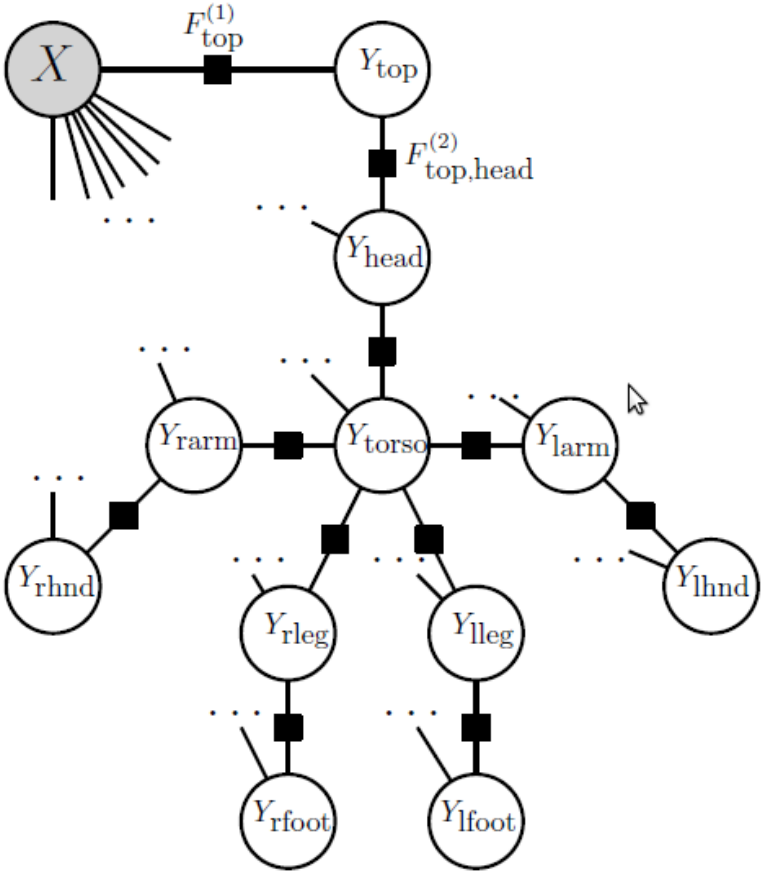
$$\log\left(\max_x p(x)\right) = \max_x \log\left(p(x)\right)$$

- Taking the logarithm replaces the products with sums
  ($\Rightarrow$ yields the max-sum algorithm)

Example: Pictorial Structures

- Find body parts

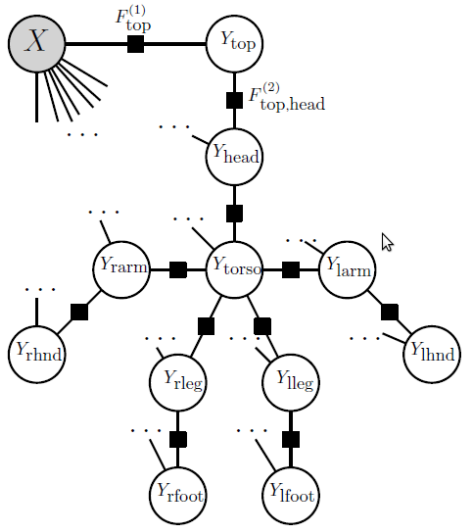




[Fischler & Elschlager, 1973] [Felsenzwalb & Huttenlocher, 2000]

Pictorial Structures

- Each body part one variable (torso,head,etc) (11 variables total)
- Each variable represented as tupel *e.g.* $y_{torso} = (x, y, s, \theta)$
  - (x, y) image coordinates
  - s scale
  - θ rotation of the part
- Discretize label space y (that is x, y, s, θ) in L states
- How many states L do we need?

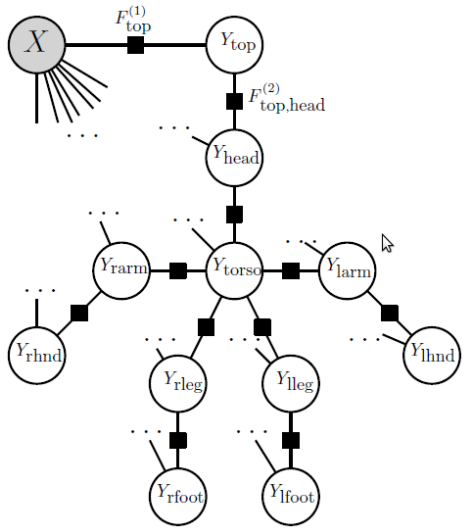


[Felsenzwalb & Huttenlocher, 2000]

Pictorial Structures

Energy Model:

- Unaries:
  $E_{torso}(y_{torso}, X), E_{rarm}(y_{rarm}, X), \ldots,$
- Pairwise: $E_{torso,rarm}(y_{torso}, y_{rarm}), \ldots,$
- Let k be the number of parts (11),
  L the size of the label space ($\approx 500,000$)
- What is the algorithmic complexity?
- Given new test image, max-product
  algorithm complexity is $\mathcal{O}(kL^2)$
- For specific energies reduction to $\mathcal{O}(kL)$



[Felsenzwalb & Huttenlocher, 2000]

Next Time ...

- Approximate Inference
- On the horizon: learning