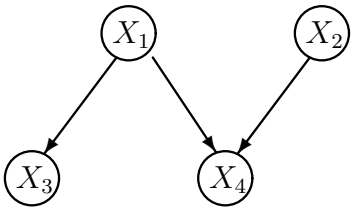
13 Efficient probabilistic inference

Poly time inference algorithms for tree-structured Bayesian networks
– marginalization, conditioning, completion

How is this done?

13.1 Example



Compute $P(X_4 = x_4)$

$$\begin{aligned} &= \sum_{x_1}\sum_{x_2}\sum_{x_3} P(X_1 = x_1, X_2 = x_2, X_3 = x_3, X_4 = x_4) \\ &= \sum_{x_1}\sum_{x_2}\sum_{x_3} P(X_1 = x_1)\ P(X_2 = x_2)\ P(X_3 = x_3|X_1 = x_1) \\ &\qquad\qquad P(X_4 = x_4|X_1 = x_1, X_2 = x_2) \end{aligned}$$

$3V^3$ multiplications, $V^3 - 1$ additions

Now think of conditional probabilities as *functions*

$$\begin{aligned} f_1(x_1) &= P(X_1 = x_1) \\ f_2(x_2) &= P(X_2 = x_2) \\ f_3(x_1, x_3) &= P(X_3 = x_3|X_1 = x_1) \\ f_4(x_1, x_2, x_4) &= P(X_4 = x_4|X_1 = x_1, X_2 = x_2) \end{aligned}$$

Then we obtain

$$
\begin{aligned}
&P(X_4 = x_4) = \\
&\quad = \sum_{x_1}\sum_{x_2}\sum_{x_3} f_1(x_1)\ f_2(x_2)\ f_3(x_1, x_3)\ f_4(x_1, x_2, x_4) \\
&\quad = \sum_{x_1} f_1(x_1) \sum_{x_3} f_3(x_1, x_3) \underbrace{\sum_{x_2} f_2(x_2)\ f_4(x_1, x_2, x_4)}_{g_2(x_1, x_4)} \\
&\quad = \sum_{x_1} f_1(x_1) \left(\sum_{x_2} f_2(x_2)\ f_4(x_1, x_2, x_4)\right) \sum_{x_3} f_3(x_1, x_3)
\end{aligned}
$$

$2V + V^2$ multiplications, $(V - 1) + 2V(V - 1)$ additions

13.2 Efficient inference in trees

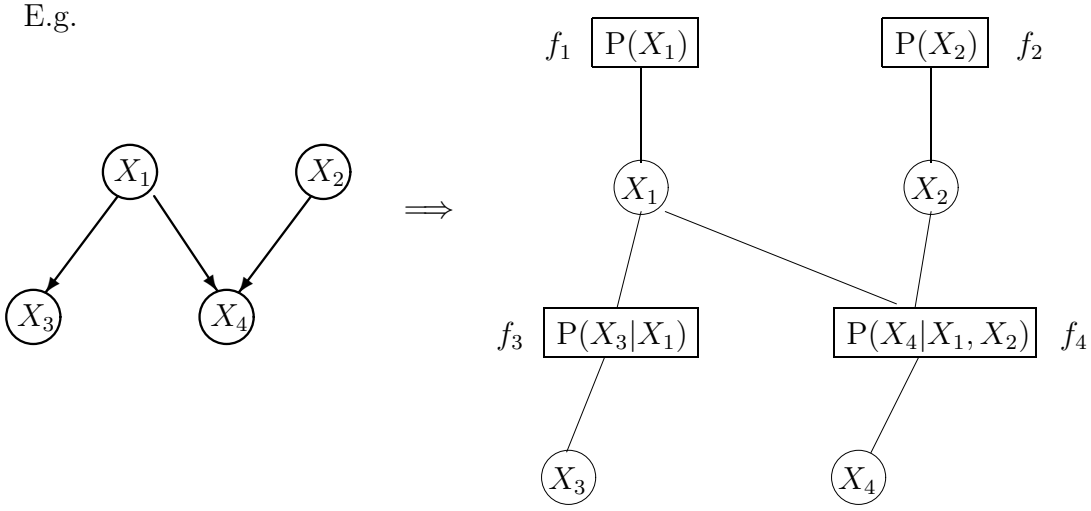
A famous algorithm achieves polynomial time inference in tree structured Bayesian networks. (This algorithm has been re-invented over a dozen times in various guises)

- hidden Markov models (forward-backward algorithm, Viterbi algorithm)
- Kalman filters
- error correcting codes
- graphical probability models
- etc.

13.3 Polynomial time marginalization algorithm for trees

Step 1 Convert the Bayesian network into a *factor graph*: an undirected graph that has one "round" node for each variable and one "square" node for each conditional probability function that is connected to each of the participating variables.

E.g.



An important property of this transformation is that if the original Bayes net is a tree then the resulting factor graph will also be a tree.

Given a factor graph with nodes $X_1, ..., X_n$ and functions $f_1, ..., f_k$, we can evaluate the probability of a complete configuration of the variables by

$$\mathrm{P}(X_1 = x_1, ..., X_n = x_n) \;\; = \;\; \prod_{\ell=1}^{k} f_\ell(\mathbf{x}_\ell)$$

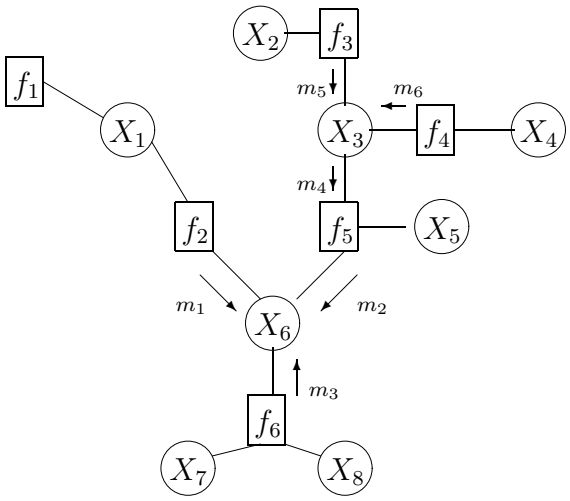
where $\mathbf{x}_\ell$ = the values of variables connected to f_ℓ.

E.g.

$$\mathrm{P}(X_1 = x_1, X_2 = x_2, X_3 = x_3, X_4 = x_4) \;\; = \;\; f_1(x_1)\; f_2(x_2)\; f_3(x_1, x_3)\; f_4(x_1, x_2, x_4)$$

Efficient marginalization for tree structured factor graphs

Consider the following tree structured factor graph and consider computing the marginal probability $\mathrm{P}(X_6 = x_6)$.



The way to think about this computation is to first note that computing the marginal probability that $\mathrm{P}(X_6 = x_6)$ requires us to sum over all possible instantiations of the other variables, but that this sum can be factored into a product of sums over variables in each independent subtree, and each of these sums can then be recursively decomposed into a product of sums over independent subtrees, and so on. So, in particular, in this case we obtain

$$
\begin{aligned}
&\mathrm{P}(X_6 = x_6) = \\
&= \sum_{x_1,x_2,x_3,x_4,x_5,x_7,x_8} f_1(x_1) f_2(x_1, x_6) f_3(x_2, x_3) f_4(x_3, x_4) f_5(x_3, x_5, x_6) f_6(x_6, x_7, x_8) \\
&= \left[\sum_{x_1} f_2(x_1, x_6) f_1(x_1) \right] \left[\sum_{x_2,x_3,x_4,x_5} f_3(x_2, x_3) f_4(x_3, x_4) f_5(x_3, x_5, x_6) \right] \\
&\quad \left[\sum_{x_7,x_8} f_6(x_6, x_7, x_8) \right] \\
&= m_1(x_6) \cdot m_2(x_6) \cdot m_3(x_6)
\end{aligned}
$$

Where now $m_2(x_6)$ further decomposes into

$$\begin{aligned} m_2(x_6) &= \sum_{x_2,x_3,x_4,x_5} f_3(x_2,x_3) f_4(x_3,x_4) f_5(x_3,x_5,x_6) \\ &= \sum_{x_3,x_5} f_5(x_3,x_5,x_6) \sum_{x_2,x_4} f_3(x_2,x_3) f_4(x_3,x_4) \\ &= \sum_{x_3,x_5} f_5(x_3,x_5,x_6) \cdot m_4(x_3) \end{aligned}$$

And $m_4(x_3)$ decomposes into

$$\begin{aligned} m_4(x_3) &= \sum_{x_2,x_4} f_3(x_2,x_3) f_4(x_3,x_4) \\ &= \left[\sum_{x_2} f_3(x_2,x_3)\right] \left[\sum_{x_4} f_4(x_3,x_4)\right] \\ &= m_5(x_3) \cdot m_6(x_3) \end{aligned}$$

And so on.

This factoring works because the sets of variables in each subtree are disjoint (in fact, this is precisely the reason why having a tree structure is significant). In general, we obtain an efficient marginalization procedure using the following message passing algorithm.

13.4 Message passing algorithm for marginalization

- Messages are vectors of real numbers $m = (u_1, ..., u_V)$, where u_k is a number that summarizes the computation for the case $x_j = k$, and of course $k = 1, ..., V$.
- Messages are passed from variable nodes to function nodes, and from function nodes to variable nodes.
- A node can send a message to its neighbor only when it has received all of the messages from its other neighbors.
- Given a tree, the algorithm can start by sending messages from each of the leaves, and stops once every node has passed a message to every neighbor. (You should convince yourself that in a tree every node will eventually send a message to every neighbor, and therefore exactly two messages will be sent across every edge (one in each direction)).

- Function to variable messages $m_{f \to X}(x)$ are computed by

$$m_{f \to X}(x) = \sum_{x_1,...,x_k} f(x, x_1, ..., x_k)\, m_{X_1 \to f}(x_1)\ \cdots m_{X_k \to f}(x_k)$$

  over all *other* variables $X_1, ..., X_k$ (besides X) connected to f. If f contains only X, then $m_{f \to X}(x) = f(x)$.

- Variable to function messages $m_{X \to f}(x)$ are computed by

$$m_{X \to f}(x) \quad = \quad \begin{cases} 1 & \text{if only } f \text{ contains } X \\ m_{f_1 \to X}(x) \cdots m_{f_k \to X}(x) & \text{otherwise} \end{cases}$$

  over all other functions $f_1, ..., f_k$ (besides f) containing X.

- Once all of the messages have been passed, then the final marginal for any variable X_i can be calculated by

$$\mathrm{P}(X_i = x_i) \quad = \quad m_{f_1 \to X_i}(x_i) \cdots m_{f_k \to X_i}(x_i)$$

  over all $f_1, ..., f_k$ containing X_j.

This algorithm is efficient because there are $n-1$ edges in an undirected tree containing n nodes (variables), $2(n-1)$ messages get sent (one in each direction along each edge), each function to variable message can be computed in time $O(V^k)$ where k is the number of function neighbors, each variable to function message can be computed in time $O(Vk)$ where k is the number of variable neighbors, and the final marginal can be computed in time $O(Vk)$. Thus, the total running time is bounded by $O(nV^k)$ where k is the maximum number of neighbors of any node in the graph. This is linear in n and polynomial in V (but exponential in k, so the maximum number of neighbors has to be bounded).

13.5 Computing the marginal of a *set*

Consider computing the marginal of one particular configuration $\mathrm{P}(X_1 = x_1, ..., X_k = x_k)$. For such a case, call the variables $X_1, ..., X_k$ *evidence variables* and call the instantiated values *observed evidence*. Then we can compute the desired probability by using the same message passing algorithm as above, except:

- An evidence-variable to function message is computed by

$$m_{X_j \to f}(x) \;=\; \begin{cases} 1 & \text{if } x = x_j \text{ (i.e. the observed evidence)} \\ 0 & \text{otherwise} \end{cases}$$

- Once all of the messages have been passed, then the final marginal can be determined by taking *any* evidence variable $X_j \in \{X_1, ..., X_k\}$ and computing

$$\mathrm{P}(X_1 = x_1, ..., X_k = x_k) \;=\; m_{f_1 \to X_j}(x_j) \cdots m_{f_k \to X_j}(x_j)$$

  over all $f_1, ..., f_k$ containing X_j.

13.6 Computing a conditional probability

To compute $\mathrm{P}(X_{k+1} = y_{k+1} | X_1 = x_1, ..., X_k = x_k)$, we can use the same message passing algorithm as above, treating $X_1, ..., X_k$ *evidence variables*, except:

- Once all of the messages have been passed, then the final conditional probability can be determined by

$$\begin{aligned} &\mathrm{P}(X_{k+1} = y_{k+1} | X_1 = x_1, ..., X_k = x_k) \\ &= \frac{m_{f_1 \to X_{k+1}}(y_{k+1}) \cdots m_{f_k \to X_{k+1}}(y_{k+1})}{Z} \end{aligned}$$

  where Z is a re-normalization constant over choices of y_{k+1}. That is,

$$Z \;=\; \sum_{y_{k+1}} m_{f_1 \to X_{k+1}}(y_{k+1}) \cdots m_{f_k \to X_{k+1}}(y_{k+1})$$

13.7 Computing the conditional probability of a *set*

To compute an arbitrary conditional probability $\mathrm{P}(\mathbf{X}_\alpha = \mathbf{y}_\alpha | \mathbf{X}_\beta = \mathbf{x}_\beta)$, where α and β are two disjoint sets of indices from $\{1, \ldots, n\}$, we can use the formula

$$\mathrm{P}(\mathbf{X}_\alpha = \mathbf{y}_\alpha | \mathbf{X}_\beta = \mathbf{x}_\beta) = \frac{\mathrm{P}(\mathbf{X}_\alpha = \mathbf{y}_\alpha, \mathbf{X}_\beta = \mathbf{x}_\beta)}{\mathrm{P}(\mathbf{X}_\beta = \mathbf{x}_\beta)}$$

and exploit the fact that we know how to calculate marginal probabilities $\mathrm{P}(\mathbf{X}_\alpha = \mathbf{y}_\alpha, \mathbf{X}_\beta = \mathbf{x}_\beta)$ and $\mathrm{P}(\mathbf{X}_\beta = \mathbf{x}_\beta)$ using the message passing algorithm outlined above.

13.8 Computing the most probable completion

To compute

$$y^*_{k+1}, ..., y^*_n = \arg \max_{y_{k+1},...,y_n} \mathrm{P}(X_{k+1}=y_{k+1}, ..., X_n=y_n | X_1=x_1, ..., X_k=x_k)$$

we can use the same message passing algorithm as above, except:

- Function to variable messages $m_{f\to X}(x)$ are computed by

$$m_{f\to X}(x) \quad = \quad \max_{x_1,...,x_k} f(x, x_1, ..., x_k) m_{X_1\to f}(x_1) \cdots m_{X_k\to f}(x_k)$$

  over all other variables $X_1, ..., X_k$ (besides X) connected to f.

- Once all of the messages have been passed, then the maximum probability completion for any free variable X_{k+j} can be calculated by

$$y^*_{k+j} \quad = \quad \arg\max_{y_{k+j}} \quad m_{f_1\to X_{k+j}}(y_{k+j}) \cdots m_{f_\ell\to X_{k+j}}(y_{k+j})$$

  over all $f_1, ..., f_\ell$ containing X_{k+j}.

Readings

Frey: Section 2.1
Dean, Allen, Aloimonos: Section 8.3
Russell and Norvig 2nd Ed: Section 14.4