STAT 535 Lecture 7

The Junction Tree Algorithm

©Marina Meilă
mmp@stat.washington.edu

1 Preliminaries

Denote by V the set of variables of interest. For any variable $X \in V$, denote by Ω_X the set of values that can take, by $r_X = |\Omega_X|$ the number of values and by $x \in \Omega_X$ a particular value. The same conventions extend to a set $C \subseteq V$, i.e we'll talk about $\Omega_C, r_C = |\Omega_C|, c \in \Omega_C$ a configuration of values for the variables in C. We continue to assume for now that all variables are discrete and finite valued.

Let $\{C\}$ the set of (maximal) cliques of a junction tree, and $\{S\}$ the set of separators. The general form of a probability distribution that factors according to a junction tree $(\{C\}, \{S\})$ is

$$P_V(v) \;=\; \frac{\prod_C P_C(c)}{\prod_S P_S(s)} \tag{1}$$

The implementation of the junction tree that we study here maintains a table ϕ_C to store each marginal P_C and another set of tables ϕ_S to store the separator marginals ϕ_S. We distinguish between marginal distributions P_C, P_S which are mathematical objects, and the data structure where they are stored ϕ_C, ϕ_S. The reason is that, during intermediate phases of various algorithms on the junction tree data structure, the values stored in the tables ϕ_C, ϕ_S will not always coincide with the marginal values. We may also call ϕ_C, ϕ_S potentials.

For two neighboring cliques C, C' connected by separator S, with $C \setminus S = U$, $C' \setminus S = U'$, we say that ϕ_C is *calibrated* with ϕ_S, denoted $\phi_C \sim \phi_S$ iff

$$\sum_u \phi_C(u, s) \;=\; \phi_S(s) \quad \text{for all } s \tag{2}$$

We define $\phi_C \sim \phi_{C'}$ as

$$\sum_u \phi_C(u,s) \;=\; \sum_{u'} \phi_{C'}(u',s) \quad \text{for all } s \tag{3}$$

We say that a **junction tree data structure is calibrated** if all neighboring potentials are calibrated.

We say that a potential ϕ_C (ϕ_S) is **normalized** iff $\sum_c \phi_C = 1$ ($\sum_s \phi_S = 1$). The **junction tree is normalized** iff all potentials are normalized.

Obviously, if every potential contains the corresponding clique or separator marginal, then the junction tree is normalized and calibrated.

2 Junction tree propagation

We start with a trivially simple case, which will help us establish the necessary concepts, then we go on with progressively more complicated cases.

2.1 The case of a single clique

Assume $V = C$ consists of a single clique, and that variable $E \in V$ (the evidence) is observed, i.e $E = e_0$. We would like to obtain a representation for $P_{V\setminus E|E=e_0}$ and from it to extract the updated **belief** $P_{X|E=e_0}$ about some variable $X \in V \setminus E$.

Let us pose the problem this way:

- P_V is the *prior* = what we know about the domain before we see the evidence
- $\delta_{E=e_0}(v)$ is the function on Ω_V that is 1 if $E = e_0$ in v and 0 otherwise. This represents the **likelihood**.
- By Bayes rule, the **posterior** $\propto$ prior $\times$ likelihood, i.e $P_{V|E=e_0} \propto P_V \delta_{E=e_0}$
- Also by Bayes rule, the **normalization constant** in the above is $Z = P_V(E = e_0)$ the prior probability of the evidence

The following (trivially simple!) algorithm implements the observations above.

"PROPAGATION" ALGORITHM 1

1. $\phi_C \leftarrow \phi_C \delta_{E=e_0}$	*Enter evidence*	$\phi_C = P_V$ $\phi_C \leftarrow P_{C \setminus E, E=e_0}$ $\phi_C \leftarrow P_{C \setminus E \mid E=e_0}$
2. Normalize $\phi_C \Rightarrow Z = P_V(E = e_0)$		

Now ϕ_C contains $P_{V|E=e_0}$. For any $X \in V$ we can obtain $P_{X|E=e_0}$ by marginalizing in ϕ_C.

2.2 The case of two cliques

Assume now that the junction tree over V has two cliques

$$C \;\text{———}\; \boxed{S} \;\text{———}\; C'$$

with $U = C \setminus S$, $U' = C' \setminus S$ and $E \in C$ is observed.

We have

- Prior: $P_V = \phi_C \phi_{C'} / \phi_S$
- Likelihood: $\delta_{E=e_0}(v)$
- Posterior: $P_{V|E=e_0} \propto P_V \delta_{E=e_0}$
- Normalization constant: $Z = P_V(E = e_0)$

The unnormalized posterior can be written like this:

$$P_V \delta_{E=e_0} = \frac{\delta_{E=e_0} \phi_C \phi_{C'}}{\phi_S} \tag{4}$$

$$= \frac{(\delta_{E=e_0} \phi_C) \left(\phi_{C'} \frac{\phi_S^{new}}{\phi_S} \right)}{\phi_S^{new}} \tag{5}$$

where $\phi_S^{new}(s) = \sum_u \phi_C(u, s)$ is the S-marginal of table ϕ_C. The operation corresponding to (5) is called **absorbtion**; we say that clique C' absorbs from C.

ABSORB$(C \to C')$
Input potentials $\phi_C, \phi_S, \phi_{C'}$
1. $\phi_S^{new}(s) = \sum_u \phi_C(u, s)$
2. $\phi_{C'} \leftarrow \phi_{C'} \frac{\phi_S^{new}}{\phi_S}$
3. $\phi_S \leftarrow \phi_S^{new}$

This operation has the following simple properties, whose proofs are left as an exercise. Denote by $\phi_V = \prod_C \phi_C / \prod_S \phi_S$ the "joint distribution" represented by the data structure consisting of the potential tables ϕ_C, ϕ_S.

After absorbtion:

- $\phi_C \sim \phi_S$
- ϕ_V is left invariant
- ϕ_C normalized before absobtion $\Rightarrow$ ϕ_S normalized after absorbtion
- $\phi_{C'} \sim \phi_S$ (and ϕ_C normalized) before absorbtion $\Rightarrow$ $\phi_{C'} \sim \phi_C$ (and normalized) after absorbtion

We can write now a new (still very simple) "propagation" algorithm.

"PROPAGATION" ALGORITHM 2

			$\phi_V = P_V$
1.	$\phi_C \leftarrow \phi_C \delta_{E=e_0}$	*Enter evidence*	$\phi_V \leftarrow P_V \delta_{E=e_0} = P_{V \setminus E, E=e_0}$
2.	Normalize ϕ_C		$\phi_V \leftarrow P_{V \setminus E \mid E=e_0}$
3.	ABSORB$(C \to C')$	*Distribute evidence*	ϕ_V invariant

The data structure that contained P_V before now contains $P_{V|E=e_0}$. For any $X \in V$ we can obtain $P_{X|E=e_0}$ by marginalizing in ϕ_C, for some clique C that contains X.

2.3 Any junction tree, evidence in one clique

Assume as before that the junction tree is normalized and calibrated, representing a valid P_V. We observe variable $E \in C$.

Figure 1: A junction tree.

To handle this case, we will think of the junction tree as a **rooted tree** with the root at clique C. One absorbtion from C to a child C' through separator S will make $\phi_C \sim \phi_S \sim \phi_{C'}$. It is easy to see that, if we repeat this operation recursively, the whole junction tree will be made calibrated. This procedure is called **distribute evidence**. Moreover, if ϕ_C is normalized before distributing the evidence, then the final tree will be normalized as well.

Propagation Algorithm 3

			$\phi_V = \phi_V$
1.	$\phi_C \leftarrow \phi_C \delta_{E=e_0}$	*Enter evidence*	$\phi_V \leftarrow P_V \delta_{E=e_0} = P_{V\setminus E, E=e_0}$
2.	Normalize ϕ_C	*Normalize*	$\phi_V \leftarrow P_{V\setminus E \mid E=e_0}$ $Z = P_E(E=e_0)$
3.	Recursively on the tree with root C starting with $\tilde{C} = C$ for all children C' of $\tilde{C}$ Absorb$(\tilde{C} \to C')$	*Distribute evidence*	ϕ_V invariant

After Propagation Algorithm 3 the data structure that contained P_V will contain $P_{V|E=e_0}$. For any $X \in V$ we can obtain $P_{X|E=e_0}$ by marginalizing in $\phi_{C'}$, for any clique C' that contains X.

Example Consider the junction tree in figure 1. **(a)** Assume $A = 0$ is observed.

$$
\begin{array}{rcll}
\phi_{ABC} & \leftarrow & \phi_{ABC}\delta_{A=0} & \textit{Enter evidence} \\
\text{normalize} & & \phi_{ABC} & \textit{Normalize} \\
\phi_{BC}^{new} & \leftarrow & \sum_a \phi_{ABC} & \textsc{Absorb}(ABC \to BCD) \\
\phi_{BCD} & \leftarrow & \phi_{BCD}\frac{\phi_{BC}^{new}}{\phi_{BC}} & \\
\phi_{BC}^{new} & \rightarrow & \phi_{BC} & \\
\phi_{B}^{new} & \leftarrow & \sum_{cd} \phi_{BCD} & \textsc{Absorb}(BCD \to BE) \\
\phi_{BE} & \leftarrow & \phi_{BE}\frac{\phi_{B}^{new}}{\phi_{B}} & \\
\phi_{B} & \leftarrow & \phi_{B}^{new} &
\end{array}
$$

(b) Assume now $B = 1$ is observed.

$$\begin{array}{rcll} \phi_{BCD} & \leftarrow & \phi_{BCD}\delta_{B=1} & \textit{Enter evidence} \\ \text{normalize} & & \phi_{BCD} & \textit{Normalize} \\ \phi^{new}_{BC} & \leftarrow & \sum_d \phi_{BCD} & \text{ABSORB}(BCD \to ABC) \\ \phi_{ABC} & \leftarrow & \phi_{ABC}\frac{\phi^{new}_{BC}}{\phi_{BC}} & \\ \phi^{new}_{BC} & \to & \phi_{BC} & \\ \phi^{new}_{B} & \leftarrow & \sum_{cd} \phi_{BCD} & \text{ABSORB}(BCD \to BE) \\ \phi_{BE} & \leftarrow & \phi_{BE}\frac{\phi^{new}_{B}}{\phi_{B}} & \\ \phi_{B} & \leftarrow & \phi^{new}_{B} & \end{array}$$

Exercise: Could one execute the first example, and then the other on the result of the first? What would the junction tree represent after propagations a. and b. are completed?

[**Analogy with Markov Chains - the Forward algorithm**]

2.4 Any junction tree, evidence in multiple cliques

Assume we observe variables $E_1 \in C_1$, $E_2 \in C_2, \ldots$ so that no single clique contains all of them. Let $E = \{E_1, E_2, \ldots\}$ and $e_0 = (e_{0,1}, e_{0,2}, \ldots)$ denote the set of observed variables, respectively the observed configuration. Let

$$\delta_{E=e_0}(v) \;=\; \delta_{E_1=e_{0,1}}(v)\delta_{E_2=e_{0,2}}(v)\ldots \;=\; \begin{cases} 1 & \text{if } E = e_0 \text{ in } v \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

PROPAGATION ALGORITHM 4 (=JUNCTION TREE ALGORITHM)

1. For every observed variable E_j find clique C_j that contains E_j. Set $\phi_{C_j} \leftarrow \phi_{C_j}\delta_{E_j=e_{0,j}}$ — *Enter evidence*
2. Choose a root clique C.
   Recursively on the rooted tree with root C, starting with $\tilde{C} = C$ — *Collect evidence*
   for all C' children of $\tilde{C}$
   i. $\tilde{C}$ calls *Collect evidence* in C'
   ii. ABSORB$(C' \to \tilde{C})$
3. Normalize ϕ_C and store $Z = \sum_{x_{C\setminus E}} \phi(x_C)$ — *Normalize*
4. Recursively on the rooted tree with root C, starting with $\tilde{C} = C$ — *Distribute evidence*
   for all children C' of $\tilde{C}$ ABSORB$(\tilde{C} \to C')$

After Propagation Algorithm 4 the data structure that contained P_V will contain $P_{V|E=e_0}$. For any $X \in V$ we can obtain $P_{X|E=e_0}$ by marginalizing in $\phi_{C'}$, for some clique C' that contains X.

It can be shown that at the end of algorithm 4, the junction tree obtained is normalized and calibrated. This is true even when the original junction tree (before entering evidence) is not normalized, nor calibrated. Therefore, one can use the Junction Tree algorithm to make calibrated a tree obtained from a Bayes net or MRF.

An example Consider again the junction tree in figure 1, **Observed** $A = 0, E = 1$

$$
\begin{array}{rcll}
\phi_{ABC} & \leftarrow & \phi_{ABC}\delta_{A=0} & \textit{Enter evidence} \\
\phi_{BE} & \leftarrow & \phi_{BE}\delta_{E=1} & \\
\textit{Choose} & & \textit{root}\, ABC & \textit{Collect evidence} \\
\phi_B^{new} & \leftarrow & \sum_e \phi_B E & \textit{Absorb}(BE \to BCD)\text{The “sum” has 1 term only!} \\
\phi_{BCD} & \leftarrow & \phi_{BCD}\frac{\phi_B^{new}}{\phi_B} & \\
\phi_B & \leftarrow & \phi_B^{new} & \\
\phi_{BC}^{new} & \leftarrow & \sum_d \phi_{BCD} & \text{ABSORB}(BCD \to ABC) \\
\phi_{ABC} & \leftarrow & \phi_{ABC}\frac{\phi_{BC}^{new}}{\phi_{BC}} & \\
\phi_{BC} & \leftarrow & \phi_{BC}^{new} & \\
\text{normalize} & & \phi_{ABC} & \textit{Normalize} \\
\phi_{BC}^{new} & \leftarrow & \sum_a \phi_{ABC} & \text{ABSORB}(ABC \to BCD) \\
\phi_{BCD} & \leftarrow & \phi_{BCD}\frac{\phi_{BC}^{new}}{\phi_{BC}} & \\
\phi_{BC} & \leftarrow & \phi_{BC}^{new} & \\
\phi_B^{new} & \leftarrow & \sum_{cd} \phi_{BCD} & \text{ABSORB}(BCD \to BE) \\
\phi_{BE} & \leftarrow & \phi_{BE}\frac{\phi_B^{new}}{\phi_B} & \\
\phi_B & \leftarrow & \phi_B^{new} &
\end{array}
$$

Comment on computer implementations: If one of the observed variables is in multiple cliques, then the separator potentials will contain zeor values. Since the algorithm has divisions by ϕ_S we need to convince ourselves that we are: (1) not going to divide by 0 and (2) not going to encounter zero over zero.

We can ensure this with the simple rule that: if in a clique potential $\phi_C(x_C) = 0$ for some value x_C, then, when C is absorbing from a neighbor clique, the respective x_C entry will not be updated. This is correct because if the entry is zero, multiplication with anything will leave it at zero. This is also sufficient, because: (i) the only divisions occur in this step of absorbtion, (ii) division is always by the "old" separator, (iii) if an "old" separator has $\phi_S(x_S) = 0$, then the destination clique will always have $\phi_C(x_C) = 0$ for all x_C calibrated with x_S. The latter is true because in the JT algorithm the "old" separator tables are either: (a) calibrated with the destination cliques, or (b) have ≥ 0 values for the configurations where the destination clique potential ϕ_C has a zero (if evidence was introduced in C), or (c) are 1 (as in the next section).

[**Analogy with the Forward-Backward algorithm for Markov Chains**]

3 Compiling a Bayes Net or MRF to a junction tree

Bayes net To compile a Bayes net into a junction tree that is it's I-map, we need to

Construct the junction tree structure

1. Moralize the graph, and transform it into an undirected graph. If the Bayes net is a decomposable model, no edges will be added.
2. Triangulate the graph. If the Bayes net is a decomposable model, no edges will be added. Construct the junction tree graph.

Parametrize, i.e. fill in the potentials ϕ_C, ϕ_S

3. Set $\phi_C \equiv 1$, $\phi_S \equiv 1$
4. For each $X \in V$,
   (a) choose a clique C that contains $X \cup pa(X)$
   (b) $\phi_C \leftarrow \phi_C P_{X|pa(X)}$
5. Run the Junction Tree Algorithm without entering evidence to make the junction tree calibrated

A few comments on the parametrization: In step 4, one can easily see that each factor $P_{X|pa(X)}$ is entered only once so that at the end of this step

$\prod_C \phi_C / \prod_S \phi_S = \prod_X P_{X|pa(X)}$. If there is one clique that contains its correct marginal, and if that clique is chosen as the root in step 5, then in step 5 one only needs to perform Distribute evidence, instead of the whole junction tree algorithms.

Markov random field Assume we have a MRF with cliques $\{\tilde{C}\}$ and clique potentials $\{\tilde{\phi}_{\tilde{C}}\}$

Construct the junction tree structure

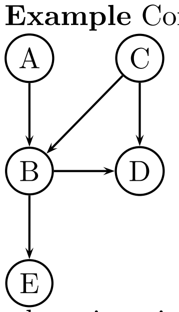
1. Triangulate the graph. If the MRF is a decomposable model, no edges will be added, and the clique potentials will be proportional to the marginals. Construct the junction tree graph.

Parametrize, i.e. fill in the potentials ϕ_C, ϕ_S

2. Set $\phi_C \equiv 1$, $\phi_S \equiv 1$
3. For each $\tilde{C}$ clique in the MRF,
   (a) choose a clique C that contains $\tilde{C}$
   (b) $\phi_C \leftarrow \phi_C \tilde{\phi}_{\tilde{C}}$
4. Run the Junction Tree Algorithm without entering evidence to make the junction tree calibrated

There are two differences w.r.t the compilation of Bayes nets: (1) no moralization is needed, and (2) there is no guarantee that any ϕ_C contains a marginal after step 3 so the junction tree algorithm in step 4 needs to both collect and distribute.

Example Consider the Bayes net



whose junction tree is depicted in figure 1. For this network, step 4 could be

$$\phi_{ABC} \leftarrow \phi_{ABC} P_A$$

$$
\begin{aligned}
\phi_{ABC} &\leftarrow \phi_{ABC} P_{B|AC} \\
\phi_{BCD} &\leftarrow \phi_{BCD} P_{C} \\
\phi_{BCD} &\leftarrow \phi_{BCD} P_{D|BC} \\
\phi_{BE} &\leftarrow \phi_{BE} P_{E|B}
\end{aligned}
$$

No clique contains its true marginal, so one must perform both a collect and a distribute step.