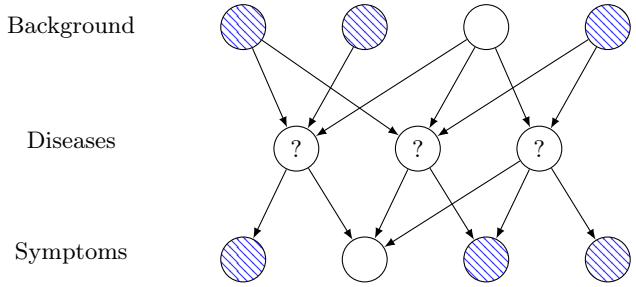
Chapter 11

Inference in Graphical Models

11.1 Introduction

Inference refers to drawing conclusions about unknown quantities based on observations and a model. In the context of graphical models assume, our goal is to learn about a set of query nodes given observed nodes.

For example, consider the following graph with nodes for background information about a patient (e.g., diet, exercise, genetics, etc.), diseases (diabetes, hypertension, etc.), and symptoms/test results (blood pressure, etc). Our goal is assign probabilities to disease based on our observations. Alternatively, we may be interested in identifying the disease that is most likely.



In such a graph, we deal with three types of nodes, evidence (observed) nodes, x_E, query nodes, x_Q, and other nodes, x_O.

Without having made any observations, we can find the probability of the query nodes through *marginalization*:

$$p(x_Q) = \sum_{x_O, x_E} p(x_Q, x_O, x_E),$$

and with observations, through *conditioning*:

$$p(x_Q|x_E) \propto \sum_{x_O} p(x_Q, x_O, x_E).$$

Since we can view the latter case as doing summation over x_E that only consists of a single set of values, from this point on, we will only consider marginalization. Note that we need to compute $\sum_{x_O} p(x_Q, x_O, x_E)$ for all values of x_Q to be able to find $p(x_Q|x_E)$.

11.2 The Elimination Algorithm

We need to find

$$p(x_4) = \sum_{x_1,x_2,x_3,x_5} p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_3)p(x_5|x_4).$$

Assume each node can take k different values. In the naive approach, we need to compute and add $O(k^4)$ terms.

We could eliminate the nodes in different orders:

$$\begin{aligned}
p(x_4) &= \sum_{x_1}\sum_{x_2}\sum_{x_3}\sum_{x_5} p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_3)p(x_5|x_4) \\
&= \sum_{x_1}\sum_{x_2}\sum_{x_3} p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_3)\sum_{x_5} p(x_5|x_4) \\
&= \sum_{x_1}\sum_{x_2}\sum_{x_3} p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_3) \\
&= \sum_{x_1}\sum_{x_2} p(x_1)p(x_2|x_1)\sum_{x_3} p(x_3|x_2)p(x_4|x_3) \\
&= \sum_{x_1}\sum_{x_2} p(x_1)p(x_2|x_1)\mu(x_2,x_4) \\
&= \sum_{x_1} p(x_1)\sum_{x_2} p(x_2|x_1)\mu(x_2,x_4) \\
&= \sum_{x_1} p(x_1)\mu(x_1,x_4) \\
&= p(x_4)
\end{aligned}$$

The computational complexity is $O(k^2)$, i.e., we need of the order of k^2 computations. In particular, computing $\mu(x_2, x_4)$ needs to be done for k different values of x_2 and each of these requires computing and adding k terms, one for each possible value of x_3.

Note that in Bayesian networks, we can ignore downstream nodes since their probability marginalizes to 1 (but not in MRFs).

We could also choose the following ordering:

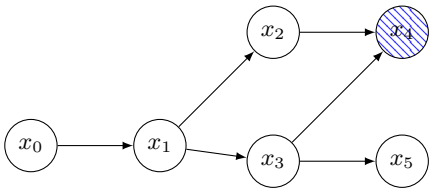
$$\begin{aligned}
p(x_4) &= \sum_{x_1}\sum_{x_3}\sum_{x_2} p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_3) \\
&= \sum_{x_1}\sum_{x_3} p(x_1)p(x_4|x_3)\sum_{x_2} p(x_2|x_1)p(x_3|x_2) \\
&= \sum_{x_1}\sum_{x_3} p(x_1)p(x_4|x_3)\mu(x_1,x_3) \\
&= \sum_{x_1} p(x_1)\sum_{x_3} p(x_4|x_3)\mu(x_1,x_3) \\
&= \sum_{x_1} p(x_1)\mu(x_1,x_4) \\
&= p(x_4).
\end{aligned}$$

Here, computing $\mu(x_1, x_3)$ has complexity $O(k^3)$.

More generally, for a Markov chain with n nodes, the complexity is $O(nk^2)$ for computing $p(x_n)$. But with the naive approach it is $O(k^n)$.

The problem of finding the best ordering for elimination is NP-hard (i.e., computationally difficult) for general graphs.

Now let us find $p(x_0|x_4)$ in the following network:



We have

$$\begin{aligned}
p(x_0|x_4) &\propto \sum_{x_1,x_2,x_3} p(x_0)p(x_1|x_0)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2,x_3) \\
&= \sum_{x_1,x_3} p(x_0)p(x_1|x_0)p(x_3|x_1) \sum_{x_2} p(x_2|x_1)p(x_4|x_2,x_3) \\
&= \sum_{x_1,x_3} p(x_0)p(x_1|x_0)p(x_3|x_1)\mu(x_1,x_3,x_4) \\
&= \sum_{x_1} p(x_0)p(x_1|x_0) \sum_{x_3} p(x_3|x_1)\mu(x_1,x_3,x_4) \\
&= \sum_{x_1} p(x_0)p(x_1|x_0)\mu(x_1,x_4) \\
&= p(x_0) \sum_{x_1} p(x_1|x_0)\mu(x_1,x_4) \\
&= p(x_0)\mu(x_0,x_4).
\end{aligned}$$

The complexity is dominated by $\mu(x_1,x_3,x_4)$, which is $O(k^3)$, assuming each node can take on k values.

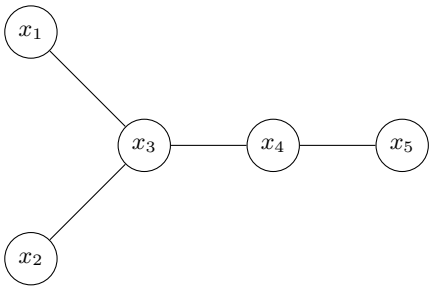
11.3 The Sum-Product Algorithm

The *sum-product algorithm*, also known as *belief propagation* and *sum-product message passing*, provides a simple way of doing exact inference on trees. It is also commonly used on graphs that are not trees since it often provides good approximations.

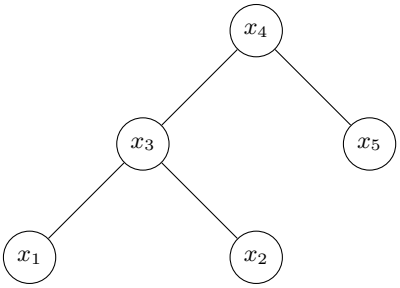
We need to clarify what we mean by trees. For Markov random fields, the algorithm works on trees, but for Bayesian networks, it works for graphs whose equivalent MRF (the moralized graph) is a tree. In particular, no node can have more than one parent. Given the straightforward equivalence between these two categories, we only consider Markov random field trees.

Consider the the following MRF, where we are interested in $p(x_4)$, with

$$p(x_1^4) \propto \psi(x_1,x_3)\psi(x_2,x_3)\psi(x_2)\psi(x_3,x_4)\psi(x_4)\psi(x_4,x_5)$$



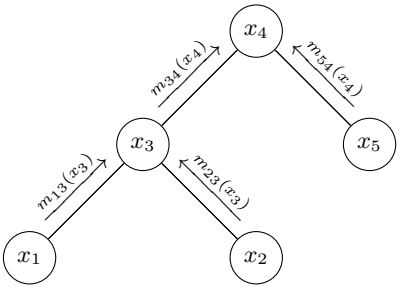
Let's look at this graph as a rooted tree,



and perform elimination starting from the leaves to the roots:

$$
\begin{aligned}
p(x_4) &\propto \sum_{x_1,x_2,x_3,x_5} \psi(x_1,x_3)\psi(x_2,x_3)\psi(x_2)\psi(x_3,x_4)\psi(x_4)\psi(x_4,x_5) \\
&= \sum_{x_3} \psi(x_3,x_4)\psi(x_4)\left(\sum_{x_1}\psi(x_1,x_3)\right)\left(\sum_{x_2}\psi(x_2,x_3)\psi(x_2)\right)\left(\sum_{x_5}\psi(x_4,x_5)\right) \\
&= \sum_{x_3} \psi(x_3,x_4)\psi(x_4)\ m_{13}(x_3)\ m_{23}(x_3)\ m_{54}(x_4) \\
&= \psi(x_4)m_{54}(x_4)\sum_{x_3}\psi(x_3,x_4)\ m_{13}(x_3)\ m_{23}(x_3) \\
&= \psi(x_4)m_{54}(x_4)m_{34}(x_4)
\end{aligned}
\tag{11.1}
$$

We can view this computation as being done on each node and then messages being passed to neighbors:



where

$$
\begin{aligned}
m_{13}(x_3) &= \sum_{x_1} \psi(x_1, x_3), \\
m_{23}(x_4) &= \sum_{x_2} \psi(x_2, x_3)\psi(x_2), \\
m_{54}(x_4) &= \sum_{x_5} \psi(x_4, x_5), \\
m_{34}(x_4) &= \sum_{x_3} \psi(x_3, x_4)\ m_{13}(x_3)\ m_{23}(x_3).
\end{aligned}
$$

and then at the root, we can find $p(x_4)$ as

$$p(x_4) \propto \psi(x_4) m_{54}(x_4) m_{34}(x_4).$$

Recall that this also works for conditioning. Specifically, if we are interested in the conditional probability $p(x_4|x_3 = a)$, we would compute

$$
\begin{aligned}
m_{34}(x_4) &= \psi(x_3 = a, x_4)\ m_{13}(a)\ m_{23}(a), \\
p(x_4) &\propto \psi(x_4) m_{54}(x_4) m_{34}(x_4).
\end{aligned}
$$

We can state the sum-product algorithm for a rooted tree as follows. At each node x_j with parent x_k,

- Product step: After receiving messages $m_{ij}(x_j)$ from all children x_i of x_j, compute the product of all messages and any potential functions containing x_j,

  $$\psi(x_j)\psi(x_j, x_k) \prod_i m_{ij}(x_j).$$

  Note that not all potentials are always present.

- Sum step: Sum over all possible values of x_j to produce the message

  $$m_{jk}(x_k) = \sum_{x_j} \psi(x_j)\psi(x_j, x_k) \prod_i m_{ij}(x_j), \tag{11.2}$$

  and send to x_k.

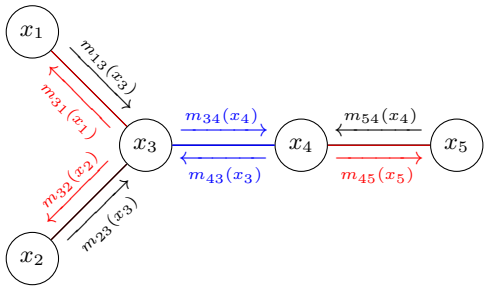
A critical point in the correctness of the sum-product algorithm is that the messages received by each node are functions of the value of that node. This is easy to see by induction. After the product step, we get a function of both the current node x_j and its parent x_k. The sum eliminates the current node and so the parent node x_k receives a message that is only a function of x_k.

Complexity of computing each message: Suppose each node can take on K different values, namely $\{1, 2, \ldots, K\}$. So the sum in (11.2) contains K terms. Furthermore, $m_{jk}(x_k)$ needs to be computed for $x_k = 1, 2, \ldots, K$. We can imagine a vector

$$\boldsymbol{m}_{jk} = \begin{pmatrix} m_{jk}(1) \\ m_{jk}(2) \\ \vdots \\ m_{jk}(K) \end{pmatrix}$$

being sent to the node x_k. So the complexity at each node is $O(K^2)$ and for n nodes the complexity is $O(nK^2)$.

Computing marginals at all nodes. We can easily extend this algorithm to computing all marginals rather than a single node. We note that the messages sent by the nodes do not depend on the location of the root. Each node sends a message when it receives messages from all but one of its neighbors. We can extend this by not sending a message only once, but sending a message to each neighbor based on the messages received by the other neighbors:



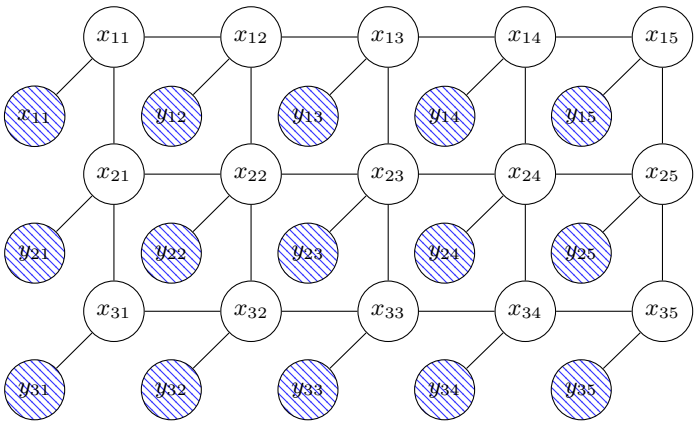
Here the order of messages is color-coded: 1, 2, 3. We can now find the marginal at each node. For example,

$$p(x_2) \propto m_{32}(x_2)\psi(x_2),$$
$$p(x_3) \propto m_{13}(x_3)m_{23}(x_3)m_{43}(x_3).$$

Example 102. An example for the sum-product algorithm is given at the end of the document.

11.4 The Max-Product Algorithm

The max-product algorithm is used to identify the configuration that has the maximum probability. Examples include part-of-speech tagging, voice recognition, decoding (communication), and image denoising. The last example is shown below:



where x_{ij} are true image pixels and y_{ij} are observed pixels, e.g., from a camera. Our goal is to find

$$\arg\max_x p(x, y).$$

Note that the local maximum-probability configuration does not necessarily agree with the global maximum-probability configuration. As an example, consider

$p(x_1, x_2)$	$x_1 = 0$	$x_1 = 1$
$x_2 = 0$	.3	.4
$x_2 = 1$	.3	0

We have

$$\arg\max_{x_1,x_2} p(x_1, x_2) = (1, 0)$$
$$\arg\max_{x_1} p(x_1) = \arg\max_{x_1}(p(x_1, x_2 = 0) + p(x_1, x_2 = 1)) = 0.$$

To see the max-product algorithm, suppose we want to find

$$\arg\max_{x_1^5} p(x_1^5)$$

for the tree given in the previous section. To solve this problem, let us start with solving

$$\max_{x_1^5} p(x_1^5)$$

We proceed similar to (11.1). For clarity, we make the partition function Z explicit, but we don't actually need to find it. We replace each summation in the previous derivation with max and write:

$$\begin{aligned}
\max p(x_1^5) &= \max_{x_1,x_2,x_3,x_4,x_5} Z\psi(x_1, x_3)\psi(x_2, x_3)\psi(x_2)\psi(x_3, x_4)\psi(x_4)\psi(x_4, x_5) \\
&= Z \max_{x_4} \max_{x_3} \psi(x_3, x_4)\psi(x_4) \left(\max_{x_1} \psi(x_1, x_3)\right) \left(\max_{x_2} \psi(x_2, x_3)\psi(x_2)\right) \left(\max_{x_5} \psi(x_4, x_5)\right) \\
&= Z \max_{x_4} \max_{x_3} \psi(x_3, x_4)\psi(x_4)\ m_{13}(x_3)\ m_{23}(x_3)\ m_{54}(x_4) \\
&= Z \max_{x_4} \psi(x_4) m_{54}(x_4) \max_{x_3} \psi(x_3, x_4)\ m_{13}(x_3)\ m_{23}(x_3) \\
&= Z \max_{x_4} \psi(x_4) m_{54}(x_4) m_{34}(x_4)
\end{aligned}$$

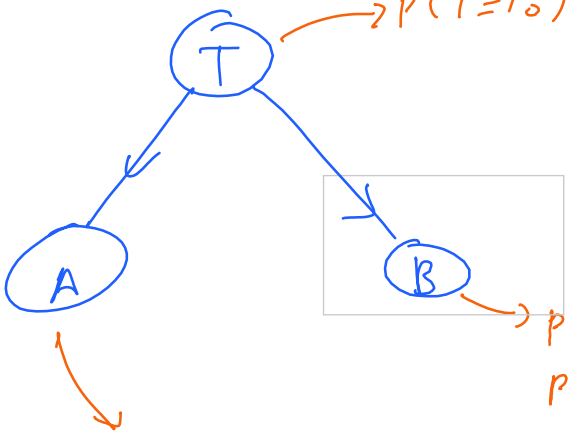
This is the same as the sum-product algorithm, except that we take the max of product terms. We can again view this as message-passing, but using max instead of sum, with the following messages:

$$\begin{aligned}
m_{13}(x_3) &= \max_{x_1} \psi(x_1, x_3), \\
m_{23}(x_4) &= \max_{x_2} \psi(x_2, x_3)\psi(x_2), \\
m_{54}(x_4) &= \max_{x_5} \psi(x_4, x_5), \\
m_{34}(x_4) &= \max_{x_3} \psi(x_3, x_4)\ m_{13}(x_3)\ m_{23}(x_3).
\end{aligned}$$

If we have Z, we can find the maximum probability. But we are interested in the values x^* of x that achieve this maximum probability (also we don't have Z). To find x^*, we simply need to keep track of which values of x_i maximize the message. Specifically, for a message $m_{ij}(x_j)$, we should know for each value of x_j what value of x_i was used to obtain the maximum value of the message. Then, when we find what value of x_4 maximizes the probability at the last step, we backtrack and find all the other x_is.

11.5 Sum-product Example

Example:



$p(T=T_0) = 0.65 = 1-p(T=T_1)$

$p(B=B_0 \mid T=T_0) = 0.82$

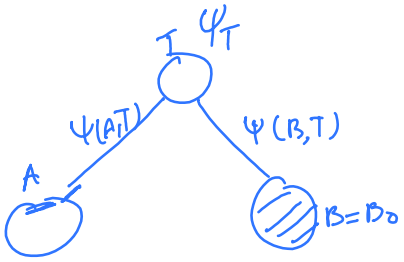
$p(B=B_0 \mid T=T_1) = 0.15$

$p(A=A_0 \mid T=T_0) = 0.9$

$p(A=A_0 \mid T=T_1) = 0.5$

$$p(A,B,T) = p(T)\,p(A \mid T)\,p(B \mid T)$$

We first convert this to an MRF



$\psi_T(T) = p(T) =$

$T=0$	$T=1$
0.65	0.35

$\psi_{BT}(B,T) = p(B \mid T) =$

	$B=0$	$B=1$
$T=0$	0.82	0.18
$T=1$	0.15	0.85

$\psi_{AT}(A,T) = p(A \mid T) =$

	$A=0$	$A=1$
$T=0$	0.9	0.1
$T=1$	0.5	0.5

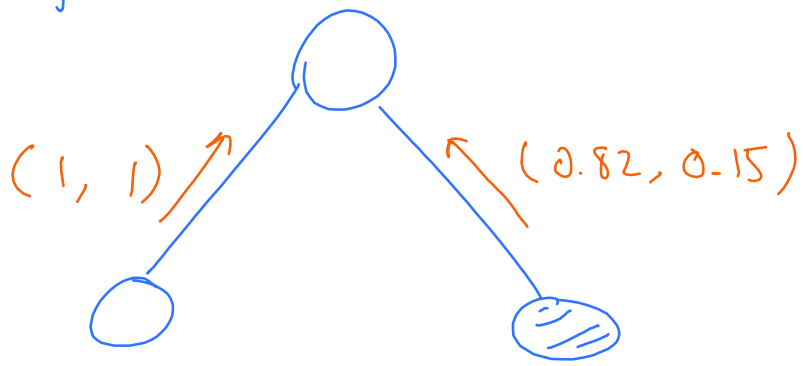
$$\mu_{BT}(T=0) = \sum_{B\in\{0\}} \psi(B, T=0) = 0.82$$

$$\mu_{BT}(T=1) = \sum_{B\in\{0\}} \psi(B, T=1) = 0.15$$

$$\psi(A, T=0) = 0.9 + 0.1 = 1$$

$$\mu_{AT}(T=0) = \sum_{A\in\{0,1\}}$$

$$\mu_{AT}(T=1) = \sum_{A\in\{0,1\}} \psi(A, T=1) = 0.5 + 0.5 = 1$$



$$P(T \mid B=B_0) \propto \mu_{AT}(T)\, \mu_{BT}(T)\, \psi_T(T)$$

$T=0$: $1 \times 0.82 \times 0.65$

$T=1$: $1 \times 0.15 \times 0.35$

Normalization :

$$P(T=0 \mid B=0) = \frac{0.82 \times 0.65}{0.82 \times 0.65 + 0.15 \times 0.35} = 0.91$$

$$\mu_{TA}(A=0) = \sum_{T \in \{0,1\}} \mu_{BT}(T)\,\psi(T, A=0)\,\psi(T)$$

$$= 0.65 \times 0.82 \times 0.9 + 0.35 \times 0.15 \times 0.5$$

$$= 0.506$$

$$\mu_{TA}(A=1) = \sum_{T \in \{0,1\}} \mu_{BT}(T)\,\psi(T, A=1)\,\psi(T)$$

$$= 0.65 \times 0.82 \times 0.1 + 0.35 \times 0.15 \times 0.5$$

$$= 0.08$$

$$P(A \mid B=0) \propto \mu_{TA}(A)$$

$A = A_0$: 0.506

$A = A_1$: 0.08

Normalization

$$P(A=0 \mid B=0) = \frac{0.506}{0.506 + 0.08} = \frac{0.506}{0.586} = 0.863$$

$$\mu_{TB}(B=0) = \sum_{T\in\{0,1\}} \mu_{AT}(T)\,\psi(T, B=0)$$

$$= 1 \times 0.82 + 1 \times 0.15$$

$$= 0.97$$

But the only case is $B=0$, so after normalization regardless of the value of $\mu_{TB}(B=0)$, we have

$$P(B=0 \mid B=0) = 1$$

Bibliography

[1] B. Hajek, *Random Processes for Engineers*. 2014.

[2] T. T. Nguyen and S. Sanner, "Algorithms for direct 0-1 loss optimization in binary classification," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, (Atlanta, GA, USA), pp. III–1085–III–1093, JMLR.org, June 2013.