Inference in Bayesian Networks with Belief Propagation

Jerod J. Weinman
weinman@grinnell.edu
Department of Computer Science
Grinnell College
Grinnell, IA 50112

Contents

1 Introduction

We have already seen how a joint probability distribution may be written quite compactly by recording only conditional probability tables and using the chain rule. How then does one make inferences (i.e., find the marginal probabilities) of arbitrary variables? How do we incorporate any variables that have been observed?

Recall the factorization of the joint probability given by the chain rule allowed us to encode the dependency among the variables as as a graph. Just as this graph structure allowed us to write the probability table more compactly, it also allows us to make inferences more quickly.

1.1 The inference task

If we have a belief network[1] with several variables, a common problem is to find the marginal probability of just one variable of particular interest. Quite often, some of the variables in the graph have also been taken as observations (evidence). In Pearl's earthquake example, we may wish to know the probability of there being an earthquake, given that John has called (but it remains unknown whether Mary called). The propositional assertion of there having been an earthquake is logically dependent on whether John called (among other things), so we cannot not simply read $P(Earthquake)$ from the CPT in the network for the answer we want. Mathematically, it is easy to determine the probability we are after by using the rules of probability; we could simply use the chain rule over all five variables to produce the joint probability and add up the probabilities for all cases of the variables except $Earthquake$ where $JohnCalled$. That doesn't seem too bad for this graph, with its 5 variables and 32 probability table entries, but if we had a dozen variables the probability table would quickly balloon to over 4000 entries. That marginal is not something you would want to calculate manually.

1.2 Inference in belief networks

Fortunately, we may take advantage of the graphical structure underlying the conditional probability tables to make the process much more efficient. Recall that the general problem is to find the probability of some variable X given the observed evidence $\mathbf{e}$ when the variables $\mathbf{Y}$ remain hidden (unobserved):

$$\mathbf{P}\left(X \mid \mathbf{e}\right) = \alpha \sum_{\mathbf{y}} \mathbf{P}\left(X, \mathbf{e}, \mathbf{y}\right). \tag{1}$$

We will consider inference tasks like this for the case of belief networks with a particular graphical structure, namely trees. This restriction means that there is only a single path between any two nodes in the graph. If we are interested in finding the probability for X, we can characterize the algorithm as propagating information through the network toward the node for X. In the next section, we begin to describe this process generally and then in some detail, before going into the actual algorithm and its mathematics in the last section. We finish with a pseudo-code representation of select parts of the algorithm.

2 Belief Propagation

2.1 The idea

The correctness of the belief propagation algorithm ultimately rests on the simple laws of probability theory and can be understood as nothing more than applications of the product and sum rules for the particular logical independence structure implied by the graphical network. However, it is very helpful to have an intuitive sense of the algorithm before delving into any details.

Everything begins with a query for the probability of a particular variable node. Because this query depends (indirectly) on the variables elsewhere in the graph, the query node asks its neighbors (both parents and children) for the information they have (i.e., biases for or against) about the query node. Because the query node's neighbors only have a small amount of information locally (e.g., the CPTs), they must in turn ask their own neighbors (aside from the query node) for information. This additional information can then be combined with the local CPT stored at the neighboring node and returned to the query in the form of a message. Thus, the query begins a recursive process of message passing that has information being sent from all throughout the network back to the query node. This recursive process can terminate in a root node (no parents), a leaf node (no children), or an evidence node (which has been observed).

[1] Note that a belief network implies a probability distribution.

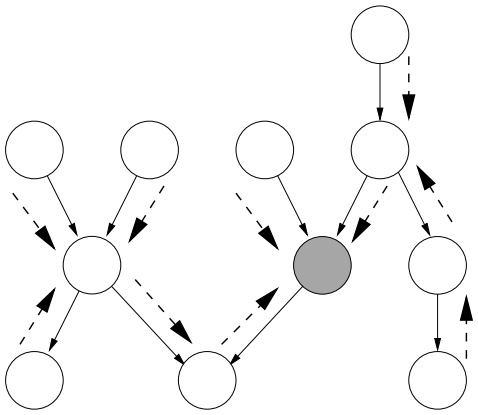

Figure 1: Flow of information through a graph with belief propagation. Messages (shown as dashed lines) are passed from every other node in the polytree toward the query node (shown in grey).

Figure 2: A simple two-node graph.

Thus, nearly every other node is gathering information from its neighbors and passing its summarized information toward the query (see Figure (1)). It should be noted that depending on the relation between the query node and this other node, that information may be passed from parent to child or from child to parent, whichever direction is toward the query. The type of message being passed will depend on this because the CPT encodes directional relations among a child and its parents. Next, we will try to gain some intuition about the forms of these messages.

2.2 Message basics

Let us begin by considering a simple two-node graph, shown in Figure 2, that may be part of a larger network. Thus, X_1 may have parents and X_2 may have children, but we ignore any other nodes for the time being.

2.2.1 Parent to child messages

If X_2 is the query node, what information might it need from X_1? Assuming that the variables are binary, we first observe the marginalization calculation made possible by the product and sum rules,

$$\mathbf{P}(X_2) = \mathbf{P}(X_2 \mid x_1) P(x_1) + \mathbf{P}(X_2 \mid \neg x_1) P(\neg x_1). \tag{2}$$

In this case, the information needed by X_2 would be the marginal for X_1, $\mathbf{P}(X_1)$. If X_1 has no parents, then this "marginal" would be given directly by the CPT stored at X_1. However, if these two nodes are part of a larger graph, and X_1 has parents, the CPT will depend on even more variables and it alone is not sufficient. Instead, a complete marginal for X_1 must be calculated. Computing the marginal for X_1 from its parents is just a repetition of the same type of problem we currently face for X_2. Rather than continue this process

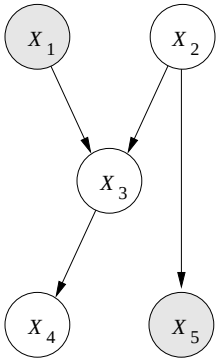

Figure 3: A simple graph with shaded nodes X_1 and X_5 observed.

explicitly now, we will simply say that the complete information from parents to a child must be captured in what we call $\boldsymbol{\pi}$ messages. These messages capture our belief about the value of the parent, and are akin to *prior* probabilities for the conditioning used in the marginalization of Eq. (2).

2.2.2 Child to parent messages

What if we wished to reason in the reverse direction? Let us assume that $X_2 = x_2$ has been observed as evidence. How do we find $\mathbf{P}(X_1 \mid x_2)$? Because we know the CPT for X_2 given X_1 , we may use Bayes rule to write the query in terms of the values we have directly,

$$\mathbf{P}(X_1 \mid x_2) \propto \mathbf{P}(x_2 \mid X_1)\,\mathbf{P}(X_1)\,. \tag{3}$$

In this case, the information needed by X_1 is the CPT values $\mathbf{P}(x_2 \mid X_1)$, which we typically think of as being "stored" at the node X_2. Sending such information from children to a parent is captured in what we call $\boldsymbol{\lambda}$ messages. These messages capture information from the child relevent to the belief about the value of the parent, and are akin to the *likelihood* term for the application of Bayes rule used to answer the query.

2.2.3 Considering the evidence

It is also important for us to see what role the evidence plays in our calculations. Because our graph is a tree, any observed variables will separate the network into two parts, relative to the query. Evidence variables that are descendants of the query may be considered "symptoms" and are therefore referred to as **diagnostic** evidence. Information about these variables arrive at the query through the "upward" $\boldsymbol{\lambda}$ (i.e., *likelihood*) messages. Other evidence variables (non-descendants of the query) are related to causes and are therefore termed **causal** evidence. Information about these arrive at the query via the "downward" $\boldsymbol{\pi}$ (i.e., *prior*) messages.

Consider the graph shown in Figure 3. Because X_1 and X_5 have been observed, our "belief" about these values changes. For instance, it is *as if* the CPTs for the two variables became

X_1
$P(X_1)$
1.00

X_5	
X_2	$P(X_5 \mid X_2)$
T	1.00
F	1.00

to reflect our newfound certainty about these values. Of course, the CPTs do not actually change, only our current beliefs about the values of the evidence variables. This update in beliefs can be accounted for in the messages sent by such evidence variables.

2.2.4 Collecting the information

While we have not yet stated how the repeated (recursive) information collection process is actually conducted, we have said just enough to figure out how to calculate the query $\mathbf{P}(X \mid \mathbf{e})$ as noted in Eq. (1). First, we separate our evidence observations $\mathbf{e}$ into those that are diagnostic, $\mathbf{e}^-$, and those that are causal, $\mathbf{e}^+$. We then rewrite our query using Bayes rule and continue with a few other transformations, which we subsequently explain,

$$\begin{aligned}
\mathbf{P}(X \mid \mathbf{e}) &= \mathbf{P}\left(X \mid \mathbf{e}^-, \mathbf{e}^+\right) && (4)\\
&\propto \mathbf{P}\left(\mathbf{e}^-, \mathbf{e}^+ \mid X\right)\mathbf{P}(X) && (5)\\
&= \mathbf{P}\left(\mathbf{e}^- \mid X\right)\mathbf{P}\left(\mathbf{e}^+ \mid X\right)\mathbf{P}(X) && (6)\\
&\propto \mathbf{P}\left(\mathbf{e}^- \mid X\right)\mathbf{P}\left(X \mid \mathbf{e}^+\right) && (7)\\
&= \boldsymbol{\lambda}\Big(X\Big)\boldsymbol{\pi}\Big(X\Big). && (8)
\end{aligned}$$

So we have it that a query probability is simply proportional to a product of the messages we have been describing—the likelihood message $\boldsymbol{\lambda}$ and the prior message $\boldsymbol{\pi}$, just as Bayes rule factors into the likelihood and the prior probabilities.

Before moving on to the details of how these messages are calculated, it is important to understand how this fact was established. In Eq. (5), we simply used Bayes rule, dropping the denominator $\mathbf{P}(\mathbf{e}^-, \mathbf{e}^+)$, which is why the equality changes to a proportion. Next is perhaps the most crucial step. In Eq. (6), we use the fact that the query variable X separates the evidence. Given X, the causal evidence $\mathbf{e}^+$ and the diagnostic evidence $\mathbf{e}^-$ are conditionally independent. Thus in Eq. (6) we write their joint probability given X as the product of two separate probabilities. From here, we use Bayes rule to reverse the query and the causal evidence in Eq. (7),

$$\mathbf{P}\left(\mathbf{e}^+ \mid X\right) = \frac{\mathbf{P}(X \mid \mathbf{e}^+)}{\mathbf{P}(X)}\mathbf{P}\left(\mathbf{e}^+\right), \tag{9}$$

allowing us to cancel the $\mathbf{P}(X)$ term and drop the $P(\mathbf{e}^+)$, which is the reason for the proportion.

The product given in Eq. (8) is a pointwise product. Each message is a tuple over the variable indicated (in this case X), and the pointwise product multiplies the respective entries in each tuple to yield a new tuple. For instance, if X could take on three values, we might have

$$\begin{aligned}
\boldsymbol{\lambda}(X) &= \left\langle \frac{1}{4}, \frac{1}{2}, \frac{1}{4} \right\rangle\\
\boldsymbol{\pi}(X) &= \left\langle \frac{1}{8}, \frac{1}{4}, \frac{5}{8} \right\rangle\\
\boldsymbol{\lambda}(X)\,\boldsymbol{\pi}(X) &= \left\langle \frac{1}{4}\cdot\frac{1}{8}, \frac{1}{2}\cdot\frac{1}{4}, \frac{1}{4}\cdot\frac{5}{8} \right\rangle\\
&= \left\langle \frac{1}{32}, \frac{1}{8}, \frac{5}{32} \right\rangle.
\end{aligned}$$

The resulting pointwise product must then be normalized to find the probability distribution $\mathbf{P}(X \mid \mathbf{e})$. With the normalization constant $\frac{1}{\alpha} = \frac{1}{32} + \frac{1}{8} + \frac{5}{32} = \frac{5}{16}$, we would have

$$\begin{aligned}
\mathbf{P}(X \mid \mathbf{e}) &= \alpha\boldsymbol{\lambda}(X)\,\boldsymbol{\pi}(X)\\
&= \frac{16}{5}\left\langle \frac{1}{32}, \frac{1}{8}, \frac{5}{32} \right\rangle\\
&= \left\langle \frac{1}{10}, \frac{4}{10}, \frac{5}{10} \right\rangle.
\end{aligned}$$

Message	Description
$\boldsymbol{\lambda}_X(X)$	Message from all the children of X
$\boldsymbol{\pi}_X(X)$	Message from all the parents of X
$\boldsymbol{\lambda}_{Y\to X}(X)$	Message from Y to its parent X
$\boldsymbol{\pi}_{U\to X}(U)$	Message from U to its child X

Table 1: Summary of the four messages types for belief propagation.

As we have seen, the "likelihood" $\boldsymbol{\lambda}(X)$ message comes from the children of X and thus depends on the diagnostic evidence, while the "prior" $\boldsymbol{\pi}(X)$ message comes from the parents of X and thus depends on the causal evidence. In the next section we see exactly how these messages are calculated.

3 Calculating Messages

As we have already said, performing inference is a recursive process of nodes asking neighbors for messages. These messages may be sent in one of two directions: from a parent to child or vice-versa. In all cases, we take a "recipient-centric" view of our messages, as it makes understanding (and perhaps implementation) easier. Toward this end, we make the recipient clear with a subscript on all messages, e.g., $\boldsymbol{\lambda}_X(X)$ and $\boldsymbol{\pi}_X(X)$, in addition to the parenthetical functional arguments that indicate the variable that the message describes.

The summary messages incorporate the information from either all the node's children—the "likelihood" $\boldsymbol{\lambda}_X(X)$—or all of its parents– the "prior" $\boldsymbol{\pi}_X(X)$. In other cases, we will need messages from an individual node to another. The message from a child node Y to its parent X, $\boldsymbol{\lambda}_{Y\to X}(X)$, passes along likelihood information from the child about the parent node's value. Recall in our earlier example (Section 2.2.2) that this constituted the idea of propagating upward the child's CPT, which is the parent's likelihood. Conversely, when a child has a CPT that requires knowing the value of a parent node, the parent must propagate downward prior information about the parent's values. Thus, the message from a parent node U to a child X, $\boldsymbol{\pi}_{U\to X}(U)$, passes along prior information about the parent to the child.

All four of these messages are summarized in Table 1. Next, we describe in more detail how all four of these messages are calculated.

3.1 Summary messages

3.1.1 Message from children

Computing the summary likelihood message from all the children is perhaps the easiest calculation of all. We simply take the product of all the messages from the individual children. Let $\{Y_i\}$ represent the set of children of a node X. Our summary message to a parent X from its children is given by

$$\boldsymbol{\lambda}_X(X) = \prod_i \boldsymbol{\lambda}_{Y_i\to X}(X). \tag{10}$$

Recall that the message from an individual child Y_i is relaying the likelihood for the parent X based on all the diagnostic evidence and any CPTs in that child's part of the graph. All the children are relaying what amounts to independent information from different parts of the graph; we simply agglomerate these as a product of independent information sources. This calculation is illustrated in Figure 4. If a node has no children, we may simply use a message consisting entirely of ones to indicate that there is no bias in likelihood one way or the other for a particular value of X.

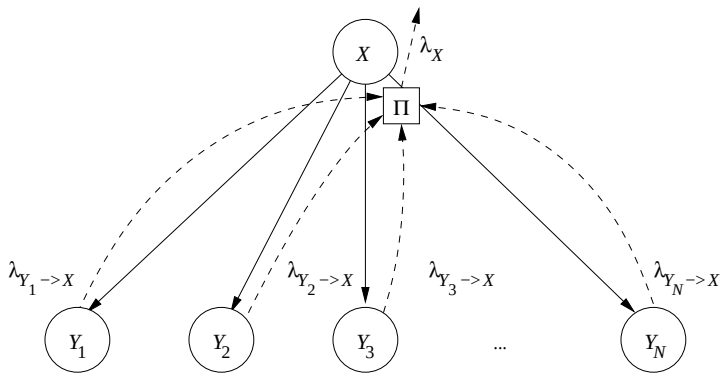

Figure 4: Calculating the summary message from a node's children.

Using Pearl's earthquake network as an example (see AIMA Figure 14.2, p. 512), we calculate the summary message from the children of $Alarm$. In the next section we will see how to compute the individual messages; for now we will take them as given. Using single letter names for the variables, the individual messages are

$$\boldsymbol{\lambda}_{J\rightarrow A}(A) = \left\langle \frac{1}{2}, \frac{1}{2} \right\rangle \tag{11}$$

$$\boldsymbol{\lambda}_{M\rightarrow A}(A) = \left\langle \frac{1}{2}, \frac{1}{2} \right\rangle. \tag{12}$$

Their pointwise product is then given by

$$\begin{aligned} \boldsymbol{\lambda}_A(A) &= \boldsymbol{\lambda}_{J\rightarrow A}(A)\,\boldsymbol{\lambda}_{M\rightarrow A}(A) \\ &= \left\langle \frac{1}{2}, \frac{1}{2} \right\rangle \times \left\langle \frac{1}{2}, \frac{1}{2} \right\rangle = \left\langle \frac{1}{4}, \frac{1}{4} \right\rangle, \end{aligned} \tag{13}$$

where we have used the operator $\times$ to indicate the pointwise multiplication of two tuples.

In practice, the message is normalized like a probability distribution (so that it sums to unity over the range of X). This helps us avoid numerical underflow, because repeated the multiplication of values less than one will produce smaller and smaller results. Because the final calculation is itself normalized as well (Eq. (8)), this does not affect the correctness of the final results.

Section 4 describes a simple pseudo-code implementation of the general computation.

3.1.2 Message from parents

Computing the summary message from parents is slightly more complicated, but simply reflects the rules of probability. Let $\{U_j\}$ represent the set of parents of a node X, then our summary message to the child X from its parents is given by

$$\boldsymbol{\pi}_X(X) = \sum_{\mathbf{u}} \mathbf{P}(X \mid \mathbf{u}) \prod_j \pi_{U_j\rightarrow X}(u_j). \tag{14}$$

What does this calculation say? Let us work through this equation from the right side to the left. If we think of the individual summary messages from each parent as the prior probability distribution for that single parent variable and assume that all the parents are independent of each other, then the product over these messages—$\prod_j \pi_{U_j\rightarrow X}(u_j)$—is simply the joint prior probability of all the parent nodes. When we multiply

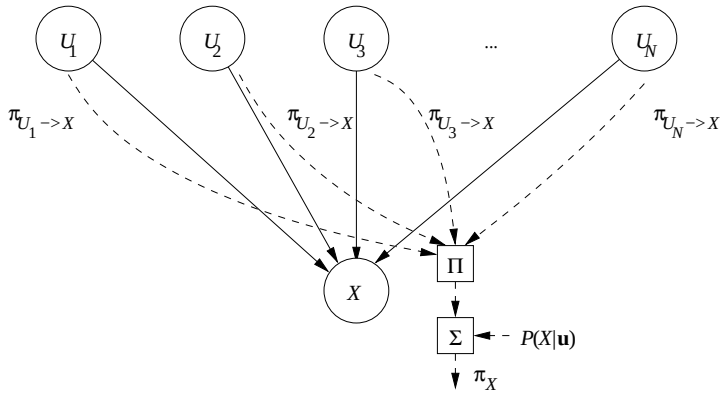

Figure 5: Calculating the summary message from a node's parents.

this prior probability of the parent nodes by the CPT of the child node given the parents—$\mathbf{P}(X \mid \mathbf{u})$—then by the product rule we have the joint probability of the child and all the parents. Finally, if we then marginalize out all the parent nodes (the sum in our expression), we are left with something like the marginal probability for the child node X, where all the information from the non-descendants has been integrated. One omission must be noted: we haven't incorporated any information from the children of X, so the resulting value is not the true marginal, but it is something marginal-like, using information from all the non-descendants of X. If a node has no parents, then the summary message is simply the prior probability table for that node.

Once again taking the earthquake network as an example, assume we have the individual messages from $Burglary$ and $Earthquake$, the parents of $Alarm$,

$$\boldsymbol{\pi}_{B\rightarrow A}(B) = \left\langle \frac{1}{1000}, \frac{999}{1000} \right\rangle \tag{15}$$

$$\boldsymbol{\pi}_{E\rightarrow A}(E) = \left\langle \frac{2}{1000}, \frac{998}{1000} \right\rangle \tag{16}$$

We must then sum over all four joint values of these two parents, multiplying the appropriate entry in the messages by the CPT for $Alarm$,

$$\begin{aligned}
\boldsymbol{\pi}_A(A) &= \mathbf{P}(A \mid b, e)\,\pi_{B\rightarrow A}(b)\,\pi_{E\rightarrow A}(e) + \mathbf{P}(A \mid b, \neg e)\,\pi_{B\rightarrow A}(b)\,\pi_{E\rightarrow A}(\neg e) + \\
&\quad \mathbf{P}(A \mid \neg b, e)\,\pi_{B\rightarrow A}(\neg b)\,\pi_{E\rightarrow A}(e) + \mathbf{P}(A \mid \neg b, \neg e)\,\pi_{B\rightarrow A}(\neg b)\,\pi_{E\rightarrow A}(\neg e) \\
&= \langle 0.95, 0.05\rangle \frac{2}{1000000} + \langle 0.94, 0.06\rangle \frac{998}{1000000} + \\
&\quad \langle 0.29, 0.71\rangle \frac{1998}{1000000} + \langle 0.001, 0.999\rangle \frac{997002}{1000000} \\
&\approx \langle 0.002516, 0.997484\rangle .
\end{aligned} \tag{17}$$

The combined information from the parents thus indicates that an alarm is exceedingly unlikely.

3.2 Individual Messages

As the definitions of the summary messages in Eqs. (10) and (14) indicate, the individual messages are at the heart of the algorithm. This is where we see information flowing from an arbitrary node back toward the query. Whenever a node needs to send a message to a child, it simply gathers the summary message from all of its parents and the individual messages from all children (*except* the recipient) and combines

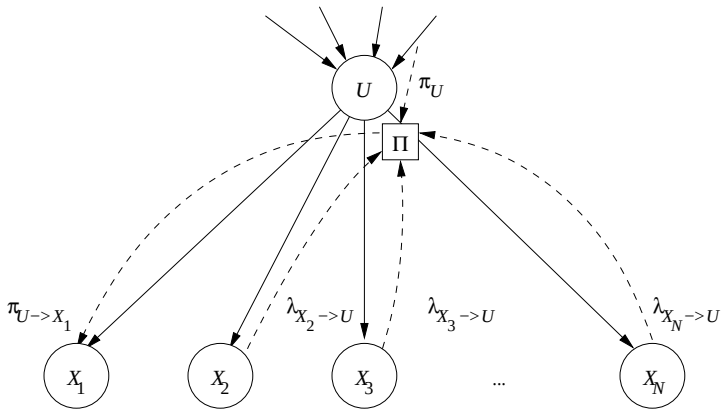

Figure 6: Calculating the individual message to a child node X_1. Parent U computes the message $\boldsymbol{\pi}_U$ from its parents, multiplying it by the messages from from the other children, $X_2, X_3, \ldots, X_N$, and sending the product to X_1.

all those messages. Similarly, when a node needs to send a message to a parent, it gathers the summary message from its children and the individual messages from all parents (*except* the recipient) and combines that information.

Because our graph is a tree, each node stands between the query and some larger portion of the graph. Therefore, each node gathers all the information from that portion of the graph and then sends it back toward the query, where another node is standing by ready to gather more information from an even larger part of the graph, sending it all on toward the query. We may also think of this process in reverse, because the original query variable starts this process of information seeking. It queries its neighbors, who then query their other neighbors, who in turn query their other neighbors. The information seeking continues until the query reaches a node that has no other neighbors. Then, the process of passing the information back along the request path begins. In the implementation, the first stage is the recursion, and the second stage is the result of combining the return values of recursive calls. Now we can describe how exactly these individual messages are calculated.

3.2.1 Message to a child

We begin with the parent-to-child message, since it is the most straightforward. In order for some node X_i to receive a message from one of its parents U, the parent must do some work. Let $\{X_j\}$ represent the set of *all* the children of U, of which X_i is just one. In order to pass a message onto X_i, U must first gather the summary message from its parents, as well as the individual messages from all its *other* children, passing on the resulting product to the child X_i,

$$\boldsymbol{\pi}_{U \to X_i}(U) = \boldsymbol{\pi}_U(U) \prod_{j \neq i} \boldsymbol{\lambda}_{X_j \to U}(U). \tag{18}$$

This process is illustrated in Figure 6. Like the summary message, the individual message is simply gathering up the beliefs about the parent U from all other information sources and agglomerating them. The calculation in Eq. (18) again relates to conveying both likelihood and prior information about the parent U to its child X_i. To prevent underflow, this message also should be normalized in practice.

Let us take the individual message from $Alarm$ to $JohnCalls$ as an example. First, we must have on hand the summary parent message to $Alarm$, $\boldsymbol{\pi}_A$, which was given in Eq. (17). We also need the individual message from its other child, $MaryCalls$; this was given in Eq. (12). We then only need to take the

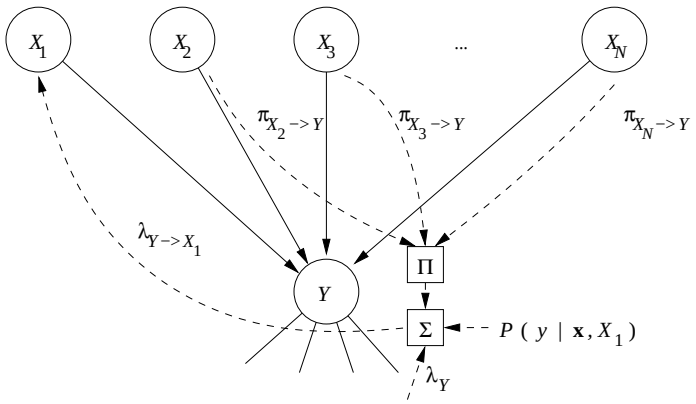

Figure 7: Calculating the individual message to a parent node X_1. Child Y multiplies the messages from its other parents, $X_2, X_3, \ldots, X_N$ as well as its CPT, marginalizing out all but X_1 and Y. The product of the marginal and the message likelihood $\boldsymbol{\lambda}_Y$ from Y's children is then further summed to eliminate Y, yielding the message to parent X_1.

pointwise product of these two messages,

$$\begin{aligned}\boldsymbol{\pi}_{A\to J}(A) &= \boldsymbol{\pi}_A(A)\,\boldsymbol{\lambda}_{M\to A}(A)\\ &= \langle 0.002516, 0.997484\rangle \cdot \left\langle \frac{1}{2}, \frac{1}{2}\right\rangle\\ &\approx \langle 0.001258, 0.498748\rangle\,.\end{aligned}$$

3.2.2 Message to a parent

Finally, we need to calculate the message to a parent X_i from a child node Y. Let $\{X_j\}$ represent the set of *all* the parents of Y, of which X_i is just one. In order to pass a message to X_i, Y must first gather the summary message from its children, as well as the individual messages from all its *other* parents, passing on the result to the parent X_i,

$$\boldsymbol{\lambda}_{Y\to X_i}(X_i) = \sum_y \lambda_Y(y) \sum_{\mathbf{x}\setminus X_i} \mathbf{P}(y \mid \mathbf{x}, X_i) \prod_{j\neq i} \pi_{X_j\to Y}(x_j)\,. \tag{19}$$

This process is illustrated in Figure 7. To understand the message to a parent, we must recognize that the factors at work are a combination of all the message interpretations we have seen thus far. Let us work our way from the right side of the equation to the left; this process will be very similar to that message from parents, Eq. (14).

The product of the other parents' messages, $\prod_{j\neq i}\pi_{X_j\to Y}(x_j)$, is like receiving prior information from those parents about their values, and these are simply multiplied together to form something like a joint prior probability over those other parents. When that prior is multiplied by the CPT for Y, $\mathbf{P}(y \mid \mathbf{x}, X_i)$, we have something like the joint probability over the variable Y and most of its parents. We then marginalize out those parents to leave something like a marginal over two variables, Y and the recipient parent X_i. The process so far is analogous to the process described in computing the summary message from parents in Eq. (14), except that we are now left with two variables instead of just one. The message from all of Y's children, $\lambda_Y(y)$, is then incorporated to ensure that we get information about Y from its descendants. In the same way that the messages from independent portions of the graph were multiplied together in Eqs. (10) and (18), we take the "marginal" over Y and X_i , which incorporates the local CPT and prior information from

the other parents of Y, and multiply that marginal prior by the likelihood information from Y's descendants. Ultimately, we do not care about Y, we merely want to pass a message to the parent X_i indicating the likelihood for X_i from this part of the graph. Therefore, we marginalize further summing over the values of Y, leaving a message over X_i alone.

We take the example of sending a message from $Alarm$ to its parent $Burglary$. This requires the summary message from all the children of $Alarm$, $\boldsymbol{\lambda}_A$, which was calculated in Eq. (13). We also need the individual message from the other parent of $Alarm$, namely $Earthquake$, $\boldsymbol{\pi}_{E\to A}$, which was given in Eq. (16). Putting these pieces together using Eq. (19), we have

$$\begin{aligned}
\boldsymbol{\lambda}_{A\to B}(B) &= \lambda_A(a)\left(\mathbf{P}(a \mid B, e)\,\pi_{E\to A}(e) + \mathbf{P}(a \mid B, \neg e)\,\pi_{E\to A}(\neg e)\right) + \\
&\quad \lambda_A(\neg a)\left(\mathbf{P}(\neg a \mid B, e)\,\pi_{E\to A}(e) + \mathbf{P}(\neg a \mid B, \neg e)\,\pi_{E\to A}(\neg e)\right) \\
&= \frac{1}{4}\left(\langle 0.95, 0.29\rangle\, 0.002 + \langle 0.94, 0.001\rangle\, 0.998\right) + \\
&\quad \frac{1}{4}\left(\langle 0.05, 0.71\rangle\, 0.002 + \langle 0.06, 0.999\rangle\, 0.998\right) \\
&= \langle 0.25, 0.25\rangle .
\end{aligned}$$

For all that work, in this case the message is conveying that, in the absence of other evidence, the $Alarm$ node has no additional information about $Burglary$ (because both values are the same).

3.3 Incorporating the evidence

A final question remains: what of the observed variables, the evidence? The answer comes into play most easily if we incorporate it at the summary message stage. The summary messages have been written in a recipient-centric fashion, so that they represent beliefs about the values of the recipient. Thus, whenever we need to calculate the message $\boldsymbol{\pi}_X(X)$ from parents to a child X for a variable X that has been observed, we simply report with certainty the value observed, which corresponds to a message with a 1 for the observed value of X and a zero everywhere else. Similarly when we calculate the message $\boldsymbol{\lambda}_X(X)$ from children to a parent X for an evidence variable, we also report with certainty the value observed for X.

We stated earlier that the process of recursively propagating query requests continues until a leaf node is reached. We can now see that this process can be short-circuited when an evidence variable is reached, because we immediately know the summary message that needs to be returned.

4 Implementations

In this section we give pseudo-code implementations of a subset of the message calculations necessary for a complete belief-propagation algorithm.

Algorithm contains the function MESSAGE-FROM-CHILDREN, for calculating the summary likelihood message $\boldsymbol{\lambda}_X(X)$ from children to a parent X. Similarly Algorithm contains the function MESSAGE-TO-CHILD, for calculating the individual message $\boldsymbol{\pi}_{U\to X}(U)$ from a parent to a child. Finally, Algorithm gives the function Compute-Beliefs, which combines MESSAGE-FROM-CHILDREN and the unspecified MESSAGE-FROM-PARENTS to answer a query $\mathbf{P}(X \mid \mathbf{e})$.

Mathematically, the process of calculating the summary message from parents $\boldsymbol{\pi}_X(X)$ is straightforward. However, the implementation of MESSAGE-FROM-PARENTS gets tricky when one must account for the fact that the "marginalizing" sums over parents for different nodes will be over different numbers of variables. Thus, simple static nested loops will not do, unless one makes restrictions on the number of parents a variable might have. Fortunately, there are ways solve this problem, but we will not cover them here.

Algorithm 1 Computing the summary message from all children for a node, $\boldsymbol{\lambda}_X(X)$.

function MESSAGE-FROM-CHILDREN(X, $\mathbf{e}$, *bn*) **returns** a message over X
  inputs: X, the query variable
    $\mathbf{e}$, observed values for variables $\mathbf{E}$
    bn, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

  if X is observed in $\mathbf{e}$ **then**
    return a message with 1 for the observed value and 0 elsewhere
  else if X has no children **then**
    return a normalized uniform message
  else
    mesgs $\leftarrow$ []
    for each Y **in** CHILDREN(X, bn) **do**
      mesgs $\leftarrow$ [MESSAGE-TO-PARENT$(Y, X, \mathbf{e}, bn)$ | *mesgs*]
    return NORMALIZE(POINTWISE-PRODUCT(*mesgs*))

Algorithm 2 Computing the message to one child from a parent node, $\boldsymbol{\pi}_{U \to X}(U)$.

function MESSAGE-TO-CHILD(U, X, $\mathbf{e}$, *bn*) **returns** a message over U
  inputs: U, the parent variable
    X, the child variable
    $\mathbf{e}$, observed values for variables $\mathbf{E}$
    bn, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

  mesgs $\leftarrow$ [MESSAGE-FROM-PARENTS$(U, \mathbf{e}, bn)$]
  for each Y **in** CHILDREN(U, bn) **do**
    if $X \neq Y$ **then** *mesgs* $\leftarrow$ [MESSAGE-TO-PARENT$(Y, U, \mathbf{e}, bn)$ | *mesgs*]

  return NORMALIZE(POINTWISE-PRODUCT(*mesgs*))

Algorithm 3 Computing the belief $\mathbf{P}(X \mid \mathbf{e})$ for a node X given evidence $\mathbf{e}$.

function COMPUTE-BELIEF(X, $\mathbf{e}$, *bn*) **returns** a distribution over X
  inputs: X, the query variable
    $\mathbf{e}$, evidence specified as an event
    bn, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

  prior $\leftarrow$ MESSAGE-FROM-PARENTS$(X, \mathbf{e}, bn)$
  likelihood $\leftarrow$ MESSAGE-FROM-CHILDREN$(X, \mathbf{e}, bn)$
  return NORMALIZE(POINTWISE-PRODUCT($[prior, likelihood]$))

5 Conclusion

The belief propagation algorithm only applies to Bayesian networks that are trees. However, the algorithm has been used in such loopy graphs to calculate *approximate* beliefs with reasonable accuracy. This is accomplished by maintaining all the messages throughout the graph all the time while iteratively updating them. This also allows information to propagate through the graph. Once an update yields no changes, the message passing has "converged" to a fixed point. Although there is no guarantee of convergence, when it does converge it seems to give reasonable answers. Such a method is called "loopy belief propagation" and is used very widely for inference in many practical (i.e., non-tree) belief networks.

Due to the form of the summary messages from parents, the algorithm is also known as "sum-product"—so called for it is a sum of products. Recognized in this form, the algorithm can be seen in other fields, such as error correcting code theory.