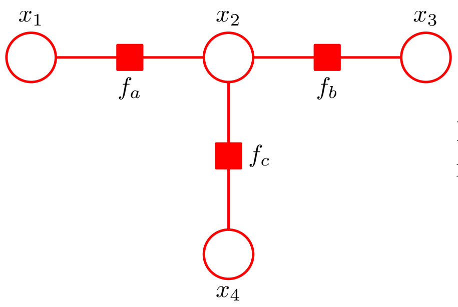
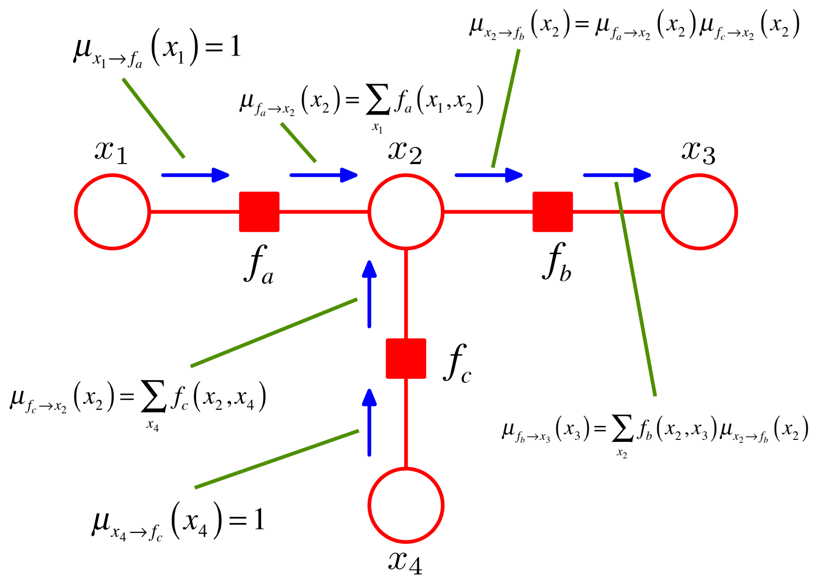
Sum-product algorithm example

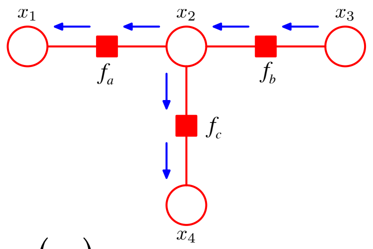
Let $\tilde{\mathrm{p}}(\mathbf{x}) = f_a(x_1,x_2) f_b(x_2,x_3) f_c(x_2,x_4)$



Declare x_3 as root node.



$\mu_{x_1 \to f_a}(x_1) = 1$
$\mu_{f_a \to x_2}(x_2) = \sum_{x_1} f_a(x_1,x_2)$
$\mu_{x_2 \to f_b}(x_2) = \mu_{f_a \to x_2}(x_2)\mu_{f_c \to x_2}(x_2)$
x_1
x_2
x_3
f_a
f_b
f_c
$\mu_{f_c \to x_2}(x_2) = \sum_{x_4} f_c(x_2,x_4)$
$\mu_{f_b \to x_3}(x_3) = \sum_{x_2} f_b(x_2,x_3)\mu_{x_2 \to f_b}(x_2)$
$\mu_{x_4 \to f_c}(x_4) = 1$
x_4



$$\mu_{x_3 \to f_b}(x_3) = 1$$
$$\mu_{f_b \to x_2}(x_2) = \sum_{x_3} f_b(x_2,x_3)$$
$$\mu_{x_2 \to f_a}(x_2) = \mu_{f_b \to x_2}(x_2)\mu_{f_c \to x_2}(x_2)$$
$$\mu_{f_a \to x_1}(x_1) = \sum_{x_2} f_a(x_1,x_2)\mu_{x_2 \to f_a}(x_2)$$
$$\mu_{x_2 \to f_c}(x_2) = \mu_{f_a \to x_2}(x_2)\mu_{f_b \to x_2}(x_2)$$
$$\mu_{f_c \to x_4}(x_4) = \sum_{x_2} f_c(x_2,x_4)\mu_{x_2 \to f_c}(x_2)$$

Handling observed variables

Usually we have observed variables (e.g., evidence).

We simply clamp those variables to their observed values.

More formally, denote hidden variables by $\mathbf{h}$, and observed ones by $\mathbf{v}$. Denote the observed value as $\hat{\mathbf{v}}$. For each observed variable, v_i, with value $\hat{v}_i$, we can introduce a factor

$$I(v_i, \hat{v}_i) = \begin{cases} 1 & \text{if } v_i = \hat{v}_i \\ 0 & \text{otherwise} \end{cases}$$

Then, $p(\mathbf{h}, \mathbf{v} = \hat{\mathbf{v}}) = p(\mathbf{h}, \mathbf{v}) \prod_i I(v_i, \hat{v}_i)$

(can be normalized to get $p(\mathbf{h} \mid \hat{\mathbf{v}})$)

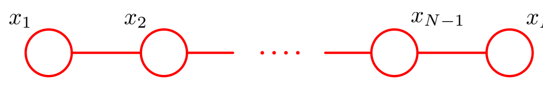
Max-sum algorithm

Method to compute.

$$\mathbf{x}^{\max} = \arg\max_{\mathbf{x}} p(\mathbf{x})$$

$$i.e.,\ p\left(\mathbf{x}^{\max}\right) = \max_{\mathbf{x}} p(\mathbf{x})$$

Recall inference on chains



$$p(\mathrm{x}) = \psi_{1,2}(x_1,x_2)\psi_{2,3}(x_2,x_3)\ \ \dots\ \ \psi_{N-2,N-1}(x_{N-2},x_{N-1})\psi_{N-1,N}(x_{N-1},x_N)$$

Naive compution of $\arg\max_{\mathbf{x}} p(\mathbf{x})$

would evauate the above for each value of x,

and take the max.

Too expensive!!

Recall speeding up marginalization

$$p(x_n) = \left(\sum_{x_{n-1}}\sum_{x_{n-2}}\cdots\sum_{x_1}\prod_{i=1}^{n-1}\psi_{n-i,\,n-i+1}(x_{n-i},x_{n-i+1})\right)\left(\sum_{x_{n+1}}\cdots\sum_{x_{N-1}}\sum_{x_N}\prod_{i=n}^{N-1}\psi_{i,\,i+1}(x_i,x_{i+1})\right)$$

$$\sum_{x_{n-1}}\sum_{x_{n-2}}\cdots\sum_{x_1}\prod_{i=1}^{n-1}\psi_{n-i,n-i+1}(x_{n-i},x_{n-i+1}) = \left\{\sum_{x_{n-1}}\psi_{n-1,n}(x_{n-1},x_n)\ \dots\ \left\{\sum_{x_3}\psi_{3,4}(x_3,x_4)\left\{\sum_{x_2}\psi_{2,3}(x_2,x_3)\left\{\sum_{x_1}\psi_{1,2}(x_1,x_2)\right\}\right\}\right\}\dots\right\}$$

and

$$\sum_{x_{n+1}}\cdots\sum_{x_{N-1}}\sum_{x_N}\prod_{i=n}^{N-1}\psi_{i,\,i+1}(x_i,x_{i+1}) = \left\{\sum_{x_{n+1}}\psi_{n,n+1}(x_n,x_{n+1})\ \dots\left\{\sum_{x_{N-1}}\psi_{N-2,N-1}(x_{N-2},x_{N-1})\left\{\sum_{x_N}\psi_{N-1,N}(x_{N-1},x_N)\right\}\right\}\dots\right\}$$

What if we could do with max() what we are doing with $\sum$?

Helpful facts

First note that.

$$\max_{\mathbf{x}} p(\mathbf{x}) = \max_{x_1}\ \max_{x_2}\ \dots\ \max_{x_M} p(\mathbf{x})$$

Second note that.

$$\max(ab,ac) = a\ \max(b,c) \qquad \text{(for a} \geq 0)$$

Max on a chain

$$\max_{\mathbf{x}} p(\mathbf{x}) = \frac{1}{Z}\ \max_{x_1}\left[\ \max_{x_2}\left[\ldots\left[\psi_{1,2}(x_1,x_2)\psi_{2,3}(x_2,x_3)\ \ldots\ \psi_{N-1,N}(x_{N-1},x_N)\right]\right]\right]$$
$$= \frac{1}{Z}\max_{x_1}\left[\psi_{1,2}(x_1,x_2)\left[\max_{x_2}\left[\psi_{2,3}(x_2,x_3)\left[\ \ldots\ \max_{x_N}\left[\psi_{N-1,N}(x_{N-1},x_N)\right]\right]\right]\right]\right]$$

The second line shows that we can do this using message passing.

(Compute max distribution, then multiply, rinse and repeat.)

Max-sum algorithm

Two steps.
1) Compute the max while remembering certain computations
2) Compute a value of **x** that achieves the max

The message passing algorithm for step (1) is clear from the analog with the “sum-product” algorithm, except that it should be called the “max-product” algorithm.

Computing long products looses precision, so we switch to log().

Max-sum algorithm

Note that

$$\ln\left(\max_{\mathbf{x}}\left(p(x)\right)\right) = \max_{\mathbf{x}}\left(\ln\left(p(x)\right)\right)$$

And we have

$$\max\left(a+b,\, a+c\right) = a + \max\left(b,c\right)$$

Max-sum algorithm

Working now in analogy with the sum-product algorithm

$$\mu_{f\to x}(x) = \max_{x_1, x_2, \ldots, x_M}\left[\ln\left(f(x, x_1, x_2, \ldots, x_M) + \sum_{m\in n(f_s)\backslash x}\mu_{x_m\to f}(x_m)\right)\right]$$

and

$$\mu_{x\to f}(x) = \sum_{l\in n(x)\backslash f}\mu_{f\to x_l}(x)$$

Max-sum algorithm

Working now in analogy with the sum-product algorithm

For initialization at leaf nodes

$\mu_{f\to x}(x) = 0$

and

$\mu_{x\to f}(x) = \ln(f(x))$

Max-sum algorithm

Working now in analogy with the sum-product algorithm

To compute the max using the choosen root node,

$$\mathrm{p}^{\max} = \max_{\mathbf{x}}\left[\sum_{s\in n(x)} \mu_{s\to x_s}(x)\right]$$

Max-sum algorithm

Now we need to find an **x** where p reaches the max.

This does not have a direct analogy in the sum-product algorithm.

Why we do not know x yet:

The factor-to-node messages takes a distribution for the maxima over the downstream variables, and multiplies it by the factor (sum using logs), and reports a new distribution.

We do not know which value in the new distribution will be part of the maximum.

Max-sum algorithm

Can passing messages backwards find the arg max?

At the root node, which is a product (sum in logs), the contributions of the maximum are factored, so messages backwards can find the configuration.

But at the factor nodes, if there are two configurations that give the same maximum, we might get pieces of each of them.

This is because the variable for the node collecting the factor-to-node messages is shared.

(We made an arbitrary choice, but did not write it down!)

Max-sum algorithm

Adjustment to forward message passing so backtracking works:

The factor-to-node operations store the dependencies for the various choices of x_i .

Then, once the node-to-factor backtracking expresses a choice, a consistent set of values for x_i for the max can be found.

Max-sum algorithm (back-tracking)

In more detail, when we compute

$$\mu_{f\to x}(x) = \max_{x_1, x_2, \ldots, x_M}\left[\ln\left(f(x, x_1, x_2, \ldots, x_M) + \sum_{m\in n(f_s)\backslash x} \mu_{x_m\to f}(x_m)\right)\right]$$

store

$$\phi(x) = \arg\max_{x_1, x_2, \ldots, x_M}\left[\ln\left(f(x, x_1, x_2, \ldots, x_M) + \sum_{m\in n(f_s)\backslash x} \mu_{x_m\to f}(x_m)\right)\right]$$

(This records the downstream choices for any upstream choice of x)

Then, once we know the overall max, we can recover a set of x_i that leads to it by backtracking.