# Multiplexer nodes in Pearl's algorithm

Kevin Murphy

4/9/99

In this section, we derive the equations that a multiplexer node needs to use to implement Pearl's algorithm [Pea88]. (A multiplexer node is a deterministic node with conditional distribution $P(X|M_1, \ldots, M_n, S = i) = 1.0$ is $X = M_i$ and is 0.0 otherwise, where $S \in \{1, \ldots, n\}$ is the switch node and $M_i \in \{1, \ldots, k\}$, is the $i$'th multiplexer input.) The derivation is similar in spirit to the one on Pearl p189 for noisy-or nodes. The key point is to avoid summing over all possible $O(k^n)$ parent values. Note that the marginals on the parents will be computed correctly, even though we do not add any moral edges to reflect their correlation (which would generate a clique of size $n + 2$).

We first restate the results from Pearl p183 for the case of a general discrete node $X$ with parents $U_i$ and children $Y_j$. Since the graph is singly connected, each edge $X \to Y$ partitions the evidence $e$ into an upstream $e^+_{X \to Y}$ and downstream $e^-_{X \to Y}$ component. In addition, we define $e^+_X$ to be the evidence upstream of node $X$ (i.e., all the evidence separated from $X$ if its incoming arcs were deleted), and $e^-_X$ to be the remaining evidence (including evidence in $X$ itself, if any). (Following Peot and Shachter [PS91], we include the evidence by letting a node $X$ send a message to itself, $\lambda_{X \to X}(x)$. However, we scale the messages as in Pearl, to simplify the derivation, and to prevent underflow.)

$$
\begin{aligned}
\mathrm{bel}_X(x) &\stackrel{\text{def}}{=} \Pr(X = x|e) = \alpha \lambda_X(x) \pi_X(x) \\
\lambda_X(x) &\stackrel{\text{def}}{=} P(e^-_X|X = x) = \lambda_{X \to X}(x) \times \prod_j \lambda_{Y_j \to X}(x) \\
\pi_X(x) &\stackrel{\text{def}}{=} \Pr(X = x|e^+_X) = \sum_u P(X = x|u) \prod_i \Pi_{U_i \to X}(u_i) \\
\lambda_{X \to U_i}(u_i) &\stackrel{\text{def}}{=} \Pr(e^-_{U_i \to X}|U_i = u) = \beta \sum_x \lambda_X(x) \left[ \sum_{u_k : k \neq i} P(X = x|u) \prod_{k \neq i} \Pi_{U_k \to X}(u_k) \right] \\
\Pi_{X \to Y_j}(x) &\stackrel{\text{def}}{=} \Pr(X = x|e^+_{X \to Y_j}) = \alpha \pi_X(x) \lambda_{X \to X}(x) \prod_{k \neq j} \lambda_{Y_k \to X}(x)
\end{aligned}
$$

where $\alpha$ is a normalizing constant and $\beta$ is an arbitrary constant.

In the special case where the graph is a tree, so each node has a unique parent, these equations can be simplified as follows.

$$
\begin{aligned}
\lambda_X(x) &= \lambda_{X \to X}(x) \prod_{c \in \mathrm{children}(X)} \lambda_{Y_c \to X}(x) \\
\pi_X(x) &= \sum_u P(X = x|u) \pi_U(u) \prod_{s \in \mathrm{siblings}(X)} \lambda_{X_s \to U}(u)
\end{aligned}
$$

where

$$\lambda_{Y_c \to X}(x) = \beta \sum_y P(Y_c = y | X = x) \lambda_{Y_c}(y)$$

# 1 Multiplexer node

We now derive the equations for $\pi_X(x)$ and $\lambda_{X \to U_i}$ in such a way as to avoid explicitely summing over all parent values.

Let us divide the parents into $U = (S, M_1, \ldots, M_n)$, and fix $n = 2$ for notational simplicity. We first derive $\pi_X(x)$ as follows.

$$\begin{aligned}
\pi_X(x) &= \sum_s \sum_{m_1, m_2} P(x | m_1, m_2, s) \Pi_{S \to X}(s) \Pi_{M_1 \to X}(m_1) \Pi_{M_2 \to X}(m_2) \\
&= \sum_{m_1, m_2} \Pi_{S \to X}(1) \delta(x, m_1) \Pi_{M_1 \to X}(m_1) \Pi_{M_2 \to X}(m_2) \\
&\quad + \sum_{m_1, m_2} \Pi_{S \to X}(2) \delta(x, m_2) \Pi_{M_1 \to X}(m_1) \Pi_{M_2 \to X}(m_2) \\
&= \Pi_{S \to X}(1) \Pi_{M_1 \to X}(x) \sum_{m_2} \Pi_{M_2 \to X}(m_2) + \Pi_{S \to X}(2) \sum_{m_1} \Pi_{M_1 \to X}(m_1) \Pi_{M_2 \to X}(x) \\
&= \Pi_{S \to X}(1) \Pi_{M_1 \to X}(x) + \Pi_{S \to X}(2) \Pi_{M_2 \to X}(x)
\end{aligned}$$

where we have used the fact that

$$\sum_{u_i} \Pi_{U_i \to X}(u_i) = \sum_{u_i} P(u_i | e^+_{U_i X}) = 1$$

In general we have

$$\pi_X(x) = \sum_{i=1}^n \Pi_{S \to X}(i) \Pi_{M_i \to X}(x)$$

Now let us define the vector

$$\Pi_{S \to X} = (\Pi_{S \to X}(1), \ldots, \Pi_{S \to X}(n))^T$$

and the matrix

$$\Pi_{M \to X} = \begin{pmatrix} \Pi_{M_1 \to X}(1) & \ldots & \Pi_{M_n \to X}(1) \\ \vdots & \ddots & \vdots \\ \Pi_{M_1 \to X}(k) & \ldots & \Pi_{M_n \to X}(k) \end{pmatrix}$$

Then we can restate the above result more concisely as

$$\pi_X = \Pi_{M \to X} \Pi_{S \to X}$$

To compute an expression for the lambda messages, we consider the term

$$W_i(x, u_i) = \sum_{u_k : k \neq i} P(x|u) \prod_{k \neq i} \Pi_{U_k \to X}(u_k)$$

We first consider the case where we avoid summing over a multiplexer input, say $U_i = M_j$. This is just like computing $\pi(x)$, except we don't sum over $m_j$ and exclude the term $\Pi_{M_j \to X}$. For concreteness, consider the case $j = 1$.

$$\begin{aligned} W_1(x, m) &= \sum_{m_2} \Pi_{S \to X}(1)\delta(x, m_1)\Pi_{M_2 \to X}(m_2) + \sum_{m_2} \Pi_{S \to X}(2)\delta(x, m_2)\Pi_{M_2 \to X}(m_2) \\ &= \Pi_{S \to X}(1)\delta(x, m_1) + \Pi_{S \to X}(2)\Pi_{M_2 \to X}(x) \end{aligned}$$

In general, we have

$$W_j(x, m) = \Pi_{S \to X} \cdot {\Pi_{M \to X}}^{j,x,m}$$

where $u \cdot v$ is the inner product between vectors $u$ and $v$, and ${\Pi_{M \to X}}^{j,x,m} = \Pi_{M \to X}(x, :)$ with the $j$'th element replaced by $\delta(x, m)$.

Then we have

$$\lambda_{X \to M_j}(m) = \beta \sum_x \lambda_X(x) W_j(x, m)$$

or

$$\lambda_{X \to M_j} = \beta W_j^T \lambda_X$$

Finally, we consider the case where we avoid summing over the switching gate, $U_i = S$.

$$\begin{aligned} W_S(x, s) &= \sum_{m_1, m_2} P(x|m_1, m_2, s)\Pi_{M_1 \to X}(m_1)\Pi_{M_2 \to X}(m_2) \\ &= \Pi_{M_s \to X}(x) \end{aligned}$$

Hence

$$\lambda_{X \to S} = \beta {\Pi_{M \to X}}^T \lambda_X$$

# 2 Two-dimensional multiplexer node

Now consider a node which has an $R \times C$ array of inputs and two gates, $R$ and $C$, to select which element gets passed to $X$.

Let us make the following definitions.

$$\Pi_{M \to X}(x) = \begin{pmatrix} \Pi_{M_{1,1} \to X}(x) & \dots & \Pi_{M_{1,c} \to X}(x) \\ \vdots & \ddots & \vdots \\ \Pi_{M_{R,1} \to X}(x) & \dots & \Pi_{M_{R,C} \to X}(x) \end{pmatrix}$$

$$\hat{\Pi}_{M\to X} = \begin{pmatrix} \mathrm{vec}(\Pi_{M\to X}(1))^T & \dots & \mathrm{vec}(\Pi_{M\to X}(1))^T \\ \vdots & \ddots & \vdots \\ \mathrm{vec}(\Pi_{M\to X}(k))^T & \dots & \mathrm{vec}(\Pi_{M\to X}(k))^T \end{pmatrix}$$

where $\mathrm{vec}A = (A_{1,1}, \dots, A_{R,1}, A_{1,2}, \dots, A_{R,C})^T$ is the columns of $A$ concatenated into a column vector with $n = R \times C$ elements.

$$\Pi_{R\to X}\Pi_{C\to X}{}^T = \begin{pmatrix} \Pi_{R\to X}(1)\Pi_{C\to X}(1) & \dots & \Pi_{R\to X}(1)\Pi_{C\to X}(C) \\ \vdots & \ddots & \vdots \\ \Pi_{R\to X}(R)\Pi_{C\to X}(1) & \dots & \Pi_{R\to X}(R)\Pi_{C\to X}(C) \end{pmatrix}$$

$$\hat{\Pi}_{RC\to X} = \mathrm{vec}(\Pi_{R\to X}\Pi_{C\to X}{}^T)$$

Hence we have

$$\pi_X = \hat{\Pi}_{M\to X}\hat{\Pi}_{RC\to X}$$

Now we consider how to compute the lambda messages. If we don't sum out over $M_{r,c}$, we get

$$W_{r,c}(x,m) = \hat{\Pi}_{RC\to X} \cdot \mathrm{vec}(\Pi_{M\to X}(x)^{r,c,x,m})$$

where $\Pi_{M\to X}(x)^{r,c,x,m} = \Pi_{M\to X}(x)$ with element $(r,c)$ replaced by $\delta(x,m)$.

Now suppose we don't sum out over $R$. For purposes of illustration, let $r = 1$ and $R = C = 2$.

$$\begin{aligned} W_R(x,1) &= \sum_{c=1}^{2}\sum_{m_{1,1}}\sum_{m_{1,2}}\sum_{m_{2,1}}\sum m_{2,2} P(x|m_{1,1}, m_{1,2}, m_{2,1}, m_{2,2}, R=1, C=c)\Pi_{C\to X}(c) \\ &\quad \times\Pi_{M_{1,1}\to X}(m_{1,1})\Pi_{M_{1,2}\to X}(m_{1,2})\Pi_{M_{2,1}\to X}(m_{2,1})\Pi_{M_{2,2}\to X}(m_{2,2}) \\ &= \sum_{m_{1,1}} \delta(x, m_{1,1})\Pi_{M_{1,1}\to X}(m_{1,1})\Pi_{C\to X}(1) + \sum_{m_{1,2}} \delta(x, m_{1,2})\Pi_{M_{1,2}\to X}(m_{1,2})\Pi_{C\to X}(2) \\ &= \Pi_{M_{1,1}\to X}(x)\Pi_{C\to X}(1) + \Pi_{M_{1,2}\to X}(x)\Pi_{C\to X}(2) \end{aligned}$$

In general,

$$W_R(x,r) = \Pi_{C\to X} \cdot \Pi_{M_{r,\cdot}\to X}$$

Hence

$$\lambda_{X\to R}(r) = \beta \sum_x \lambda_X(x) W_R(x,r)$$

becomes

$$\lambda_{X\to R} = \beta W_R^T \lambda_X$$

Similarly,

$$W_C(x, c) = \Pi_{R \to X} \cdot \Pi_{M_{\cdot, c} \to X}$$

and

$$\lambda_{X \to C} = \beta W_C^T \lambda_X$$

## References


[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[PS91] M. Peot and R. Shachter. Fusion and propogation with multiple observations in belief networks. *Artificial Intelligence*, 48:299–318, 1991.