Machine Learning

4771

Instructor: Tony Jebara
Topic 23

• Review: The Junction Tree Algorithm
• Collect & Distribute
• Algorithmic Complexity
• ArgMax Junction Tree Algorithm
• Loopy Propagation
• Hidden Markov Models
**Review: Junction Probabilities**

- We now have a valid Junction Tree!
- What does that mean?
- Recall probability for undirected graphs:
  \[ p(X) = p(x_1, \ldots, x_M) = \frac{1}{Z} \prod_c \psi(X_c) \]
- Can write junction tree as potentials of its cliques:
  \[ p(X) = \frac{1}{Z} \prod_c \tilde{\psi}(X_c) \]
- Alternatively: clique potentials over separator potentials:
  \[ p(X) = \frac{1}{Z} \prod_c \psi(X_c) \prod_s \phi(X_s) \]

- Running JTA converts clique potentials & separator potentials into **consistent** marginals over their variables
  \[ \psi(A, B, D) \rightarrow p(A, B, D) \]
  \[ \phi(B, D) \rightarrow p(B, D) \]
  \[ \psi(B, C, D) \rightarrow p(B, C, D) \]
Review: Junction Tree Algorithm

- Send message from each clique \textit{to} its separators of what it thinks the submarginal on the separator is.
- Normalize each clique by incoming message \textit{from} its separators so it agrees with them

\[
V = \{A, B\} \quad S = \{B\} \quad W = \{B, C\}
\]

If agree:
\[
\sum_{V \setminus S} \psi_V = \phi_s = p(S) = \phi_s = \sum_{W \setminus S} \psi_W \quad \text{...Done!}
\]

\begin{align*}
\phi_s^* &= \sum_{V \setminus S} \psi_V \\
\psi_w^* &= \frac{\phi_s^*}{\phi_s} \psi_w \\
\psi_V^* &= \psi_V
\end{align*}

Send message From V to W...

\begin{align*}
\phi_s^{**} &= \sum_{W \setminus S} \psi_w^* \\
\psi_w^{**} &= \frac{\phi_s^{**}}{\phi_s^*} \psi_w \\
\psi_V^{**} &= \psi_V
\end{align*}

Send message From W to V...

\begin{align*}
\sum_{V \setminus S} \psi_V^{**} &= \sum_{V \setminus S} \frac{\phi_s^{**}}{\phi_s^*} \psi_V \\
 &= \frac{\phi_s^{**}}{\phi_s^*} \sum_{V \setminus S} \psi_V
\end{align*}

Now they Agree...Done!

\begin{align*}
\phi_s &= \sum_{W \setminus S} \psi_w \\
\psi_w &= \psi_w \\
\psi_V &= \psi_V
\end{align*}
Review: Junction Tree Algorithm

• When “Done”, all clique potentials are marginals and all separator potentials are submarginals!
• Note that \( p(X) \) is unchanged by message passing step:

\[
\phi^*_S = \sum_{V \setminus S} \psi_V
\]
\[
\psi^*_W = \frac{\phi^*_S}{\phi_S} \psi_W
\]
\[
\psi^*_V = \psi_V
\]

\[
p(X) = \frac{1}{Z} \sum_{V} \psi^*_V \psi^*_W = \frac{1}{Z} \sum_{V} \psi_V \frac{\phi^*_S}{\phi_S} \psi_W = \frac{1}{Z} \psi_V \psi_W
\]

• Example: if potentials are poorly initialized... get corrected!

\[
\psi_{AB} = p(B \mid A) p(A)
= p(A, B)
\]

\[
\phi^*_B = \sum_A \psi_{AB} = \sum_A p(A, B) = p(B)
\]

\[
\psi^*_{BC} = \frac{\phi^*_S}{\phi_S} \psi_{BC} = \frac{p(B)}{1} p(C \mid B) = p(B, C)
\]
Junction Tree Algorithm

• Example: if evidence is observed... i.e. random var A:=1

Initialize as before...

\[ \psi_{AB} = p(A, B) \quad \psi_{BC} = p(C \mid B) \quad \phi_B = 1 \]

Update with slice...

\[ \phi_B^* = \sum_A \psi_{AB} \delta(A = 1) = \sum_A p(A, B) \delta(A = 1) = p(A = 1, B) \]

\[ \psi_{BC}^* = \frac{\phi_B^*}{\phi_S} \psi_{BC} = \frac{p(A = 1, B)}{1} p(C \mid B) = p(A = 1, B, C) \]

\[ \psi_{AB}^* = \psi_{AB} = p(A = 1, B) \]

To get conditionals...

\[ p(B, C \mid A = 1) = \frac{\psi_{BC}^*}{\sum_{B,C} \psi_{BC}^*} \]

• Problem: if send message to neighbor & he changes, we must re-update! Could keep looping for a long time.
JTA: Collect & Distribute

- Trees: recursive, no need to reiterate messages mindlessly!
- Send a message only after hearing from all neighbors...

initialize(DAG){
    Pick root
    Set all variables as:
    \[ \psi_C = p(x_i | \pi_i) \quad \forall \ i \]
    \[ \phi_S = 1 \quad \forall \ S \]
    \[ Z = 1 \]
}

collectEvidence(node) {
    for each child of node {
        update(node, collectEvidence(child));
    }
    return(node);
}

distributeEvidence(node) {
    for each child of node {
        update(child, node);
        distributeEvidence(child);
    }
}

update(node, evidence) {
    \[ \psi_C^* = \frac{\phi_S^*}{\sum_{C \neq S} \psi_C} \psi_C \]
}
Junction Tree Algorithm

- JTA: 1) Initialize  2) Collect  3) Distribute
Algorithmic Complexity

- The 5 steps of JTA are all efficient:

  **OFFLINE**
  1) Moralization
     Polynomial in # of nodes
  2) Introduce Evidence (fixed or constant)
     Polynomial in # of nodes (convert pdf to slices)
  3) Triangulate
     Suboptimal=Polynomial, Optimal=NP
  4) Construct Junction Tree
     Polynomial, Kruskal
  5) Propagate Probabilities (Junction Tree Algorithm)
     Polynomial in # of cliques, *Exponential* in Clique Size

**ONLINE** (for each query, new evidence, etc.)
**ArgMax Junction Tree Algorithm**

- We can also use JTA for finding the max not the sum over the joint to get argmax of marginals & conditionals
- Say have some evidence: \( p(X_F, \bar{X}_E) = p(x_1, \ldots, x_n, \bar{x}_{n+1}, \ldots, \bar{x}_N) \)
- Most likely (highest p) \( X_F \)? \( X_F^* = \arg \max_{X_F} p(X_F, \bar{X}_E) \)
- What is most likely state of patient with fever & headache?
  \[
p_F^* = \max_{x_2, x_3, x_4, x_5} p(x_1 = 1, x_2, x_3, x_4, x_5, x_6 = 1)
  = \max_{x_2} p(x_2 | x_1 = 1) p(x_1 = 1) \max_{x_3} p(x_3 | x_1 = 1)
  \max_{x_4} p(x_4 | x_2) \max_{x_5} p(x_5 | x_3) p(x_6 = 1 | x_2, x_5)
\]
- Solution: update in JTA uses max instead of sum:
  \[
  \phi_s^* = \max_{V \setminus S} \psi_V \quad \psi_s^* = \frac{\phi_s^*}{\phi_s} \psi_w \quad \psi_v^* = \psi_v
  \]
- Final potentials aren’t marginals: \( \psi(X_C) = \max_{U \setminus C} p(X) \)
- Highest value in potential is most likely: \( X_C^* = \arg \max_C \psi(X_C) \)
Loopy Propagation

• We *could* run junction tree algorithm on non-trees... but...
  a) no guaranteed convergence
  b) might get inexact marginals
  c) might iterate indefinitely (not polynomial time)
• Called Loopy Propagation since messages loop indefinitely
• Example: Markov random field for images...

Just find cliques
Don’t triangulate
Draw junction graph
Keep iterating JTA...
Hidden Markov Models

- A great application of Junction Tree Algorithm with EM
- So far, we have dealt with mixture models with IID:

\[
q_t \quad y_t
\]

\[
t = 0 \ldots T
\]

\[
\begin{align*}
q_0 & \quad q_1 & \quad q_2 & \quad q_3 & \quad q_4 \\
y_0 & \quad y_1 & \quad y_2 & \quad y_3 & \quad y_4
\end{align*}
\]

\[
p(q_t | \theta) = [0.5 \quad 0.2 \quad 0.3]
\]

- Variable \( q \) was a multinomial, flip a coin or roll a die
- Then sample appropriate Gaussian (or other observation)
- Maybe other mixtures, mix of multinomials, Poisson, etc.
- What if we had a dependency and not IID?
- Have a memory that keeps track of last roll of dice \( q_{t-1} \)
- Change weight of dice \( q_t \) depending on what was rolled...
- I.e. casino changes the dice if we roll \( q_{t-1}=6 \)...
- Now, order of \( y^0, \ldots, y^T \) matters (if IID order doesn’t matter)
Hidden Markov Models

- Since next roll of the dice depends on previous one...

Order of $y^0, \ldots, y^T$ matters
Temporal or sequence model!

- Add left-right arrows. This is a hidden Markov model

- Markov: \( \text{future} \parallel \text{past} \mid \text{present} \)

\[
p(q^t \mid q^{t-1}, q^{t-2}, \ldots, q^1, q^0) = p(q^t \mid q^{t-1})
\]

- From graph, have the following general pdf:

\[
p(X_U) = p(q^0) \prod_{t=1}^{T} p(q^t \mid q^{t-1}) \prod_{t=0}^{T} p(y^t \mid q^t)
\]

- So \( p(q_t) \) depends on previous outcome \( q_{t-1} \) ...

\[
p(q^t \mid q^{t-1} = 1) \quad p(q^t \mid q^{t-1} = 2) \quad p(q^t \mid q^{t-1} = 3)
\]