Project1: Belief propagation in Bayesian Network

Zeming Fang
RIN 661958922

1. Introduction to Inference

The probabilistic graphical model (PGM) is a model that describes the relationships between variables. However, in terms of observability, not all variables are created equally. It is easy to observe some variables while directly accessing to the other can be intractable. A circuitous way to gain information about the target unobservable variables is to guess based on the observed variables and the well-built relationship model (PGM). This guess is called *inference*. Usually, the inference may not precisely provide us the status of those unobservable variables. Still, it can help us narrow down the uncertainty (entropy) of the probability distribution of the target variables. A good example of inference is the human's mind-reading behavior. It is impossible to enter others' mental world to unveil what he is thinking at the current time step. Nevertheless, through others' objective behaviors and our intuitive psychology engine (a PGM that characterize correspondence between behaviors and motivation), we can confine others' possible motivations to a narrow space. The mind-reading inference is the prerequisite for efficient human social activity.

Inference is to get more insights to the unobservable variables (denoted as X_u) given the observed evidence and the relationships provided by PGM. So usually the first step of inference is to calculate the joint posterior of unknown variables given evidence (X_e), which is $p(X_u|X_e)$. However, we usually want to query a specific subset X_q rather than to know all the X_u. To characterize the inference of X_q, there are three ways:

- Posterior probability inference: it is also called sum-product method. This method calculates the conditional distribution given evidence of the query variables by marginalizing all the non-query variables X_q/X_u:

$$p(X_q|X_e) = \sum_{X_u \backslash x_q} p(x_u|X_e) \propto \sum_{X_u \backslash x_q} p(X_u, X_e)$$

As for the name sum-product, I believe the product is referred to obtaining this terms $p(X_u|X_e)$. The term is the product of different factors using the chain rule. The sum means marginal.

- Maximum a Posteriori Inference (MAP): it is also called max-sum method. This inference can be see as an extension of the sum-product inference, where we first obtain the posterior distribution $p(X_q|X_e)$ and then choose the value x_q that maximize the posterior distribution.

$$\begin{aligned} x_q^* &= \arg\max_{x_q} p(x_q|X_e) = \arg\max_{x_q} \sum_{x_u \backslash x_q} p(x_u|X_e) \\ &= \arg\max_{x_q} \sum_{x_u \backslash x_q} p(X_u, X_e) \end{aligned}$$

  The max of max-sum means $\arg\max$ operation, and the sum here means the posterior probability inference. Maybe we should call it max-sum-product.

- Most Probable Explanation Inference (MPE): the max product method. We do not focus on a specific query set in this inference, but rather on the all unknown variables. We directly take the max of $p(X_u|X_e)$, just like what we did in the MAP inference:

$$x_u^* = \arg\max_{x_u} p(x_u|X_e)$$

  The max product here is similar the max-sum-product without marginalization over all non-query variables.

In summary, though with distinct names, these three inferences share many similarities, the most significant one among which is they all heavily rely on calculating the $p(X_u|X_e)$. Based on the $p(X_u|X_e)$, we can use different way to approach $p(X_q|X_e)$ either $\sum_{X_u \backslash X_q} p(X_u|X_e)$ or $\max_{X_u \backslash X_q} p(X_u|X_e)$.

2. Belief propagation.

In section 1, we mentioned inference is to compute some unknown variables given evidence $p(X_q|X_e)$. However, when the structure of a Bayesian network include no loop, we can expand the equation:

$$\begin{aligned} p(X_q|X_e) =& p(X_q|X_e^{\pi}, X_e^{\lambda}) \\ \propto& p(X_q, X_e^{\pi}, X_e^{\lambda}) \\ =& p(X_e^{\lambda}|X_q, X_e^{\pi})p(X_q|X_e^{\pi})p(X_e^{\pi}) \\ \propto& p(X_e^{\lambda}|X_q)p(X_q|X_e^{\pi}) \\ =& \pi(X_q)\lambda(X_q) \end{aligned} \tag{1}$$

Where X_e^{π} means prior evidence (or parents' message) from the root direction, and $X^{\lambda}e$ means likelihood evidence (or children's message) from the leaves direction. The equation 1 implies, we can easily do inference once we know the $\pi(X_q)$ and $\lambda(X_q)$.

Due to the properties of the non-loopy tree structure, we may further find that both parents' and children's message can be a part of parents' and children's message of other nodes. One natural idea is to save these parents' and children's message. When we need to infer other nodes' distribution, we can directly use the cached value instead of recomputing everything from scratch.

Build upon these ideas, the Belief propagation method (BP) was formalized in 1982 (Judea, 1982). In this project, we will focus on the sum-product inference and max-product inference. Note that the BP algorithm discussed in his project is not general enough to solve all inference problems. All theories, key equations, and pseudo-codes are confined to solve two project problems.

2.1 Key equations in sum-product inference

The crux of formalizing the BP algorithm is defining how to compute $\pi(X_q)$ and $\lambda(X_q)$ based on other nodes' cached messages than from scratch. In this section, we reviews these definitions.

Start from computing parents' messages $\pi(X_q)$. The contribution of the tot parents' message $\pi(X_q)$ of each parent node $V_i \in \pi(X_q)$ should follow the CPT:

$$\pi(X_q) \quad = \sum_{V_i,\ldots,V_K} p(X_q|V_i,\ldots,V_K)\prod_{k=1}^{K}\pi_{V_k}(X_q) \tag{2}$$

Equation 2 is derived based on the idea Variables elimination (VE). To reduce the computational complexity, we can first do some summation before doing others.

$$\pi_{V_k}(X_q) = \pi(V_k)\prod_{C\in\mathrm{child}(V_k)\backslash X_q}\lambda_C(V_k) \tag{3}$$

The equation is also a result of implementing the VE algorithm. A new dilemma is that we are now continuously creating new functions without being able to terminate it. A solution to this dilemma is to continue expand these equation until the occurrence of some kind of recursive expression.

$$\lambda_C(V_k) = \sum_{c}\lambda(c)\sum_{u_1,\ldots,u_P} p(c|V_k,u_1,\ldots u_P)\prod_{p=1}^{P}\pi_{u_p}(c) \tag{4}$$

Where $U_p \in \mathrm{Parents}(C)\backslash V_k$ means the parents that not include V_k. At equation 4, we found another occurrence of a $\pi_{\mathrm{par}}(\mathrm{target})$ form, which means we can use equation 3 and equation 4 alternatively to solve $\pi(X_q)$.

The solution to the children's message is very similar to that to the parents' message. We first expand the $\lambda(X_q)$:

$$\lambda(X_q) = \prod_{j=1}^{J}\lambda_{Y_j}(X) \tag{5}$$

Equation 5 shows that the children's message can be computed by factorizing all the likelihood message from a single child. Note that the likelihood message from a single child follows the $\lambda_{\mathrm{child}}(\mathrm{target})$ form, which is equation 4. This means that we can use equation 4 and equation 3 alternatively to calculate the children's message.

Given parents' message and children's message, we can now compute the belief:

$$\begin{aligned}\mathrm{BEL}(X_q) =& p(X_q|X_e) \\ =& a\pi(X)\lambda(X) \\ =& \frac{\pi(X_q)\lambda(X_q)}{\sum_{x_q}\pi(x_q)\lambda(x_q)}\end{aligned} \tag{6}$$

We may run the BP algorithm until converges. The answer to sum-product inference is：

$$p(X_q|X_e) \quad = \mathrm{BEL}(X_q) \tag{7}$$

2.2 Key equations in max-product inference

The belief propagation for max-product inference is extremely similar to that in sum-product, due to they are all using VE idea to eliminate variables. The difference are while sum-product using sum operation to eliminate variables, the max-product use max operation.

It is easy to generalize from sum-product to the max-product inference. Thus, we merely listed the key equations without extra explanations.

- $\pi(X_q) \quad = \max_{V_i,\ldots,V_K} p(X_q|V_i,\ldots,V_K)\prod_{k=1}^{K}\pi_{V_k}(X_q)$
- $\pi_{V_k}(X_q) = \pi(V_k)\prod_{C\in\mathrm{child}(V_k)\setminus X_q}\lambda_C(V_k)$
- $\lambda(X_q) = \prod_{j=1}^{J}\lambda_{Y_j}(X)$
- $\lambda_C(X_q) = \max_{y_j}\lambda(y_i)\max_{u_1,\ldots,u_P} p(y_j|X_q,u_1,\ldots u_P)\prod_{p=1}^{P}\pi_{u_p}(y_i)$
- $\mathrm{BEL}(X_q) = a\pi(X)\lambda(X) = \frac{\pi(X_q)\lambda(X_q)}{\sum_{xq}\pi(x_q)\lambda(x_q)}$
- $x_q^* = \arg\max_{x_q}\mathrm{BEL}(X_q)$

2.3 Initialization of the algorithm

To run BP algorithm, we need to initiate π and λ message of each node.

In most of the case, the initialization of an algorithm will merely affect the speed of convergence but not its asymptotic performance. However, an inappropriate initialization of the BP algorithm will result in an incorrect inference.

One reason is that the evidence is added to the BN through initialization. The value of the evidence node should be assigned to 1 while forcing the other value as 0. For example, a variable X has value space as $\{1, 2, 3\}$. If we receive the evidence as $X = 2$, we should initiate both its $\pi(X)$ and $\lambda(X)$ as $[0, 1, 0]$ to ensure $p(X) = [0, 1, 0]$. During the belief propagation, these evidence will not be updated.

Incorrect initialization will affect BP's asymptotic performance through both the root nodes and leaf nodes of the network. The π message of the root node will not be updated during the propagation. So the pi information of the root nodes should follow the prior distribution to ensure the BN runs with correct prior knowledge. Similarly, the λ distribution of leaf nodes will also not be updated by the BP algorithm. The λ of leaf node should be initiate as a non-informative likelihood.

To guarantee correct inference, the general principle to initiate the algorithm are:

1. For nodes in evidence $X_i = e_i \in X_e$:
   - $\lambda(x_i) = 1$ where $x_i = e_i$; 0 otherwise
   - $\pi(x_i) = 1$ where $x_i = e_i$; 0 otherwise
2. For root nodes without parents:
   - $\pi(x_i) = p(x_i)$ follows the prior probabilities
   - $\lambda(x_i) \quad = 1$
3. For leaf nodes without children:
   - $\lambda(x_i) \quad = 1$ uniformly, serve as non-informative distribution.

2.4 Pseudo Code of both algorithms

Algorithm 1 Belief Propagation

```
Input A Bayesian Graph $G = \{X, E\}$,
Input A query set $X_q$
Input An evidence set $X_e = E$
Select An infer method $I$, either "sum-product" or "max-product"
for each node $X_i$ in the graph do
    if $X_i$ is evidence then
        $\lambda(x_i) = 1$ wherever $x_i = e_i$; 0 otherwise
        $\pi(x_i) = 1$ wherever $x_i = e_i$; 0 otherwise
    else if $X_i$ is root node then
        Initiate $\pi(X_i)$ as prior
    else if $X_i$ is leaf node then
        Initiate $\lambda(x_i)$ as an uniform distribution
    end if

    while "not converge" do
        for Each node $X_i$ in $X_q$ do
            Compute the $\pi_{\text{parent}}(X_i)$ message from each of its parents using the selected infer method
            Calculate the total parent message $\pi(X_i)$ using the selected infer method $I$
            Compute the $\lambda_{\text{child}}(X_i)$ message from each of its children using the selected infer method $I$
            Compute the total children $\lambda$ message using the selected infer method $I$
            Update the belief $\text{BEL}(X_i) = \lambda(X_i)\pi(X_i)$ and normalize
        end for
    end while
    if $I$ is "sum-product" method then
        for Each node $X_i$ in $X_q$ do
            Output $p(X_i|X_e = E) = \text{BEL}(X_i)$
        end for
    else if $I$ is "sum-product" method then
        for Each node $X_i$ in $X_q$ do
            Output $x_i^* = \arg\max \text{BEL}(X_i)$
        end for
    end if
```

3. Experiment setting

In this project, we are instructed to solve inference problem of a given toy non-loopy BN using BP algorithm. See figure 1 for the structure of this BN and the corresponding CPT:

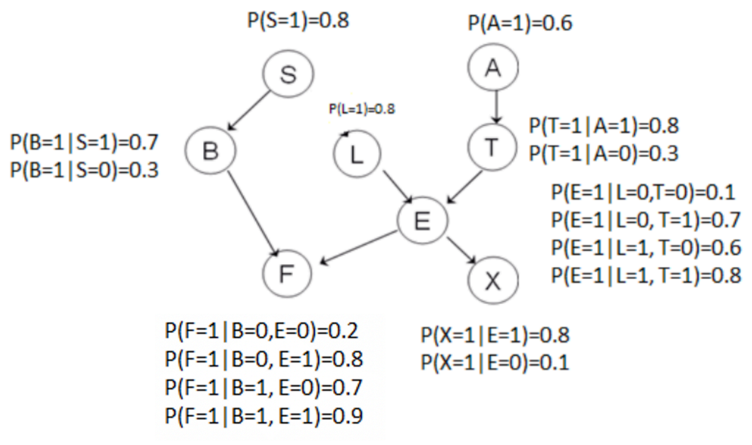

Figure1: (Adopt from project1 instruction with modifications) BN with the conditional probability tables (CPT) that declare the relationship between different nodes.

We are instructed to solve two inference problem based on this BN.

1. Posterior probability inference problem: compute $p(X_q|F=1, X=0)$, where $X_q \in \{S, A, B, L, T, E\}$ using sum-product inference.
2. Most possible explanation (MPE) inference problem: compute $s^*, a^*, l^*, t^*, e^* = \arg\max_{s,a,b,l,t,e} p(s, a, b, l, t, e|F=1, X=0)$ using max-product inference.

4. Results

In this section, we will show the performance of belief propagation algorithm on our experiment setting.

4.1 Results for sum-product algorithm

The sum-product BP algorithm is used to conduct posterior probability inference. In our experiment, we need to calculate the posterior probability of each query variable $x_q \in \{S, A, B, L, T, E\}$. The BP algorithm is very sensitive to the propagating sequence, and finding the best sequence is an NP-hard problem. For simplicity, in this project, we fix the propagation sequence as $\{S, A, B, L, T, E\}$.

To evaluate the algorithm's convergence performance, we initialized the targets' believes $p(X_q|X_e)$ as 0. After each iteration, we measured the summed L1 distance between the believes in the previous iteration and the updated believes. The belief propagation is asserted to convergence when the summed L1 distance is smaller than the tolerance $\tau = 1e - 3$.

The experiment data shows that the algorithm converged at the 4th iteration, see figure 1. Note that we forced the algorithm to run extra two iterations after it converges for the purpose of visualization.

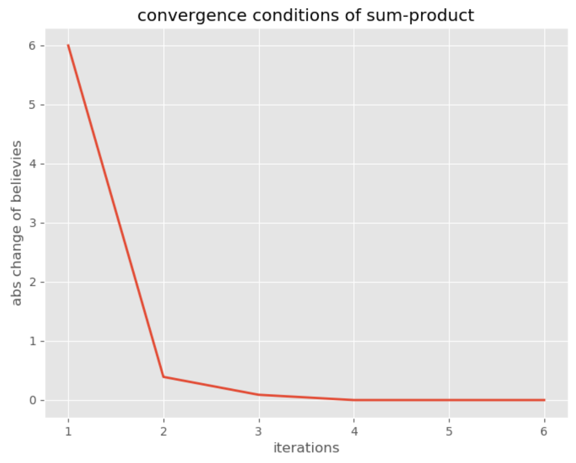

Figure 2: Convergence condition of sum-product algorithm.

Table 1 shows the converged believes the algorithm reached for each query. All values are rounded up by four digits. See appendix for the converged π and λ message for sum-product algorithm.

Table1: Output of sum-product algorithm

Query variable	$p(X_q = 0 \vert F = 1, X = 0)$	$p(X_q = 1 \vert F = 1, X = 0)$
$X_q = S$	0.1611	0.8389
$X_q = A$	0.4360	0.5640
$X_q = B$	0.2367	0.7633
$X_q = L$	0.2446	0.7554

$X_q = T$	0.4720	0.5280
$X_q = E$	0.5696	0.4304

4.1 Results for sum-product algorithm

The max-product algorithm is to find the most probable explanation for the unobserved variables. In this project the specific question is to calculate $s^*, a^*, l^*, t^*, e^* = \arg\max_{s,a,b,l,t,e} p(s,a,b,l,t,e|F=1, X=0)$. Like what we did in the sum-product algorithm, we follow the sequence $\{S, A, B, L, T, E\}$ to do inference.

Follow the same process with sum-product, we evaluated the convergence performance of max-product algorithm. Though max-product algorithm also converged at the 4th iterations, the convergence speed of max-product is slightly lower that the sum-product, see figure 2.

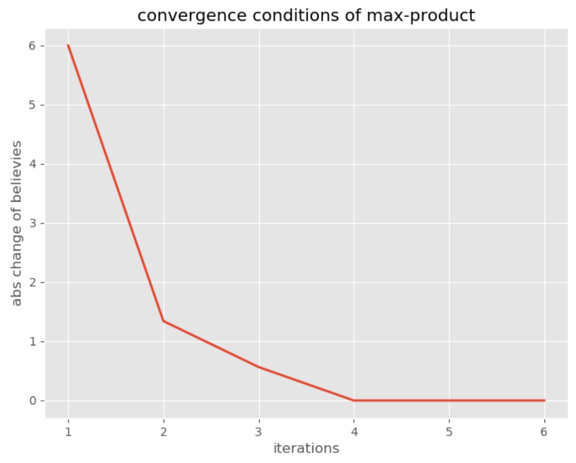

Figure 3: the convergence performance of max-product algorithm.

The output of max-product algorithm are:

$$s^* = \arg\max_s \mathrm{Bel}(S) = 1$$
$$a^* = \arg\max_a \mathrm{Bel}(A) = 0$$
$$b^* = \arg\max_b \mathrm{Bel}(B) = 1$$
$$l^* = \arg\max_l \mathrm{Bel}(L) = 1$$
$$t^* = \arg\max_t \mathrm{Bel}(T) = 0$$
$$e^* = \arg\max_e \mathrm{Bel}(E) = 0$$

See appendix for the converged π and λ messages.

5. Discussion

5.1 Difference between taking argmax of posterior and MPE.

As mentioned in the class, directly taking argmax to the result of the sum-product algorithm will not return the same value as the MPE. Sill, I wanted to have empirical experience of these. Table 2 shows the comparison between taking argmax to the sum-product output and max-product.

Table2: Comparison between $\arg\max$ to sum-product inference and max-product inference

Variable	$\arg\max$ **sum-product**	**max-product**
s^*	1	1
a^x	1	0
b^*	1	1
l^*	1	1
t^*	1	0
e^*	0	0

As table2 shows, the results obtained through these two methods do not match. The explanation of this observations is that the most possible variables returns by sum-product inference are calculated through the marginal posterior distribution of one variable while MPE is calculated by the joint distribution of all query variables

5.2 Towards a more general BP algorithm

During the BP algorithm implementation, I found that the sum-product algorithm in this project is designed only to solve the marginal posterior of one single variable, which means $|X_q| = 1$. Besides the implementation, both the key equations we walked through on the class and the pseudo-code are specific to the marginal posterior solution.

Perhaps, a more general BP algorithm requires something more. We may need to divide the query variables into groups and find these groups' parents and children to run BP algorithm.

6. Conclusions

Both sum-product and max-product BP algorithm work well in this toy experiment problem.

7. Appendix:

7.1 π and λ message after convergence

For sum-product:

Variables	$\lambda(X_q = 0)$	$\lambda(X_q = 1)$	$\lambda(X_q = 0)$	$\lambda(X_q = 1)$
S	0.2155	0.2806	0.2	0.8
A	0..2916	0.2515	0.4	0.6
B	0.1666	0.3294	0.38	0.62
L	0.3272	0.2526	0.2	0.8
T	0.3157	0.2355	0.4	0.6
E	0.4590	0.1724	0.3320	0.6680

For max-product:

Variables	$\lambda(X_q = 0)$	$\lambda(X_q = 1)$	$\lambda(X_q = 0)$	$\lambda(X_q = 1)$
S	0.0344	0.0395	0.2	0.8
A	0.0790	0.0516	0.4	0.6
B	0.0492	0.0564	0.24	0.56
L	0.0889	0.0395	0.2	0.8
T	0.1129	0.0645	0.28	0.48
E	0.3528	0.1008	0.0896	0.3072