Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
6.438 Algorithms for Inference
Fall 2014

11 The Max-Product Algorithm

In the previous lecture, we introduced the MAP elimination algorithm for computing an MAP configuration for any undirected graph. Today, we specialize the algorithm to the case of trees. The resulting algorithm is the Max-Product algorithm, which recovers the *max-marginal* at each node of a graph over random variables $\mathsf{x}_1, \ldots, \mathsf{x}_N$:

$$\bar{p}_{\mathsf{x}_i}(x_i) = \max_{x_j : j \neq i} p_{\mathbf{x}}(\mathbf{x}).$$

If each $x_i^* \in \text{argmax}_{x_i} \bar{p}_{\mathsf{x}_i}(x_i)$ is uniquely attained, i.e., there are no ties at each node, then the vector $\mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_N^*)$ is the unique global MAP configuration. Otherwise, keeping track of "backpointers" indicating what maximizing values neighbors took is needed to recover a global MAP configuration. Note that computing x_i^* is computationally not the same as first computing the marginal distribution of x_i and then looking at what value x_i maximizes $p_{\mathsf{x}_i}(x_i)$ because we are actually maximizing over all other x_j for $j \neq i$ as well!

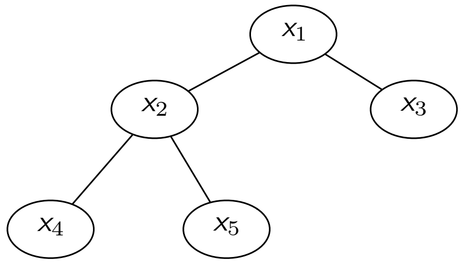
Max-Product relates to MAP elimination in the same way Sum-Product relates to elimination. In particular, for trees, we can just root the tree at an arbitrary node and eliminate from the leaves up to the root at which point we can obtain the max-marginal at the root node. We pass messages from the root back to the leaves to obtain max-marginals at all the nodes.

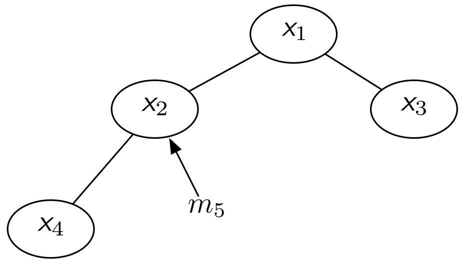
11.1 Max-Product for Undirected Trees

As usual, for undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = \{1, 2, \ldots, N\}$, we define a distribution over $\mathsf{x}_1, \ldots, \mathsf{x}_N$ via factorization:

$$p_{\mathbf{x}}(\mathbf{x}) \propto \prod_{i \in \mathcal{V}} \phi_i(x_i) \prod_{(i,j) \in \mathcal{E}} \psi_{ij}(x_i, x_j).$$

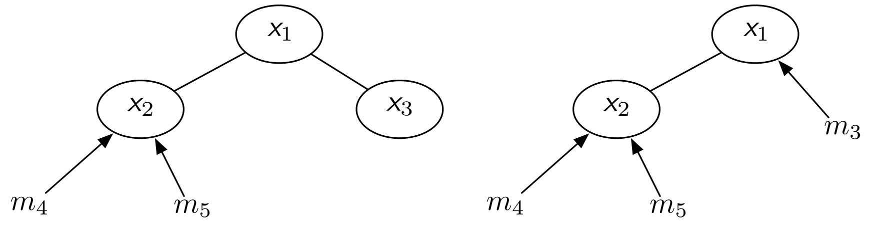
We work through an example to illustrate what happens when $\mathcal{G}$ is a tree. The intuition is nearly identical to that of Sum-Product. Consider an undirected graphical model shown below. We run the MAP elimination algorithm with ordering $(5, 4, 3, 2, 1)$, i.e., effectively we're rooting the tree at node 1:





Eliminating nodes 5, 4, and 3, we obtain messages:

$$m_5(x_2) = \max_{x_5 \in \mathcal{X}} \phi_5(x_5)\psi_{25}(x_2, x_5)$$
$$m_4(x_2) = \max_{x_4 \in \mathcal{X}} \phi_4(x_4)\psi_{24}(x_2, x_4)$$
$$m_3(x_1) = \max_{x_3 \in \mathcal{X}} \phi_3(x_3)\psi_{13}(x_1, x_3)$$



Eliminating node 2 yields message:

$$m_2(x_1) = \max_{x_2} \phi_2(x_2)\psi_{12}(x_1, x_2)m_4(x_2)m_5(x_2).$$

Finally, we obtain the max-marginal at node 1:

$$\bar{p}_{\mathrm{x}_1}(x_1) = \phi_1(x_1)m_2(x_1)m_3(x_1),$$

with maximal point

$$x_1^* \in \operatorname*{argmax}_{x_1 \in \mathcal{X}} \bar{p}_{\mathrm{x}_1}(x_1) = \operatorname*{argmax}_{x_1 \in \mathcal{X}} \phi_1(x_1)m_2(x_1)m_3(x_1).$$

To obtain max-marginals for all nodes, we could imagine that we instead choose a different node to act as the root and eliminate from leaves to the root. As with the Sum-Product case, the key insight is that messages computed previously could be reused. The bottom line is that it suffices to use the same serial schedule as for Sum-Product: pass messages from leaves to the root and then from the root back to the leaves.

Using the above serial schedule for message passing, in general the messages are computed as:

$$m_{i\to j}(x_j) = \max_{x_i \in \mathcal{X}} \phi_i(x_i)\psi_{ij}(x_i, x_j) \prod_{k \in N(i)\setminus j} m_{k\to i}(x_i). \quad (1)$$

Once all messages have been computed in both passes, then we compute max-marginals at each node:

$$\bar{p}_{\mathrm{x}_i}(x_i) = \phi_i(x_i) \prod_{k \in N(i)} m_{k\to i}(x_i). \quad (2)$$

As mentioned previously, in general we need to backtrack to obtain a global configuration, which requires us to store the argmax for each message:

$$\delta_{i\to j}(x_j) = \operatorname*{argmax}_{x_i \in \mathcal{X}} \phi_i(x_i)\psi_{ij}(x_i, x_j) \prod_{k \in N(i)\setminus j} m_{k\to i}(x_i).$$

Then the backtracking works as follows: Let $x_i^* \in \operatorname{argmax}_{x_i} \overline{p}_{\mathsf{x}_i}(x_i)$ for an arbitrary node i. For each $j \in N(i)$, assign $x_j^* = \delta_{j\to i}(x_i^*)$. Recurse until all nodes have been assigned a configuration.

Parallel Max-Product for Undirected Trees

As with Sum-Product, we have a parallel or *flood* schedule for message passing:

1. Initialize all messages $m^0_{i\to j}(x_j) = 1$ for all $(i, j) \in \mathcal{E}$.
2. Iteratively apply the update for $t = 0, 1, 2, \ldots$ until convergence:

$$m^{t+1}_{i\to j}(x_j) = \max_{x_i \in \mathcal{X}} \phi_i(x_i)\psi_{ij}(x_i, x_j) \prod_{k \in N(i)\setminus j} m^t_{k\to i}(x_i).$$

3. Compute max-marginals:

$$\overline{p}_{\mathsf{x}_i}(x_i) = \phi_i(x_i) \prod_{k \in N(i)} m^{t+1}_{k\to i}(x_i).$$

4. Backtrack to recover an MAP configuration.

As with parallel Sum-Product, we can reduce the amount of computation by precomputing at the beginning of each iteration of step 2:

$$\widetilde{m}^t_i(x_i) = \prod_{k \in N(i)} m^t_{k\to i}(x_i).$$

Then the update equation in step 2 becomes

$$m^{t+1}_{i\to j}(x_j) = \max_{x_i \in \mathcal{X}} \phi_i(x_i)\psi_{ij}(x_i, x_j) \frac{\widetilde{m}^t_i(x_i)}{m^t_{j\to i}(x_i)}.$$

Numerical stability, Max-Sum, and Min-Sum

In the message passing and max-marginal computation equations, we are multiplying many potential function values, which can lead to numerical issues. For example, if the potential functions represent probabilities, e.g., if our undirected graph is actually obtained from a DAG so the potentials are conditional probability distributions, then multiplying many probabilities can result in extremely small values below machine precision. A remedy for this is to work in the log domain.

Denote $\widetilde{b}_{i\to j}(x_j) = \log m_{i\to j}(x_j)$. Then taking the log of equations (1) and (2) results in updates for the Max-Sum algorithm, summarized below.

Message-passing:

$$\widetilde{b}_{i\to j}(x_j) = \max_{x_i \in \mathcal{X}} \left\{ \log \phi_i(x_i) + \log \psi_{ij}(x_i, x_j) + \sum_{k \in N(i)\backslash j} \widetilde{b}_{k\to i}(x_i) \right\}.$$

Max-marginals:

$$\overline{p}_{\mathsf{x}_i}(x_i) = \exp \left\{ \log \phi_i(x_i) + \sum_{k \in N(i)} \widetilde{b}_{k\to i}(x_i) \right\}.$$

We could also take the negative log instead of the log, which results in the Min-Sum algorithm. A rationale for why this might make sense is that if the potentials represent probabilities, then log probabilities take on negative values. In this case, if we wish to deal with strictly non-negative entries, then we we would use Min-Sum. Denote $b_{i\to j}(x_j) = -\log m_{i\to j}(x_j)$. Taking the negative log of equations (1) and (2), we obtain the update rules below.

Message-passing:

$$b_{i\to j}(x_j) = \min_{x_i \in \mathcal{X}} \left\{ -\log \phi_i(x_i) - \log \psi_{ij}(x_i, x_j) + \sum_{k \in N(i)\backslash j} b_{k\to i}(x_i) \right\}.$$

Max-marginals:

$$\overline{p}_{\mathsf{x}_i}(x_i) = \exp \left\{ \log \phi_i(x_i) - \sum_{k \in N(i)} b_{k\to i}(x_i) \right\}.$$

Lastly, we remark that if there are ties in MAP configurations, one way to remove the ties is to perturb the potential functions' values with a little bit of noise. If the model is numerically well-behaved, perturbing the potential functions' values with a little bit of noise should not drastically change the model but may buy us node max-marginals that all have unique optima, implying that the global MAP configuration is unique as well.

11.2 Max-Product for Factor Trees

The Max-Product algorithm for factor trees is extremely similar to that of undirected trees, so we just present the parallel schedule version of Max-Product on factor tree $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$:

1. Initialize all messages $m^0_{i\to a}(x_i) = m^0_{a\to i}(x_i) = 1$ for all $i \in \mathcal{V}, a \in \mathcal{F}$.
2. Iteratively apply the update for $t = 0, 1, 2, \ldots$ until convergence:

$$m^{t+1}_{a\to i}(x_i) = \max_{x_j : j\in N(a)\setminus i} f_a(\{x_i\} \cup \{x_j : j \in N(a) \setminus i\}) \prod_{j\in N(a)\setminus i} m^t_{j\to a}(x_j)$$
$$m^{t+1}_{i\to a}(x_i) = \prod_{b\in N(i)\setminus a} m^{t+1}_{b\to i}(x_i)$$

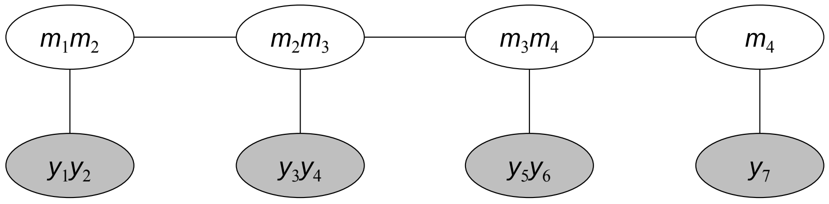
3. Compute max-marginals:

$$\bar{p}_{\mathsf{x}_i}(x_i) = \prod_{a\in N(i)} m^{t+1}_{a\to i}(x_i).$$

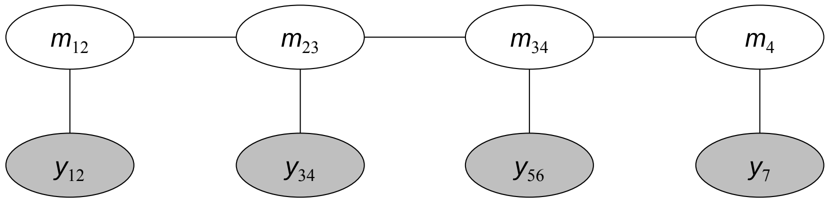
4. Backtrack to recover an MAP configuration.

11.3 Max-Product for Hidden Markov Models

We revisit our convolution code HMM example from Lecture 9, truncating it to have four hidden states:



To ease the notation a bit for what's to follow, let's denote $\mathsf{m}_i\mathsf{m}_{i+1} \triangleq \mathsf{m}_{i,i+1}$ and $\mathsf{y}_j\mathsf{y}_{j+1} \triangleq \mathsf{y}_{j,j+1}$. We thus obtain:

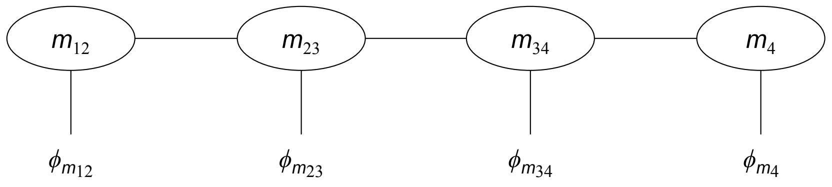


Fix $\varepsilon \in (0, 1/2)$. Let $w_\varepsilon = \log\left(\frac{1-\varepsilon}{\varepsilon}\right) > 0$. Then the potential functions are given by:

$$\begin{aligned} \psi_{\mathsf{m}_{i,i+1},\mathsf{m}_{i+1,i+2}}(\text{“}ab\text{”}, \text{“}cd\text{”}) &= \mathbf{1}(b = c) && \text{for } i \in \{1, 2\}, \\ \psi_{\mathsf{m}_{34},\mathsf{m}_4}(\text{“}ab\text{”}, \text{“}c\text{”}) &= \mathbf{1}(b = c), \\ \psi_{\mathsf{m}_{i,i+1},\mathsf{y}_{2i-1,2i}}(\text{“}ab\text{”}, \text{“}uv\text{”}) &= \exp\left\{\mathbf{1}(a = u)w_\varepsilon + \mathbf{1}(a \oplus b = v)w_\varepsilon\right\} && \text{for } i \in \{1, 2, 3\}, \\ \psi_{\mathsf{m}_4,\mathsf{y}_7}(\text{“}a\text{”}, \text{“}u\text{”}) &= \exp\left\{\mathbf{1}(a = u)w_\varepsilon\right\}, \end{aligned}$$

where $a, b, c, d, u, v \in \{0, 1\}$, "$ab$" denotes a length-2 bit-string, and $\mathbf{1}(A)$ is the indicator function that is 1 when event A is true and 0 otherwise.

Suppose we observe $\mathbf{y} = (0, 0, 0, 1, 0, 0, 0)$ and want to infer the MAP configuration for $\mathbf{m} = (m_1, m_2, m_3, m_4)$. Let's solve this using Max-Sum. The first thing to do is to incorporate the data into the model by plugging in the observed values; this will cause the resulting graphical model to just be a line:



The new singleton potentials are:

$$\begin{aligned}
\phi_{m_{12}}(\text{"}ab\text{"}) &= \exp\{\mathbf{1}(a = 0)w_\varepsilon + \mathbf{1}(a \oplus b = 0\}w_\varepsilon\} \\
\phi_{m_{23}}(\text{"}ab\text{"}) &= \exp\{\mathbf{1}(a = 0)w_\varepsilon + \mathbf{1}(a \oplus b = 1\}w_\varepsilon\} \\
\phi_{m_{34}}(\text{"}ab\text{"}) &= \exp\{\mathbf{1}(a = 0)w_\varepsilon + \mathbf{1}(a \oplus b = 0\}w_\varepsilon\} \\
\phi_{m_4}(\text{"}a\text{"}) &= \exp\{\mathbf{1}(a = 0)w_\varepsilon\}
\end{aligned}$$

The pairwise potentials remain the same.

Since we're going to use Max-Sum, we write out the log-potentials:

$$\begin{aligned}
\log \phi_{m_{12}}(\text{"}ab\text{"}) &= \mathbf{1}(a = 0)w_\varepsilon + \mathbf{1}(a \oplus b = 0\}w_\varepsilon \\
\log \phi_{m_{23}}(\text{"}ab\text{"}) &= \mathbf{1}(a = 0)w_\varepsilon + \mathbf{1}(a \oplus b = 1\}w_\varepsilon \\
\log \phi_{m_{34}}(\text{"}ab\text{"}) &= \mathbf{1}(a = 0)w_\varepsilon + \mathbf{1}(a \oplus b = 0\}w_\varepsilon \\
\log \phi_{m_4}(\text{"}a\text{"}) &= \mathbf{1}(a = 0)w_\varepsilon \\
\log \psi_{m_{i,i+1}, m_{i+1,i+2}}(\text{"}ab\text{"}, \text{"}cd\text{"}) &= \begin{cases} 0 & \text{if } b = c \\ -\infty & \text{otherwise} \end{cases} \qquad \text{for } i \in \{1, 2\} \\
\log \psi_{m_{34}, m_4}(\text{"}ab\text{"}, \text{"}c\text{"}) &= \begin{cases} 0 & \text{if } b = c \\ -\infty & \text{otherwise} \end{cases}
\end{aligned}$$

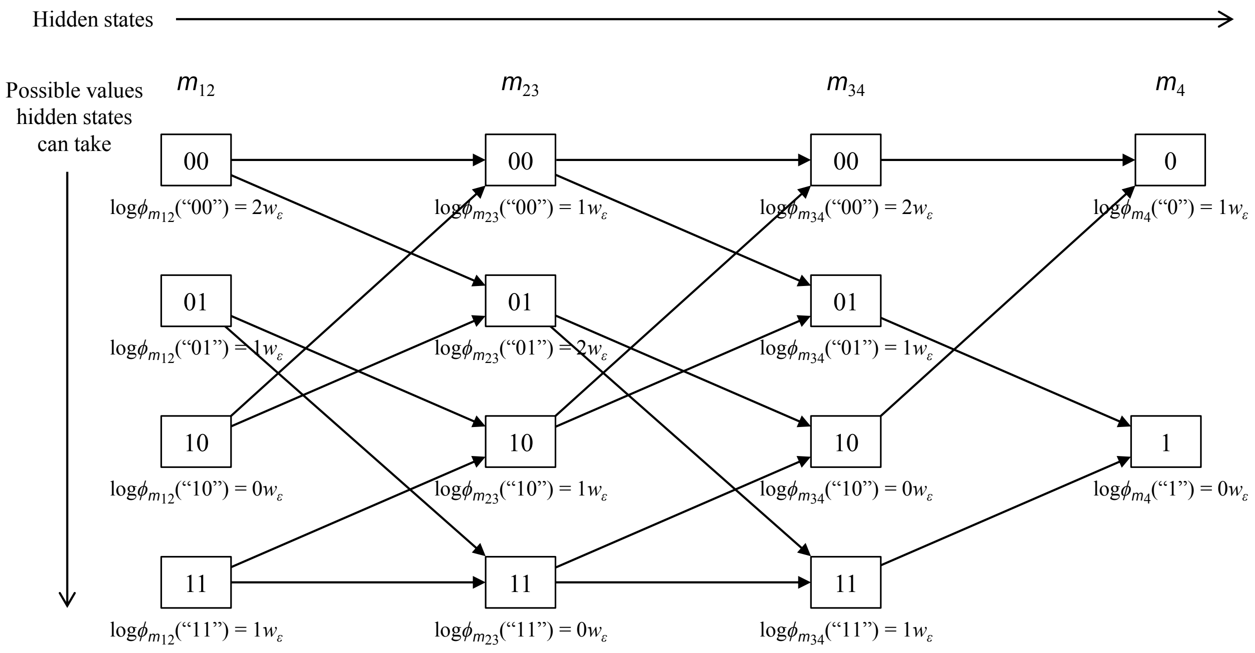
Using the Max-Sum message update equation,

$$\begin{aligned}
&\log m_{(i,i+1)\to(i+1,i+2)}(\text{"}cd\text{"}) \\
&= \max_{\text{"}ab\text{"} \in \{0,1\}^2} \left\{\log \phi_{m_{i,i+1}}(\text{"}ab\text{"}) + \log \psi_{m_{i,i+1}, m_{i+1,i+2}}(\text{"}ab\text{"}, \text{"}cd\text{"}) + \log m_{(i-1,i)\to(i,i+1)}(\text{"}ab\text{"})\right\} \\
&= \max_{a \in \{0,1\}} \left\{\log \phi_{m_{i,i+1}}(\text{"}ac\text{"}) + \log m_{(i-1,i)\to(i,i+1)}(\text{"}ac\text{"})\right\},
\end{aligned}$$

where the last equality uses the fact that what the pairwise potentials really are saying is that we only allow transitions where the last bit of the previous state is the first bit of the next state. This means that if we know the next state is "*cd*", then the previous state must end in "*c*", so we shouldn't bother optimizing over all bit-strings of length 2.

Note that if we pass messages from the left-most node to the right-most node, then what $\log m_{(i,i+1)\to(i+1,i+2)}(\text{“}cd\text{”})$ represents is the highest "score" achievable that ends in $m_{i+1,i+2} = \text{“}cd\text{”}$ but that does not include the contribution of the observation associated with $m_{i+1,i+2}$. The goal of Max-Sum would then be to find the MAP configuration with the highest overall score, with contributions from all nodes' singleton potentials and pairwise potentials.

We can visualize this setup using a *trellis diagram*:


Hidden states
Possible values hidden states can take
m_{12}
m_{23}
m_{34}
m_4
00
01
10
11
0
1
$\log\phi_{m_{12}}(\text{“00”}) = 2w_\varepsilon$
$\log\phi_{m_{12}}(\text{“01”}) = 1w_\varepsilon$
$\log\phi_{m_{12}}(\text{“10”}) = 0w_\varepsilon$
$\log\phi_{m_{12}}(\text{“11”}) = 1w_\varepsilon$
$\log\phi_{m_{23}}(\text{“00”}) = 1w_\varepsilon$
$\log\phi_{m_{23}}(\text{“01”}) = 2w_\varepsilon$
$\log\phi_{m_{23}}(\text{“10”}) = 1w_\varepsilon$
$\log\phi_{m_{23}}(\text{“11”}) = 0w_\varepsilon$
$\log\phi_{m_{34}}(\text{“00”}) = 2w_\varepsilon$
$\log\phi_{m_{34}}(\text{“01”}) = 1w_\varepsilon$
$\log\phi_{m_{34}}(\text{“10”}) = 0w_\varepsilon$
$\log\phi_{m_{34}}(\text{“11”}) = 1w_\varepsilon$
$\log\phi_{m_4}(\text{“0”}) = 1w_\varepsilon$
$\log\phi_{m_4}(\text{“1”}) = 0w_\varepsilon$

For each hidden state's possible value, incoming edges correspond to which possible previous states are possible. The MAP configuration is obtained by summing across scores (i.e., the log singleton potential values in this case) along a path that goes from one of the left-most nodes to the right-most node of the trellis diagram. If we keep track of which edges we take that maximizes the score along the path we traverse, then we can backtrack once we're at the end to figure out what the best configuration is, as was done in the MAP elimination algorithm from last lecture.

The key idea for the Max-Sum algorithm is that we basically iterate through the vertical layers (i.e., hidden states) in the trellis diagram one at a time from left to right. At each vertical layer, we store the highest possible score achievable that arrives at each possible value that the hidden state can take. Note that for this example, the edges do not have scores associated with them; in general, edges may have scores as well.

With this intuition, we run the Max-Sum with the modification that we keep track of which optimizing values give rise to the "highest score so far" at each hidden state's possible value. At the end, we backtrack to obtain the global MAP configuration $(00, 00, 00, 0)$, which achieves a overall score of $6w_{\varepsilon}$. Note that this algorithm of essentially using Max-Product on an HMM while keeping track of optimizing values to enable backtracking at the end is called the *Viterbi algorithm*, developed in 1967.