PROBABILISTIC GRAPHICAL MODELS

Sergey Nikolenko

Harbour Space University, Barcelona, Spain
March 29, 2017

GRAPHICAL MODELS

DIRECTED GRAPHICAL MODELS

- Graphical models show the dependence/independence relations between the variables.
- Example — consider the joint distribution of three variables:

$$p(x, y, z) = p(x \mid y, z)p(y \mid z)p(z).$$

- Let us plot it as a graph.
- A complete graph can describe any distribution $p(x_1, \dots, x_n)$.
- But if some edges are missing, this simplifies (restricts) the distribution.

DIRECTED GRAPHICAL MODELS

- Consider a directed acyclic graph $x_1, \dots, x_k$ with distributions $p(x_i \mid \text{pa}(x_i))$ at every node. Such a graph is called a *directed graphical model* (Bayesian network) for joint probability

$$p(x_1, \dots, x_k) = \prod_{i=1}^{k} p(x_i \mid \text{pa}(x_i)).$$

- In other words, it's good to be able to decompose a large distribution into a product of small manageable distributions.

DIRECTED GRAPHICAL MODELS

- Example — learning distribution parameters after several experiments:

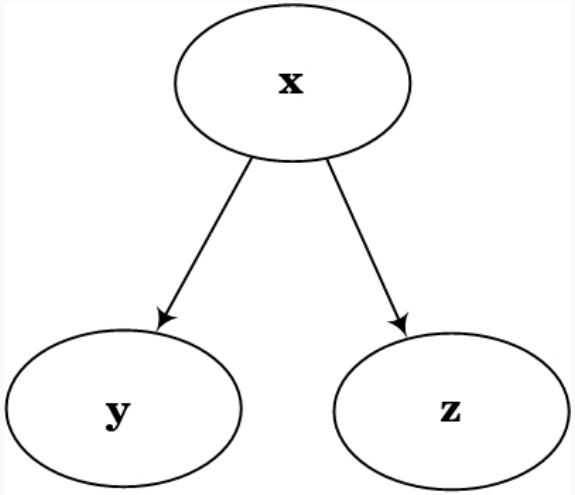
$$p(x_1, \dots, x_n, \theta) = p(\theta) \prod_{i=1}^{n} p(x_i \mid \theta).$$

- What can we say about the (in) dependence of x_i and x_j?
- Inference on graphical models: we obtain certain *evidence* and want to recompute the distributions at other vertices.
- Both learning parameters and making Bayesian predictions can be expressed in this way.

- d-separability — conditional independence expressed via graph structure:
  - sequential connection, $p(x, y, z) = p(x)p(y \mid x)p(z \mid y)$:
    - if y is not observed then
      $p(x, z) = p(x) \int p(y \mid x)p(z \mid y)\mathrm{d}y = p(x)p(z \mid x)$;
    - if y is observed then
      $p(x, z \mid y) = \frac{p(x,y,z)}{p(y)} = \frac{p(x)p(y|x)p(z|y)}{p(y)} = p(x \mid y)p(z \mid y)$, we get conditional independence.

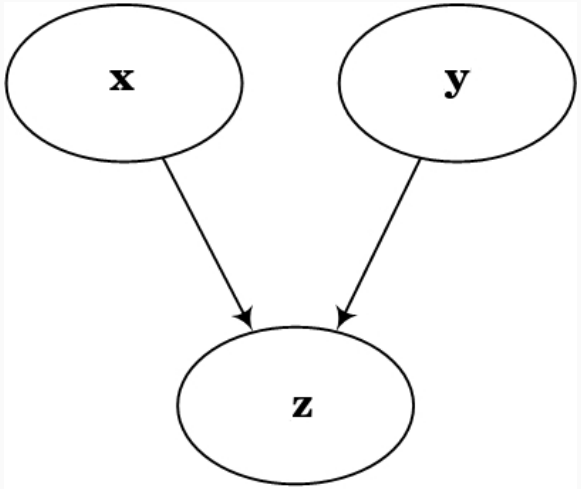
DIRECTED GRAPHICAL MODELS

- diverging connection, $p(x, y, z) = p(x)p(y \mid x)p(z \mid x)$, – так же:
  - if y is not observed then
    $p(x, z) = p(x)p(z \mid x) \int p(y \mid x)\mathrm{d}y = p(x)p(z \mid x)$;
  - if y is observed then
    $p(x, z \mid y) = \frac{p(x,y,z)}{p(y)} = \frac{p(x)p(y|x)p(z|x)}{p(y)} = p(x \mid y)p(z \mid y)$, we get conditional independence.



Расходящаяся связь

- Interesting case – converging connection, $p(x, y, z) = p(x)p(y)p(z \mid x, y)$:
  - if z is not observed then $p(x, y) = p(x)p(y)$, they are independent;
  - if z is observed then $p(x, y \mid z) = \frac{p(x,y,z)}{p(z)} = \frac{p(x)p(y)p(z|x,y)}{p(z)}$, there is no conditional independence.



Сходящаяся связь

Generalization: if we observe at least one descendant of z, independence between x and y may be violated.

- General statement on conditional independence: in a graph with evidence at vertices from Z two vertices x and y (not from Z) are conditionally independent given the set Z if any (undirected) path between x and y:
  - either passes through a vertex $z \in Z$ with evidence with a sequential or diverging connection;
  - or passes through a vertex with converging connection where neither the vertex nor its descendants belong to Z.

DIRECTED GRAPHICAL MODELS

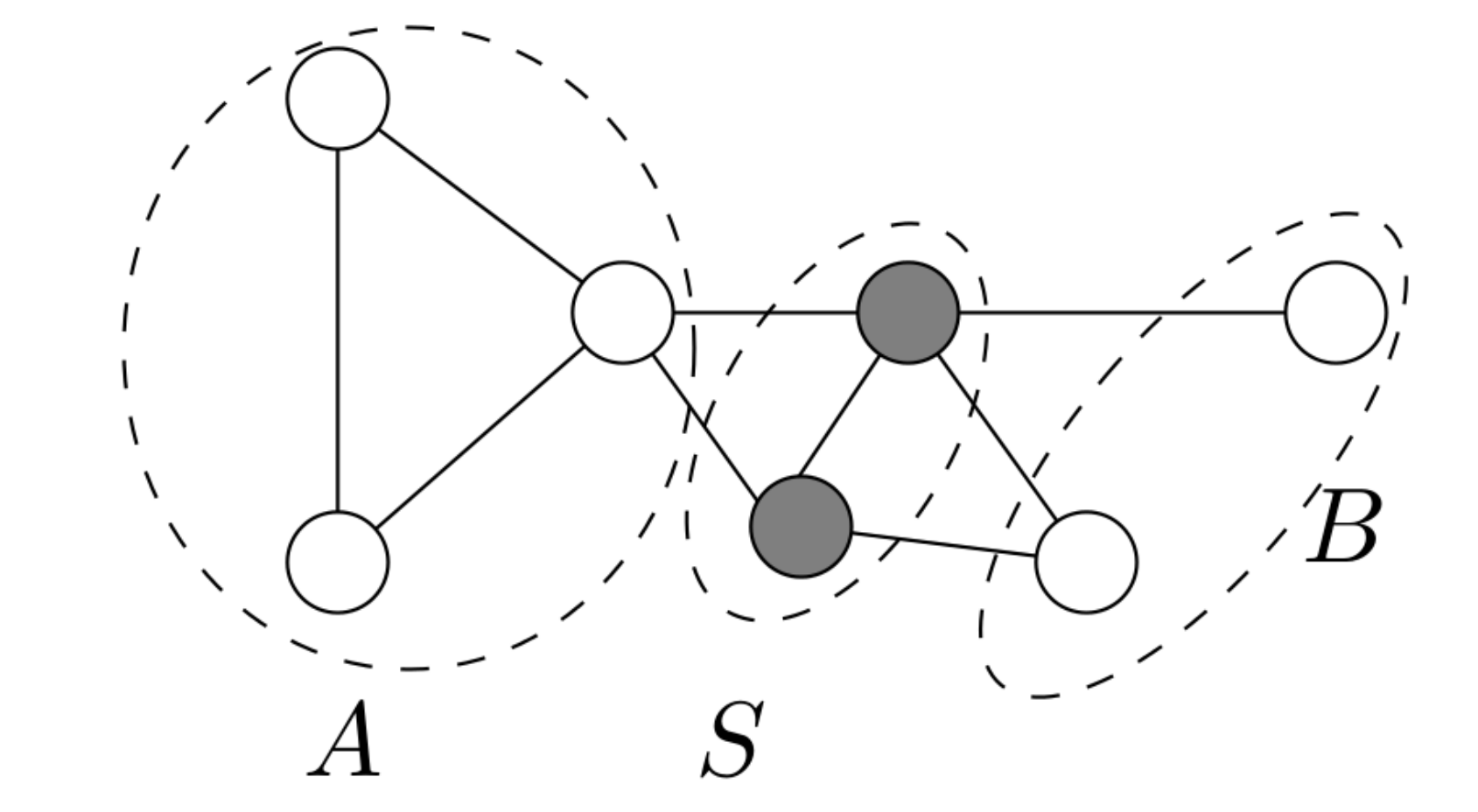
- A graph specifies a set of distributions by specifying restrictions on conditional independence.
- Theorem: this family of distributions exactly coincides with the family of distributions that can be decomposed into

$$p(x_1, \dots, x_k) = \prod_{i=1}^{k} p(x_i \mid \mathsf{pa}(x_i)).$$

UNDIRECTED GRAPHICAL MODELS

- We can also make the independence condition more local.
- Let's define models with undirected graphs, with a natural condition: X is conditionally independent of Y given Z if any path from X to Y passes through Z.
- In particular, $p(x_i, x_j \mid x_{k \neq i,j}) = p(x_i \mid x_{k \neq i,j}) p(x_j \mid x_{k \neq i,j})$ if and only if x_i and x_j are not connected.
- Such models are called *Markov random fields*, or undirected graphical models..

CONDITIONAL INDEPENDENCE IN UNDIRECTED MODELS



UNDIRECTED GRAPHICAL MODELS

- In undirected models, local distributions correspond to cliques in the graph, and they factor as

$$p(x_1, \dots, x_k) = \frac{1}{Z} \prod \psi_C(x_C),$$

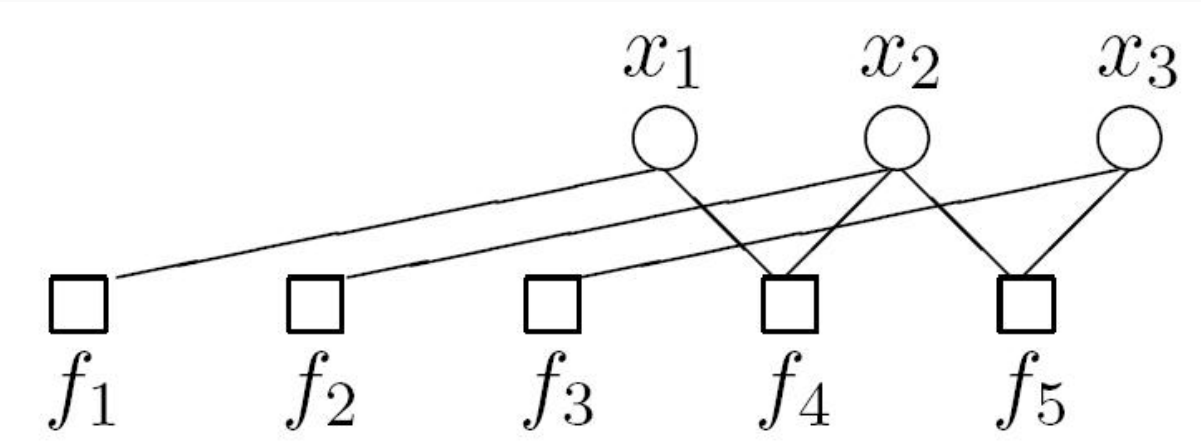
  where C are maximal cliques, ψ_C are nonnegative functions (*potentials*), and Z is the normalizing constant (called *partition function*).
- Since $\psi_C \geq 0$, they are usually represented as exponents:

$$\psi_C(x_C) = \exp\left(-E_C(x_C)\right),$$

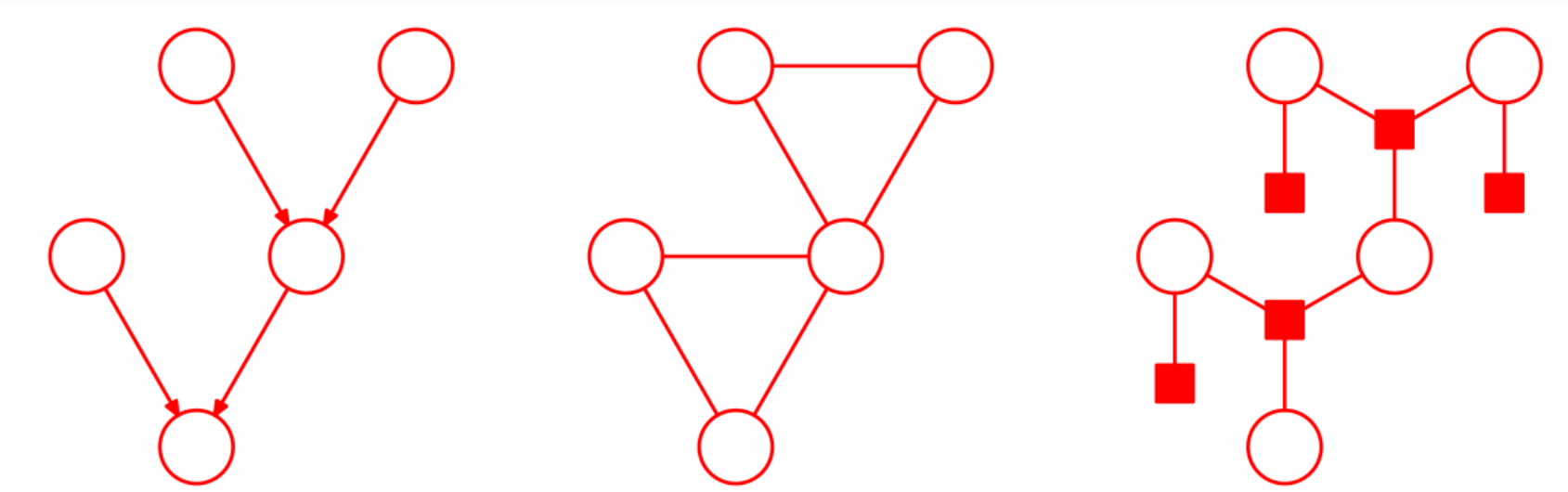
  E_C – are energy functions, they sum into the full energy of the system (similar to statistical physics).
- Directed and undirected models often can be converted into one another, but not always.

FACTOR GRAPHS

- Yet another graphical model: a *factor graph*.
- A factor graph is a bipartite graph of functions and variables.
- It corresponds to the product of all its functions, i.e., represents the decomposition.
- For example, for $p(x_1, x_2, x_3) = f_1(x_1)f_2(x_2)f_3(x_3)f_4(x_1, x_2)f_5(x_2, x_3)$ we have



THREE REPRESENTATIONS



MESSAGE PASSING

GENERAL FUNCTION

- Generally speaking, consider a function

$$p^*(X) = \prod_{j=1}^{m} f_j(X_j),$$

  where $X = \{x_i\}_{i=1}^{n}$, $X_j \subseteq X$.
- I.e., we consider a function that decomposes into a product of several functions.

PROBLEMS

- Normalization problem: find $Z = \sum_X \prod_{j=1}^{m} f_j(X_j)$.
- Marginalization problem: find

$$p_i^*(x_i) = \sum_{k \neq i} p^*(X)$$

  (sometimes also $p_{i_1 i_2}$ and so on).
- Likelihood maximization:

$$\mathbf{x}^* = \arg\max_X p(X).$$

- All of these problems are NP-complete, but we can often solve special cases and/or approximate.

EXAMPLE

- We begin with a graph as an (undirected) chain:

$$p(x_1, \dots, x_n) = \frac{1}{Z}\psi_{1,2}(x_1, x_2) \dots \psi_{n-1,n}(x_{n-1}, x_n).$$

- We want to find

$$p(x_k) = \sum_{x_1} \dots \sum_{x_{k-1}} \sum_{x_{k+1}} \dots \sum_{x_n} p(x_1, \dots, x_n).$$

EXAMPLE

- Obviously, we can simplify a lot here; e.g., from right to left:

$$\sum_{x_n} p(x_1, \dots, x_n) =$$

$$= \frac{1}{Z} \psi_{1,2}(x_1, x_2) \dots \psi_{n-2,n-1}(x_{n-2}, x_{n-1}) \sum_{x_n} \psi_{n-1,n}(x_{n-1}, x_n).$$

- And similar from left to right.

EXAMPLE

- The process will converge in the node x_k that receives two "messages": from the left

$$\mu_\alpha(x_k) = \sum_{x_{k-1}} \psi_{k-1,k}(x_{k-1}, x_k) \left[\dots \sum_{x_2} \psi_{2,3}(x_2, x_3) \left[\sum_{x_1} \psi_{1,2}(x_1, x_2) \right] \dots \right],$$

  and from the right

$$\mu_\beta(x_k) = \sum_{x_{k+1}} \psi_{k,k+1}(x_k, x_{k+1}) \left[\dots \left[\sum_{x_n} \psi_{n-1,n}(x_{n-1}, x_n) \right] \dots \right].$$

- Each partial sum can be viewed as a "message" from a node to its neighbor; a message is a function of that neighbor.

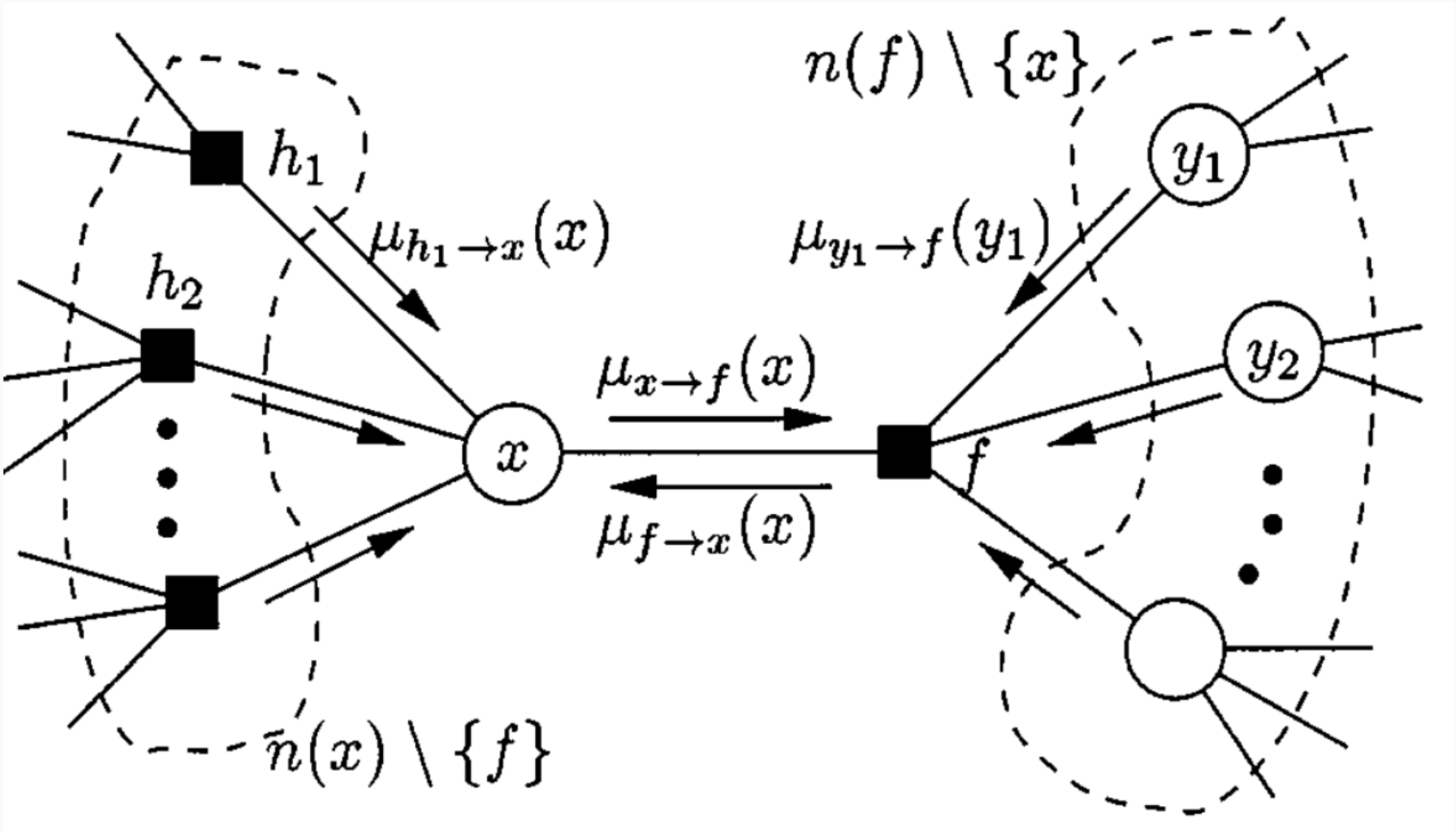
MESSAGE PASSING ALGORITHM

- To generalize, consider a factor graph; suppose it's a tree.
- The message passing algorithm solves the marginalization problem for a function of the form

$$p(x_1, \dots, x_n) = \prod_s f_s(X_s)$$

  defined as a factor graph.
- We pass messages towards the necessary node along the edges.

MESSAGE PASSING



MESSAGE PASSING ALGORITHM

- To find $p(x_k)$, we write $p(x_1, \dots, x_n) = \prod_{s \in \neq(x_k)} F_s(x_k, X_s)$, where X_s are variables from the subtree with root f_s. Then

$$p(x_k) = \sum_{x_{i \neq k}} p(x_1, \dots, x_n) = \prod_{s \in \neq(x_k)} \left[\sum_{X_s} F_s(x_k, X_s) \right] =$$

$$= \prod_{s \in \neq(x_k)} \mu_{f_s \to x_k}(x_k),$$

where $\mu_{f_s \to x_k}(x_k)$ are messages from adjacent functions to variable x_k.

MESSAGE PASSING ALGORITHM

- To find $\mu_{f_s \to x_k}(x_k)$, note that $F_s(x_k, X_s)$ can also be decomposed w.r.t. the corresponding subgraph:

$$F_s(x_k, X_s) = f_s(x_k, Y_s) \prod_{y \in Y_s} G_y(y, X_{s,y}),$$

  where Y_s are variables immediately connected to f_s (except x_k), $X_{s,y}$ are the corresponding subtrees.
- Overall we get

$$\mu_{f_s \to x_k}(x_k) = \sum_{Y_s} f_s(x_k, Y_s) \prod_{y \in Y_s} \left(\sum_{X_{s,y}} G_y(y, X_{s,y}) \right) = \\ = \sum_{Y_s} f_s(x_k, Y_s) \prod_{y \in Y_s} \mu_{y \to f_s}(y).$$

- Similarly, $\mu_{y \to f_s}(y) = \prod_{f \in \neq(y)\ f_s} \mu_{f \to y}(y)$.

MESSAGE PASSING ALGORITHM

- Thus, we get a clear algorithm:
  - as soon as a node has received messages from all neighbors except one, it begins to transmit to this neighbor;
  - a message on an edge between a function and a variable is a function of this variable;
  - a variable node x transmits message

$$\mu_{x \to f}(x) = \prod_{g \in \neq(x)\ f} \mu_{g \to x}(x);$$

  - a function node $f(x, Y)$ transmits message

$$\mu_{f \to x}(x) = \sum_{y \in Y} f(x, Y) \prod_{y \in Y} \mu_{y \to f}(y);$$

  - initial messages at the leaves are $\mu_{x \to f}(x) = 1$, $\mu_{f \to x}(x) = f(x)$.

MESSAGE PASSING ALGORITHM

- When messages come from all neighbors to some variable x_k, we will be able to compute

$$p(x_k) = \prod_{f \in \neq(x_k)} \mu_{f \to x_k}(x_k).$$

- When messages come from all neighbors to some factor $f_s(X_s)$, we will be able to compute the joint distribution

$$p(X_s) = f_s(X_s) \prod_{y \in \neq(f_s)} \mu_{y \to f_s}(y).$$

- In two passes (there and back again along each edge) we will be able to compute the marginals in all nodes.

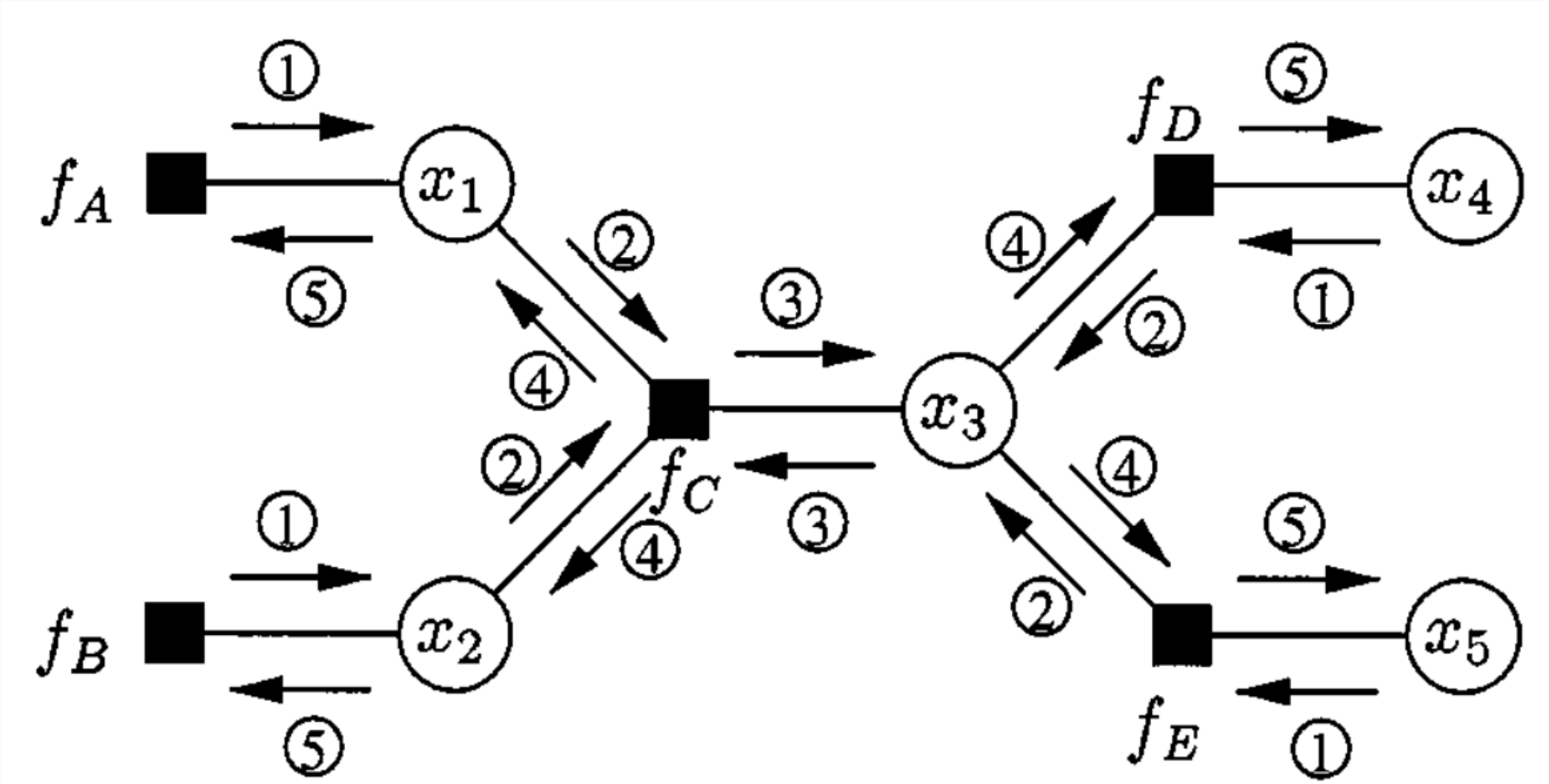
MESSAGE PASSING ALGORITHM

- This is called the sum-product algorithm because a message is computed as

$$\mu_{f \to x}(x) = \sum_{y \in Y} f(x, Y) \prod_{y \in Y} \mu_{y \to f}(y).$$

- The maximization problem $\arg\max_x p(x_1, \ldots, x_n)$ can be solved with a similar *max-sum algorithm*, with $\max$ instead of sum and sum instead of product.

MESSAGE PASSING

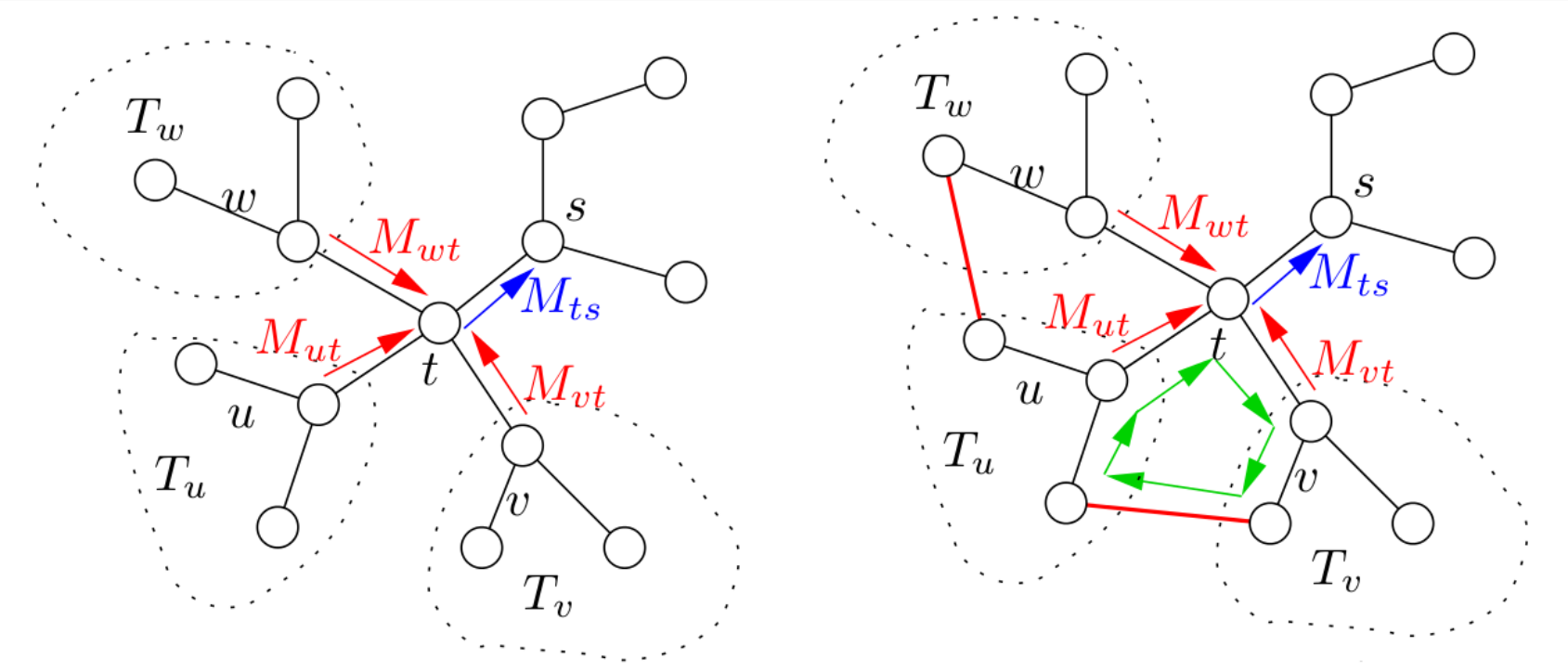


APPROXIMATE INFERENCE

COMPLEX GRAPH STRUCTURE

- What do we do if there are loops in the graph? A very common thing.
- If the loops are small, we can remove them with exponential blowup.
- If they are large, sum-product does not work, so one way is to apply sum-product. :)
- Simply repeat it until convergence; often works.

LOOPY MESSAGE PROPAGATION



But the better approaches are methods for approximate inference: *variational approximations* and *sampling*. We will consider them later.

- If the structure is simple but the factors are hard (can't integrate a message analytically), we can approximate the factors too. If in the factorized distribution

$$p(\theta \mid D) = \frac{1}{p(D)} \prod_i f_i(\theta)$$

f_i are too hard, we replace them with simpler ones (e.g., Gaussians)

$$q(\theta \mid D) = \frac{1}{Z} \prod_i \hat{f}_i(\theta).$$

and also minimize the Kullback–Leibler divergence between p and q.

EXPECTATION PROPAGATION

- For one factor it would be simply moments matching.
- For many factors we have to approximate all $\hat{f}_i$ simultaneously.
- Expectation Propagation — in fact we can do it by sequentially passing the messages:
  1. run message passing, but on every step instead of $\mu_{f_s \to x_k}(x_k)$ pass its approximation $\hat{\mu}_{f_s \to x_k}(x_k)$ from some simpler family;
  2. repeat message passing until convergence.
- Won't prove that it works, but it does.

THANK YOU!

Thank you for your attention!