Loopy Belief Propagation

Kayhan Batmanghelich

Inference Problems

- Compute the likelihood of observed data
- Compute the marginal distribution $p(x_A)$ over a particular subset of nodes $A \subset V$
- Compute the conditional distribution $p(x_A|x_B)$ for disjoint subsets *A* and *B*
- Compute a mode of the density $\hat{x} = \arg\max_{x \in \mathcal{X}^m} p(x)$

- Methods we have

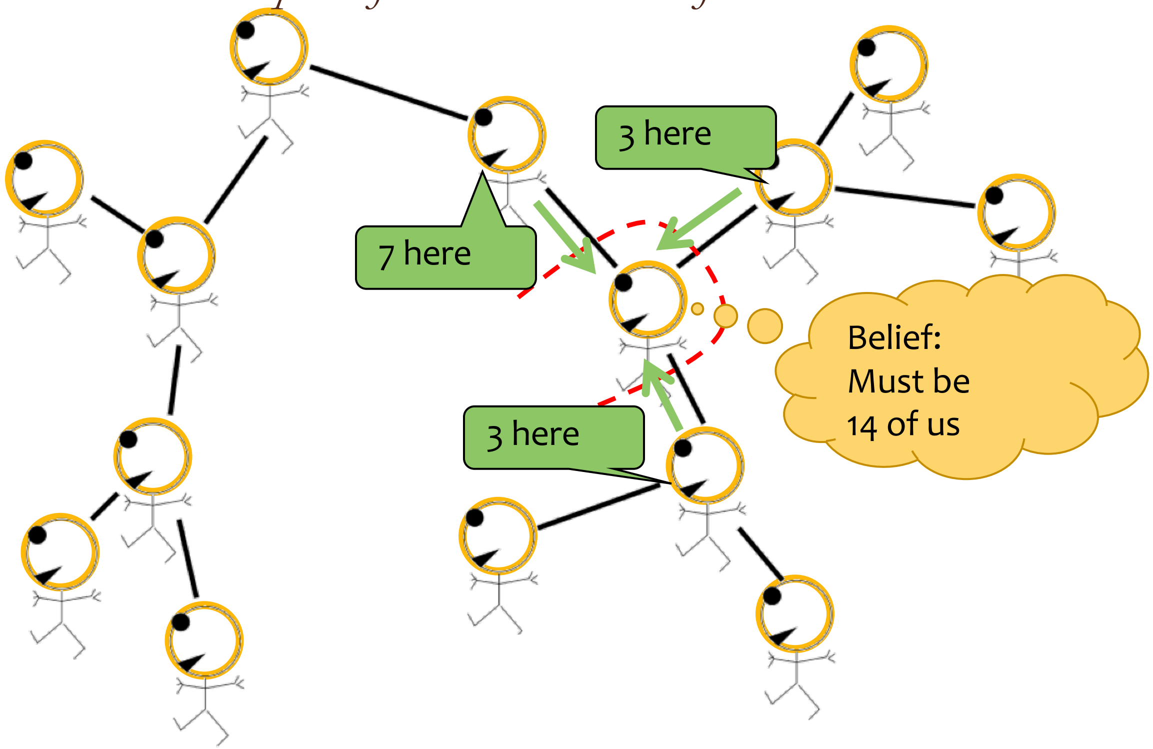
Brute force | **Elimination** ⇒ **Message Passing** **(Forward-backward , Max-product /BP, Junction Tree)**

Individual computations independent

Sharing intermediate terms

Great Ideas in ML: Message Passing

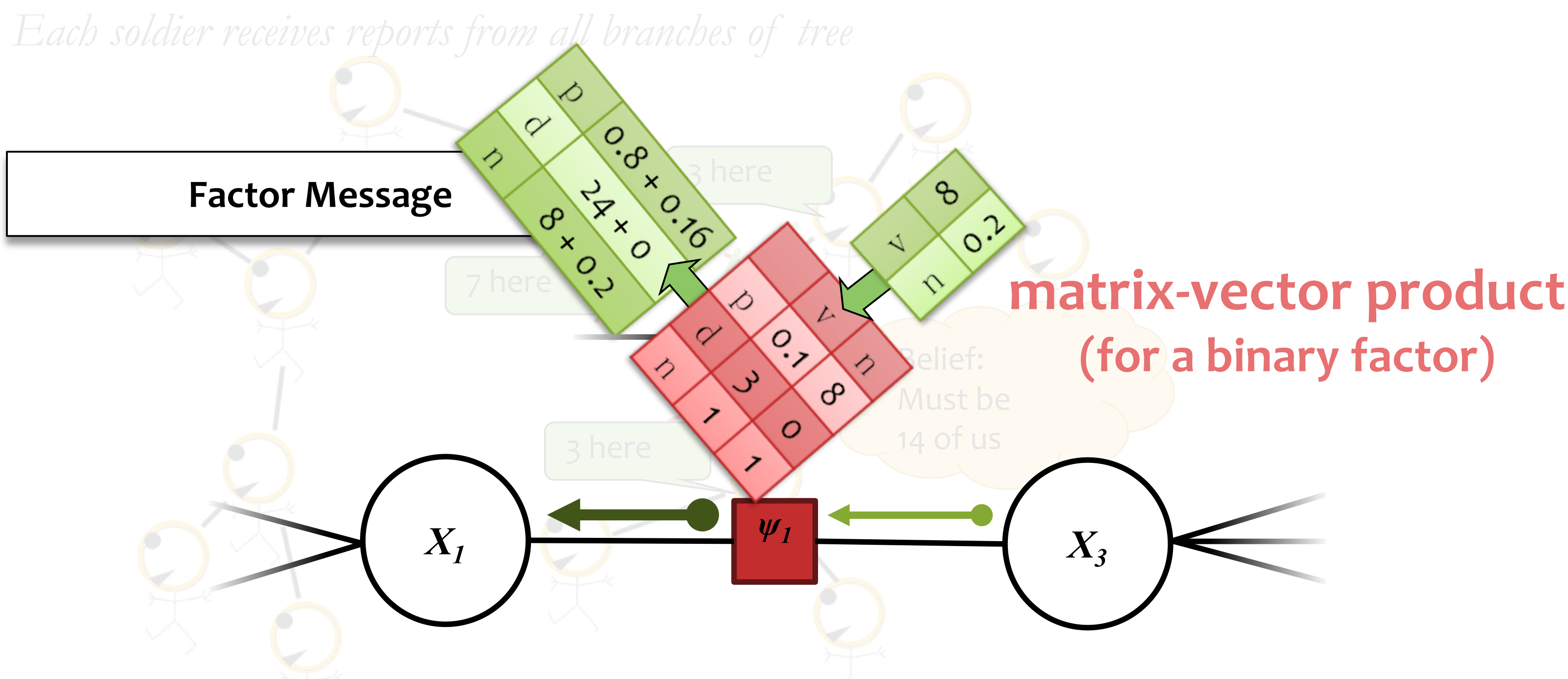
Each soldier receives reports from all branches of tree



Slides adapted from Matt Gormley (2016)

adapted from MacKay (2003) textbook

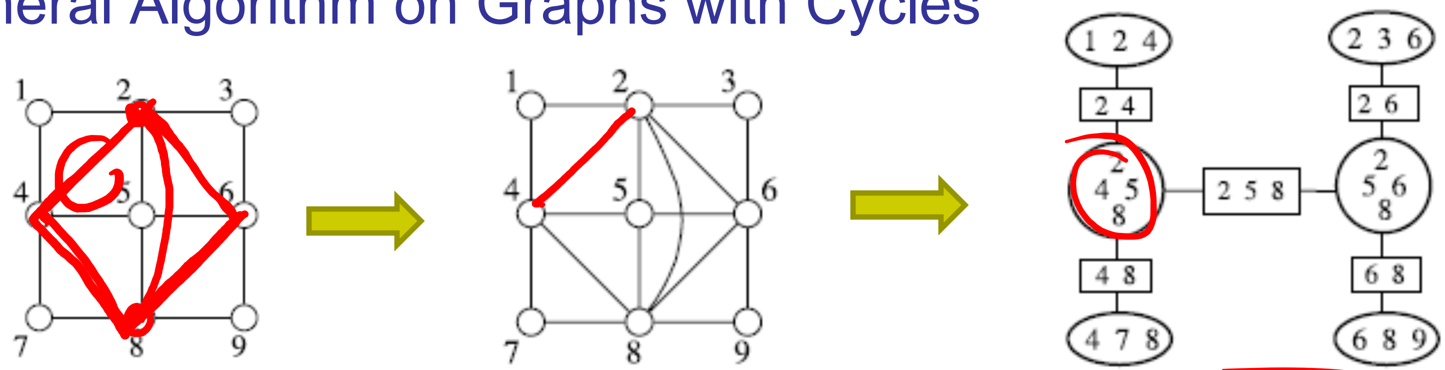
Sum-Product Belief Propagation



$$\mu_{\alpha\to i}(x_i) = \sum_{\boldsymbol{x}_\alpha : \boldsymbol{x}_\alpha[i]=x_i} \psi_\alpha(\boldsymbol{x}_\alpha) \prod_{j\in\mathcal{N}(\alpha)\setminus i} \mu_{j\to\alpha}(\boldsymbol{x}_\alpha[i])$$

Junction Tree Revisited

- General Algorithm on Graphs with Cycles

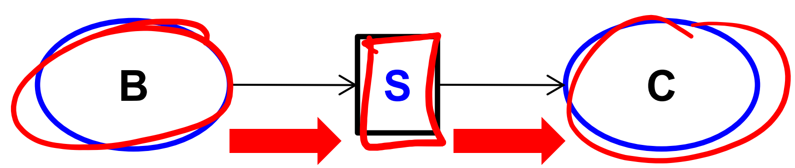


  - Steps: => **Triangularization** => **Construct JTs**

=> Message Passing on Clique Trees

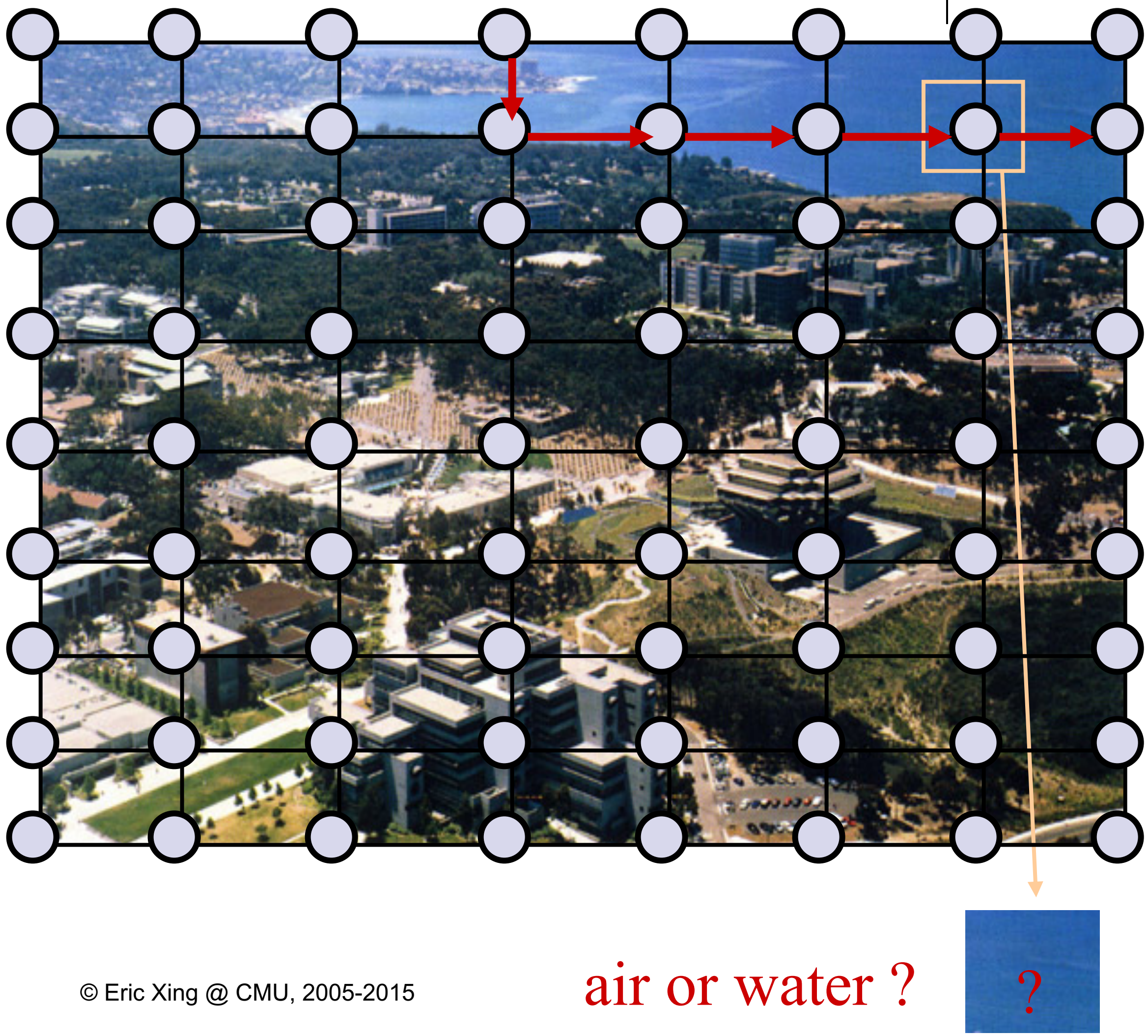
$$\tilde{\phi}_S(x_S) \leftarrow \sum_{x_{B \setminus S}} \phi_B(x_B)$$

$$\phi_C(x_C) \leftarrow \frac{\tilde{\phi}_S(x_S)}{\phi_S(x_S)} \phi_C(x_C)$$

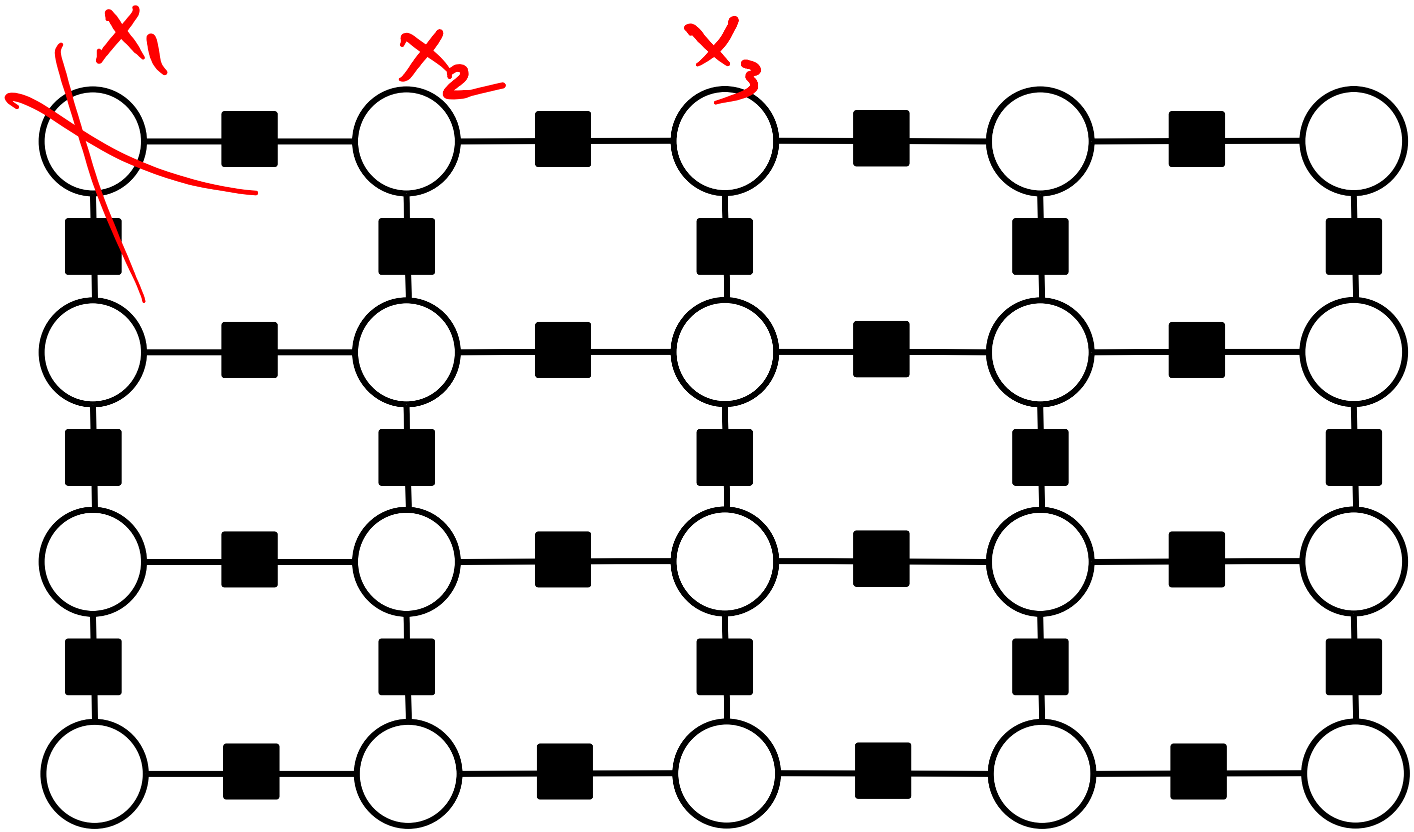


An Ising model on 2-D image

- Nodes encode hidden information (patch-identity).
- They receive local information from the image (brightness, color).
- Information is propagated though the graph over its edges.
- Edges encode 'compatibility' between nodes.

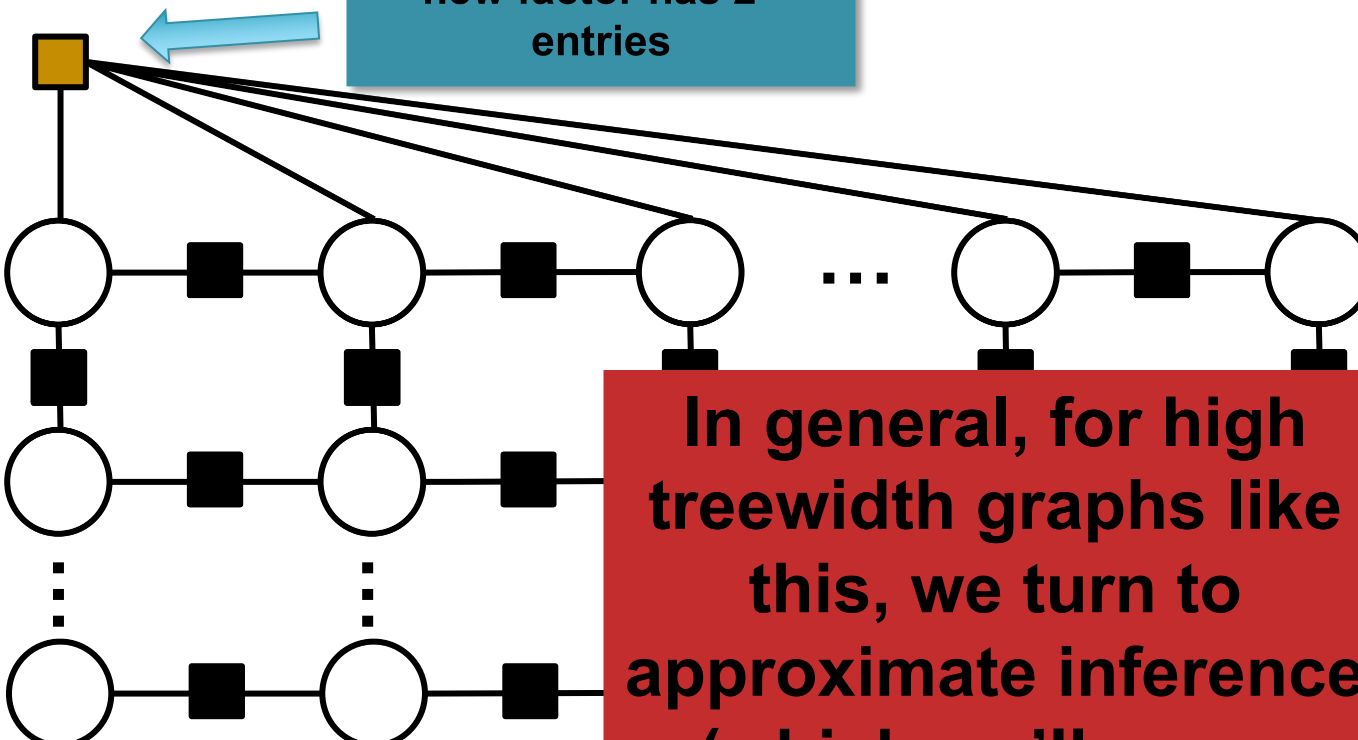


Why Approximate Inference?



$$p(X) = \frac{1}{Z} \exp\{\sum_{i<j} \theta_{ij} x_i x_j + \sum_i \theta_{i0} x_i\}$$

Why Approximate Inference?



For an MxM grid the new factor has 2^M entries

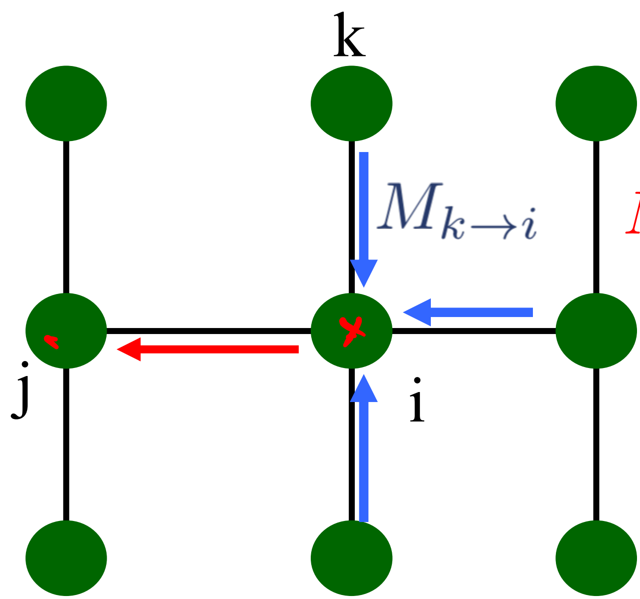
In general, for high treewidth graphs like this, we turn to approximate inference (which we'll cover soon!)

Approaches to inference

- Exact inference algorithms
  - The elimination algorithm
  - Message-passing algorithm (sum-product, belief propagation)
  - The junction tree algorithms
- Approximate inference techniques
  - Variational algorithms
    - Loopy belief propagation
    - Mean field approximation
  - Stochastic simulation / sampling methods
  - Markov chain Monte Carlo methods

Recap of Belief Propagation

Recap: Belief Propagation



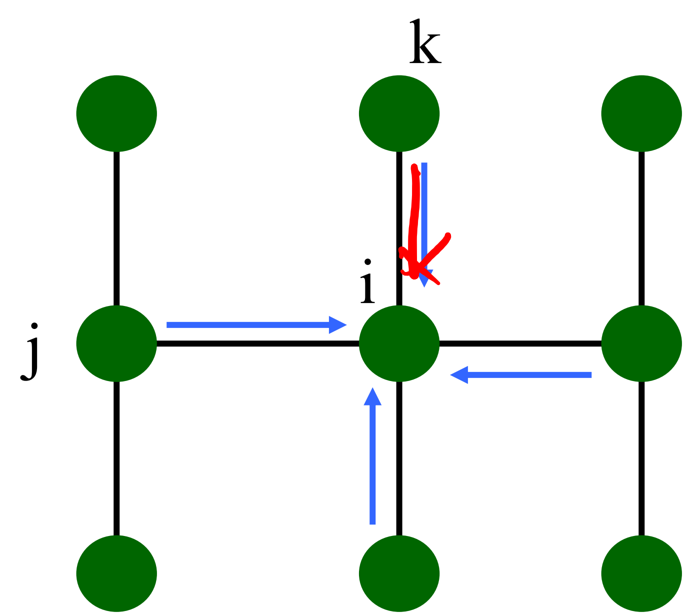
bin uni

$$M_{k\to i}(x_j) \propto \sum_{x_i} \psi_{ij}(x_i, x_j)\,\psi_i(x_i) \prod_k M_{k\to i}(x_i)$$

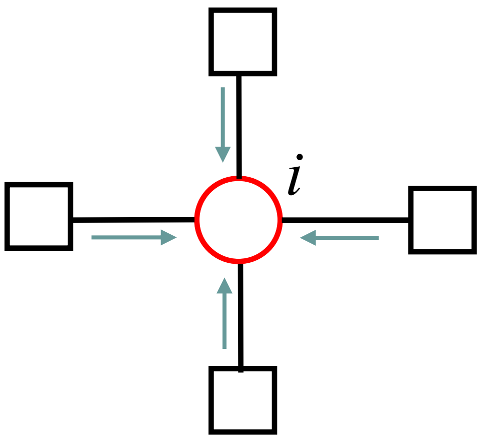
Interactions

External evidence

$$b_i(x_i) \propto \psi(x_i) \prod_k M_{k\to i}(x_i)$$



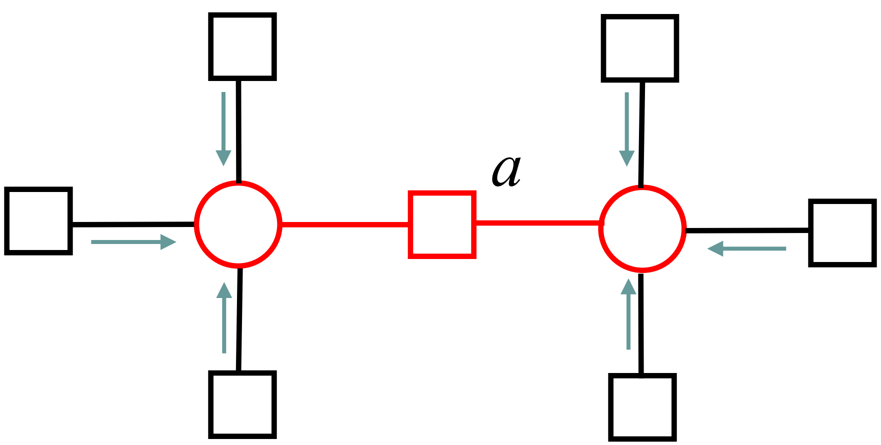
Beliefs and messages in FG



$$m_{i\to a}(x_i) = \prod_{c\in N(i)\setminus a} m_{c\to i}(x_i)$$

$$b_i(x_i) \propto \prod_{a\in N(i)} m_{a\to i}(x_i)$$

"beliefs" "messages"



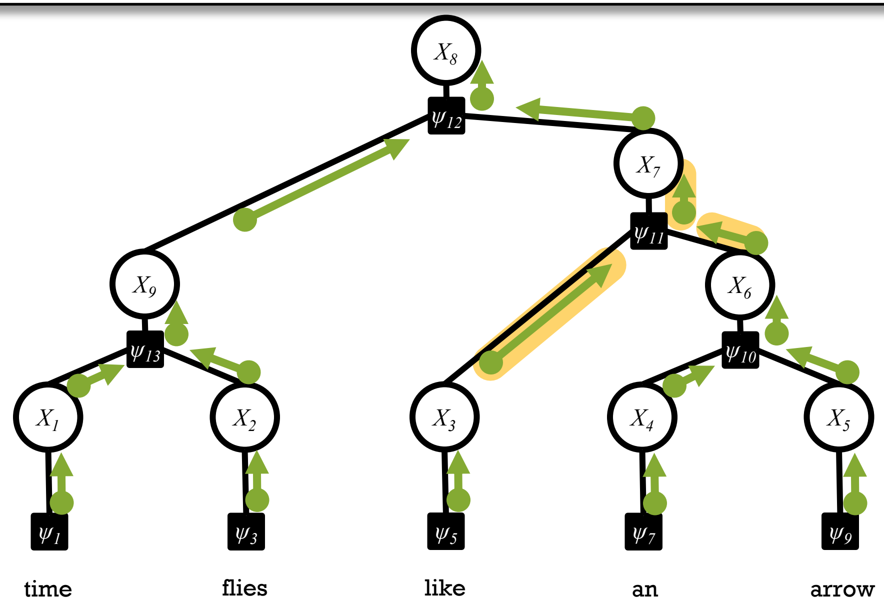
$$b_a(X_a) \propto f_a(X_a) \prod_{i\in N(a)} m_{i\to a}(x_i)$$

$$m_{a\to i}(x_i) = \sum_{X_a\setminus x_i} f_a(X_a) \prod_{j\in N(a)\setminus i} m_{j\to a}(x_j)$$

(Acyclic) Belief Propagation

In a factor graph with no cycles:
1. Pick any node to serve as the root.
2. Send messages from the **leaves** to the **root**.
3. Send messages from the **root** to the **leaves**.

A node computes an outgoing message along an edge
only after it has received incoming messages along all its other edges.



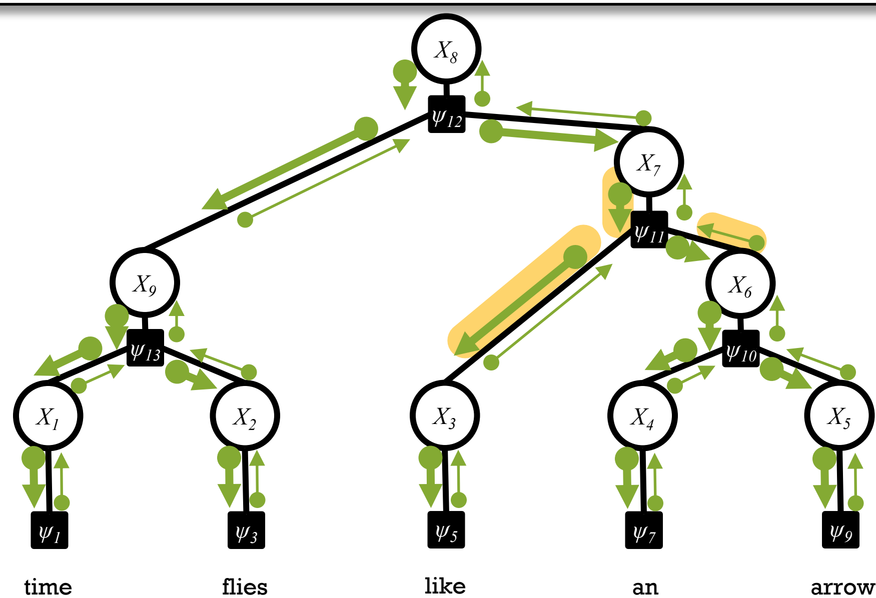
Slides adapted from Matt Gormley (2016)

(Acyclic) Belief Propagation

In a factor graph with no cycles:

1. Pick any node to serve as the root.
2. Send messages from the **leaves** to the **root**.
3. Send messages from the **root** to the **leaves**.

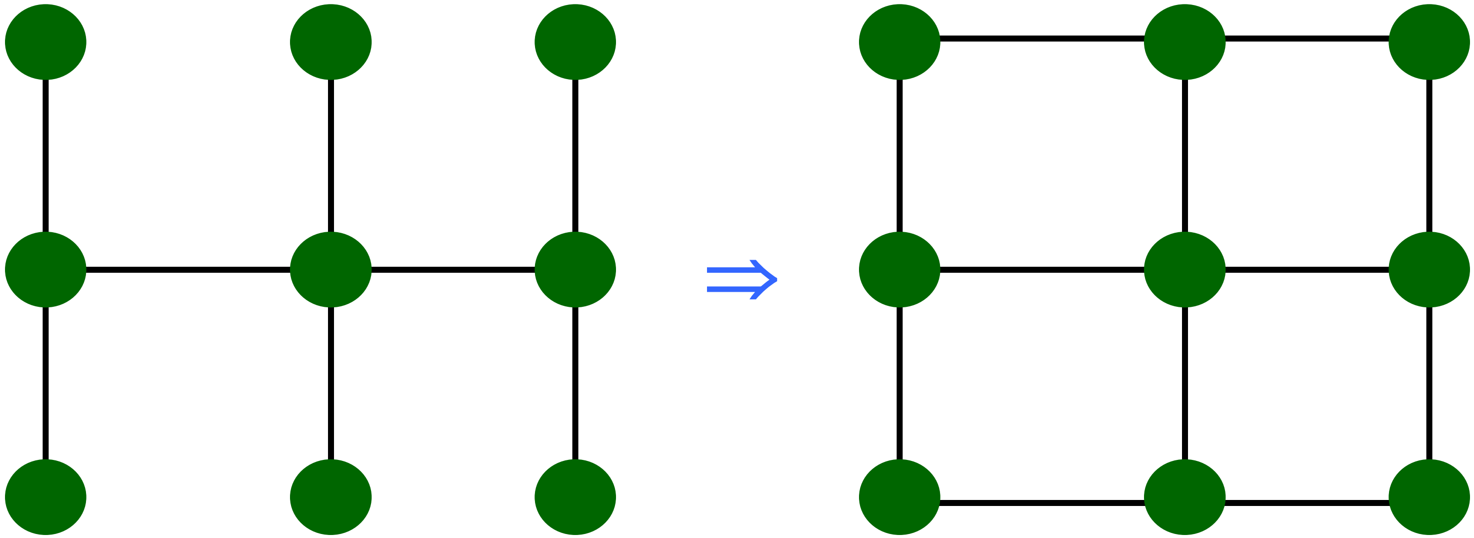
A node computes an outgoing message along an edge
only after it has received incoming messages along all its other edges.



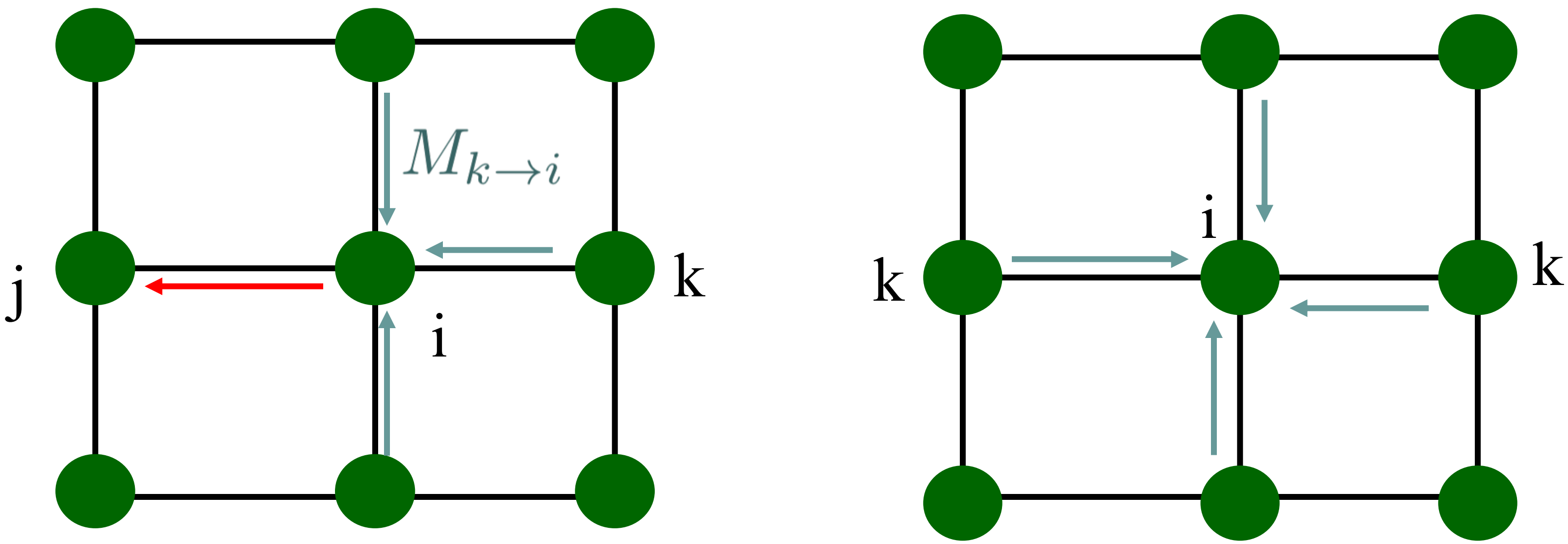
Slides adapted from Matt Gormley (2016)

What if there is a loop?

What if the graph is loopy?



Belief Propagation on loopy graphs



- BP Message-update Rules

$$M_{i\to j}(x_j) \propto \sum_{x_i} \psi_{ij}(x_i, x_j)\psi_i(x_i)\prod_k M_{k\to i}(x_i) \qquad b_i(x_i) \propto \psi_i(x_i)\prod_k M_k(x_k)$$

Compatibilities (interactions)

external evidence

- May not converge or converge to a wrong solution

Loopy Belief Propagation

- A fixed point iteration procedure that tries to minimize F_{bethe}
- Start with random initialization of messages and beliefs
  - While not converged do

$$b_i(x_i) \propto \prod_{a \in N(i)} m_{a \to i}(x_i) \qquad b_a(X_a) \propto f_a(X_a) \prod_{i \in N(a)} m_{i \to a}(x_i)$$

$$m_{i \to a}^{new}(x_i) = \prod_{c \in N(i) \backslash a} m_{c \to i}(x_i) \qquad m_{a \to i}^{new}(x_i) = \sum_{X_a \backslash x_i} f_a(X_a) \prod_{j \in N(a) \backslash i} m_{j \to a}(x_j)$$

  - At convergence, stationarity properties are guaranteed
  - However, not guaranteed to converge!

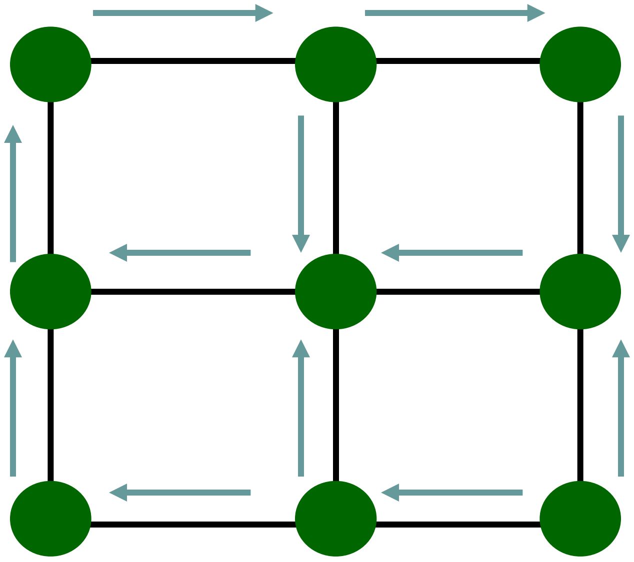
Loopy Belief Propagation

- If BP is used on graphs with loops, messages may circulate indefinitely
- But let's run it anyway and hope for the best … ☺
- How to stop it?
  - Stop after fixed # of iterations
  - Stop when no significant change in beliefs
  - If solution is not oscillatory but converges, it usually is a good approximation

Loopy-belief Propagation for Approximate Inference: An Empirical Study
Kevin Murphy, Yair Weiss, and Michael Jordan.
***UAI '99 (Uncertainty in AI).*]**

So what is going on?

- Is it a dirty hack that you bet your luck?



$P(x_A \mid \text{Data})$

approximate $q(x_A) = 42$

How to measure how close we to the correct answer?

$D(Q_2, Q_1) = KL(Q_1 || Q_2) + KL(Q_2 || Q_1)$

Approximate Inference

$D(Q_1, Q_2) \leq$

$D(Q_1, Q_3)$

$+ D(Q_3, Q_2)$

- Let us call the actual distribution P

$$P(X) = 1/Z \prod_{f_a \in F} f_a(X_a)$$

- We wish to find a distribution Q such that Q is a "good" approximation to P
- Recall the definition of KL-divergence

$$KL(Q_1 || Q_2) = \sum_X \underline{Q_1(X) \log(\frac{Q_1(X)}{Q_2(X)})}$$

- $KL(Q_1||Q_2)>=0$ ✓
- $KL(Q_1||Q_2)=0$ iff $Q_1=Q_2$
- We can therefore use KL as a scoring function to decide a good Q
- But, $KL(Q_1||Q_2) \neq KL(Q_2||Q_1)$

Which KL?

- Computing KL(P||Q) requires inference!
- But KL(Q||P) can be computed without performing inference on P

$$KL(Q \| P) = \sum_X Q(X) \log\left(\frac{Q(X)}{P(X)}\right)$$

$$= \sum_X Q(X) \log Q(X) - \sum_X Q(X) \log P(X) = \mathbb{E}[\log Q(X)]$$

$$= -H_Q(X) - E_Q \log P(X)$$

$\mathbb{E}[\log P(X)]$

- Using $P(X) = 1/Z \prod_{f_a \in F} f_a(X_a)$

$$KL(Q \| P) = -H_Q(X) - E_Q \log(1/Z \prod_{f_a \in F} f_a(X_a))$$

$$= -H_Q(X) - \log 1/Z - \sum_{f_a \in F} E_Q \log f_a(X_a)$$

Optimization function

p(X)

Q

$$KL(Q\|P) = \underbrace{-H_Q(X) - \sum_{f_a \in F} E_Q \log f_a(X_a)}_{F(P,Q)} + \log Z$$

Q = P

- We will call $F(P,Q)$ the "Free energy" *
- $F(P,P)$ =?
- F(P,Q) >= ~~F(P,P)~~

0

*Gibbs Free Energy

The Energy Functional

- Let us look at the functional

$$F(P,Q) = \boxed{-H_Q(X)} - \boxed{\sum_{f_a \in F} E_Q \log f_a(X_a)}$$

- $\sum_{f_a \in F} E_Q \log f_a(X_a)$ can be computed if we have marginals over each f_a
- $H_Q = -\sum_X Q(X) \log Q(X)$ is harder! Requires summation over all possible values
- Computing F, is therefore hard in general.
- Our goals is to:

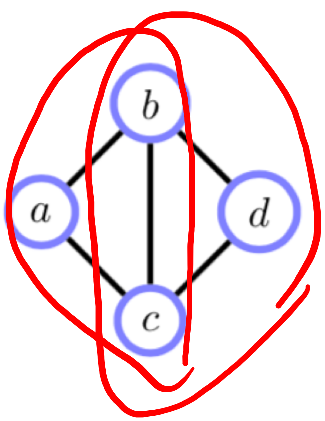
Can we approximate it?

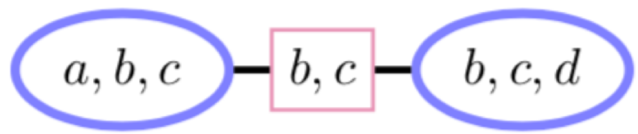
$$Q^* = \arg\max_{Q \in \boxed{\mathcal{Q}}} \boxed{F(P,Q)}$$

Can we suggest an easy family?

Work out a simple case

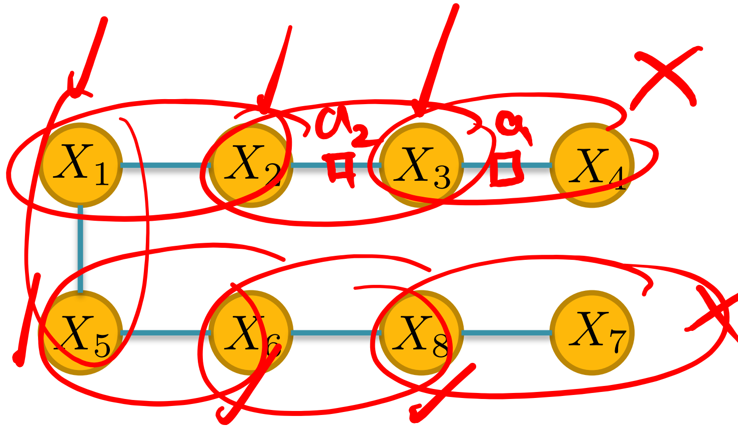
Do you remember this from lecture 6 ?





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

A Tree example



$$F(P,Q) = -H_Q(X) - \sum_{f_a \in F} E_Q \log f_a(X_a)$$

$$p(X_1, ..., X_8) = \frac{1}{Z}\phi_{43}(X_4, X_3)\phi_{32}(X_3, X_2)\phi_{21}(X_2, X_1)\phi_{15}(X_1, X_5)\phi_{56}(X_5, X_6)\cdots$$

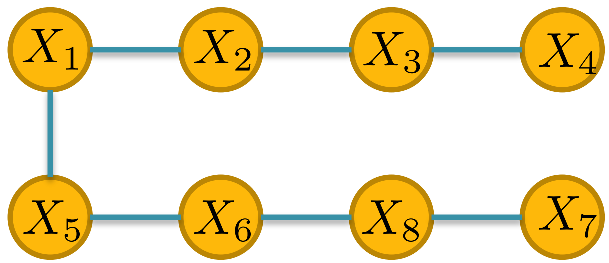
$P(X_3, X_4)\, P(X_2, X_3)\, P(X_1, X_2) \dots$

$P(X_3)\, P(X_2) \dots$

$$H(X_1, \cdots, X_8) = -\sum_{a \in \text{num}} \mathbb{E}[\log p(X_a)] + \sum_{i \in \text{den}} \mathbb{E}[\log p(x_i)]$$

$$F(X_1, \cdots, X_8) = -\sum_{a \in \text{num}} \mathbb{E}[\log \frac{p(X_a)}{f(X_a)}] + \sum_{i \in \text{den}} \mathbb{E}[\log \frac{p(X_i)}{f(X_i)}]$$

For a general tree



$$F(P,Q) = -H_Q(X) - \sum_{f_a \in F} E_Q \log f_a(X_a)$$

- The probability can be written as: $b(\mathbf{x}) = \underbrace{\prod_a b_a(\mathbf{x}_a)}_{\text{num}} \underbrace{\prod_i b_i(x_i)^{1-d_i}}_{\text{den}}$

$$H_{tree} = -\sum_a \sum_{\mathbf{x}_a} b_a(\mathbf{x}_a) \ln b_a(\mathbf{x}_a) + \sum_i (d_i - 1) \sum_{\mathbf{x}_i} b_i(\mathbf{x}_i) \ln b_i(\mathbf{x}_i)$$

$$F_{Tree} = \sum_a \sum_{\mathbf{x}_a} b_a(\mathbf{x}_a) \ln \frac{b_a(\mathbf{x}_a)}{f_a(\mathbf{x}_a)} + \sum_i (1 - d_i) \sum_{\mathbf{x}_i} b_i(\mathbf{x}_i) \ln b_i(\mathbf{x}_i)$$

Degree of the node

  - involves summation over edges and vertices and is therefore easy to compute

$$= F_{12} + F_{23} + .. + F_{67} + F_{78} - F_1 - F_5 - F_2 - F_6 - F_3 - F_7$$

Let’s extend it to a general graph

Bethe Approximation to Gibbs Free Energy

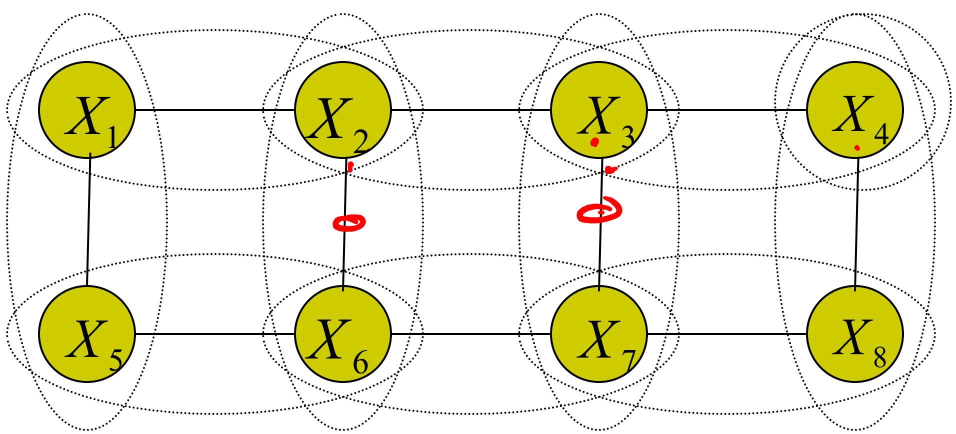
$F(Q,P)$ $\hat{F}$

- For a general graph, choose $\hat{F}(P,Q) = F_{Betha}$

$$H_{Bethe} = -\sum_a \sum_{\mathbf{x}_a} b_a(\mathbf{x}_a)\ln b_a(\mathbf{x}_a) + \sum_i (d_i - 1)\sum_{\mathbf{x}_i} b_i(\mathbf{x}_i)\ln b_i(\mathbf{x}_i)$$

$$F_{Bethe} = \sum_a \sum_{\mathbf{x}_a} b_a(\mathbf{x}_a)\ln\frac{b_a(\mathbf{x}_a)}{f_a(\mathbf{x}_a)} + \sum_i (1 - d_i)\sum_{\mathbf{x}_i} b_i(\mathbf{x}_i)\ln b_i(\mathbf{x}_i) = -\langle f_a(\mathbf{x}_a)\rangle - H_{betha}$$

- Called “Bethe approximation” after the physicist Hans Bethe



$$F_{bethe} = F_{12} + F_{23} + .. + F_{67} + F_{78} - F_1 - F_5 - 2F_2 - 2F_6 .. - F_8$$

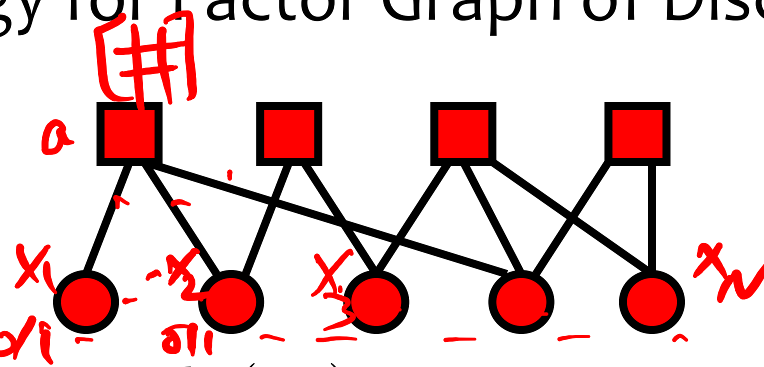
- Equal to the exact Gibbs free energy when the factor graph is a tree
- In general, H_{Bethe} is **not** the same as the H of a tree

Bethe Approximation

- Pros:
  - Easy to compute, since entropy term involves sum over pairwise and single variables
- Cons:
  - $\hat{F}(P,Q) = F_{bethe}$ **may or may not** be well connected to $F(P,Q)$
  - It could, in general, be greater, equal or less than $F(P,Q)$!!
- Optimize each $b(\mathbf{x}_a)$'s.
  - For discrete belief, constrained opt. with *Lagrangian* multiplier
  - For continuous belief, not yet a general formula
  - Not always converge

Bethe Free Energy for Factor Graph of Discrete RVs

Factor →



$x \rightarrow d_i$

$$F_{\text{Bethe}} = \sum_a \sum_{\mathbf{x}_a} \boxed{b_a(\mathbf{x}_a)} \ln \frac{b_a(\mathbf{x}_a)}{f_a(\mathbf{x}_a)} + \sum_i (1 - d_i) \sum_{\mathbf{x}_i} b_i(\mathbf{x}_i) \ln b_i(\mathbf{x}_i)$$

$$H_{\text{Bethe}} = -\sum_a \sum_{\mathbf{x}_a} b_a(\mathbf{x}_a) \ln b_a(\mathbf{x}_a) + \sum_i (d_i - 1) \sum_{\mathbf{x}_i} b_i(\mathbf{x}_i) \ln b_i(\mathbf{x}_i)$$

$$F_{\text{Bethe}} = -\sum_a \langle f_a(\mathbf{x}_a) \rangle - H_{\text{Bethe}}$$

$$\sum_{a \setminus j} b(x_a) = b_j(x_j)$$

How about optimizing this:

$b(x_{1,2})$

$a = 1, 2$

$$\min_{b_a(\mathbf{x}_a),\, b_i(\mathbf{x}_i)} F_{\text{Bethe}}$$

Subject to:

$$\sum_{x_a} b_a(x_a) = 1 \qquad \sum_{x_i} b_i(x_i) = 1 \qquad b_i(x_i) \geq 0$$

$b_a(x_a) = \exp(u)$

Minimizing the Bethe Free Energy

- $$L = F_{Bethe} + \sum_i \gamma_i \{1 - \sum_{x_i} b_i(x_i)\}$$
$$+ \sum_a \sum_{i \in N(a)} \sum_{x_i} \lambda_{ai}(x_i) \left\{ b_i(x_i) - \sum_{X_a \setminus x_i} b_a(X_a) \right\}$$

- Set derivative to zero

Constrained Minimization of the Bethe Free Energy

$$L = F_{Bethe} + \sum_i \gamma_i \{\sum_{x_i} b_i(x_i) - 1\}$$

$$+ \sum_a \sum_{i \in N(a)} \sum_{x_i} \lambda_{ai}(x_i) \left\{ \sum_{X_a \setminus x_i} b_a(X_a) - b_i(x_i) \right\}$$

$$\frac{\partial L}{\partial b_i(x_i)} = 0 \quad \Rightarrow \quad b_i(x_i) \propto \exp\left(\frac{1}{d_i - 1} \sum_{a \in N(i)} \lambda_{ai}(x_i) \right)$$

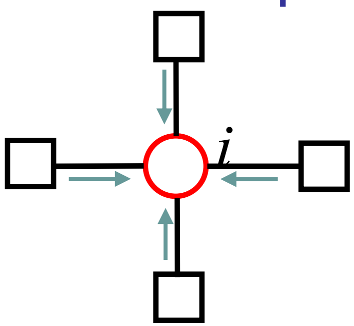
$$\frac{\partial L}{\partial b_a(X_a)} = 0 \quad \Rightarrow \quad b_a(X_a) \propto \exp\left(-E_a(X_a) + \sum_{i \in N(a)} \lambda_{ai}(x_i) \right)$$

Bethe = BP on FG

- We had:

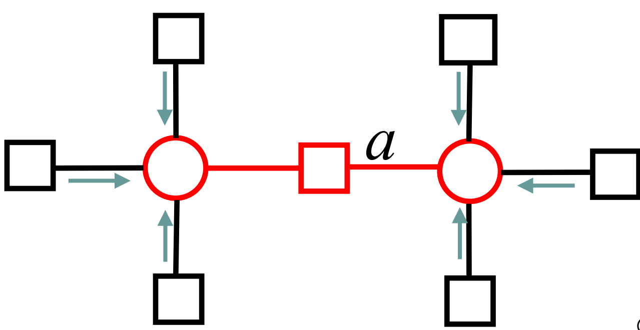
$$b_i(x_i) \propto \exp\left(\frac{1}{d_i - 1}\sum_{a \in N(i)} \lambda_{ai}(x_i)\right) \qquad b_a(X_a) \propto \exp\left(-\log f_a(X_a) + \sum_{i \in N(a)} \lambda_{ai}(x_i)\right)$$

- Identify $\lambda_{ai}(x_i) = \log(m_{i\to a}(x_i)) = \log \prod_{b \in N(i) \neq a} m_{b \to i}(x_i)$
- to obtain BP equations:



$$b_i(x_i) \propto f_i(x_i) \prod_{a \in N(i)} m_{a \to i}(x_i)$$

"beliefs" "messages"



$$b_a(X_a) \propto f_a(X_a) \prod_{i \in N(a)} \prod_{c \in N(i) \backslash a} m_{c \to i}(x_i)$$

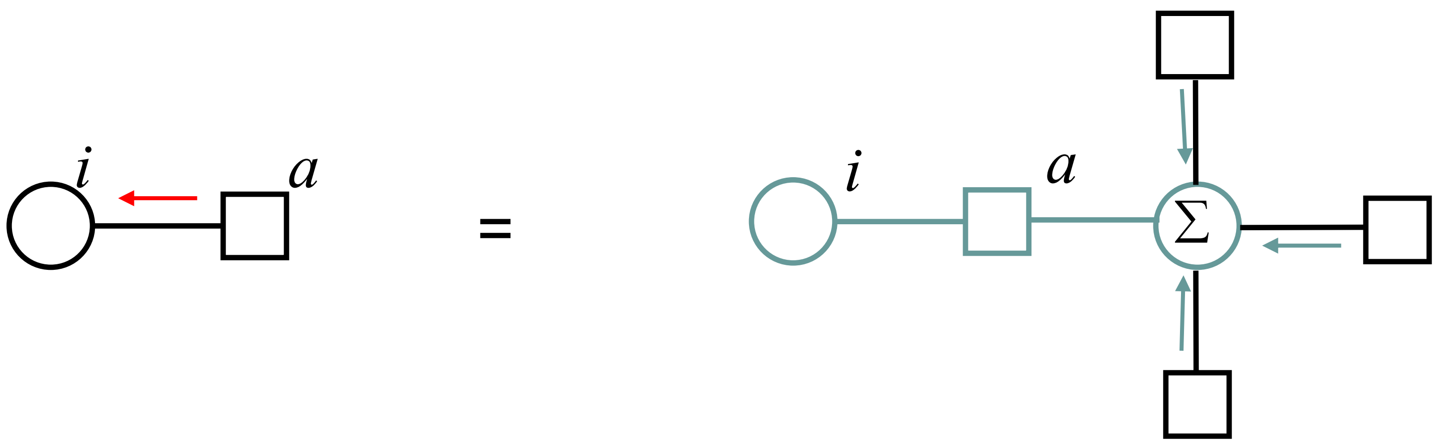
The "belief" is the BP approximation of the marginal probability.

BP Message-update Rules

Using $b_{a\to i}(x_i) = \sum_{X_a \setminus x_i} b_a(X_a)$, we get

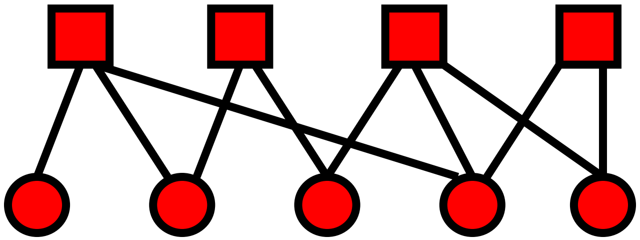
$$\boxed{m_{a\to i}(x_i) = \sum_{X_a \setminus x_i} f_a(X_a) \prod_{j\in N(a)\setminus i} \prod_{b\in N(j)\setminus a} m_{b\to j}(x_j)}$$

(A sum product algorithm)



Summary so far

- $P(X) = 1/Z \prod_{f_a \in F} f_a(X_a)$

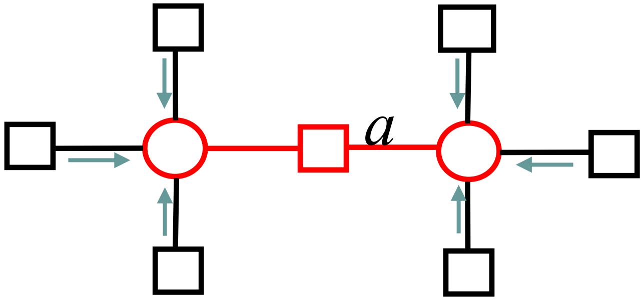


$$F(P,Q) = -H_Q(X) - \sum_{f_a \in F} E_Q \log f_a(X_a)$$

$$\hat{F}(P,Q) = \sum_a \sum_{\mathbf{x}_a} b_a(\mathbf{x}_a) \log \frac{f_a(\mathbf{x}_a)}{b_a(\mathbf{x}_a)} + \sum_i (1-d_i) \sum_{\mathbf{x}_i} b_i(\mathbf{x}_i) \log b_i(\mathbf{x}_i)$$

$$b_a(X_a) \propto \exp\left(-\log f_a(X_a) + \sum_{i \in N(a)} \lambda_{ai}(x_i) \right)$$

$$b_i(x_i) \propto \exp\left(\frac{1}{d_i - 1} \sum_{a \in N(i)} \lambda_{ai}(x_i) \right)$$



The Theory Behind LBP

$$\boxed{\begin{array}{c} F(Q,P) \\ q \in Q \end{array}}$$

- For a distribution $p(X|\theta)$ associated with a complex graph, computing the marginal (or conditional) probability of arbitrary random variable(s) is intractable

- Variational methods
  - formulating probabilistic inference as an optimization problem:

$$q^* = \arg\min_{q \in S} \left\{ F_{Betha}(p,q) \right\}$$

$$F_{Bethe} = \sum_a \sum_{\mathbf{x}_a} b_a(\mathbf{x}_a) \ln \frac{b_a(\mathbf{x}_a)}{f_a(\mathbf{x}_a)} + \sum_i (1-d_i) \sum_{\mathbf{x}_i} b_i(\mathbf{x}_i) \ln b_i(\mathbf{x}_i) = -\langle f_a(\mathbf{x}_a) \rangle - H_{bethe}$$

Optimizing the marginal in the Bethe energy is **a** way to make q tractable !

$Ax \geq b \quad x_i \in 0/1 \qquad X = (x_1, \dots, x_N)$

The Theory Behind LBP

- But we do not optimize $q(\mathbf{X})$ explicitly, focus on the set of beliefs
  - e.g., $b = \{ b_{i,j} = \tau(x_i, x_j),\ b_i = \tau(x_i) \}$

$P = \begin{bmatrix} P_1 \\ \vdots \\ P_N \end{bmatrix}$

- Relax the optimization problem
  - approximate objective: $H_q \approx F(b)$
  - relaxed feasible set: $\mathcal{M} \to \mathcal{M}_o \quad (\mathcal{M}_o \supseteq \mathcal{M})$

$$b^* = \arg\min_{b \in \mathcal{M}_o} \left\{ \langle E \rangle_b + F(b) \right\}$$

- The loopy BP algorithm:
  - a fixed point iteration procedure that tries to solve b^*

$Ax = \begin{bmatrix} a_1 \\ \vdots \\ a_N \end{bmatrix} x$

$a_2 x \geq 0 \qquad a_1 x \geq 0$

The Theory Behind LBP

- But we do not optimize $q(\mathbf{X})$ explicitly, focus on the set of beliefs
  - $e.g., \quad b = \{b_{i,j} = \tau(x_i, x_j), \;\; b_i = \tau(x_i)\}$
- Relax the optimization problem
  - approximate objective: $H_{Betha} = H(b_{i,j}, b_i)$
  - relaxed feasible set: $\mathcal{M}_o = \{ \; \tau \geq 0 \mid \sum_{x_i} \tau(x_i) = 1, \sum_{x_i} \tau(x_i, x_j) = \tau(x_j) \; \}$

$$b^* = \arg\min_{b \in \mathcal{M}_o} \; \{ \; \langle E \rangle_b + F(b) \; \}$$

- The loopy BP algorithm:
  - a fixed point iteration procedure that tries to solve b^*