Graphical Models

Lecture 12: Belief Update Message Passing

Andrew McCallum
mccallum@cs.umass.edu

Thanks to Noah Smith and Carlos Guestrin for slide materials.
Today’s Plan

• Quick Review: Sum Product Message Passing (also known as “Shafer-Shenoy”)

• Today: Sum Product Divide Message Passing (also known as “belief update” message passing “Lauritzen-Spiegelhalter” and “belief propagation”)

• Mathematically equivalent, but different intuitions.

• Moving toward approximate inference.
Quick Review

• \{H, R, S, A, F\}
Message Passing (One Root)

- Input: clique tree $\mathcal{T}$, factors $\Phi$, root $C_r$
- For each clique $C_i$, calculate $\nu_i$
- While $C_r$ is still waiting on incoming messages:
  - Choose a $C_i$ that has received all of its incoming messages.
  - Calculate and send the message from $C_i$ to $C_{\text{upstream-neighbor}(i)}$:
    
    $$\delta_{i \rightarrow j} = \sum_{C_i \setminus S_{i,j}} \nu_i \prod_{k \in \text{Neighbors}_i \setminus \{j\}} \delta_{k \rightarrow i}$$
    
    $$\beta_r = \nu_r \prod_{k \in \text{Neighbors}_r} \delta_{k \rightarrow r}$$
    
    $$\sum_{X \setminus C_i} \prod_{\phi \in \Phi} \phi$$
    
    $$= Z \cdot P(C_r)$$
different roots; same messages
Sum-Product Message Passing

• Each clique tree vertex $C_i$ passes messages to each of its neighbors once it’s ready to do so.

• At the end, for all $C_i$:

$$\beta_i = \nu_i \prod_{k \in \text{Neighbors}_i} \delta_{k \rightarrow i}$$

– This is the unnormalized marginal for $C_i$. 
Calibrated Clique Tree

- Two adjacent cliques $C_i$ and $C_j$ are calibrated when:
  \[
  \sum_{C_i \setminus S_{i,j}} \beta_i = \sum_{C_j \setminus S_{i,j}} \beta_j = \mu_{i,j}(S_{i,j})
  \]
Calibrated Clique Tree as a

• Original (unnormalized) factor model and calibrated clique tree represent the same (unnormalized) measure:

\[
\prod_{\phi \in \Phi} \phi = \frac{\prod_{C \in \text{Vertices}(T)} \beta_C}{\prod_{S \in \text{Edges}(T)} \mu_S}
\]
Inventory of Factors

- original factors $\phi$
- initial potentials $\nu$
- messages $\delta$
- intermediate factors $\psi$ (no longer explicit)
- clique beliefs $\beta$
- sepset beliefs $\mu$
Inventory of Factors

- original factors $\phi$
- initial potentials $v$
- messages $\delta$
- intermediate factors $\psi$ (no longer explicit)
- clique beliefs $\beta$
- sepset beliefs $\mu$

New algorithm collapses everything into beliefs!
Another Operation: Factor Division

- $0 / 0$ is defined to be $0$
- $a / 0$ is undefined when $a > 0$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>$\phi_1(A, B, C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7000</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>500</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
<th>$\phi_2(B, C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>100</td>
</tr>
</tbody>
</table>

\[
\frac{\phi_1(A, B, C)}{\phi_2(B, C)} = \phi_3(A, B, C)
\]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>$\phi_3(A, B, C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>70</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2.5</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>
Messages

• When computing the message $\delta_{i \rightarrow j}$ from $i$ to $j$, we multiply together all incoming messages to $i$ except the one from $j$ to $i$, $\delta_{j \rightarrow i}$.

• Alternative: multiply all messages, and divide out the one from $j$ to $i$.

$$
\beta_i = \nu_i \prod_{k \in \text{Neighbors}_i} \delta_{k \rightarrow i}
$$

$$
\delta_{i \rightarrow j} = \sum_{C_{i \setminus S_{i,j}}} \nu_i \prod_{k \in \text{Neighbors}_i \setminus \{j\}} \delta_{k \rightarrow i}
$$

$$
\delta_{i \rightarrow j} = \frac{\sum_{C_{i \setminus S_{i,j}}} \beta_i}{\delta_{j \rightarrow i}}
$$
Key Idea

• We can “forget” the initial potentials \( v \).
• We do not need to calculate the messages \( \delta \) explicitly.

• Store a partially calculated \( \beta \) on each vertex and a partially calculated \( \mu \) on each edge; *update* whenever new information comes in.
A Single Belief Update

• At any point in the algorithm:

\[
\sigma_{i \rightarrow j} \leftarrow \sum_{C_i \setminus S_{i,j}} \beta_i
\]

\[
\beta_j \leftarrow \beta_j \times \frac{\sigma_{i \rightarrow j}}{\mu_{i,j}}
\]

\[
\mu_{i,j} \leftarrow \sigma_{i \rightarrow j}
\]
Belief Update Message Passing

• Maintain beliefs at each vertex ($\beta$) and edge ($\mu$).
• Initialize each $\beta_i$ to $v_i$.
• Initialize each $\mu_{i,j}$ to 1.
• Pass belief update messages.

$$\sigma_{i \rightarrow j} \leftarrow \sum_{C_i \setminus S_{i,j}} \beta_i$$

$$\beta_j \leftarrow \beta_j \times \frac{\sigma_{i \rightarrow j}}{\mu_{i,j}}$$

$$\mu_{i,j} \leftarrow \sigma_{i \rightarrow j}$$
Three Clique Example

\( \beta_1 \)  
\( C_1 = \{A, B\} \)

\( \beta_3 \)  
\( C_3 = \{C, D\} \)

\( \beta_2 \)  
\( C_2 = \{B, C\} \)

\( \mu_{1,2} \)
\( \mu_{2,3} \)

\[ \sigma_{i \rightarrow j} \leftarrow \sum_{C_i \backslash S_{i,j}} \beta_i \]
\[ \beta_j \leftarrow \beta_j \times \frac{\sigma_{i \rightarrow j}}{\mu_{i,j}} \]
\[ \mu_{i,j} \leftarrow \sigma_{i \rightarrow j} \]
Worries

• Does the order of the messages matter?

• What if we pass the same message twice?

• What if we pass a message based on partial information?
Claims

• At convergence, we will have a calibrated clique tree.

\[
\sum_{C_i \setminus S_{i,j}} \beta_i = \sum_{C_j \setminus S_{i,j}} \beta_j = \mu_{i,j}(S_{i,j})
\]

• Invariant: throughout the algorithm:

\[
\prod_{\phi \in \Phi} \phi = \prod_{C \in \text{Vertices}(T)} \nu_C = \frac{\prod_{C \in \text{Vertices}(T)} \beta_C}{\prod_{S \in \text{Edges}(T)} \mu_S}
\]
Equivalence

• Sum product message passing and sum product divide message passing lead to the same result: a calibrated clique tree.
  – *SumProduct* lets you calculate beliefs at the very end.
  – *BeliefUpdate* has beliefs from the start, and keeps them around the whole time.
Complexity

• Linear in number of cliques, total size of all factors.
• Can accomplish convergence in the same upward/downward passes we used for the earlier version.
Dealing with Evidence
Advantage: Incremental Updates

• Naively, if we have evidence, we can alter the initial potentials at the start, then calibrate using message passing.

• Better: think of evidence as a newly arrived factor into some clique tree node(s).
Example

\[ \begin{align*}
\beta_1 &= \{A, B\} \\
\beta_2 &= \{C, D\} \\
\beta_3 &= \{B, C\}
\end{align*} \]

\[ \begin{align*}
\mu_{1,2} &= \sum_{C_i \not= S_{i,j}} \beta_i \\
\mu_{i,j} &= \sigma_{i\rightarrow j} \\
\beta_j &= \beta_j \times \frac{\sigma_{i\rightarrow j}}{\mu_{i,j}} \\
D &= 1
\end{align*} \]
Advantage: Incremental Updates

• Naively, if we have evidence, we can alter the initial potentials at the start, then calibrate using message passing.

• Better: think of evidence as a newly arrived factor into some clique tree node i.

• Recalibrate: pass messages out from node i. Single pass!

• Retraction: can’t recover anything multiplied by zero.
Queries across cliques
Advantage: Queries across Cliques

• Naively: enforce that all query variables are in some clique.
  – Every query might need its own clique tree!

• Better: variable elimination in a calibrated clique tree.
  – Bonus: only have to use a subtree that includes all query variables.
Multi-Clique Queries

• Find a subtree of $T$ that includes all query variables $Q$. Call it $T'$ and its scope $S$.
• Pick a root node $r$ in $T'$.
• Run variable elimination of $S \setminus Q$ with factors (for all $I$ in $T'$):

$$\phi_i = \frac{\beta_i}{\mu_{i, \text{upstream}(i)}}$$
Example: $Z \cdot P(B, D)$

- $C_1 = \{A, B\}$
- $C_2 = \{B, C\}$
- $C_3 = \{C, D\}$
- $\beta_1$
- $\beta_2$
- $\beta_3$
- $\mu_{1,2}$
- $\mu_{2,3}$
- $D=1$
Advantage: Multiple Queries

• Suppose we want the marginal for every *pair* of variables X, Y.

• Naïve: construct a clique tree so all nodes pair together. (Very bad.)

• Naïve: run VE n-choose-2 times.

• Better: dynamic programming.
Dynamic Programming for All Pairs

• Construct a table so that $A_{i,j}$ contains $U(C_i, C_j) = Z \cdot P(C_i, C_j)$.

• Base case: $C_i$ and $C_j$ are neighboring cliques.

\[
A_{i,j} = U(C_i, C_j) \\
= U(C_j | C_i)U(C_i) \\
= \frac{\beta_j}{\mu_{i,j}} \beta_i
\]

• Proceed to farther more distant pairs recursively.
Dynamic Programming for All Pairs

- $C_i$ and $C_j$ are independent given $C_l$.
- We already have $U(C_i, C_l)$ and $U(C_l, C_j)$.

\[
A_{i,j} = U(C_i, C_j) \\
= \sum_{C_l \setminus C_j} U(C_i, C_l) U(C_j | C_l) \\
= \sum_{C_l \setminus C_j} A_{i,l} \frac{\beta_j}{\mu_{j,l}}
\]
Pros and Cons: Message Passing in Clique Trees

- Multiple queries
- Incremental updates
- Calibration operation has transparent complexity.

But:
- Complexity can be high (space!)
- Slower than VE for a single query
- Local factor structure is lost
Summary:
Message Passing in Clique Trees

• How to construct a clique tree (from VE elimination order or triangulated chordal graph)
• Marginal queries for \textit{all} variables solved in only twice the time of one query!
• Belief update version: clique potentials are reparameterized so that the clique tree invariant always holds.
• Runtime is linear in number of cliques, exponential in size of the largest clique (# variables; induced width).