Inference in Graphical Models (Foundations)

Jiang Guo

2014.03.24

Reading Group #1

Contents

- Backgrounds
- Inference algorithm

Keywords

Graphical models

Inference

Factor graph

Message passing

Sum-product

Belief propagation

Max-sum

Graphs

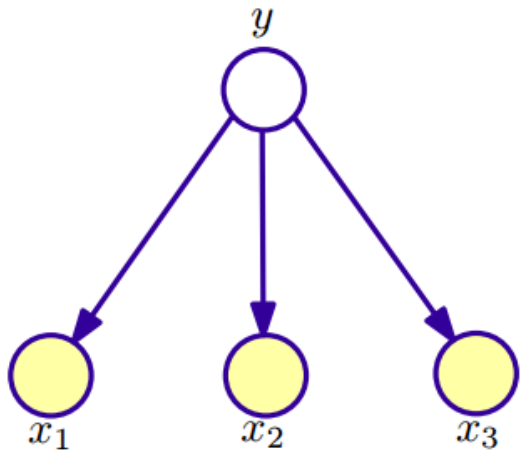
Equations

Backgrounds

- Graphical models in NLP
  - Examples
  - Inference
- General graphical models
  - Definitions
  - Inference

Graphical models in NLP

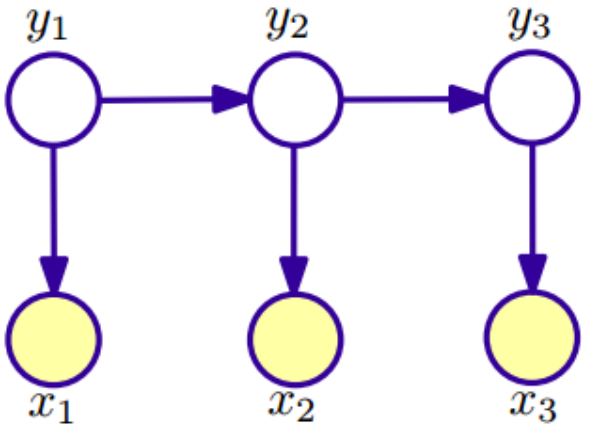
- Directed graphical models
  - Naïve Bayes Classifier



  - $P(y, x_1, \ldots x_d) = p(y) \prod_{i=1}^{d} p(x_i|y)$
  - Used extensively for classification problems
    - Document classification

Graphical models in NLP

- Directed graphical models
  - Hidden Markov Model



  - $P(\boldsymbol{y}, \boldsymbol{x}) = \prod_{i=1}^{d} p(y_i|y_{i-1})\, p(x_i|y_i)$
  - Used for sequence labeling problems
    - Part-of-Speech Tagging

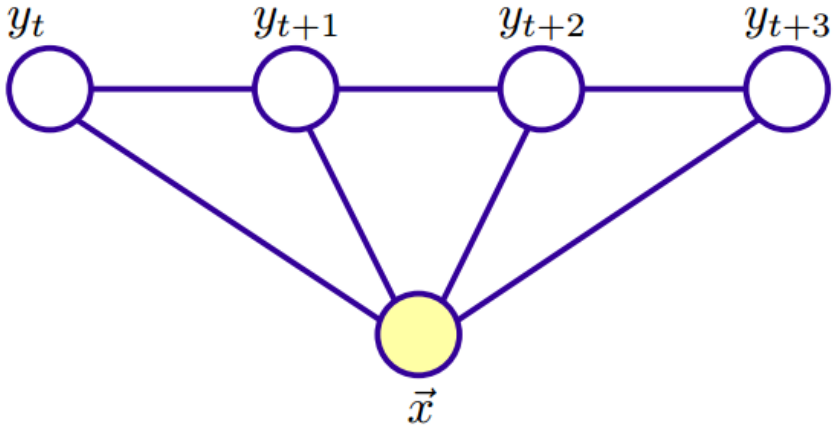
Graphical models in NLP

- Undirected graphical model
  - Maximum entropy classifier

  - $p(y|x) = \frac{1}{Z}\exp(\sum_{j=1}^{m} \lambda_j f_j(x, y))$
  - Used extensively for discriminative/feature-rich classification problems

Graphical models in NLP

- Undirected graphical model
  - Conditional random fields (Lafferty et al. 2001)



  - $p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z}\prod_{i=1}^{d}\exp(\sum_{j=1}^{m}\lambda_j f_j(x, y_{i-1}, y_i))$
  - Used extensively for discriminative/feature-rich sequence labeling

Learning and Inference

- Learning
  - Structure learning
    - Given the nodes, estimate the edges
  - Parameter estimation
    - Given the structure, estimate the weights of edges

- Inference (two situations)
  1. Marginal probability of a variable (node) or a set of variables (nodes)
  2. Find the variable configuration that has the maximum joint probability
     - Find the most likely y given observed x

Learning

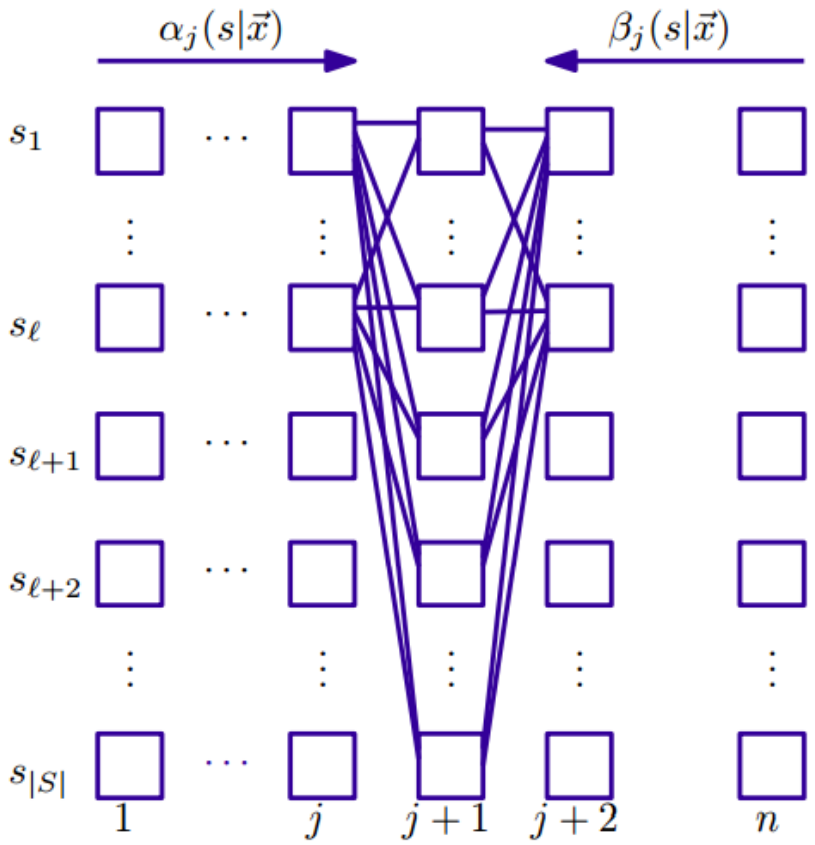
- Objective function
  - MLE
  - Max-margin
- Parameter estimation
  - SGD
  - LBFGS
  - CG
  - CD
  - …

Inference

- Maximum Entropy Classifier
  - Marginal probability
  - Argmax
    - $y^* = \underset{y}{\operatorname{argmax}}\, p(y, x) = p(x) \underset{y}{\operatorname{argmax}}\, p(y|x)$
    - Calculate a score for each y, select the maximum one, typically a Brute-force procedure.

Inference

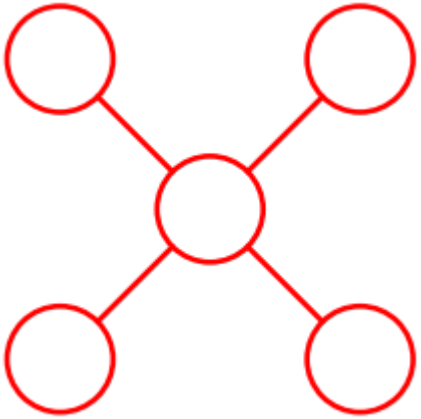
- Conditional Random Fields
  - Marginal probability
    - Forward-Backward

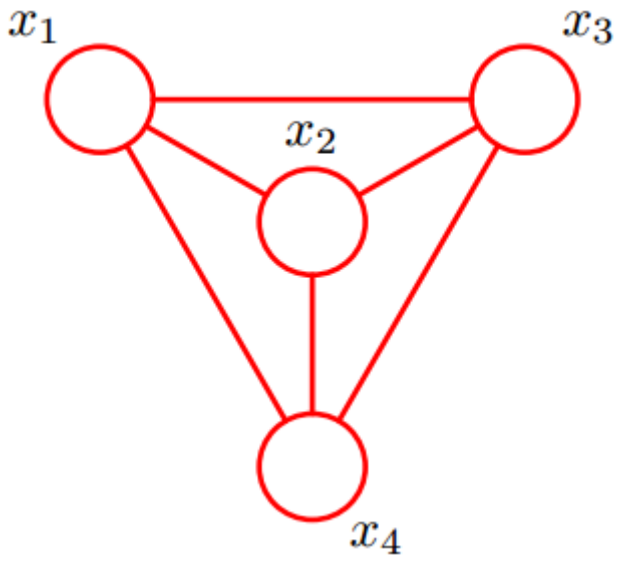


Inference

- Conditional Random Fields
  - Argmax
    - $\boldsymbol{y}$ is a sequence, which has exponential number of values.
    - Dynamic programming: Viterbi

For Trees, or more general graphs, how to do inference?



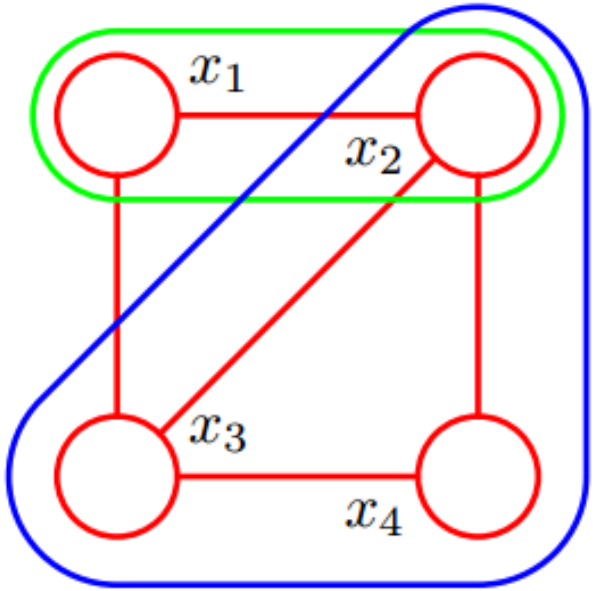


General Graphical Models

- Def.1 Cliques

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

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

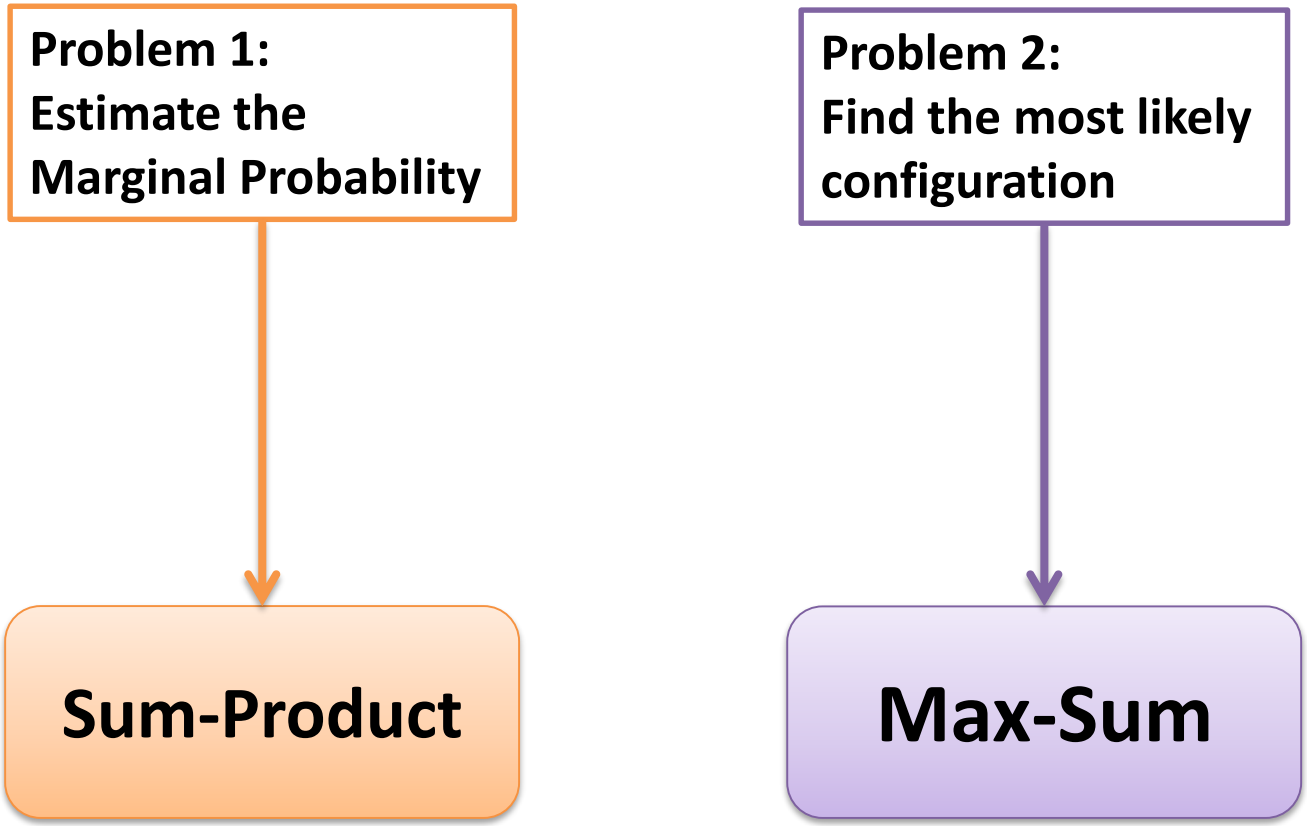


- Def.2 Potential function
  - Restrictions for UGM: $\psi_C(\mathbf{x}_C) \geqslant 0$

$$\psi_C(\mathbf{x}_C) = \exp\{-E(\mathbf{x}_C)\}$$

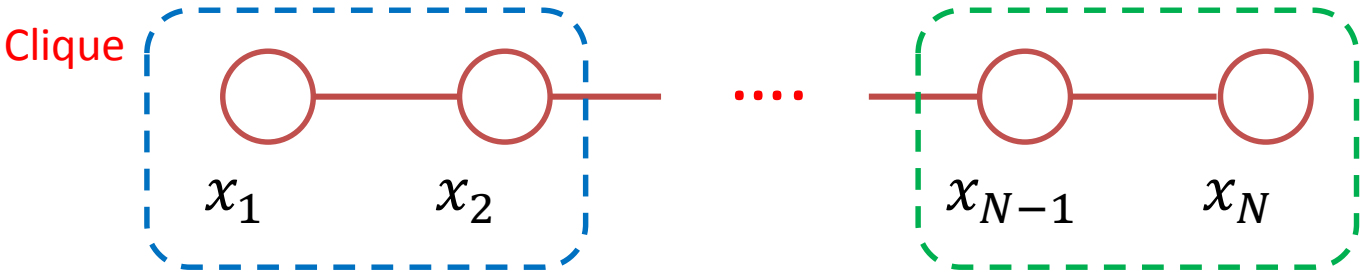
Energy Function

Inference


Problem 1:
Estimate the
Marginal Probability
Problem 2:
Find the most likely
configuration
Sum-Product
Max-Sum

Inference on a Chain

- Markov Chain



- Joint distribution over all variables

Potential

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

- Marginal probability

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

K^N computations!

Inference on a Chain

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

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

$$p(x_n) = \frac{1}{Z}$$

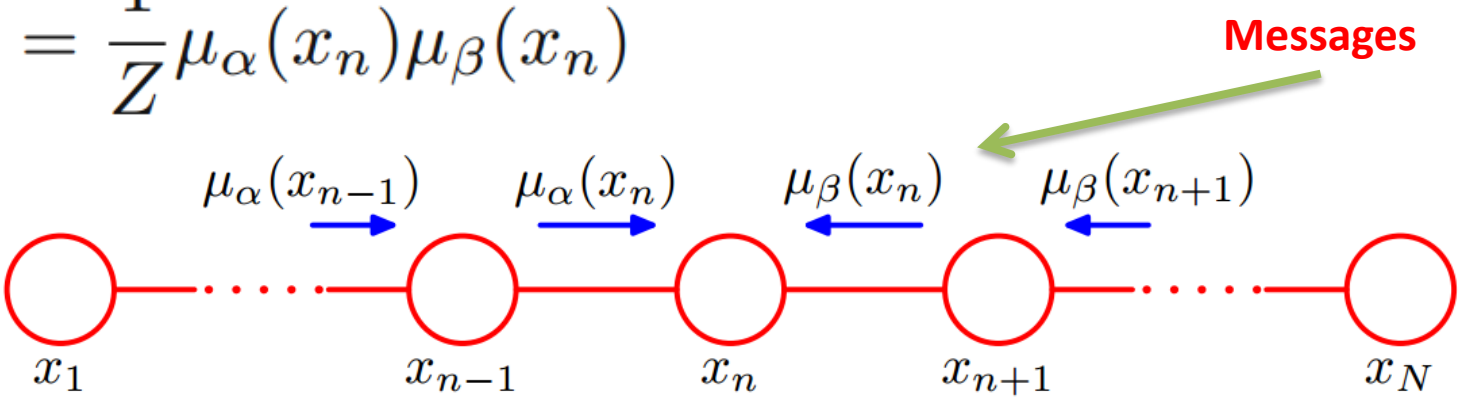
$O(NK^2)$

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

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

Inference as "Message Passing"

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



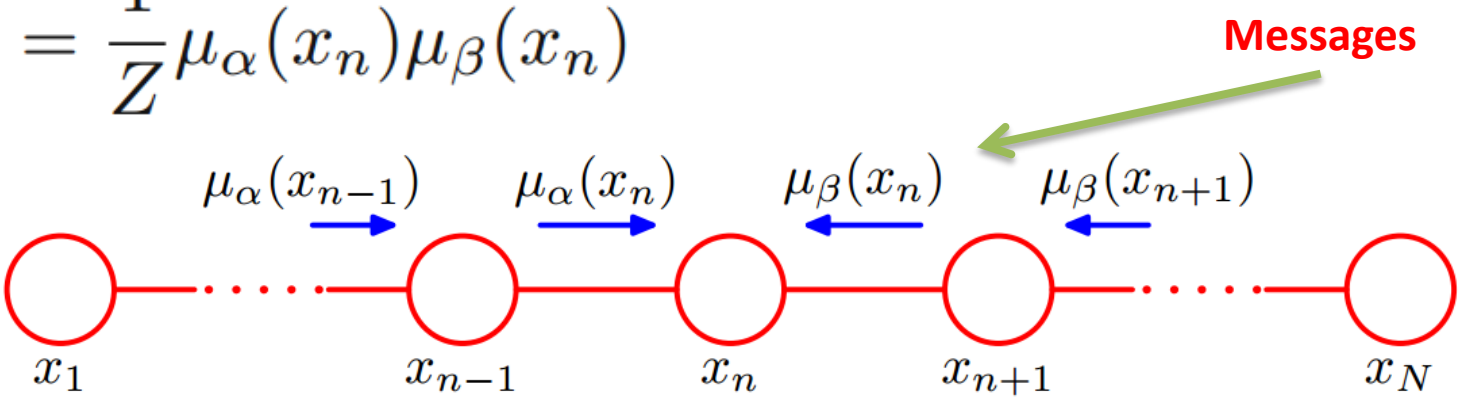
$$\mu_\alpha(x_n) = \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \left[\sum_{x_{n-2}} \cdots \right]$$

$$= \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \mu_\alpha(x_{n-1}).$$

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

Inference as "Message Passing"

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



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

$$= \sum_{x_{n+1}} \psi_{n+1,n}(x_{n+1}, x_n) \mu_\beta(x_{n+1}).$$

Inference as “Message Passing”

- If we wish to calculate the joint distribution $p(x_{n-1}, x_n)$

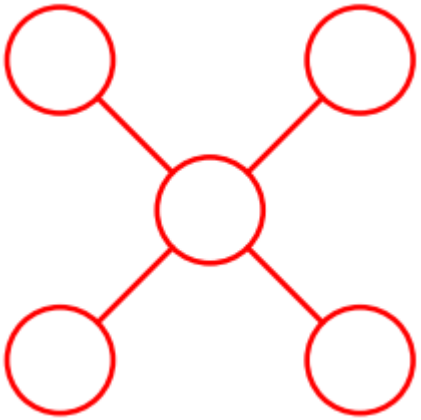
$$p(x_{n-1}, x_n) = \frac{1}{Z}\mu_\alpha(x_{n-1})\psi_{n-1,n}(x_{n-1}, x_n)\mu_\beta(x_n)$$

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

- Exactly the same as the Forward-Backward algorithm used in CRF

Inference on a Tree

- Tree-structured Graph
  - Chain is a particular kind of Tree



Inference on a Tree

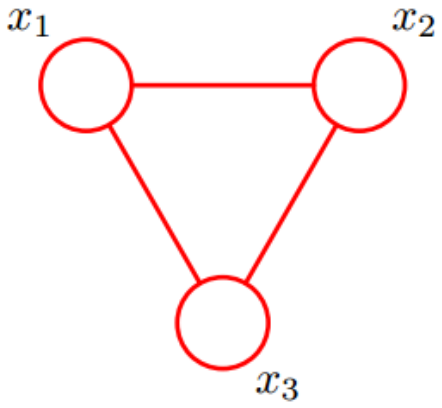
- Factor Graph
- Sum-product Algorithm
  - “Belief Propagation”
- Max-sum Algorithm

Factor Graph

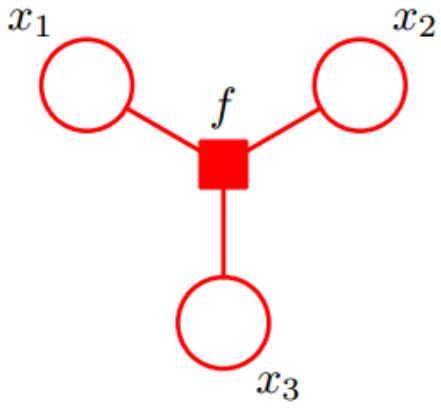
$$p(\mathbf{x}) = \prod_s f_s(\mathbf{x}_s)$$

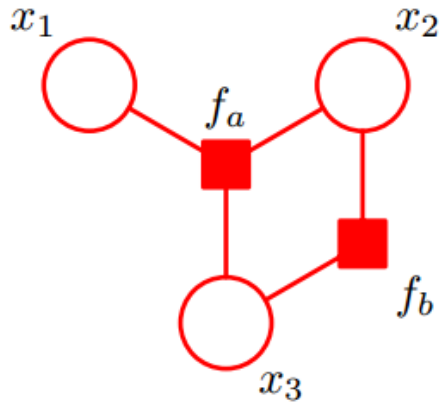
$\mathbf{x}_s$ is a subset of variables

$\psi(x_1, x_2, x_3)$



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





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

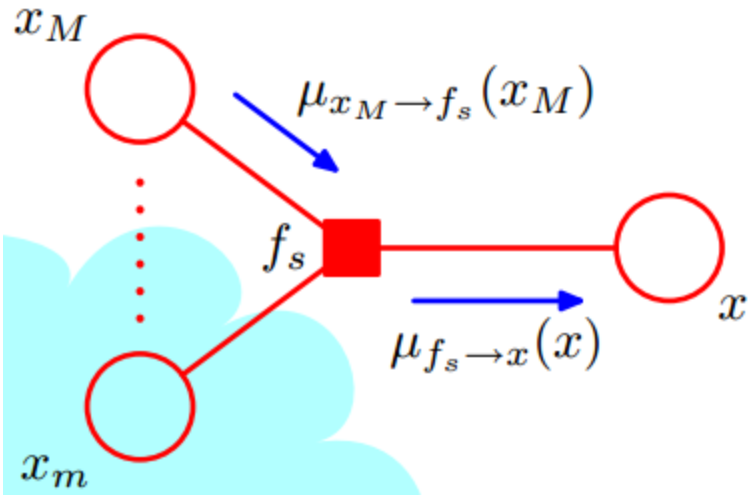
A graph may have different factor graphs

The sum-product algorithm

- Also known as belief propagation (BP)
- Exact if the graph is a tree; otherwise known as “loopy BP”, approximate
- The algorithm involves *passing messages* on the factor graph

Messages

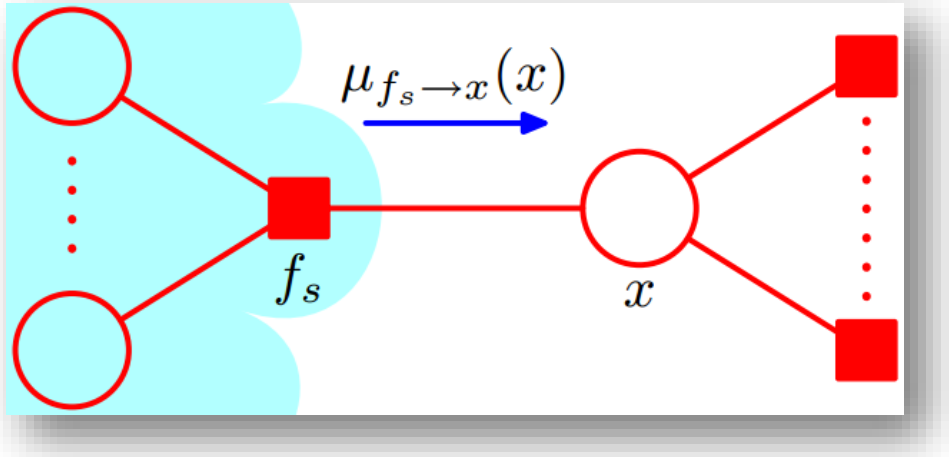
- A message is a vector of length *K,* where *K* is the number of values x takes.
- There are two types of messages:
  - $\mu_{f \to x}$: message from a factor node f to a variable node x
  - $\mu_{x \to f}$: message from a variable node x to a factor node f



The sum-product algorithm

- Marginal Probability

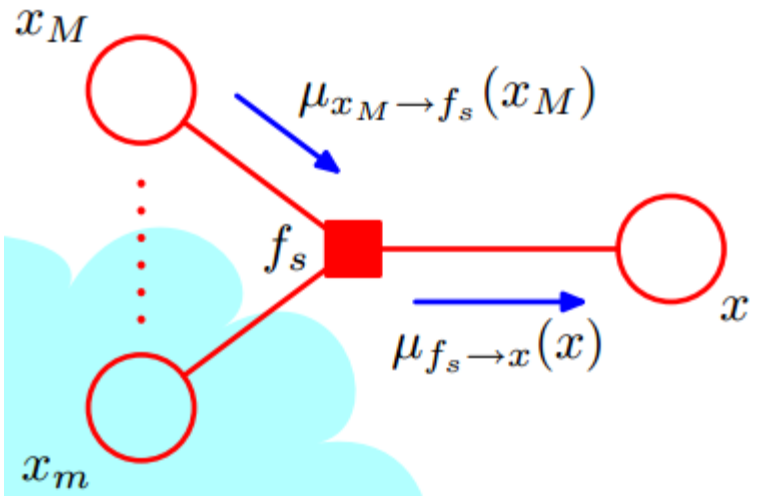
$$p(x) = \prod_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x)$$



The sum-product algorithm

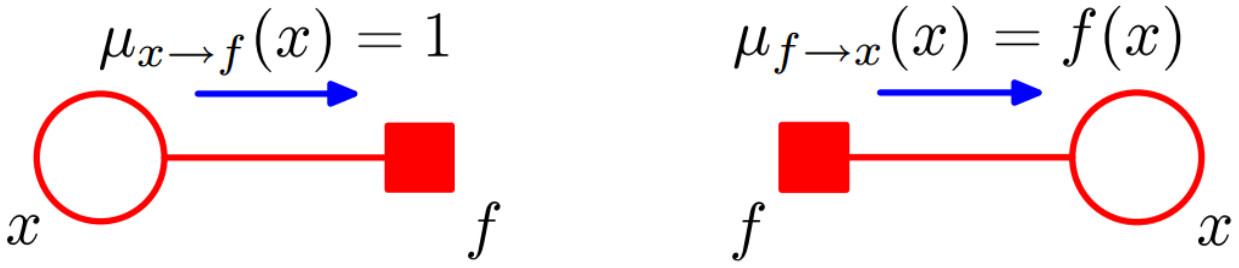
$$\mu_{x_m \to f_s}(x_m) = \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m) \qquad \boldsymbol{\mu_{x \to f}}$$

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



The sum-product algorithm

- The sum-product algorithm begins with messages sent by the leaf nodes, which depend on whether the leaf node is a variable node, or a factor node.

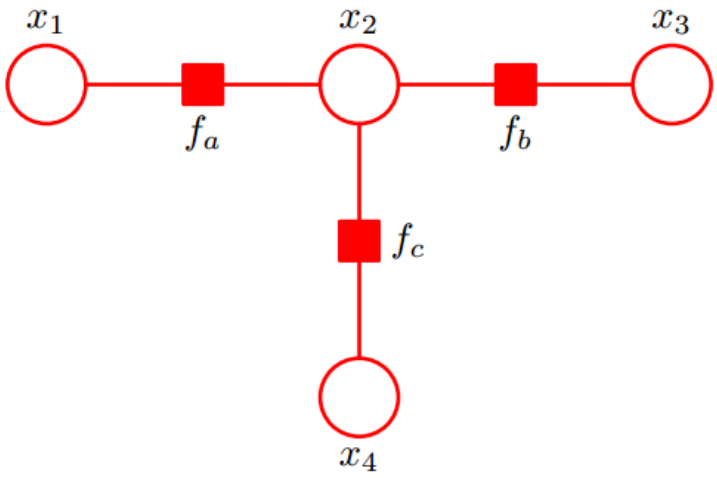


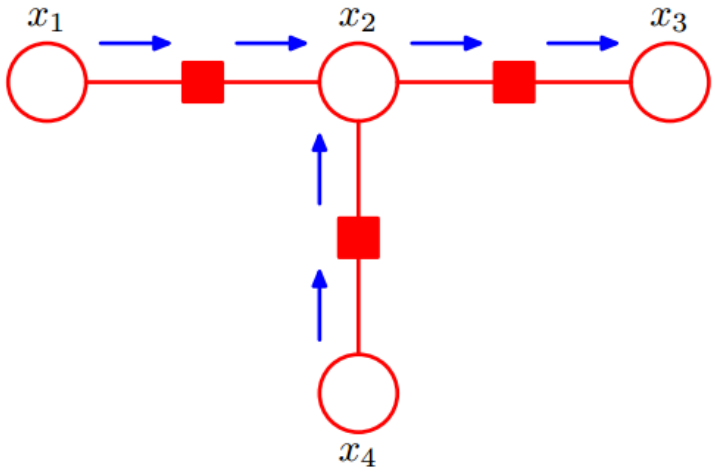
The sum-product algorithm

- Problem: How to efficiently calculate the marginal probabilities for all variables in the graph?
  - Perform sum-product for each node/variable?
    - Too wasteful!
  - Only needs two passes
    - From leaf nodes to root
    - Back from root to leaf nodes

Note that the root node is arbitrarily selected, often one of the leaf nodes.

A simple example





Unnormalized Joint Distribution

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

Designate node x_3 as the root

Message Passing from leaf nodes to root

$$\mu_{x_1 \to f_a}(x_1) = 1$$

$$\mu_{f_a \to x_2}(x_2) = \sum_{x_1} f_a(x_1, x_2)$$

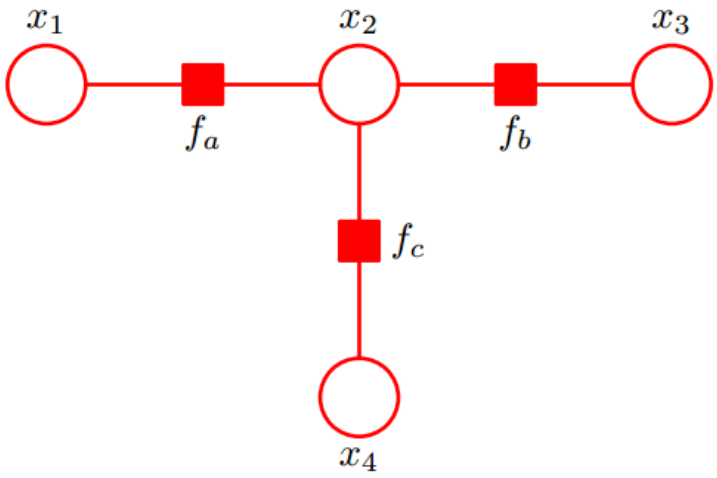
$$\mu_{x_4 \to f_c}(x_4) = 1$$

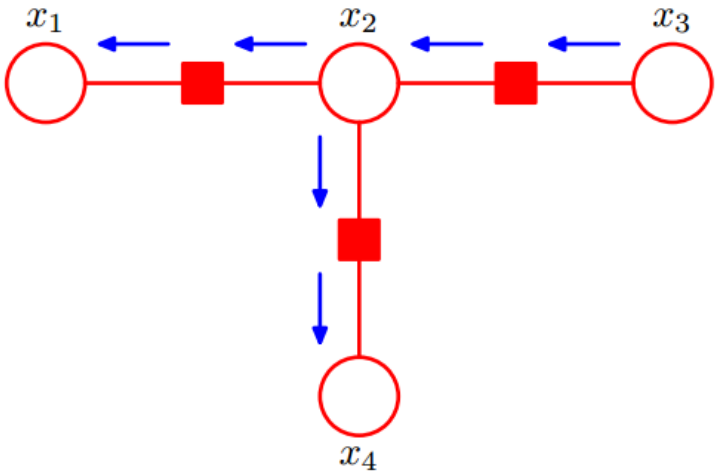
$$\mu_{f_c \to x_2}(x_2) = \sum_{x_4} f_c(x_2, x_4)$$

$$\mu_{x_2 \to f_b}(x_2) = \mu_{f_a \to x_2}(x_2) \mu_{f_c \to x_2}(x_2)$$

$$\mu_{f_b \to x_3}(x_3) = \sum_{x_2} f_b(x_2, x_3) \mu_{x_2 \to f_b}.$$

A simple example





Message Passing from root to leaf nodes

$$\mu_{x_3 \to f_b}(x_3) = 1$$

$$\mu_{f_b \to x_2}(x_2) = \sum_{x_3} f_b(x_2, x_3)$$

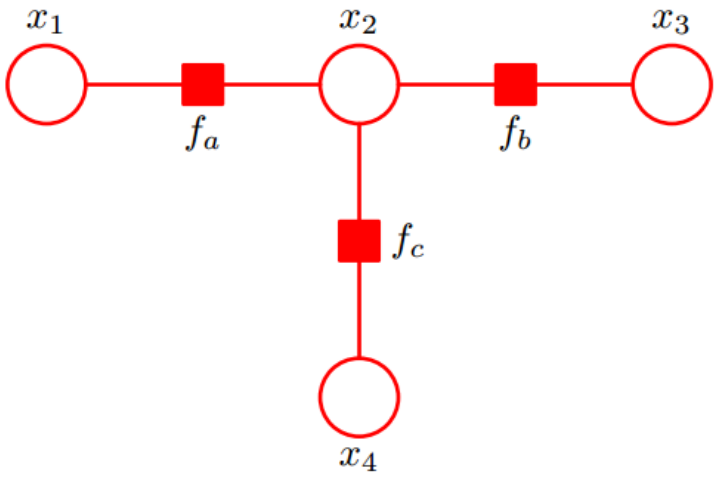
$$\mu_{x_2 \to f_a}(x_2) = \mu_{f_b \to x_2}(x_2)\mu_{f_c \to x_2}(x_2)$$

$$\mu_{f_a \to x_1}(x_1) = \sum_{x_2} f_a(x_1, x_2)\mu_{x_2 \to f_a}(x_2)$$

$$\mu_{x_2 \to f_c}(x_2) = \mu_{f_a \to x_2}(x_2)\mu_{f_b \to x_2}(x_2)$$

$$\mu_{f_c \to x_4}(x_4) = \sum_{x_2} f_c(x_2, x_4)\mu_{x_2 \to f_c}(x_2)$$

A simple example --- verification



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

The max-sum algorithm

- The max-sum algorithm generalizes the Viterbi algorithm for inferencing on Chains.
- Problem:

$$\mathbf{x}^{\max} = \arg\max_{\mathbf{x}} p(\mathbf{x})$$

$$p(\mathbf{x}^{\max}) = \max_{\mathbf{x}} p(\mathbf{x})$$

The max-sum algorithm

$$\max_{\mathbf{x}} p(\mathbf{x}) = \max_{x_1} \ldots \max_{x_M} p(\mathbf{x})$$

$$\max_{\mathbf{x}} p(\mathbf{x}) = \frac{1}{Z} \max_{x_1} \cdots \max_{x_N} \left[\psi_{1,2}(x_1, x_2) \cdots \psi_{N-1,N}(x_{N-1}, x_N)\right]$$

$$= \frac{1}{Z} \max_{x_1} \left[\psi_{1,2}(x_1, x_2) \left[\cdots \max_{x_N} \psi_{N-1,N}(x_{N-1}, x_N)\right]\right].$$

Decoding a Markov Chain, Viterbi

The max-sum algorithm

- To avoid the probability under flow problem
  - Use logarithm of the joint distribution
- Replace ‘sum’ with ‘max’ while computing $\boldsymbol{\mu_{x\to f}}$ and $\boldsymbol{\mu_{f\to x}}$

$$\mu_{f\to x}(x) = \max_{x_1,\ldots,x_M}\left[\ln f(x,x_1,\ldots,x_M) + \sum_{m\in \mathrm{ne}(f_s)\setminus x} \mu_{x_m\to f}(x_m)\right]$$

$$\mu_{x\to f}(x) = \sum_{l\in \mathrm{ne}(x)\setminus f} \mu_{f_l\to x}(x).$$

Thank you!