Max-sum algorithm

Method to compute.

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

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

Helpful facts

First, note that.

$$\max_{\mathbf{x}} p(\mathbf{x}) = \max_{x_1}\ \max_{x_2} \ldots \max_{x_M} p(\mathbf{x}) \qquad \text{(any order you like)}$$

Second, note that.

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

So, as you might expect

$$\max_{x_i,y_j}(x_i y_j) = \left(\max_{x_i}(x_i)\right)\left(\max_{y_j}(y_j)\right) \qquad (\text{for } x_i, y_j \geq 0)$$

Recall inference on chains

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

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

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

where

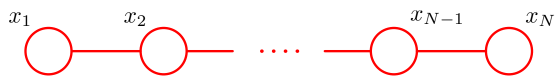
$$\sum_{x_{n-1}}\sum_{x_{n-2}} \cdots \sum_{x_1}\prod_{i=1}^{n-1}\psi_{n-i,n-i+1}(x_{n-i},x_{n-i+1}) = \left\{\sum_{x_{n-1}}\psi_{n-1,n}(x_{n-1},x_n) \ldots \left\{\sum_{x_3}\psi_{3,4}(x_3,x_4)\left\{\sum_{x_2}\psi_{2,3}(x_2,x_3)\left\{\sum_{x_1}\psi_{1,2}(x_1,x_2)\right\}\right\}\right\}\ldots\right\}$$

and

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

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

Max on chains



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

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

$$= \max_{x_1}\left[\max_{x_2}\left(\ldots\max_{x_{N-1}}\left(\max_{x_N}\left(\prod_{i=1}^{N-1}\psi_{i,i+1}(x_i,x_{i+1})\right)\right)\right)\right]$$

$$= \max_{x_1}\left[\max_{x_2}\left(\ldots\max_{x_{N-1}}\left(\prod_{i=1}^{N-2}\psi_{i,i+1}(x_i,x_{i+1})\left(\max_{x_N}\left(\psi_{N-1,N}(x_{N-1},x_N)\right)\right)\right)\right)\right]$$

(move products not involving x_N outside $\max_{x_N}\left(\psi_{N-1,N}(x_{N-1},x_N)\right)$)

$$= \max_{x_1}\left[\max_{x_2}\left(\psi_{1,2}(x_1,x_2)\bullet\max_{x_3}\left(\ldots\max_{x_{N-1}}\left(\psi_{N-2,N-1}(x_{N-2},x_{N-1})\bullet\max_{x_N}\left(\psi_{N-1,N}(x_{N-1},x_N)\right)\right)\right)\right)\right]$$

(continue moving factors out)

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 would then be called the "max-product" algorithm.

Computing long products looses precision (*), so we switch to log(), and call it the max-sum algorithm.

(*) Less of an issue with marginalization.

Max-sum algorithm (preliminaries)

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

$$\begin{aligned}\ln\left(\max(ab,ac)\right) &= \max\left(\log(ab),\log(ac)\right)\\ &= \max\left(\ln(a)+\ln(b),\ln(a)+\ln(c)\right)\\ &= \ln(a)+\max\left(\ln(b)+\ln(c)\right)\end{aligned}$$

(In general, $\max(x+y,x+z) = x+\max(y,z)$)

Can also get this from taking logs of product version,
namely: $\max(ab,ac) = a\cdot\max(b,c)$, for $a \geq 0$)

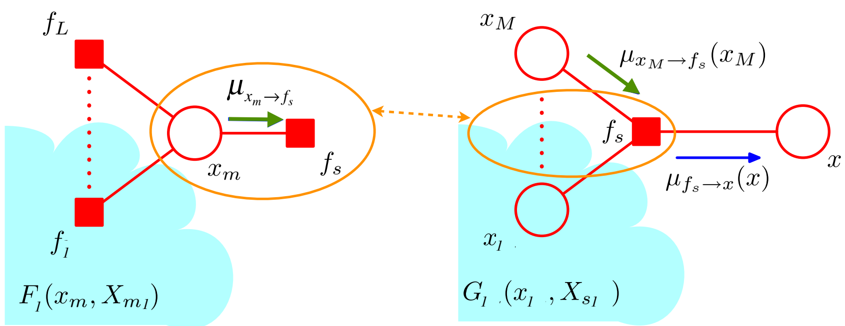
Max-sum algorithm

Develop using the analogy with the sum-product algorithm

Sums become max() and products become sums (over logs)

We will use the same notation for messages, but the semantics is a bit different (as above) and the quantities are always logs.

Sum product on a slide



$$\mu_{x_m \to f_s}(x_m) = \prod_{l \in n(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m)$$

$$\mu_{f_s \to x}(x) = \sum_{x_1} \sum_{x_2} \cdots \sum_{x_M} f_s(x, x_1, x_2, \ldots, x_M) \prod_{m \in ne(f_s) \setminus x} \mu_{x_m \to f_s}(x_m)$$

Max-sum algorithm

Develop using the analogy with the sum-product algorithm

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

and

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

Recall that in sum-product: $\underbrace{\mu_{f_s \to x}(x)}_{\text{factor} \to \text{node}} = \sum_{x_1} \ldots \sum_{x_M} f(x, x_1, \ldots, x_M) \prod_{m \in n(f_s) \setminus x} \underbrace{\mu_{x_m \to f_s}(x_m)}_{\text{node} \to \text{factor}}$

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,

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

Max-sum algorithm

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

This does not have an exact 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 upstream variables, and multiplies it by the factor (sum using logs), and reports a new distribution.

We do not yet know which value in the new distribution will be part of the maximum (it is not necessarily argmax of the reported distribution).

Max-sum algorithm

Can passing messages backwards find the arg max?

At the root node, which is a product (sum in logs), when we find the maximum, we can easily record the argmax for that node's variable, and it will be a valid for a particular maximizing configuration.

In analogy with sum-product, we might be tempted to send messages backwards to "finish the job" to get values for the other nodes.

But this can fail if there is more than one maximal configuration.
You can get pieces of each one!
We are only sure that the value is part of **some** maximal configuration.
You could end up with an inconsistent set of values.

Max-sum algorithm

Adjustment to forward message passing for "backtracking:"

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.

For example, suppose the root node could choose either setting its variable to 2 or 3, 2. Then it sends to the incoming nodes, the value "2". Those nodes need to know how their incoming links have maximized to get the value for "2" passed to the root.

Max-sum algorithm (back-tracking)

In more detail, when we compute

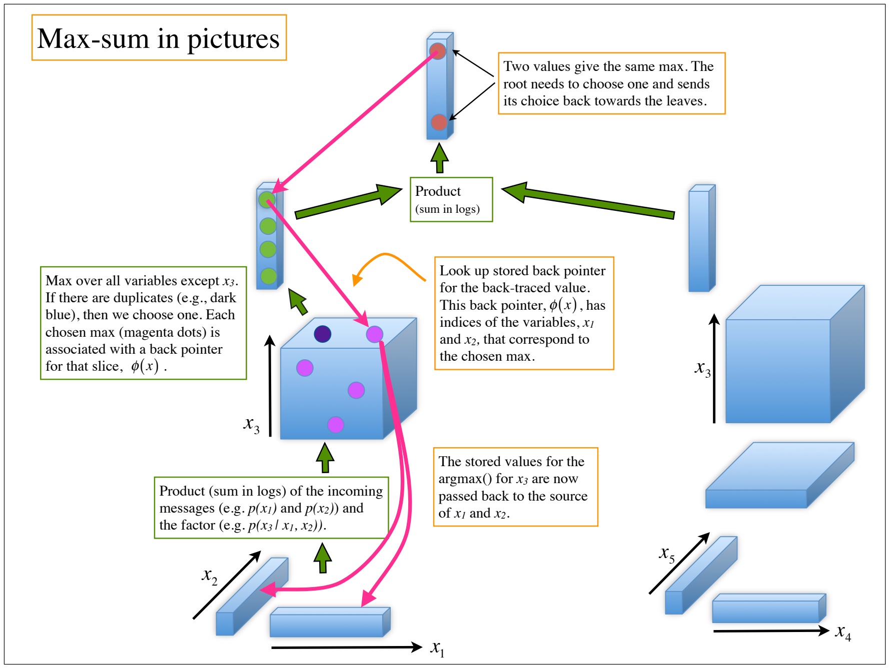
$$\mu_{f\to x}(x) = \max_{x_1, x_2, \ldots, x_M}\left[\ln\left(f(x, x_1, x_2, \ldots, x_M) + \sum_{m\in n(f_s)\backslash x} \mu_{x_m\to f}(x_m)\right)\right]$$

store

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

This records the downstream choices for any upstream choice of x.
$\phi(x)$ stored M values for each value of x.

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


Max-sum in pictures
Two values give the same max. The root needs to choose one and sends its choice back towards the leaves.
Product
(sum in logs)
Max over all variables except x_3. If there are duplicates (e.g., dark blue), then we choose one. Each chosen max (magenta dots) is associated with a back pointer for that slice, $\phi(x)$.
Look up stored back pointer for the back-traced value. This back pointer, $\phi(x)$, has indices of the variables, x_1 and x_2, that correspond to the chosen max.
x_3
Product (sum in logs) of the incoming messages (e.g. $p(x_1)$ and $p(x_2)$) and the factor (e.g. $p(x_3 | x_1, x_2)$).
The stored values for the argmax() for x_3 are now passed back to the source of x_1 and x_2.
x_2
x_1
x_3
x_5
x_4