

4. Sum-product algorithm

- Elimination algorithm
- Sum-product algorithm on a line
- Sum-product algorithm on a tree

Inference tasks on graphical models

consider an undirected graphical model (a.k.a. Markov random field)

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

where \mathcal{C} is the set of all maximal cliques in G

we want to

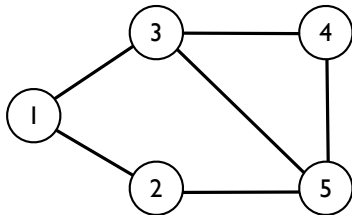
- calculate marginals: $\mu(x_A) = \sum_{x_{V \setminus A}} \mu(x)$
- calculating conditional distributions

$$\mu(x_A | x_B) = \frac{\mu(x_A, x_B)}{\mu(x_B)}$$

- calculation maximum a posteriori estimates: $\arg \max_{\hat{x}} \mu(\hat{x})$
- calculating the partition function Z
- sample from this distribution

Elimination algorithm for calculating marginals

Elimination algorithm is exact but can require $O(|\mathcal{X}|^{|V|})$ operations



$$\mu(x) \propto \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) \psi_{25}(x_2, x_5) \psi_{345}(x_3, x_4, x_5)$$

- we want to compute $\mu(x_1)$
- brute force marginalization:

$$\mu(x_1) \propto \sum_{x_2, x_3, x_4, x_5 \in \mathcal{X}} \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) \psi_{25}(x_2, x_5) \psi_{345}(x_3, x_4, x_5)$$

- requires $O(|\mathcal{C}| \cdot |\mathcal{X}|^5)$ operations, where big-O denotes that it is upper bounded by $c |\mathcal{C}| |\mathcal{X}|^5$ for some constant c and $|\mathcal{C}|$ is # of cliques

- consider an elimination ordering (5, 4, 3, 2)

$$\begin{aligned}
\mu(x_1) &\propto \sum_{x_2, x_3, x_4, x_5 \in \mathcal{X}} \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) \psi_{25}(x_2, x_5) \psi_{345}(x_3, x_4, x_5) \\
&= \sum_{x_2, x_3, x_4 \in \mathcal{X}} \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) \underbrace{\sum_{x_5 \in \mathcal{X}} \psi_{25}(x_2, x_5) \psi_{345}(x_3, x_4, x_5)}_{\equiv m_5(x_2, x_3, x_4)} \\
&= \sum_{x_2, x_3, x_4 \in \mathcal{X}} \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) m_5(x_2, x_3, x_4) \\
&= \sum_{x_2, x_3 \in \mathcal{X}} \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) \underbrace{\sum_{x_4 \in \mathcal{X}} m_5(x_2, x_3, x_4)}_{\equiv m_4(x_2, x_3)} \\
&= \sum_{x_2, x_3 \in \mathcal{X}} \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) m_4(x_2, x_3) \\
&= \sum_{x_2 \in \mathcal{X}} \psi_{12}(x_1, x_2) \underbrace{\sum_{x_3 \in \mathcal{X}} \psi_{13}(x_1, x_3) m_4(x_2, x_3)}_{\equiv m_3(x_1, x_2)}
\end{aligned}$$

$$\begin{aligned}
\mu(x_1) &\propto \sum_{x_2 \in \mathcal{X}} \psi_{12}(x_1, x_2) \underbrace{\sum_{x_3 \in \mathcal{X}} \psi_{13}(x_1, x_3) m_4(x_2, x_3)}_{\equiv m_3(x_1, x_2)} \\
&= \sum_{x_2 \in \mathcal{X}} \psi_{12}(x_1, x_2) m_3(x_1, x_2) \\
&\equiv m_2(x_1)
\end{aligned}$$

- normalize $m_2(\cdot)$ to get $\mu(x_1)$

$$\mu(x_1) = \frac{m_2(x_1)}{\sum_{\hat{x}_1} m_2(\hat{x}_1)}$$

- computational complexity depends on the elimination ordering
- how do we know which ordering is better?

Computational complexity of elimination algorithm

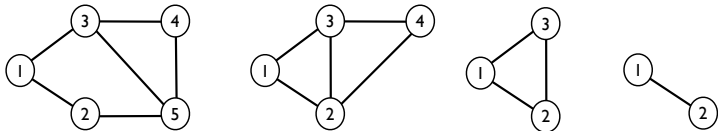
$$\begin{aligned}
 \mu(x_1) &\propto \sum_{x_2, x_3, x_4, x_5 \in \mathcal{X}} \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) \psi_{25}(x_2, x_5) \psi_{345}(x_3, x_4, x_5) \\
 &= \sum_{x_2, x_3, x_4 \in \mathcal{X}} \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) \underbrace{\sum_{x_5 \in \mathcal{X}} \psi_{25}(x_2, x_5) \psi_{345}(x_3, x_4, x_5)}_{\equiv m_5(S_5), S_5 = \{x_2, x_3, x_4\}, \Psi_5 = \{\psi_{25}, \psi_{345}\}} \\
 &= \sum_{x_2, x_3 \in \mathcal{X}} \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) \underbrace{\sum_{x_4 \in \mathcal{X}} m_5(x_2, x_3, x_4)}_{\equiv m_4(S_4), S_4 = \{x_2, x_3\}, \Psi_4 = \{m_5\}} \\
 &= \sum_{x_2 \in \mathcal{X}} \psi_{12}(x_1, x_2) \underbrace{\sum_{x_3 \in \mathcal{X}} \psi_{13}(x_1, x_3) m_4(x_2, x_3)}_{\equiv m_3(S_3), S_3 = \{x_1, x_2\}, \Psi_3 = \{\psi_{13}, m_4\}} \\
 &= \underbrace{\sum_{x_2 \in \mathcal{X}} \psi_{12}(x_1, x_2) m_3(x_1, x_2)}_{\equiv m_2(S_2), S_2 = \{x_1\}, \Psi_2 = \{\psi_{12}, m_3\}} = m_2(x_1)
 \end{aligned}$$

Total complexity: $\sum_i O(|\Psi_i| \cdot |\mathcal{X}|^{1+|S_i|}) = O(|V| \cdot \max_i |\Psi_i| \cdot |\mathcal{X}|^{1+\max_i |S_i|})$

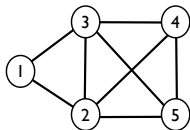
Sum-product algorithm

Induced graph

elimination algorithm as transformation of graphs



- **induced graph** $\mathcal{G}(G, I)$ for a graph G and an elimination ordering I
 - ▶ is the union of (the edges of) all the transformed graphs
 - ▶ or equivalently, start from G and for each $i \in I$ connect all pairs in S_i



- **theorem:** every maximal clique in $\mathcal{G}(G, I)$ corresponds to a domain of a message $S_i \cup \{i\}$ for some i
- size of the largest clique in $\mathcal{G}(G, I)$ is $1 + \max_i |S_i|$
- different orderings I 's give different cliques, resulting in varying

- **theorem:** finding optimal elimination ordering is NP-hard
- any suggestions?
- greedy heuristic gives $I = (4, 5, 3, 2, 1)$
- for Bayesian networks
 - ▶ the same algorithm works with conditional probabilities instead of compatibility functions
 - ▶ complexity analysis can be done on moralized undirected graph
 - ▶ intermediate messages do not correspond to a conditional distribution

Elimination algorithm

- **input:** $\{\psi_c\}_{c \in \mathcal{C}}$, alphabet \mathcal{X} , subset $A \subseteq V$, elimination ordering I
 - **output:** marginal $\mu(x_A)$
1. initialize active set Ψ to be the set of input compatibility functions
 2. **for** node i in I that is not in A **do**
 - let S_i be the set of nodes, not including i , that share a compatibility function with i
 - let Ψ_i be the set of compatibility functions in Ψ involving x_i
 - compute $m_i(x_{S_i}) = \sum_{x_i} \prod_{\psi \in \Psi_i} \psi(x_i, x_{S_i})$
 - remove elements of Ψ_i from Ψ
 - add m_i to Ψ**end**
 3. normalize $\mu(x_A) = \prod_{\psi \in \Psi} \psi(x_A) / \sum_{x_A} \prod_{\psi \in \Psi} \psi(x_A)$

Role of the messages $m_i(x_{S_i})$

- messages capture local information about what one side of the edge needs to know about the other side of the edge
- this intuition is exactly correct when the graph is a tree
- we provide a heuristic (called sum-product algorithm or belief propagation) that is accurate when the graph is a tree
- but also provides surprisingly good approximate solutions, even when the underlying graph has loops

$$m_i(x_i) = \sum_{x_a \in \mathcal{X}} \left\{ \psi_{i,a}(x_i, x_a) \sum_{x_A} \psi_A(x_a, x_A) \right\}$$

Role of the messages

$$\begin{aligned} m_i(x_i) &= \sum_{x_a, x_b \in \mathcal{X}, x_A, x_B} \psi_{i,a}(x_i, x_a) \psi_A(x_a, x_A) \psi_{i,b}(x_i, x_b) \psi_B(x_b, x_B) \\ &= \left\{ \sum_{x_a \in \mathcal{X}} \{ \psi_{i,a}(x_i, x_a) \sum_{x_A} \psi_A(x_a, x_A) \} \right\} \left\{ \sum_{x_b \in \mathcal{X}} \{ \psi_{i,b}(x_i, x_b) \sum_{x_B} \psi_B(x_b, x_B) \} \right\} \end{aligned}$$

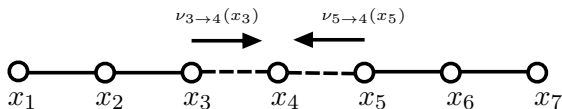
Belief Propagation for approximate inference

- given a pairwise MRF: $\mu(x) = \frac{1}{Z} \prod_{(i,j) \in E} \psi_{i,j}(x_i, x_j)$
- compute marginal: $\mu(x_i)$ for all $i \in V$
- message update: set of $2|E|$ messages on each (directed) edge $\{\nu_{i \rightarrow j}(x_i)\}_{(i,j) \in E}$, where $\nu_{i \rightarrow j} : \mathcal{X} \rightarrow \mathbb{R}^+$ encoding our belief (or approximate $\mathbb{P}(x_i)$)
 - ▶ $\nu_{i \rightarrow j}^{(t+1)}(x_i) = \prod_{k \in \partial i \setminus j} \left\{ \sum_{x_k} \nu_{k \rightarrow i}^{(t)}(x_k) \psi_{i,k}(x_i, x_k) \right\}$
 - ▶ $O(d_i |\mathcal{X}|^2)$ computations
- decision:

$$\nu_i^{(t)}(x_i) = \prod_{k \in \partial i} \left\{ \sum_{x_k} \nu_{k \rightarrow i}^{(t)}(x_k) \psi_{i,k}(x_i, x_k) \right\}$$

$$\text{▶ } \hat{\mu}(x_i) = \frac{\nu_i(x_i)}{\sum_{x'} \nu_i(x')}$$

Sum-product algorithm on a line



Remove node i and recursively compute marginals on sub-graphs

$$\mu(x) = \frac{1}{Z} \prod_{i=1}^{n-1} \psi_{i,i+1}(x_i, x_{i+1})$$

$$\mu(x_i) = \sum_{x_{[n] \setminus \{i\}}} \mu(x)$$

$$\propto \sum_{x_{[n] \setminus \{i\}}} \underbrace{\psi(x_1, x_2) \cdots \psi(x_{i-2}, x_{i-1})}_{\mu_{(i-1) \rightarrow i}: \text{joint dist. on a sub-graph}} \psi(x_{i-1}, x_i) \psi(x_i, x_{i+1}) \underbrace{\psi(x_{i+1}, x_{i+2}) \cdots \psi(x_{n-1}, x_n)}_{\mu_{(i+1) \rightarrow i}}$$

$$\propto \sum_{x_{i-1}, x_{i+1}} \underbrace{\nu_{(i-1) \rightarrow i}(x_{i-1})}_{\text{marginal dist. on a subgraph}} \psi(x_{i-1}, x_i) \psi(x_i, x_{i+1}) \nu_{(i+1) \rightarrow i}(x_{i+1})$$

$$\nu_{(i-1) \rightarrow i}(x_{i-1}) \equiv \sum_{x_1, \dots, x_{i-2}} \mu_{(i-1) \rightarrow i}(x_1, \dots, x_{i-1})$$

$$\nu_{(i+1) \rightarrow i}(x_{i+1}) \equiv \sum_{x_{i+2}, \dots, x_n} \mu_{(i+1) \rightarrow i}(x_{i+1}, \dots, x_n)$$

$$\nu_i(x_i) \equiv \sum_{x_{i-1}, x_{i+1}} \nu_{(i-1) \rightarrow i}(x_{i-1}) \psi(x_{i-1}, x_i) \psi(x_i, x_{i+1}) \nu_{(i+1) \rightarrow i}(x_{i+1})$$

$$\mu(x_i) = \frac{\nu_i(x_i)}{\sum_{x_i} \nu_i(x_i)}$$

definitions: of joint distribution and marginal on sub-graphs

$$\mu_{(i-1) \rightarrow i}(x_1, \dots, x_{i-1}) \equiv \frac{1}{Z_{(i-1) \rightarrow i}} \prod_{k \in [i-2]} \psi(x_k, x_{k+1})$$

$$\nu_{(i-1) \rightarrow i}(x_{i-1}) \equiv \sum_{x_1, \dots, x_{i-2}} \mu_{(i-1) \rightarrow i}(x_1, \dots, x_{i-1})$$

$$\mu_{(i+1) \rightarrow i}(x_{i+1}, \dots, x_n) \equiv \frac{1}{Z_{(i+1) \rightarrow i}} \prod_{k \in \{i+2, \dots, n\}} \psi(x_{k-1}, x_k)$$

$$\nu_{(i+1) \rightarrow i}(x_{i+1}) \equiv \sum_{x_{i+2}, \dots, x_n} \mu_{(i+1) \rightarrow i}(x_{i+1}, \dots, x_n)$$

how can we compute the messages ν , recursively?

$$\begin{aligned}\mu_{i \rightarrow i+1}(x_1, \dots, x_i) &\propto \mu_{i-1 \rightarrow i}(x_1, \dots, x_{i-1}) \psi(x_{i-1}, x_i) \\ \nu_{i \rightarrow i+1}(x_i) &= \sum_{x_1, \dots, x_{i-1}} \mu_{i \rightarrow i+1}(x_1, \dots, x_i) \\ &\propto \sum_{x_1, \dots, x_{i-1}} \mu_{i-1 \rightarrow i}(x_1, \dots, x_{i-1}) \psi(x_{i-1}, x_i) \\ &= \sum_{x_{i-1}} \nu_{i-1 \rightarrow i}(x_{i-1}) \psi(x_{i-1}, x_i) \\ \nu_{1 \rightarrow 2}(x_1) &= 1/|\mathcal{X}| \\ \nu_{2 \rightarrow 3}(x_2) &\propto \sum_{x_1} \frac{1}{|\mathcal{X}|} \psi(x_1, x_2)\end{aligned}$$

how many operations are required?

- $O(n |\mathcal{X}|^2)$ operations to compute one marginal $\mu(x_i)$

what if we want all the marginals?

- compute all the messages forward and backward $O(n |\mathcal{X}|^2)$
- then compute all the marginals in $O(n |\mathcal{X}|^2)$ operations

Computing partition function is as easy as computing marginals

recall

$$\begin{aligned}\mu_{(i-1) \rightarrow i}(x_1, \dots, x_{i-1}) &\equiv \frac{1}{Z_{(i-1) \rightarrow i}} \prod_{k \in [i-2]} \psi(x_k, x_{k+1}) \\ \nu_{(i-1) \rightarrow i}(x_{i-1}) &\equiv \sum_{x_1, \dots, x_{i-2}} \mu_{(i-1) \rightarrow i}(x_1, \dots, x_{i-1})\end{aligned}$$

computing the partition function from the messages

$$\begin{aligned}Z_{i \rightarrow (i+1)} &= \sum_{x_1, \dots, x_i} \prod_{k \in [i-1]} \psi(x_k, x_{k+1}) \\ &= \sum_{x_i, x_{i-1}} Z_{(i-1) \rightarrow i} \nu_{(i-1) \rightarrow i}(x_{i-1}) \psi(x_{i-1}, x_i) \\ Z_1 &= 1 \\ Z_n &= Z\end{aligned}$$

how many operations do we need?

• $O(n |\mathcal{X}|^2)$ operations
Sum-product algorithm

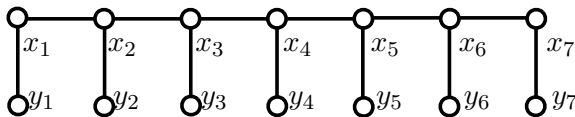
Sum-product algorithm for hidden Markov models

- hidden Markov model

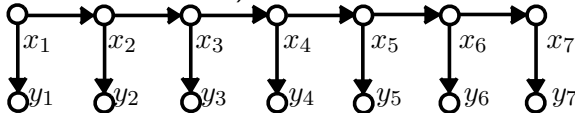
Sequence of r.v.'s $\{(X_1, Y_1); (X_2, Y_2); \dots; (X_n, Y_n)\}$

hidden state $\{X_i\}$ Markov Chain $\mathbb{P}\{x\} = \mathbb{P}\{x_1\} \prod_{i=1}^{n-1} \mathbb{P}\{x_{i+1}|x_i\}$

$\{Y_i\}$ noisy observations $\mathbb{P}\{y|x\} = \prod_{i=1}^n \mathbb{P}\{y_i|x_i\}$



(equivalent directed form)



- time homogeneous hidden Markov models

Sequence of r.v.'s $\{(X_1, Y_1); (X_2, Y_2); \dots; (X_n, Y_n)\}$

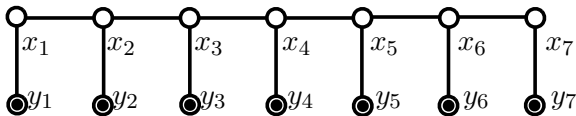
$\{X_i\}$ Markov Chain $\mathbb{P}\{x\} = q_0(x_1) \prod_{i=1}^{n-1} q(x_i, x_{i+1})$

$\{Y_i\}$ noisy observations $\mathbb{P}\{y|x\} = \prod_{i=1}^n r(x_i, y_i)$

$$\mu(x, y) = \frac{1}{Z} \prod_{i=1}^{n-1} \psi_i(x_i, x_{i+1}) \prod_{i=1}^n \tilde{\psi}_i(x_i, y_i),$$

$$\psi_i(x_i, x_{i+1}) = q(x_i, x_{i+1}), \quad \tilde{\psi}_i(x_i, y_i) = r(x_i, y_i).$$

- we want to compute marginals of the following graphical model on a line (q_0 uniform)



$$\mu_y(x) = \mathbb{P}\{x|y\} \stackrel{\text{Bayes thm}}{=} \frac{1}{Z(y)} \prod_{i=1}^{n-1} q(x_i, x_{i+1}) \prod_{i=1}^n r(x_i, y_i).$$

$$\mu_y(x) = \frac{1}{Z(y)} \prod_{i=1}^{n-1} q(x_i, x_{i+1}) \prod_{i=1}^n r(x_i, y_i)$$

$$= \frac{1}{Z(y)} \prod_{i=1}^{n-1} \psi_i(x_i, x_{i+1})$$

$$\psi_i(x_i, x_{i+1}) = q(x_i, x_{i+1}) r(x_i, y_i) \quad (\text{for } i < n - 1)$$

$$\psi_{n-1}(x_{n-1}, x_n) = q(x_{n-1}, x_n) r(x_{n-1}, y_{n-1}) r(x_n, y_n).$$

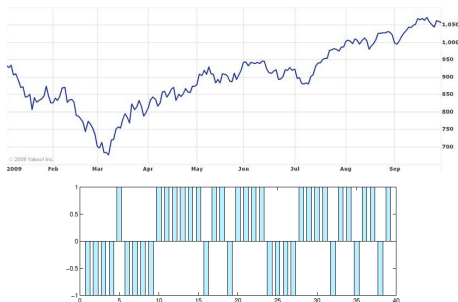
- apply sum-product algorithm to compute marginals in $O(n|\mathcal{X}|^2)$ time

$$\nu_{i \rightarrow (i+1)}(x_i) \propto \sum_{x_{i-1} \in \mathcal{X}} q(x_{i-1}, x_i) r(x_{i-1}, y_{i-1}) \nu_{(i-1) \rightarrow i}(x_{i-1}),$$

$$\nu_{(i+1) \rightarrow i}(x_i) \propto \sum_{x_{i+1} \in \mathcal{X}} q(x_i, x_{i+1}) r(x_i, y_i) \nu_{(i+2) \rightarrow (i+1)}(x_{i+1}).$$

- known as forward-backward algorithm
 - ▶ a special case of the sum-product algorithm
 - ▶ BCJR algorithm for convolutional codes ([Bahl, Cocke, Jelinek and Raviv 1974])
 - ▶ cannot find the maximum likelihood estimate (cf. Viterbi algorithm)
 - ▶ this requires max-product algorithm
- implement sum-product algorithm for HMM [Exercise 4.3]
- consider an extension of inference on HMM [Exercise 4.4]

Exercise 4.3



- S&P 500 index over a period of time
- For each week, measure the price movement relative to the previous week: +1 indicates up and -1 indicates down
- a hidden Markov model in which x_t denotes the economic state (good or bad) of week t and y_t denotes the price movement (up or down)
- $x_{t+1} = x_t$ with probability 0.8
- $\mathbb{P}_{Y_t|X_t}(y_t = +1|x_t = \text{'good'}) = \mathbb{P}_{Y_t|X_t}(y_t = -1|x_t = \text{'bad'}) = q$

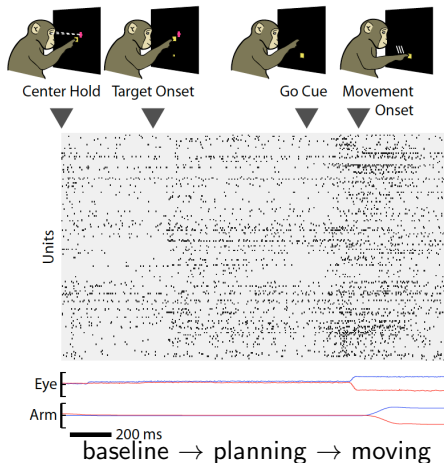
Example: Neuron firing patterns

Hypothesis

Assemblies of neurones activate in a coordinate way in correspondence to specific cognitive functions. Performing of the function corresponds sequence of these activity states.

Approach

Firing process	\leftrightarrow	Observed variables
Activity states	\leftrightarrow	Hidden variables



single subject
ensemble neural spikes

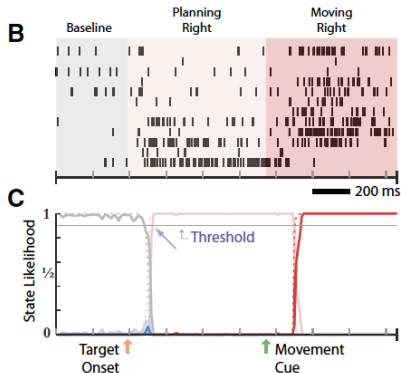
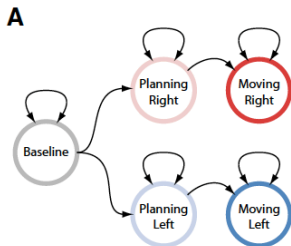
horizontal eye position
vertical eye position

baseline → planning → moving

- ▶ automatically detect (as opposed to manually specify) baseline, plan, and perimovement epochs of neural activity
- ▶ detect target movement in advance
- ▶ goal: neural prosthesis to help patients with spinal cord injury or neurodegenerative disease and significantly impaired motor control

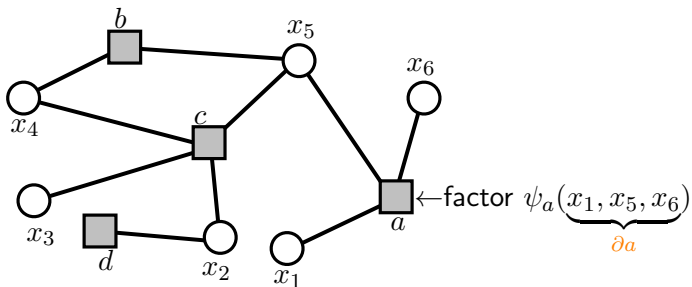
[C. Kemere, G. Santhanam, B. M. Yu, A. Afshar, S.I. Ryu, T. H. Meng and

Sum-product algorithm for neural decoding, *Neural Systems* 100:2441-2452 (2008)]



- ▶ discrete time 10ms
- ▶ $\mathbb{P}(x_{t+1}|x_t) = A_{ij}$,
 $\mathbb{P}(\# \text{ spikes for measurement } k = d|x_t = i) \propto e^{-\lambda_{k,i}} \lambda_{k,i}^d$
- ▶ likelihood: $\frac{\mathbb{P}(x_t=s)}{\sum_{s'} \mathbb{P}(x_t=s')}$

Belief propagation for factor graphs



$$\mu(x) = \frac{1}{Z} \prod_{a \in F} \psi_a(x_{\partial a})$$

- variable nodes i, j , etc.; factor nodes a, b , etc.
- set of messages $\{\nu_{i \rightarrow a}\}_{(i,a) \in E}$ and $\{\tilde{\nu}_{a \rightarrow i}\}_{(a,i) \in E}$
- messages from variables to factors

$$\nu_{i \rightarrow a}(x_i) = \prod_{b \in \partial i \setminus \{a\}} \tilde{\nu}_{b \rightarrow i}(x_i)$$

- messages from factors to variables

$$\tilde{\nu}_{a \rightarrow i}(x_i) = \sum_{x_{\partial a \setminus \{i\}}} \psi_a(x_{\partial a}) \prod_{j \in \partial a \setminus \{i\}} \nu_{j \rightarrow a}(x_j)$$

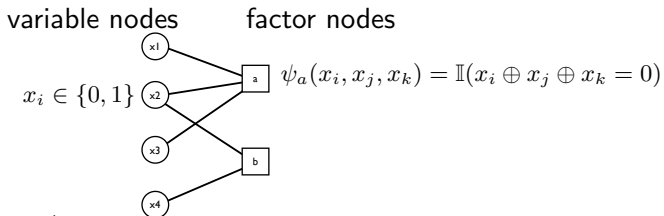
- messages from variables to factors

$$\nu_i(x_i) = \prod_{b \in \partial i} \tilde{\nu}_{b \rightarrow i}(x_i)$$

- this includes belief propagation (=sum-product algorithm) on (general) Markov random fields
- exact on factor trees

Example: decoding LDPC codes

- LDPC code is defined by a factor graph model



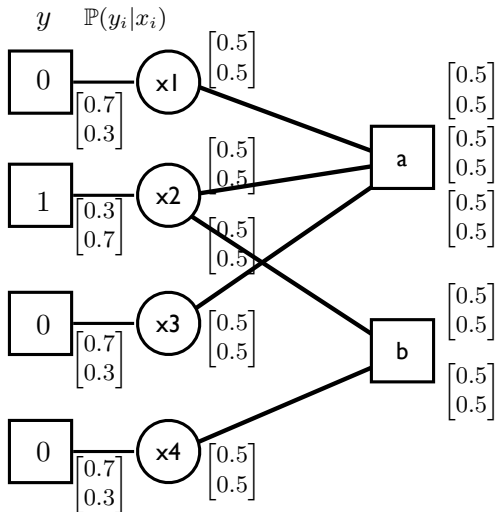
- ▶ block length $n = 4$
 - ▶ number of factors $m = 2$
 - ▶ allowed messages = $\{0000, 0111, 1010, 1101\}$
- decoding using belief propagation (for BSC with $\epsilon = 0.3$)

$$\mu_y(x) = \frac{1}{Z} \prod_{i \in V} \mathbb{P}_{Y|X}(y_i|x_i) \prod_{a \in F} \mathbb{I}(\oplus x_{\partial a} = 0)$$

- use (parallel) sum-product algorithm to find $\mu(x_i)$ and let

$$\hat{x}_i = \arg \max \mu(x_i)$$

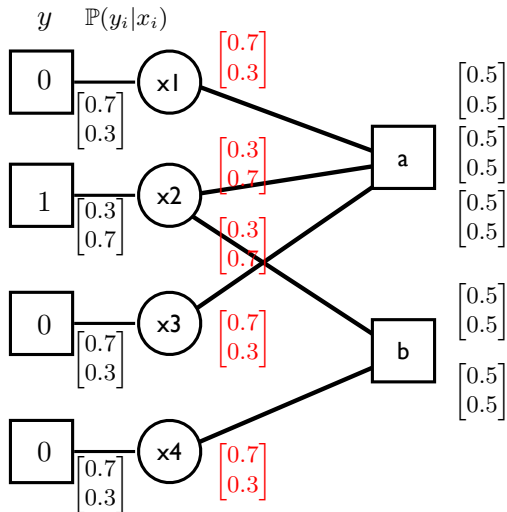
Decoding by sum-product algorithm



$$\nu_{i \rightarrow a}^{(t+1)}(x_i) = \mathbb{P}(y_i|x_i) \prod_{b \in \partial i \setminus \{a\}} \tilde{\nu}_{b \rightarrow i}^{(t)}(x_i)$$

$$\tilde{\nu}_{a \rightarrow i}^{(t+1)}(x_i) = \sum_{x_{\partial a \setminus \{i\}}} \prod_{j \in \partial a \setminus \{i\}} \nu_{j \rightarrow a}(x_j) \mathbb{I}(\oplus x_{\partial a} = 0)$$

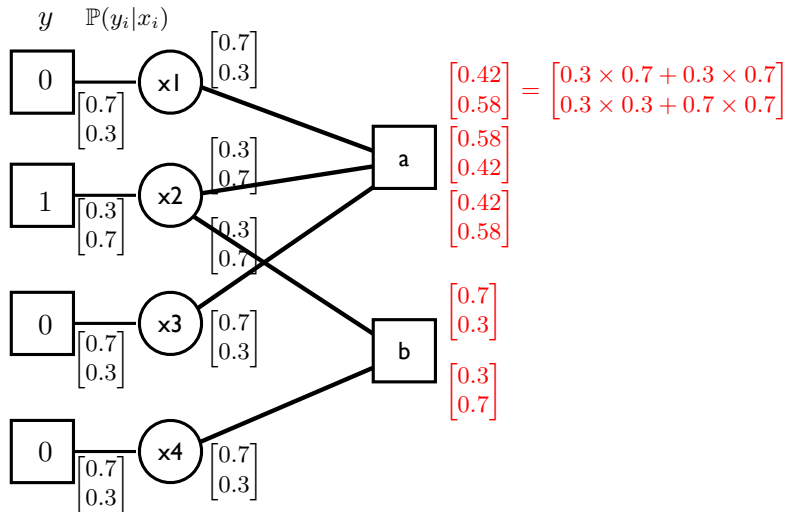
Decoding by sum-product algorithm



$$\nu_{i \rightarrow a}^{(t+1)}(x_i) = \mathbb{P}(y_i | x_i) \prod_{b \in \partial i \setminus \{a\}} \tilde{\nu}_{b \rightarrow i}^{(t)}(x_i)$$

$$\tilde{\nu}_{a \rightarrow i}^{(t+1)}(x_i) = \sum_{x_{\partial a \setminus \{i\}}} \prod_{j \in \partial a \setminus \{i\}} \nu_{j \rightarrow a}(x_j) \mathbb{I}(\oplus x_{\partial a} = 0)$$

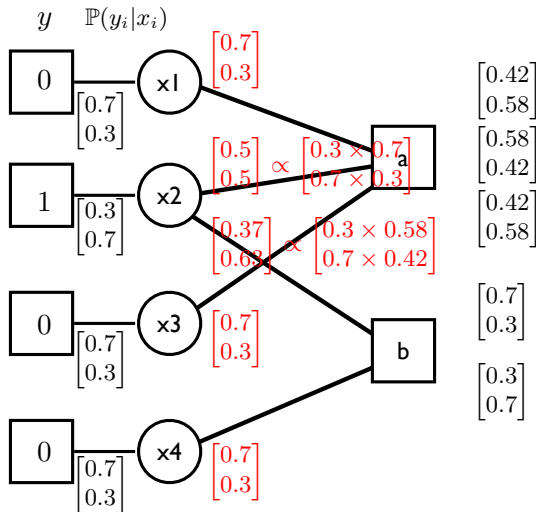
Decoding by sum-product algorithm



$$\nu_{i \rightarrow a}^{(t+1)}(x_i) = \mathbb{P}(y_i|x_i) \prod_{b \in \partial i \setminus \{a\}} \tilde{\nu}_{b \rightarrow i}^{(t)}(x_i)$$

$$\tilde{\nu}_{a \rightarrow i}^{(t+1)}(x_i) = \sum_{x_{\partial a \setminus \{i\}}} \prod_{j \in \partial a \setminus \{i\}} \nu_{j \rightarrow a}(x_j) \mathbb{I}(\oplus x_{\partial a} = 0)$$

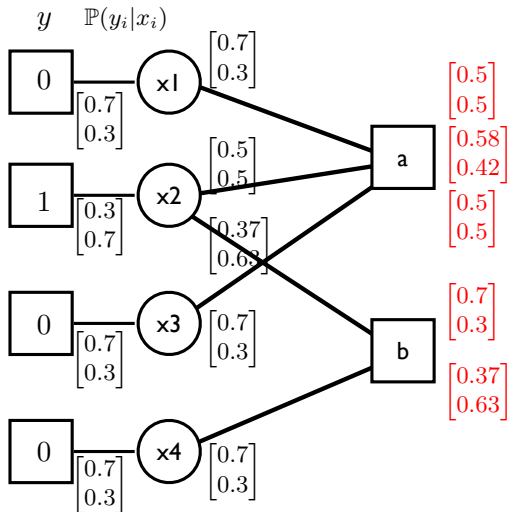
Decoding by sum-product algorithm



$$\nu_{i \rightarrow a}^{(t+1)}(x_i) = \mathbb{P}(y_i|x_i) \prod_{b \in \partial i \setminus \{a\}} \tilde{\nu}_{b \rightarrow i}^{(t)}(x_i)$$

$$\tilde{\nu}_{a \rightarrow i}^{(t+1)}(x_i) = \sum_{x_{\partial a \setminus \{i\}}} \prod_{j \in \partial a \setminus \{i\}} \nu_{j \rightarrow a}(x_j) \mathbb{I}(\oplus x_{\partial a} = 0)$$

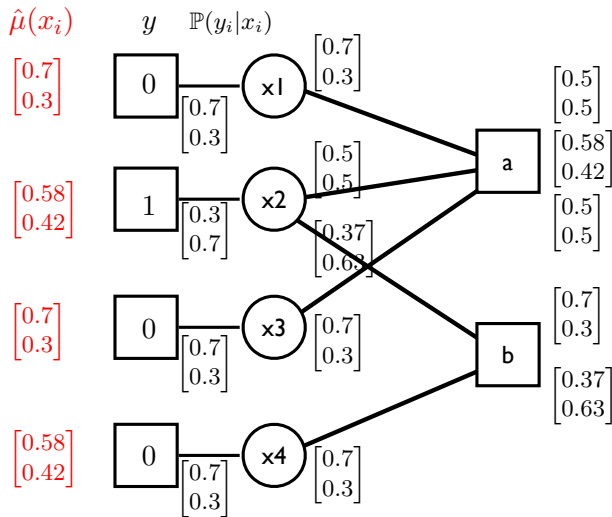
Decoding by sum-product algorithm



$$\nu_{i \rightarrow a}^{(t+1)}(x_i) = \mathbb{P}(y_i|x_i) \prod_{b \in \partial i \setminus \{a\}} \tilde{\nu}_{b \rightarrow i}^{(t)}(x_i)$$

$$\tilde{\nu}_{a \rightarrow i}^{(t+1)}(x_i) = \sum_{x_{\partial a \setminus \{i\}}} \prod_{j \in \partial a \setminus \{i\}} \nu_{j \rightarrow a}(x_j) \mathbb{I}(\oplus x_{\partial a} = 0)$$

Decoding by sum-product algorithm



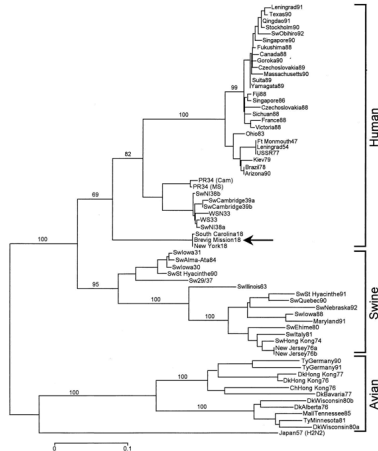
$$\nu_{i \rightarrow a}^{(t+1)}(x_i) = \mathbb{P}(y_i|x_i) \prod_{b \in \partial i \setminus \{a\}} \tilde{\nu}_{b \rightarrow i}^{(t)}(x_i)$$

$$\tilde{\nu}_{a \rightarrow i}^{(t+1)}(x_i) = \sum_{x_{\partial a \setminus \{i\}}} \prod_{j \in \partial a \setminus \{i\}} \nu_{j \rightarrow a}(x_j) \mathbb{I}(\oplus x_{\partial a} = 0)$$

Sum-product algorithm on trees

a motivating example: influenza virus

complete sequence of the gene of the 1918 influenza virus

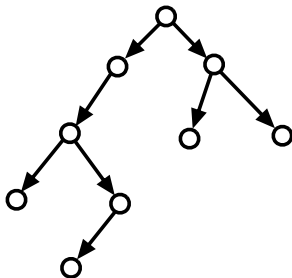


[A.H. Reid, T.G. Fanning, J.V. Hultin, and J.K. Taubenberger,

Proc. Natl. Acad. Sci. 96 (1999) 1651-1656]

Sum-product algorithm

- challenges in phylogeny
 - ▶ **phylogeny reconstruction:** given DNA sequences at vertices (only at leaves), infer the underlying tree $T = (V, E)$.
 - ▶ **phylogeny evaluation:** given a tree $T = (V, E)$ evaluate the probability of observed DNA sequences at vertices (only at leaves).
- Bayesian network model for **phylogeny evaluation**



$T = (V, D)$ directed graph, DNA sequences $x = (x_i)_{i \in V} \in \mathcal{X}^V$

$$\mu_T(x) = q_o(x_o) \prod_{(i,j) \in D} q_{i,j}(x_i, x_j),$$

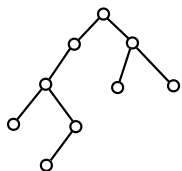
$q_{i,j}(x_i, x_j) =$ Probability that the descendent is x_j if ancestor is x_i .

- simplified model: $\mathcal{X} = \{+1, -1\}$

$$q_o(x_o) = \frac{1}{2}$$

$$q(x_i, x_j) = \begin{cases} 1 - q & \text{if } x_j = x_i \\ q & \text{if } x_i \neq x_j \end{cases}$$

MRF representation: $q(x_i, x_j) \propto e^{\theta x_i x_j}$ with $\theta = \frac{1}{2} \log \frac{q}{1 - q}$

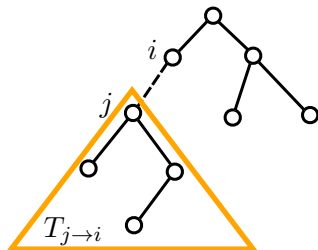


probability of certain tree of mutations x :

$$\mu_T(x) = \frac{1}{Z_\theta(T)} \prod_{(i,j) \in E} e^{\theta x_i x_j}$$

- **problem:** for given T , compute marginal $\mu_T(x_i)$
- we prove the correctness of sum-product algorithm for this model, but the same proof holds for any general pairwise MRF (and also for general MRF and FG)

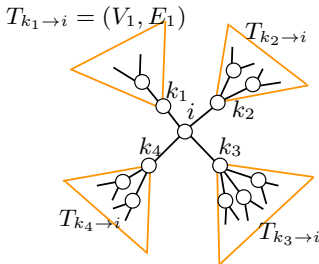
- define graphical model on sub-trees



$$T_{j \rightarrow i} = (V_{j \rightarrow i}, E_{j \rightarrow i}) \quad \equiv \quad \text{Subtree rooted at } j \text{ and excluding } i$$

$$\mu_{j \rightarrow i}(x_{V_{j \rightarrow i}}) \quad \equiv \quad \frac{1}{Z(T_{j \rightarrow i})} \prod_{(u,v) \in E_{j \rightarrow i}} e^{\theta x_u x_v}$$

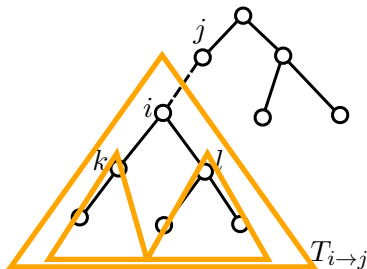
$$\nu_{j \rightarrow i}(x_j) \quad \equiv \quad \sum_{x_{V_{j \rightarrow i} \setminus \{j\}}} \mu_{j \rightarrow i}(x_{V_{j \rightarrow i}})$$



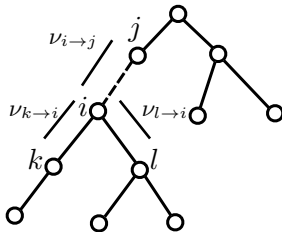
the messages from neighbors k_1, k_2, k_3, k_4 are sufficient to compute the marginal $\mu(x_i)$

$$\begin{aligned}
 \mu_T(x_i) &\propto \sum_{x_{V \setminus \{i\}}} \prod_{(u,v) \in E} e^{\theta x_u x_v} \\
 &= \sum_{x_{V_1}, x_{V_2}, x_{V_3}, x_{V_4}} \prod_{\ell=1}^4 \left\{ e^{\theta x_i x_{k_\ell}} \prod_{(u,v) \in E_\ell} e^{\theta x_u x_v} \right\} \\
 &= \prod_{\ell=1}^4 \sum_{x_{k_\ell}, x_{V_\ell \setminus \{k_\ell\}}} \left\{ e^{\theta x_i x_{k_\ell}} \prod_{(u,v) \in E_\ell} e^{\theta x_u x_v} \right\} \\
 &\propto \prod_{\ell=1}^4 \left\{ \sum_{x_{k_\ell}} e^{\theta x_i x_{k_\ell}} \underbrace{\sum_{x_{V_\ell \setminus \{k_\ell\}}} \mu_{k_\ell \rightarrow i}(x_{V_\ell})}_{\nu_{k_\ell \rightarrow i}(x_{k_\ell})} \right\}
 \end{aligned}$$

- recursion on sub-trees to compute the messages ν



$$\begin{aligned}
 \mu_{i \rightarrow j}(x_{V_{i \rightarrow j}}) &= \frac{1}{Z(T_{i \rightarrow j})} \prod_{(u,v) \in E_{i \rightarrow j}} e^{\theta x_u x_v} \\
 &= \frac{1}{Z(T_{i \rightarrow j})} e^{\theta x_i x_k} e^{\theta x_i x_l} \left\{ \prod_{(u,v) \in E_{k \rightarrow i}} e^{\theta x_u x_v} \right\} \left\{ \prod_{(u,v) \in E_{l \rightarrow i}} e^{\theta x_u x_v} \right\} \\
 &\propto e^{\theta x_u x_v} e^{\theta x_i x_l} \left\{ \prod_{(u,v) \in E_{k \rightarrow i}} e^{\theta x_u x_v} \right\} \left\{ \prod_{(u,v) \in E_{l \rightarrow i}} e^{\theta x_u x_v} \right\} \\
 &\propto e^{\theta x_i x_k} e^{\theta x_i x_l} \mu_{k \rightarrow i}(x_{V_{k \rightarrow i}}) \mu_{l \rightarrow i}(x_{V_{l \rightarrow i}})
 \end{aligned}$$



$$\begin{aligned}
\nu_{i \rightarrow j}(x_i) &= \sum_{x_{V_{i \rightarrow j} \setminus i}} \mu_{i \rightarrow j}(x_{V_{i \rightarrow j}}) \\
&\propto \sum_{x_{V_{i \rightarrow j} \setminus i}} e^{\theta x_i x_k} e^{\theta x_i x_l} \mu_{k \rightarrow i}(x_{V_{k \rightarrow i}}) \mu_{l \rightarrow i}(x_{V_{l \rightarrow i}}) \\
&\propto \left\{ \sum_{x_{V_{k \rightarrow i}}} e^{\theta x_i x_k} \mu_{k \rightarrow i}(x_{V_{k \rightarrow i}}) \right\} \left\{ \sum_{x_{V_{l \rightarrow i}}} e^{\theta x_i x_l} \mu_{l \rightarrow i}(x_{V_{l \rightarrow i}}) \right\} \\
&= \left\{ \sum_{x_k} e^{\theta x_i x_k} \sum_{x_{V_{k \rightarrow i} \setminus \{k\}}} \mu_{k \rightarrow i}(x_{V_{k \rightarrow i}}) \right\} \left\{ \sum_{x_l} e^{\theta x_i x_l} \sum_{x_{V_{l \rightarrow i} \setminus \{l\}}} \mu_{l \rightarrow i}(x_{V_{l \rightarrow i}}) \right\} \\
&\propto \left\{ \sum_{x_k} e^{\theta x_i x_k} \nu_{k \rightarrow i}(x_k) \right\} \left\{ \sum_{x_l} e^{\theta x_i x_l} \nu_{l \rightarrow i}(x_l) \right\}
\end{aligned}$$

with uniform initialization $\nu_{i \rightarrow j}(x_i) = \frac{1}{|\mathcal{X}|}$ for all leaves i

- sum-product algorithm (for our example)

$$\begin{aligned}\nu_{i \rightarrow j}(x_i) &\propto \prod_{k \in \partial i \setminus j} \left\{ \sum_{x_k} e^{\theta x_i x_k} \nu_{k \rightarrow i}(x_k) \right\} \\ \nu_i(x_i) &\equiv \prod_{k \in \partial i} \left\{ \sum_{x_k} e^{\theta x_i x_k} \nu_{k \rightarrow i}(x_k) \right\} \\ \mu_T(x_i) &= \frac{\nu_i(x_i)}{\sum_{x_i} \nu_i(x_i)}\end{aligned}$$

- what if we want all the marginals?
 - ▶ choose an arbitrary root ϕ
 - ▶ compute all the messages towards the root ($|E|$ messages)
 - ▶ then compute all the messages outwards from the root ($|E|$ messages)
 - ▶ then compute all the marginals (n marginals)
- how many operations are required?
 - ▶ naive implementation requires $O(|\mathcal{X}|^2 \sum_i d_i^2)$
 - ★ if i has degree d_i , then computing $\nu_{i \rightarrow j}$ requires $d_i |\mathcal{X}|^2$ operations
 - ★ d_i messages start at each node i , each require $d_i |\mathcal{X}|^2$ operations
 - ★ total computation for $2|E|$ messages is $\sum_i \left\{ d_i \cdot (d_i |\mathcal{X}|^2) \right\}$
 - ▶ however, we can compute all marginals in $O(n |\mathcal{X}|^2)$ operations

- let $D = \{(i, j), (j, i) \mid (i, j) \in E\}$ be the directed version of E (cf. $|D| = 2|E|$)
- **(sequential) sum-product algorithm**
 1. initialize $\nu_{i \rightarrow j}(x_i) = 1/|\mathcal{X}|$ for all leaves i
 2. recursively over $(i, j) \in D$ compute (from leaves)

$$\nu_{i \rightarrow j}(x_i) = \prod_{k \in \partial i \setminus j} \left\{ \sum_{x_k} \psi_{ik}(x_i, x_k) \nu_{k \rightarrow i}(x_k) \right\}$$

3. for each $i \in V$ compute marginal

$$\begin{aligned} \nu_i(x_i) &= \prod_{k \in \partial i} \left\{ \sum_{x_k} \psi_{ik}(x_i, x_k) \nu_{k \rightarrow i}(x_k) \right\} \\ \mu_T(x_i) &= \frac{\nu_i(x_i)}{\sum_{x_i} \nu_i(x_i)} \end{aligned}$$

- **(parallel) sum-product algorithm**

1. initialize $\nu_{i \rightarrow j}^{(0)}(x_i) = 1/|\mathcal{X}|$ for all $(i, j) \in D$
2. for $t \in \{0, 1, \dots, t_{\max}\}$
for all $(i, j) \in D$ compute

$$\nu_{i \rightarrow j}^{(t+1)}(x_i) = \prod_{k \in \partial i \setminus j} \left\{ \sum_{x_k} \psi_{ik}(x_i, x_k) \nu_{k \rightarrow i}^{(t)}(x_k) \right\}$$

3. for each $i \in V$ compute marginal

$$\begin{aligned} \nu_i(x_i) &= \prod_{k \in \partial i} \left\{ \sum_{x_k} \psi_{ik}(x_i, x_k) \nu_{k \rightarrow i}^{(t_{\max}+1)}(x_k) \right\} \\ \mu_T(x_i) &= \frac{\nu_i(x_i)}{\sum_{x_i} \nu_i(x_i)} \end{aligned}$$

- also called **belief propagation**

- when t_{\max} is larger than the diameter of the tree (the length of the longest path), this converges to the correct marginal [Homework 2.5]
- more operations than the sequential version ($O(n|\mathcal{X}|^2 \cdot \text{diam}(T))$)
 - ▶ a naive implementation requires $O(|\mathcal{X}|^2 \cdot \text{diam}(T) \cdot \sum d_i^2)$
- **naturally extends to general graphs** but no proof of exactness

Sum-product algorithm on general graphs

- **(loopy) belief propagation**

1. initialize $\nu_{i \rightarrow j}(x_i) = 1/|\mathcal{X}|$ for all $(i, j) \in D$
2. for $t \in \{0, 1, \dots, t_{\max}\}$
for all $(i, j) \in D$ compute

$$\nu_{i \rightarrow j}^{(t+1)}(x_i) = \prod_{k \in \partial i \setminus j} \left\{ \sum_{x_k} \psi_{ik}(x_i, x_k) \nu_{k \rightarrow i}^{(t)}(x_k) \right\}$$

3. for each $i \in V$ compute marginal

$$\begin{aligned} \nu_i(x_i) &= \prod_{k \in \partial i} \left\{ \sum_{x_k} \psi_{ik}(x_i, x_k) \nu_{k \rightarrow i}^{(t_{\max}+1)}(x_k) \right\} \\ \mu_T(x_i) &= \frac{\nu_i(x_i)}{\sum_{x_i} \nu_i(x_i)} \end{aligned}$$

- computes ‘approximate’ marginals in $O(n|\mathcal{X}|^2 \cdot t_{\max})$ operations
- generally it does not converge; even if it does, it might be incorrect
- folklore about loopy BP
 - ▶ works better when G has few short loops
 - ▶ works better when $\psi_{ij}(x_i, x_j) = \psi_{ij,1}(x_i)\psi_{ij,2}(x_j) + \text{small}(x_i, x_j)$
 - ▶ nonconvex variational principle

Exercise: partition function on trees

- using the recursion for messages:

$$\begin{aligned}\nu_{i \rightarrow j}(x_i) &= \prod_{k \in \partial i \setminus j} \left\{ \sum_{x_k} \psi_{ik}(x_i, x_k) \nu_{k \rightarrow i}(x_k) \right\} \\ \nu_i(x_i) &= \prod_{k \in \partial i} \left\{ \sum_{x_k} \psi_{ik}(x_i, x_k) \nu_{k \rightarrow i}(x_k) \right\}\end{aligned}$$

it follows that we can easily compute the partition function as

$$Z(T) = \sum_{x_i} \nu_i(x_i)$$

- alternatively, if we had a black box that computes marginals for any tree, then we can use it to compute partition functions efficiently

$$\begin{aligned}Z(T_{i \rightarrow j}) &= \sum_{x_i \in \mathcal{X}} \prod_{k \in \partial i \setminus j} \left\{ \sum_{x_k \in \mathcal{X}} \psi_{ik}(x_i, x_k) \cdot Z(T_{k \rightarrow i}) \cdot \mu_{k \rightarrow i}(x_k) \right\} \\ Z(T) &= \sum_{x_i \in \mathcal{X}} \prod_{k \in \partial i} \left\{ \sum_{x_k \in \mathcal{X}} \psi_{ik}(x_i, x_k) \cdot Z(T_{k \rightarrow i}) \cdot \mu_{k \rightarrow i}(x_k) \right\}\end{aligned}$$

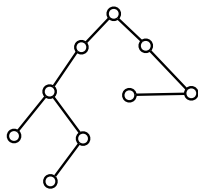
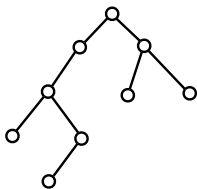
- this recursive algorithm naturally extends to general graphs
- Sum-product algorithm

Why would one want to compute the partition function?

Suppose you observe

$$x = (+1, +1, +1, +1, +1, +1, +1, +1, +1)$$

and you know this comes from either of



$$\mu(x) = \frac{1}{Z(T)} \prod_{(i,j) \in E} \psi(x_i, x_j)$$

(e.g. coloring)

which one has highest likelihood?

Exercise: sampling on the tree

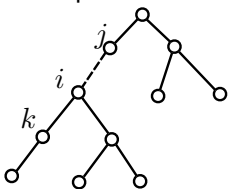
- if we have a black-box for computing marginals on any tree, we can use it to sample from any distribution on a tree

SAMPLING(Tree $T = (V, E)$, $\psi = \{\psi_{ij}\}_{(ij) \in E}$)

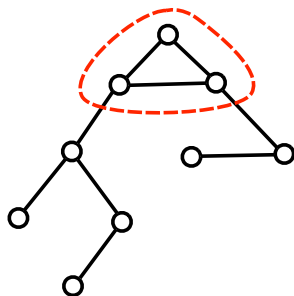
- 1: Choose a root $o \in V$;
 - 2: Sample $X_o \sim \mu_o(\cdot)$;
 - 2: Recursively over $i \in V$ (from root to leaves):
 - 3: Compute $\mu_{i|\pi(i)}(x_i | x_{\pi(i)})$;
 - 4: Sample $X_i \sim \mu_{i|\pi(i)}(\cdot | x_{\pi(i)})$;
-

$\pi(i)$ is the parent of node i in the rooted tree T_o

- we use the black-box to compute the conditional distribution

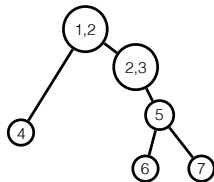
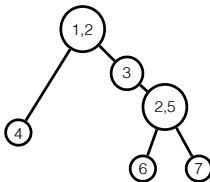
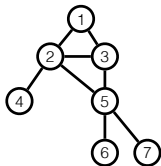

$$\mu_T(x_{V_{i \rightarrow j}} | x_j) \propto \psi_{ij}(x_i x_j) \mu_{i \rightarrow j}(x_{V_{i \rightarrow j}})$$
$$\mu_T(x_i | x_j) \propto \psi_{ij}(x_i x_j) \mu_{i \rightarrow j}(x_i)$$

Tree decomposition



- when we don't have a tree we can create an **equivalent** tree graph
- by enlarging the alphabet $\mathcal{X} \rightarrow \mathcal{X}^k$
- $\text{Treewidth}(G) \equiv \text{Minimum such } k$
- it is NP-hard to determine the treewidth of a graph
- problem: in general $\text{Treewidth}(G) = \Theta(n)$

Tree decomposition of $G = (V, E)$



A tree $T = (V_T, E_T)$ and a mapping $V : V_T \rightarrow \text{SUBSETS}(V)$ s.t.:

- For each $i \in V$ there exists at least one $u \in V_T$ with $i \in V(u)$.
- For each $(i, j) \in E$ there exists at least one $u \in V_T$ with $i, j \in V(u)$.
- If $i \in V(u_1)$ and $i \in V(u_2)$, then $i \in V(w)$ for any w on the path between u_1 and u_2 in T .