Probabilistic Graphical Models

Brown University CSCI 2950-P, Spring 2013
Prof. Erik Sudderth

Lecture 5:
Belief Propagation & Factor Graphs

Some figures courtesy Michael Jordan's draft textbook,
An Introduction to Probabilistic Graphical Models

Inference in Graphical Models

$x_E \longrightarrow$ observed *evidence* variables (subset of nodes)

$x_F \longrightarrow$ unobserved *query* nodes we'd like to infer

$x_R \longrightarrow$ remaining variables, *extraneous* to this query but part of the given graphical representation

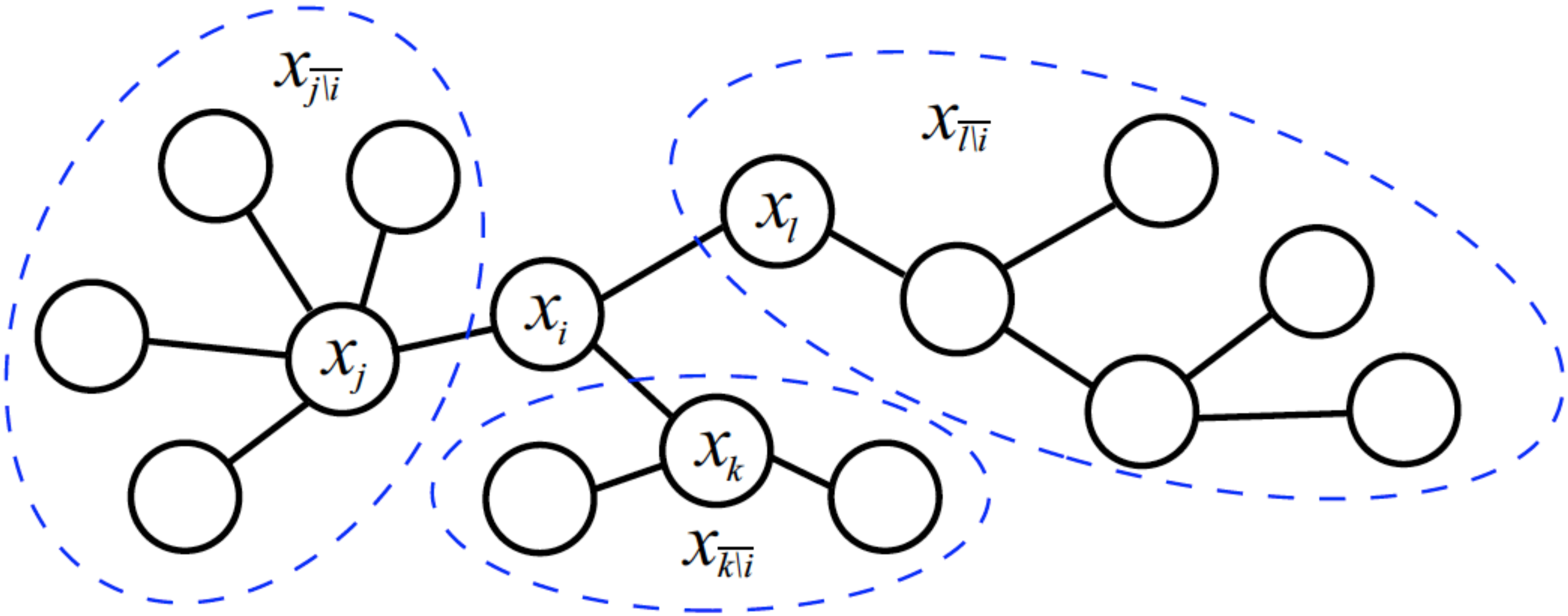
$$p(x_E, x_F) = \sum_{x_R} p(x_E, x_F, x_R) \qquad R = V \setminus \{E, F\}$$

Posterior Marginal Densities

$$p(x_F \mid x_E) = \frac{p(x_E, x_F)}{p(x_E)} \qquad p(x_E) = \sum_{x_F} p(x_E, x_F)$$

- Provides Bayesian estimators, confidence measures, and sufficient statistics for iterative parameter estimation
- The *elimination algorithm* assumed a single query node. What if we want the marginals for *all* unobserved nodes?

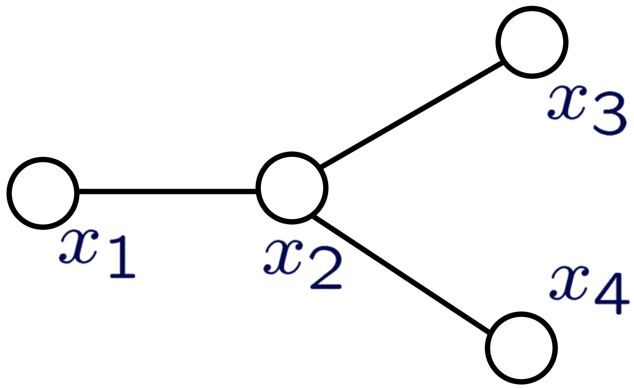
Inference in Undirected Trees



- For a tree, the maximal cliques are always pairs of nodes:

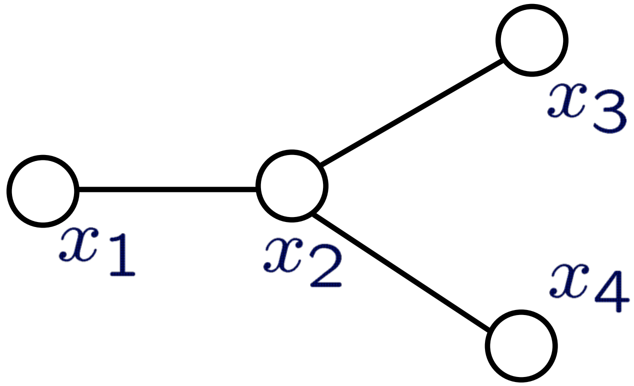
$$p(x) = \frac{1}{Z} \prod_{(s,t)\in\mathcal{E}} \psi_{st}(x_s, x_t) \prod_{s\in\mathcal{V}} \psi_s(x_s)$$

Inference via the Distributed Law



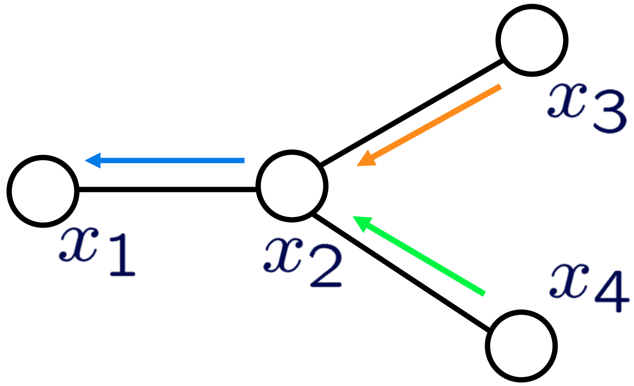
$$
\begin{aligned}
p_1(x_1) &= \sum_{x_2,x_3,x_4} \psi_1(x_1)\psi_{12}(x_1,x_2)\psi_2(x_2)\psi_{23}(x_2,x_3)\psi_3(x_3)\psi_{24}(x_2,x_4)\psi_4(x_4) \\
&= \psi_1(x_1) \sum_{x_2,x_3,x_4} \psi_{12}(x_1,x_2)\psi_2(x_2)\psi_{23}(x_2,x_3)\psi_3(x_3)\psi_{24}(x_2,x_4)\psi_4(x_4)
\end{aligned}
$$

Inference via the Distributed Law



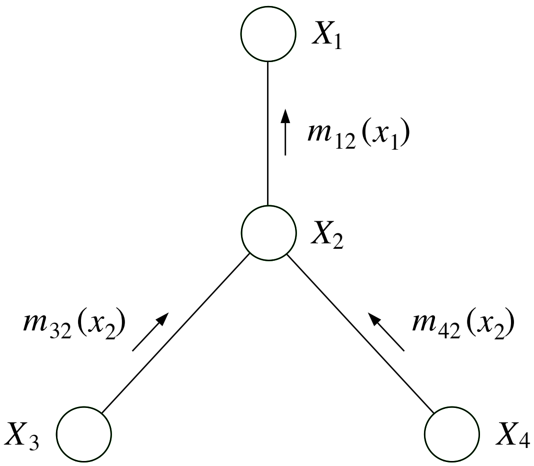
$$
\begin{aligned}
p_1(x_1) &= \sum_{x_2,x_3,x_4} \psi_1(x_1)\psi_{12}(x_1,x_2)\psi_2(x_2)\psi_{23}(x_2,x_3)\psi_3(x_3)\psi_{24}(x_2,x_4)\psi_4(x_4) \\
&= \psi_1(x_1) \sum_{x_2,x_3,x_4} \psi_{12}(x_1,x_2)\psi_2(x_2)\psi_{23}(x_2,x_3)\psi_3(x_3)\psi_{24}(x_2,x_4)\psi_4(x_4) \\
&= \psi_1(x_1) \sum_{x_2} \psi_{12}(x_1,x_2)\psi_2(x_2) \sum_{x_3,x_4} \psi_{23}(x_2,x_3)\psi_3(x_3)\psi_{24}(x_2,x_4)\psi_4(x_4)
\end{aligned}
$$

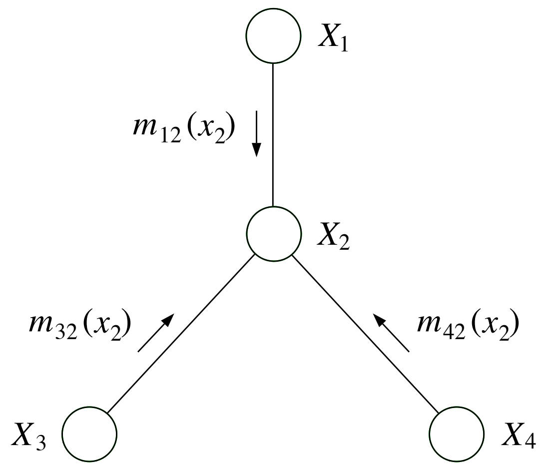
Inference via the Distributed Law

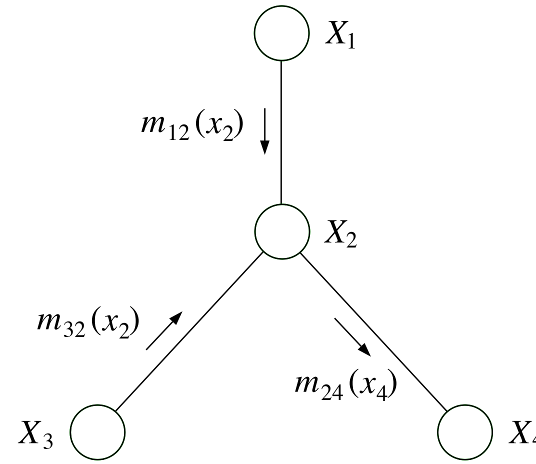


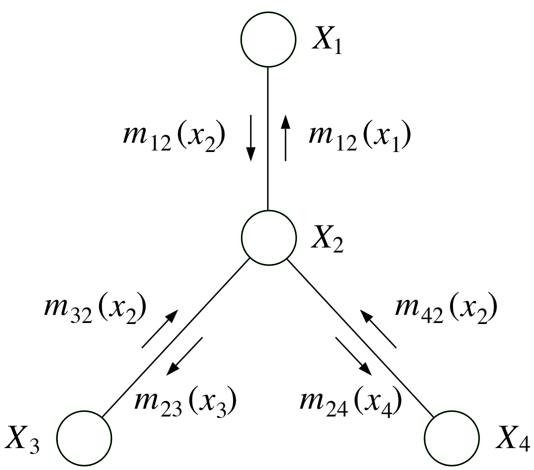
$$\begin{aligned}
p_1(x_1) &= \sum_{x_2,x_3,x_4} \psi_1(x_1)\psi_{12}(x_1,x_2)\psi_2(x_2)\psi_{23}(x_2,x_3)\psi_3(x_3)\psi_{24}(x_2,x_4)\psi_4(x_4) \\
&= \psi_1(x_1) \sum_{x_2,x_3,x_4} \psi_{12}(x_1,x_2)\psi_2(x_2)\psi_{23}(x_2,x_3)\psi_3(x_3)\psi_{24}(x_2,x_4)\psi_4(x_4) \\
&= \psi_1(x_1) \sum_{x_2} \psi_{12}(x_1,x_2)\psi_2(x_2) \sum_{x_3,x_4} \psi_{23}(x_2,x_3)\psi_3(x_3)\psi_{24}(x_2,x_4)\psi_4(x_4) \\
&= \psi_1(x_1) \underbrace{\sum_{x_2} \psi_{12}(x_1,x_2)\psi_2(x_2) \underbrace{\left[\sum_{x_3} \psi_{23}(x_2,x_3)\psi_3(x_3)\right]}_{m_{32}(x_2)} \cdot \underbrace{\left[\sum_{x_4} \psi_{24}(x_2,x_4)\psi_4(x_4)\right]}_{m_{42}(x_2)}}_{m_{21}(x_1) = \sum_{x_2} \psi_{12}(x_1,x_2)\psi_2(x_2)m_{32}(x_2)m_{42}(x_2)}
\end{aligned}$$

Computing Multiple Marginals





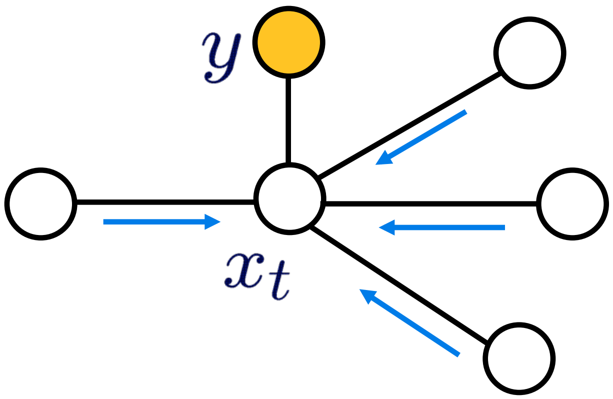




- Can compute all marginals, at all nodes, by combining incoming messages from adjacent edges
- Each message must only be computed once, via some *message update schedule*

Belief Propagation (Sum-Product)

BELIEFS: Posterior marginals

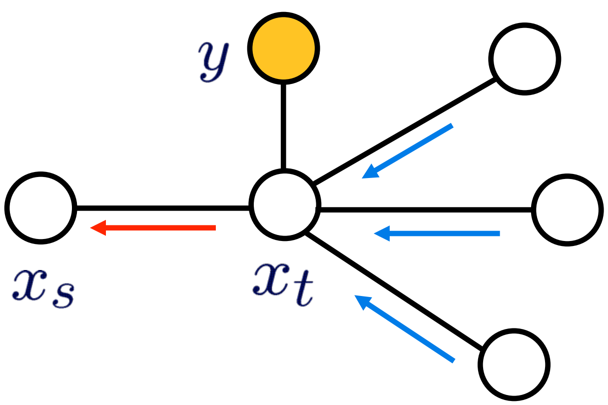


$$q_t(x_t \mid y) = \alpha \psi_t(x_t, y) \prod_{u \in \Gamma(t)} m_{ut}(x_t)$$

$\Gamma(t) \longrightarrow$ neighborhood of node t (adjacent nodes)

MESSAGES: Sufficient statistics

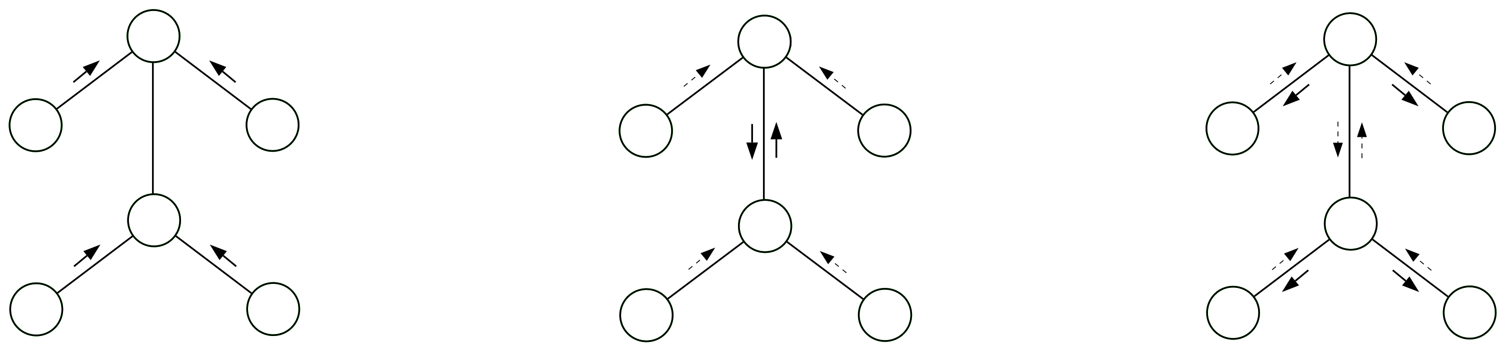
$$m_{ts}(x_s) = \alpha \sum_{x_t} \psi_{st}(x_s, x_t) \psi_t(x_t, y) \prod_{u \in \Gamma(t) \setminus s} m_{ut}(x_t)$$



I) Message Product
II) Message Propagation

Message Update Schedules

- *Message Passing Protocol:* *A node can send a message to a neighboring node when, and only when, it has received incoming messages from all of its other neighbors*



- *Synchronous Parallel Schedule:* *At each iteration, every node computes all outputs for which it has needed inputs*
- *Global Sequential Schedule:* *Choose some node as the root of the true. Pass messages from the leaves to the root, and then from the root back to the leaves.*
- *Asynchronous Parallel Schedule:* *Initialize messages arbitrarily. At each iteration, all nodes compute all outputs from all current inputs. Iterate until convergence.*

Belief Propagation for Trees

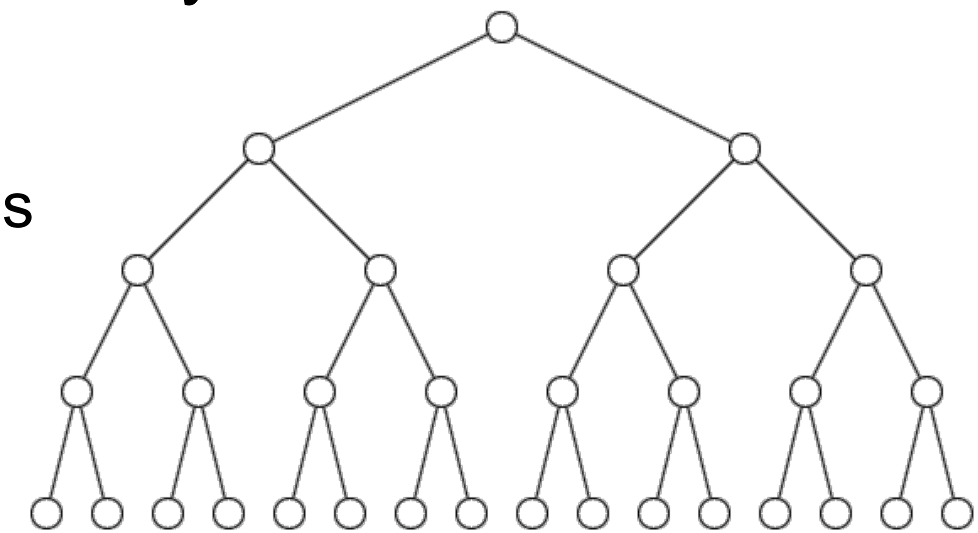
- Dynamic programming algorithm which exactly computes all marginals
- On Markov chains, BP equivalent to alpha-beta or forward-backward algorithms for HMMs
- Sequential message schedules require each message to be updated only once
- Computational cost:

$N \longrightarrow$ number of nodes

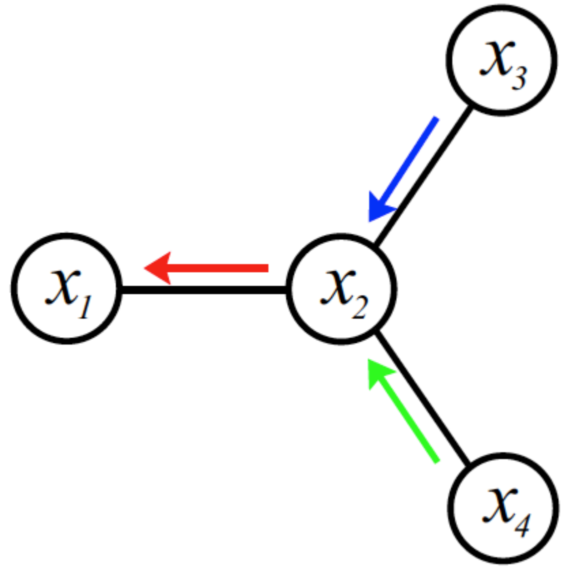
$M \longrightarrow$ discrete states for each node

Belief Prop: $\mathcal{O}(NM^2)$

Brute Force: $\mathcal{O}(M^N)$



BP for Continuous Variables

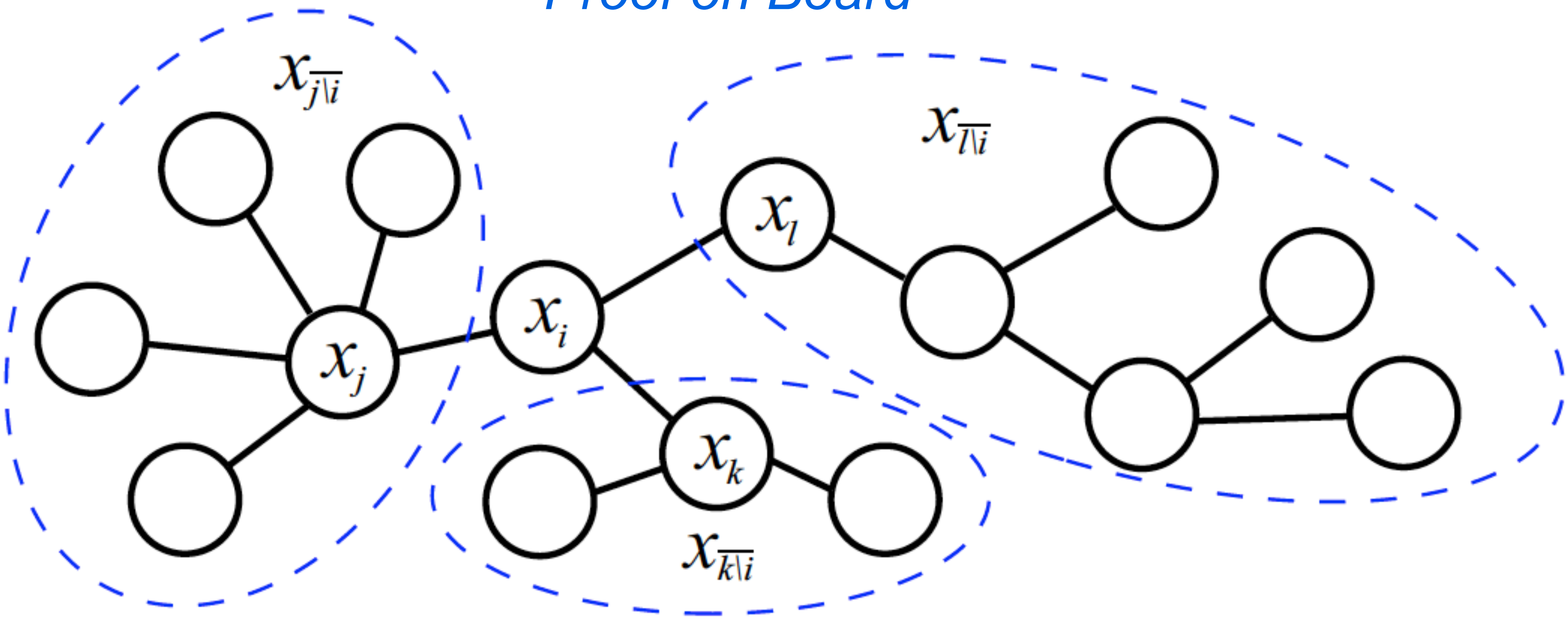


$$
\begin{aligned}
p(x_1) &\propto \iiint \psi_1(x_1)\psi_{12}(x_1,x_2)\psi_2(x_2)\psi_{23}(x_2,x_3)\psi_3(x_3)\psi_{24}(x_2,x_4)\psi_4(x_4)\,dx_4\,dx_3\,dx_2 \\
&\propto \psi_1(x_1)\iiint \psi_{12}(x_1,x_2)\psi_2(x_2)\psi_{23}(x_2,x_3)\psi_3(x_3)\psi_{24}(x_2,x_4)\psi_4(x_4)\,dx_4\,dx_3\,dx_2 \\
&\propto \psi_1(x_1)\int \psi_{12}(x_1,x_2)\psi_2(x_2)\left[\iint \psi_{23}(x_2,x_3)\psi_3(x_3)\psi_{24}(x_2,x_4)\psi_4(x_4)\,dx_4\,dx_3\right]dx_2 \\
&\propto \psi_1(x_1)\int \psi_{12}(x_1,x_2)\psi_2(x_2)\underbrace{\left[\int \psi_{23}(x_2,x_3)\psi_3(x_3)\,dx_3\right]}_{m_{32}(x_2)}\cdot\underbrace{\left[\int \psi_{24}(x_2,x_4)\psi_4(x_4)\,dx_4\right]}_{m_{42}(x_2)}dx_2
\end{aligned}
$$

$$
m_{21}(x_1) \propto \int \psi_{12}(x_1,x_2)\psi_2(x_2)m_{32}(x_2)\,m_{42}(x_2)\ dx_2
$$

BP is Exact for Trees

Proof on Board

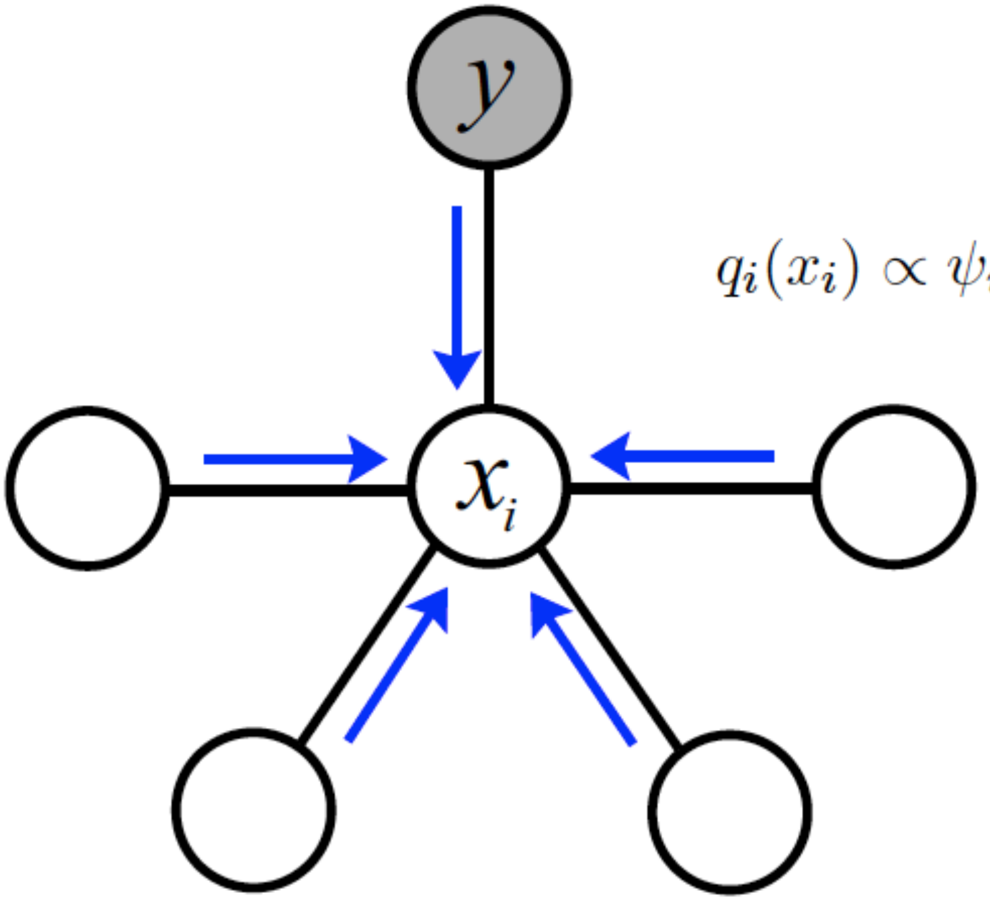


$$\overline{j \setminus i} \triangleq \{j\} \cup \{k \in \mathcal{V} \mid \text{no path from } k \to j \text{ intersects } i\}$$

$$\Psi_{\mathcal{A}}(x_{\mathcal{A}}) \triangleq \prod_{(i,j)\in\mathcal{E}(\mathcal{A})} \psi_{ij}(x_i, x_j) \prod_{i\in\mathcal{A}} \psi_i(x_i, y) \qquad \mathcal{A} \subset \mathcal{V}$$

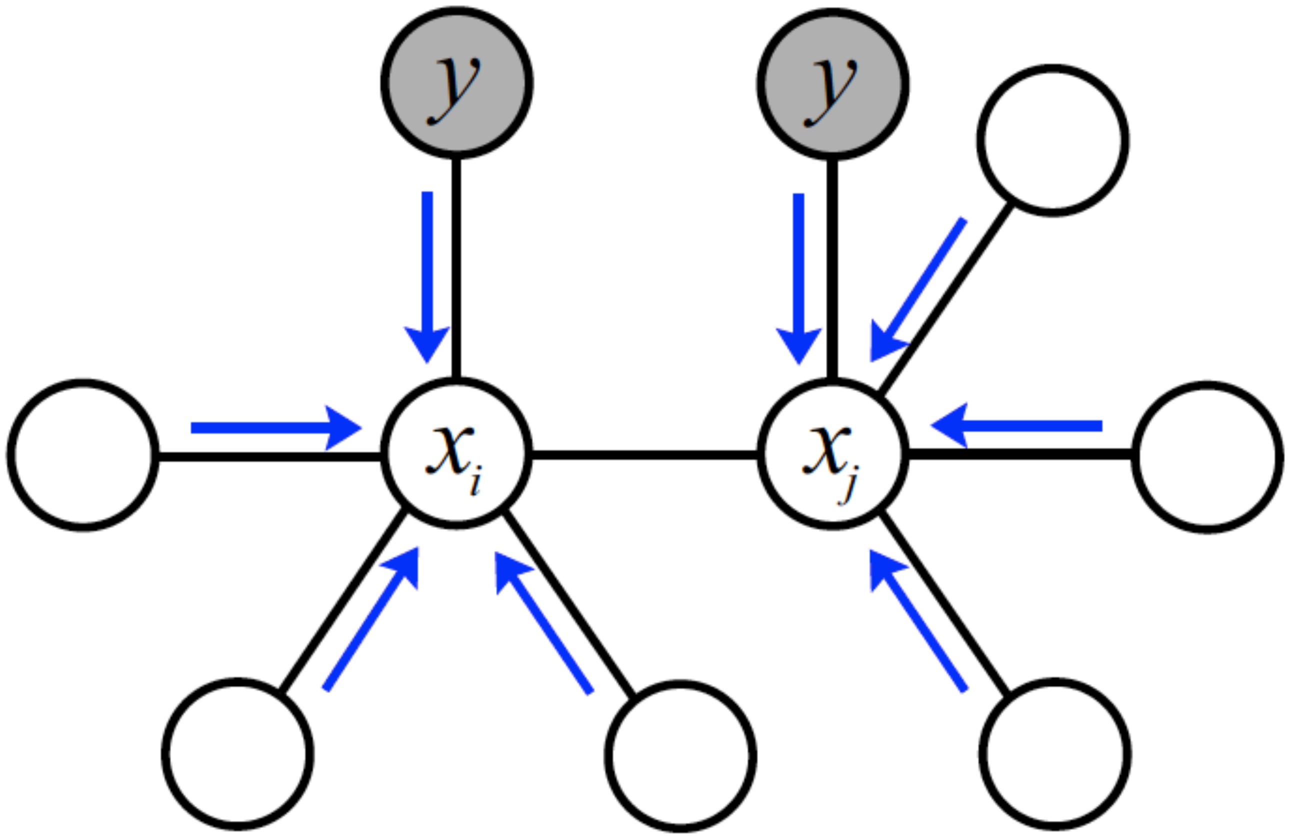
$$\mathcal{E}(\mathcal{A}) \triangleq \{(i,j) \in \mathcal{E} \mid i, j \in \mathcal{A}\}$$

BP Algorithm



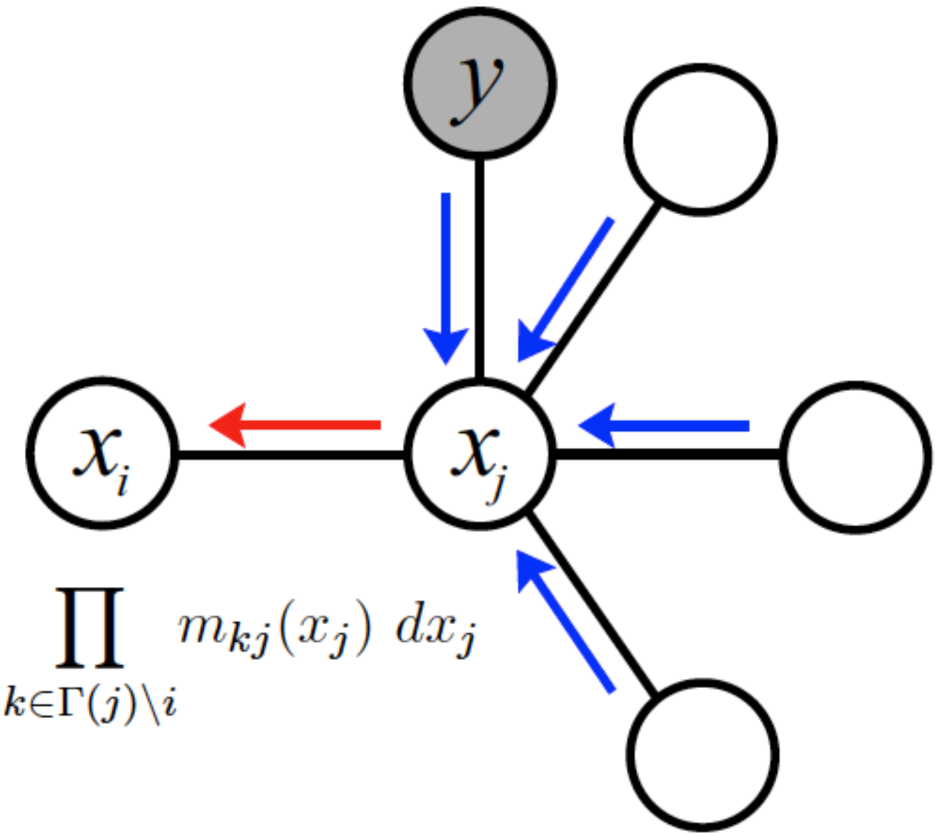
$$q_i(x_i) \propto \psi_i(x_i, y) \prod_{j \in \Gamma(i)} m_{ji}(x_i)$$

BP Algorithm



$$q_{ij}(x_i, x_j) \propto \psi_{ij}(x_i, x_j)\, \psi_i(x_i, y)\, \psi_j(x_j, y) \prod_{\ell \in \Gamma(i) \setminus j} m_{\ell i}(x_i) \prod_{k \in \Gamma(j) \setminus i} m_{kj}(x_j)$$

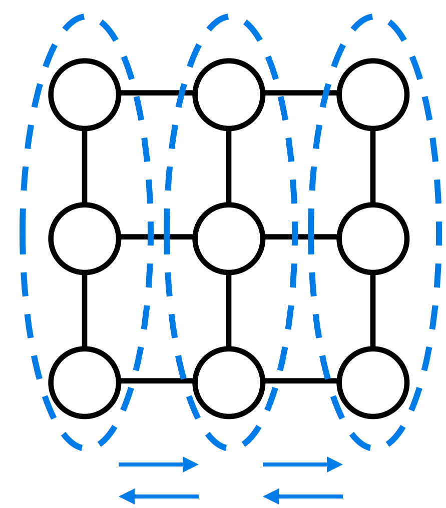
BP Algorithm



$$m_{ji}(x_i) \propto \int_{\mathcal{X}_j} \psi_{ij}(x_i, x_j)\, \psi_j(x_j, y) \prod_{k \in \Gamma(j) \setminus i} m_{kj}(x_j)\; dx_j$$

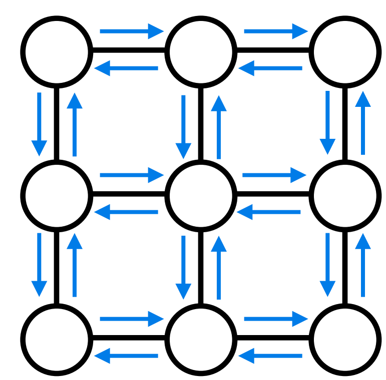
Inference for Graphs with Cycles

- For graphs with cycles, the dynamic programming BP derivation breaks



Junction Tree Algorithm

- Cluster nodes to break cycles
- Run BP on the tree of clusters
- Exact, but often intractable



Loopy Belief Propagation

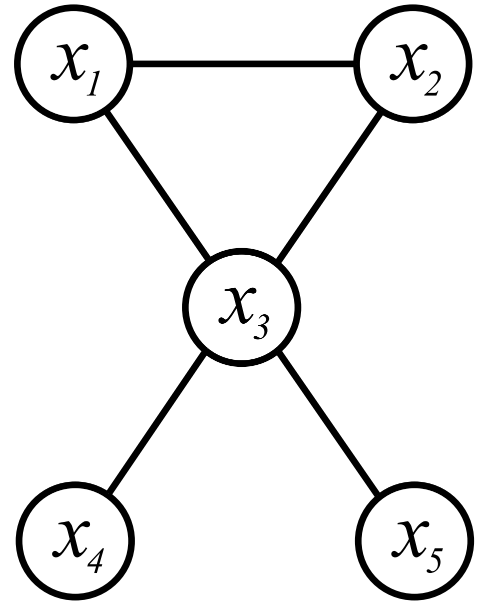
- Iterate local BP message updates on the graph with cycles
- Hope beliefs converge
- Empirically, often very effective…

A Brief History of Loopy BP

- 1993: Turbo codes (and later LDPC codes, rediscovered from Gallager's 1963 thesis) revolutionize error correcting codes (Berrou et. al.)
- 1995-1997: Realization that turbo decoding algorithm is equivalent to loopy BP (MacKay & Neal)
- 1997-1999: Promising results in other domains, & theoretical analysis via computation trees (Weiss)
- 2000: Connection between loopy BP & variational approximations, using ideas from statistical physics (Yedidia, Freeman, & Weiss)
- 2001-2007: Many results interpreting, justifying, and extending loopy BP

Pairwise Markov Random Fields

$$p(x) = \frac{1}{Z} \prod_{(s,t)\in\mathcal{E}} \psi_{st}(x_s, x_t) \prod_{s\in\mathcal{V}} \psi_s(x_s)$$



- Simple parameterization, but still expressive and widely used in practice
- Guaranteed Markov with respect to graph

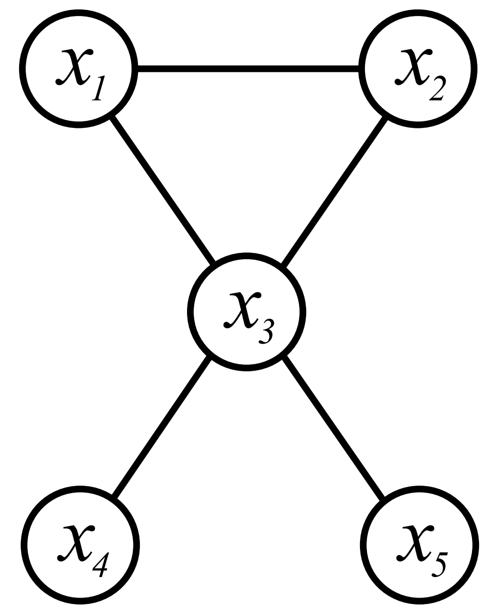
$\mathcal{E} \longrightarrow$ set of undirected edges (s,t) linking pairs of nodes

$\mathcal{V} \longrightarrow$ set of N nodes or vertices, $\{1, 2, \ldots, N\}$

$Z \longrightarrow$ normalization constant (partition function)

Undirected Graphical Models

$$p(x) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \psi_c(x_c)$$



- Parameterization exactly captures those non-degenerate distributions which are Markov with respect to this graph
- Sometimes restricted to maximal cliques, but this is not necessary

$\mathcal{C} \longrightarrow$ set of cliques (fully connected subsets) of nodes

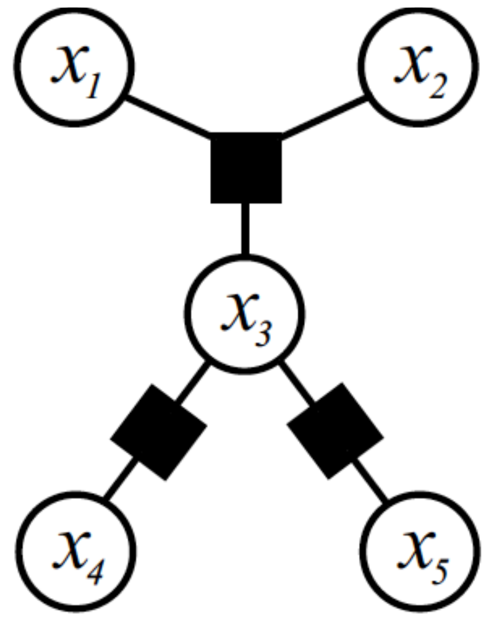
$\mathcal{E} \longrightarrow$ set of undirected edges (s,t) linking pairs of nodes

$\mathcal{V} \longrightarrow$ set of N nodes or vertices, $\{1, 2, \ldots, N\}$

$Z \longrightarrow$ normalization constant (partition function)

Factor Graphs

$$p(x) = \frac{1}{Z} \prod_{f \in \mathcal{F}} \psi_f(x_f)$$



- In a *hypergraph*, the *hyperedges* link arbitrary subsets of nodes (not just pairs)
- Visualize by a bipartite graph, with square (usually black) nodes for hyperedges
- A *factor graph* associates a non-negative potential function with each hyperedge
- Motivation: *factorization key to computation*

$\mathcal{F} \longrightarrow$ set of hyperedges linking subsets of nodes $f \subseteq \mathcal{V}$

$\mathcal{V} \longrightarrow$ set of N nodes or vertices, $\{1, 2, \ldots, N\}$

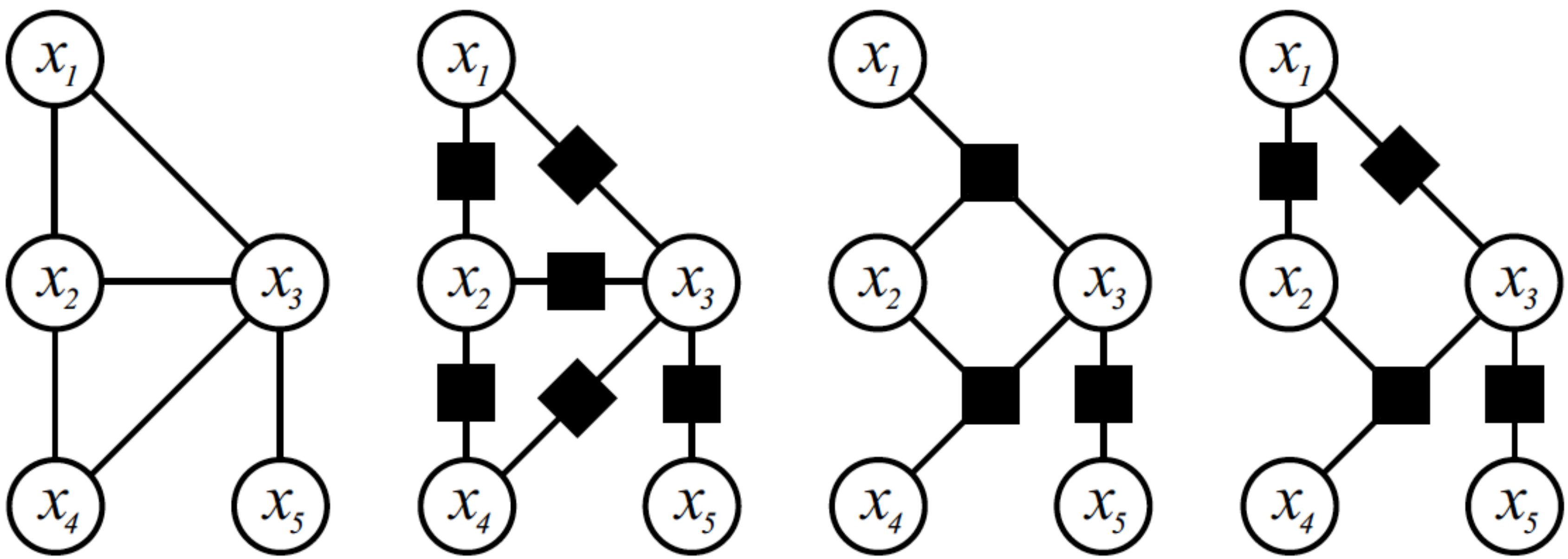
$Z \longrightarrow$ normalization constant (partition function)

Factor Graphs & Factorization

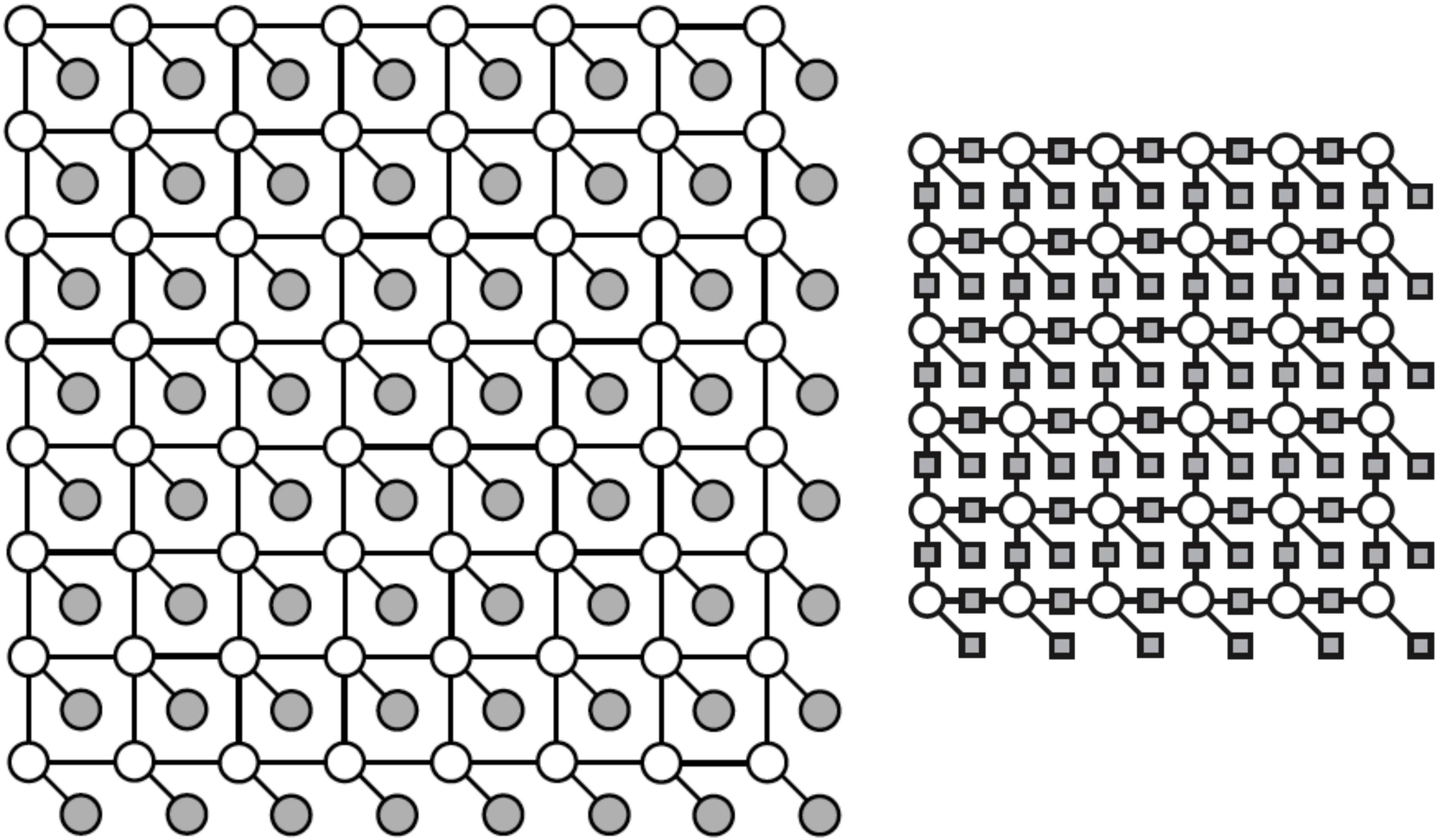
$$p(x) = \frac{1}{Z} \prod_{f \in \mathcal{F}} \psi_f(x_f)$$

- For a given undirected graph, there exist distributions which have equivalent Markov properties, but different factorizations and different inference/learning complexities:

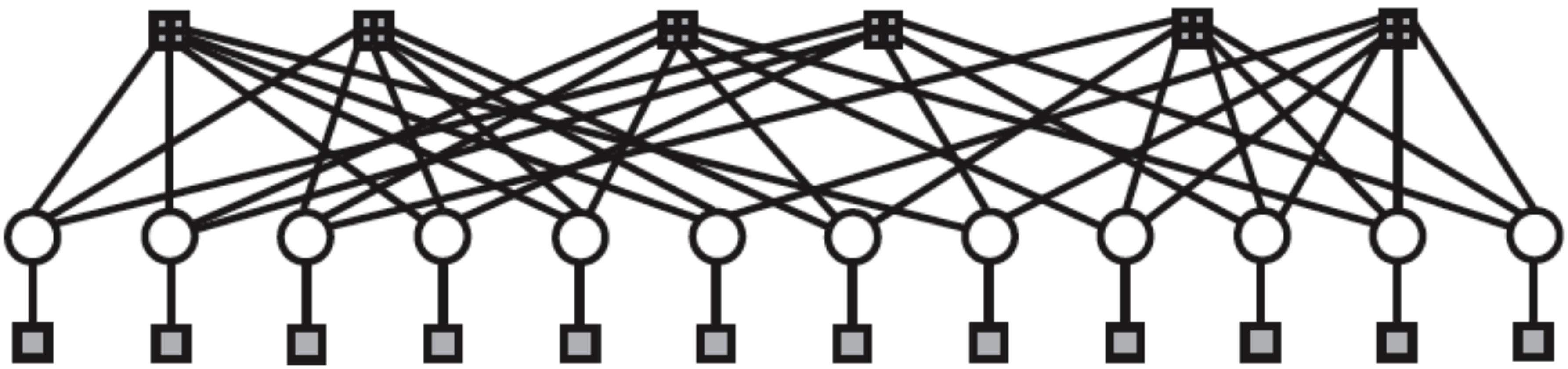
Undirected Graphical Model | *Pairwise (edge) Potentials* | *Potentials on Maximal Cliques* | *Alternative Factorization*



Pairwise Nearest-Neighbor MRF



Low Density Parity Check (LDPC) Code



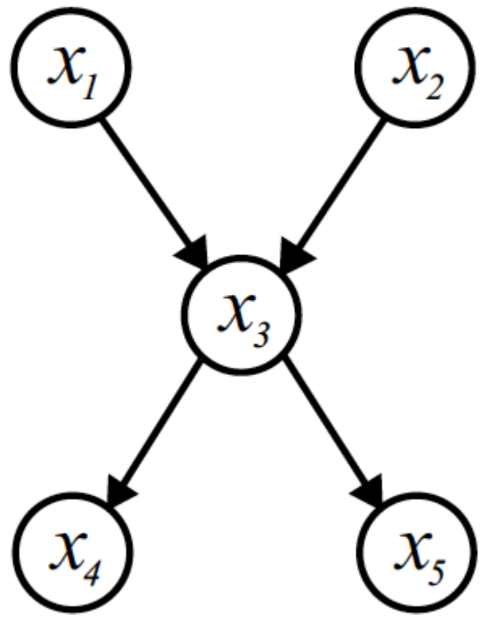
Directed Graphs as Factor Graphs

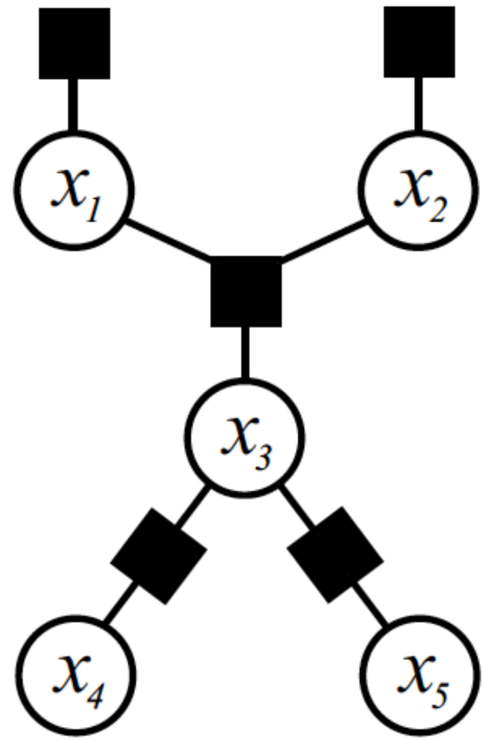
Directed Graphical Model:

$$p(x) = \prod_{i=1}^{N} p(x_i \mid x_{\Gamma(i)})$$

Corresponding Factor Graph:

$$p(x) = \prod_{i=1}^{N} \psi_i(x_i, x_{\Gamma(i)})$$





- Associate one factor with each node, linking it to its parents and defined to equal the corresponding conditional distribution
- Information lost: Directionality of conditional distributions, and fact that global partition function $Z = 1$