Lecture 8: February 9, 2015
cs 573: Probabilistic Reasoning
Professor Nevatia
Spring 2015

Review

- Assignment #2 due today
- Last lecture:
  - Conversions between BN and MN
    - Properties of chordal graphs
  - Energy functions and Log-linear models
  - Some simple models
    - Pairwise MRFs
    - Conditional random fields (CRFs)
    - Ising energy model
- Today's objective
  - Inference in graphical models
    - Variable elimination algorithm

Inference in Probabilistic Graphs

- Types of Queries
  - **Conditional probability** query P (**Y**| **E**= **e**)
  - **MAP** query, MAP (**W**|**e**) = arg $\max_w$ P(**w**, **e**)
  - We focus on the former first
- P (**Y**| **E**= **e**) = P (**Y**, **e**) / P (**e**)
- P(**y**,**e**) = $\sum_w$ P(**y**,**e**,**w**) where **W = X – Y – E**
  - Note that the quantity inside summation is just an entry in the joint distribution (may not be given explicitly but can be computed by using the chain rule)
- P (**e**) = $\sum_Y$ P(**y**,**e**)
  - Compute by summing out the joint distribution or from the calculation above
- Compute P(**y**,**e**) for various values of **Y** and then normalize to add to one (don't need to compute P (**e**) explicitly)

Worst-Case Complexity

- Solving by enumeration is exponential in the number of variables
- To decide whether $P(X = x) > 0$ is NP-hard
  - Shown by equivalence to satisfiability of a propositional logic formula
- To decide whether $P(X = x)$ is #P hard (harder than NP unless P=NP)
- Even *approximate* inference, with a ***guaranteed bound*** on the *relative error* (precise definition in the book) is NP-hard
- In practice, for moderate size networks, many queries can be answered exactly in reasonable time. Also, good approximate algorithms for larger networks.
  - Chapters 9 and 10 are about exact inference, chapters 11 and 12 about approximations

Inference on a Chain

- Study a chain first $A \rightarrow B \rightarrow C \rightarrow D$; assume CPDs are given
- First compute $P(B) = \sum_a P(a)\, P(B|a)$
  - Note, we compute the entire distribution of B
  - Let A have *k* values, B have *m* values. Consider computing probability of one value of B, say $P(b^1)$
    - Need *k* multiplications and *k-1* additions to compute
  - Repeat for each value of B, *m* times
  - Total complexity is $O(k \times m)$
- Next compute $P(C) = \sum_b P(b)\, P(C|b)$
- Then, $P(D) = \sum_c P(c)\, P(D|c)$
- Method generalizes to chain of *n* variables.
  - If each variable is k-valued, complexity is $O(nk^2)$
  - Joint distribution has k^n entries
- How about computing $P(C|d^1)$?

Another Cut: Expand the Formula

- $P(A,B,C,D) = P(A)\,P(B|A)P(C|B)P(D|C)$, by chain rule
- Sum over A, B & C, to get:

$$P(D) = \sum_C \sum_B \sum_A P(A)\,P(B|A)P(C|B)P(D|C);$$

- Compute for $D = d^1$ and $D = d^2$

$$P(d^1) = \sum_C \sum_B \sum_A P(A)\,P(B|A)P(C|B)P(d^1|C);$$

$$P(d^2) = \sum_C \sum_B \sum_A P(A)\,P(B|A)P(C|B)P(d^2|C);$$

- Now, sum over A: consider $A= a^1$ and $A= a^2$

- $P(d^1)= \sum_C \sum_B P(a^1)\,P(B|a^1)P(C|B)\,P(d^1|C) +$
  $\sum_C \sum_B P(a^2)\,P(B|a^2)P(C|B)P(d^1|C)$

Similar expression for $P(d^2)$

Expanded Formulas

P(d¹)=

$$
\begin{aligned}
& P(a^1) \quad P(b^1 \mid a^1) \quad P(c^1 \mid b^1) \quad P(d^1 \mid c^1) \\
+ \; & P(a^2) \quad P(b^1 \mid a^2) \quad P(c^1 \mid b^1) \quad P(d^1 \mid c^1) \\
+ \; & P(a^1) \quad P(b^2 \mid a^1) \quad P(c^1 \mid b^2) \quad P(d^1 \mid c^1) \\
+ \; & P(a^2) \quad P(b^2 \mid a^2) \quad P(c^1 \mid b^2) \quad P(d^1 \mid c^1) \\
+ \; & P(a^1) \quad P(b^1 \mid a^1) \quad P(c^2 \mid b^1) \quad P(d^1 \mid c^2) \\
+ \; & P(a^2) \quad P(b^1 \mid a^2) \quad P(c^2 \mid b^1) \quad P(d^1 \mid c^2) \\
+ \; & P(a^1) \quad P(b^2 \mid a^1) \quad P(c^2 \mid b^2) \quad P(d^1 \mid c^2) \\
+ \; & P(a^2) \quad P(b^2 \mid a^2) \quad P(c^2 \mid b^2) \quad P(d^1 \mid c^2)
\end{aligned}
$$

Note that the first two lines have common terms (third and fourth entries), so we do not need to compute twice; see below for factorized form.

$$
\begin{aligned}
& (P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)) \quad P(c^1 \mid b^1) \quad P(d^1 \mid c^1) \\
+ \; & (P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)) \quad P(c^1 \mid b^2) \quad P(d^1 \mid c^1) \\
+ \; & (P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)) \quad P(c^2 \mid b^1) \quad P(d^1 \mid c^2) \\
+ \; & (P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)) \quad P(c^2 \mid b^2) \quad P(d^1 \mid c^2)
\end{aligned}
$$

Similar expression for $P(d^2)$

Factorized Expressions

$$
\begin{aligned}
P(d^1)= \quad & (P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)) \quad P(c^1 \mid b^1) \quad P(d^1 \mid c^1) \\
+ \; & (P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)) \quad P(c^1 \mid b^2) \quad P(d^1 \mid c^1) \\
+ \; & (P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)) \quad P(c^2 \mid b^1) \quad P(d^1 \mid c^2) \\
+ \; & (P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)) \quad P(c^2 \mid b^2) \quad P(d^1 \mid c^2)
\end{aligned}
$$

$$
\begin{aligned}
P(d^2)= \quad & (P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)) \quad P(c^1 \mid b^1) \quad P(d^2 \mid c^1) \\
+ \; & (P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)) \quad P(c^1 \mid b^2) \quad P(d^2 \mid c^1) \\
+ \; & (P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)) \quad P(c^2 \mid b^1) \quad P(d^2 \mid c^2) \\
+ \; & (P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)) \quad P(c^2 \mid b^2) \quad P(d^2 \mid c^2)
\end{aligned}
$$

Note that sums are repeated several times: call the first sum, row1 above $\tau_1(b^1)$, second row$\tau_1(b^2)$; together $\tau_1(B)$; note that each is repeated four times. Rewritten on next slide.

In terms of τ_1 (B)

$$P(d^1) = \begin{array}{rlll} & \tau_1(b^1) & P(c^1 \mid b^1) & P(d^1 \mid c^1) \\ + & \tau_1(b^2) & P(c^1 \mid b^2) & P(d^1 \mid c^1) \\ + & \tau_1(b^1) & P(c^2 \mid b^1) & P(d^1 \mid c^2) \\ + & \tau_1(b^2) & P(c^2 \mid b^2) & P(d^1 \mid c^2) \end{array}$$

$$P(d^2) = \begin{array}{rlll} & \tau_1(b^1) & P(c^1 \mid b^1) & P(d^2 \mid c^1) \\ + & \tau_1(b^2) & P(c^1 \mid b^2) & P(d^2 \mid c^1) \\ + & \tau_1(b^1) & P(c^2 \mid b^1) & P(d^2 \mid c^2) \\ + & \tau_1(b^2) & P(c^2 \mid b^2) & P(d^2 \mid c^2) \end{array}$$

Now, factorize $P(d^1|c^1)$, $P(d^1|c^2)$, $P(d^2|c^1)$, $P(d^2|c^2)$

Factorize Again

$$P(d^1) = \begin{array}{ll} & (\tau_1(b^1)P(c^1 \mid b^1) + \tau_1(b^2)P(c^1 \mid b^2)) \quad P(d^1 \mid c^1) \\ + & (\tau_1(b^1)P(c^2 \mid b^1) + \tau_1(b^2)P(c^2 \mid b^2)) \quad P(d^1 \mid c^2) \end{array}$$

$$P(d^2) = \begin{array}{ll} & (\tau_1(b^1)P(c^1 \mid b^1) + \tau_1(b^2)P(c^1 \mid b^2)) \quad P(d^2 \mid c^1) \\ + & (\tau_1(b^1)P(c^2 \mid b^1) + \tau_1(b^2)P(c^2 \mid b^2)) \quad P(d^2 \mid c^2) \end{array}$$

Note repeated sums again; define $\tau_2(C)$ to simplify and avoid repeated computations

$$\begin{aligned} \tau_2(c^1) &= \tau_1(b^1)P(c^1 \mid b^1) + \tau_1(b^2)P(c^1 \mid b^2) \\ \tau_2(c^2) &= \tau_1(b^1)P(c^2 \mid b^1) + \tau_1(b^2)P(c^2 \mid b^2) \end{aligned}$$

In terms of τ_2 (C)

$$P(d^1) = \begin{array}{ll} & \tau_2(c^1) \quad P(d^1 \mid c^1) \\ + & \tau_2(c^2) \quad P(d^1 \mid c^2) \end{array}$$

$$P(d^2) = \begin{array}{ll} & \tau_2(c^1) \quad P(d^2 \mid c^1) \\ + & \tau_2(c^2) \quad P(d^2 \mid c^2) \end{array}$$

Note: total number of operations (for binary variables) is 18:
4 multiplies and 2 adds for each of τ_1(B), for τ_2(C) , and for P(D);
computation of joint distribution requires 48 multiplies and 14 adds.

A Numerical Example

- Let P(A) = <.6, .4>, P(B|A) = <.8, .2>, <.3, .7>, P(C|B) = <.9, .1>, <.2, .8>, P(D|C) = <.6, .4>, <.1, .9>
- $\tau_1 (b^1) = .6 \text{ x } .8 + .4 * .3 = .6, \tau_1 (b^2) = .4$
- $\tau_2 (c^1) = .6 \text{ x } .9 + .4 * .2 = .62, \tau_2 (c^2) = .38$
- *etc*.

VE Algorithm on a Chain: Compact Form

- $P(D) = \sum_C \sum_B \sum_A P(A)\ P\ (B|A)P(C|B)P(D|C)$
- Rewrite as $\sum_C P(D|C) \sum_B P(C|B) \sum_A P(A)\ P\ (B|A)$
- Let $\psi_1\ (A,B)= P(A)\ P\ (B|A)$, $\tau_1(B) = \sum_A \psi_1\ (A,B)$
- $\psi_2\ (B,C)= \tau_1(B)\ P\ (C|B)$, $\tau_2(C) = \sum_B \psi_2\ (B,C)$
- $\psi_3\ (C, D)= \tau_2(C)\ P\ (D|C)$, $P(D) = \sum_C \psi_3\ (C,D)$
- Dynamic programming: computes inner expressions first, "inside out"

Factor Marginalization

- $\psi(\mathbf{X}) = \sum_Y \phi(\mathbf{X}, Y)$; $\mathbf{X}$ is set of variables, Y is a single variable (not in set $\mathbf{X}$)

a^1	b^1	c^1	0.25
a^1	b^1	c^2	0.35
a^1	b^2	c^1	0.08
a^1	b^2	c^2	0.16
a^2	b^1	c^1	0.05
a^2	b^1	c^2	0.07
a^2	b^2	c^1	0
a^2	b^2	c^2	0
a^3	b^1	c^1	0.15
a^3	b^1	c^2	0.21
a^3	b^2	c^1	0.09
a^3	b^2	c^2	0.18

a^1	c^1	0.33
a^1	c^2	0.51
a^2	c^1	0.05
a^2	c^2	0.07
a^3	c^1	0.24
a^3	c^2	0.39

Add terms that "match up"

Factor Operations

- $\phi_1 . \phi_2 = \phi_2 . \phi_1$
- $\sum_X \sum_Y \phi = \sum_Y \sum_X \phi$
- $(\phi_1 . \phi_2) . \phi_3 = \phi_1 . (\phi_2 . \phi_3)$
- $\sum_X (\phi_1 . \phi_2) = \phi_1 . \sum_X \phi_2$, if X is not in scope[ϕ_1]

VE Algorithm (General Version)

- $P(A,B,C,D) = \phi_A \cdot \phi_B \cdot \phi_C \cdot \phi_D$;
  nodes are not necessarily arranged in a chain
- $P(D) = \sum_C \sum_B \sum_A P(A, B,C,D)$; marginalize the joint distribution
  $= \sum_C \sum_B \sum_A \phi_A \cdot \phi_B \cdot \phi_C \cdot \phi_D$; express joint as a factor product
  $= \sum_C \sum_B \phi_C \cdot \phi_D (\sum_A \phi_A \cdot \phi_B)$; sum out A
  $= \sum_C \phi_D \cdot (\sum_B \phi_C \cdot (\sum_A \phi_A \cdot \phi_B))$; sum out B then C

Note: any order of variable elimination gives correct answer.

In general, compute $\sum_Z \Pi_{\phi \varepsilon \Phi}\, \phi$; Z is the set of variables to be eliminated

Sum-Product algorithm (we have reversed the order of product and sum from the original definition)

Sum Product VE Algorithm (9.1)

Algorithm 9.1 Sum-product variable elimination algorithm

Procedure Sum-Product-VE (
  Φ, // Set of factors
  $\boldsymbol{Z}$, // Set of variables to be eliminated
  $\prec$ // Ordering on $\boldsymbol{Z}$
)

Let $Z_1, \ldots, Z_k$ be an ordering of $\boldsymbol{Z}$ such that
  $Z_i \prec Z_j$ if and only if $i < j$
for $i = 1, \ldots, k$
  $\Phi \leftarrow$ Sum-Product-Eliminate-Var(Φ, Z_i)
$\phi^* \leftarrow \prod_{\phi \in \Phi} \phi$
return ϕ^*

Procedure Sum-Product-Eliminate-Var (
  Φ, // Set of factors
  Z // Variable to be eliminated
)

$\Phi' \leftarrow \{\phi \in \Phi \,:\, Z \in Scope[\phi]\}$
$\Phi'' \leftarrow \Phi - \Phi'$
$\psi \leftarrow \prod_{\phi \in \Phi'} \phi$
$\tau \leftarrow \sum_Z \psi$
return $\Phi'' \cup \{\tau\}$

Note: A factor is used only once

Theorem 9.5

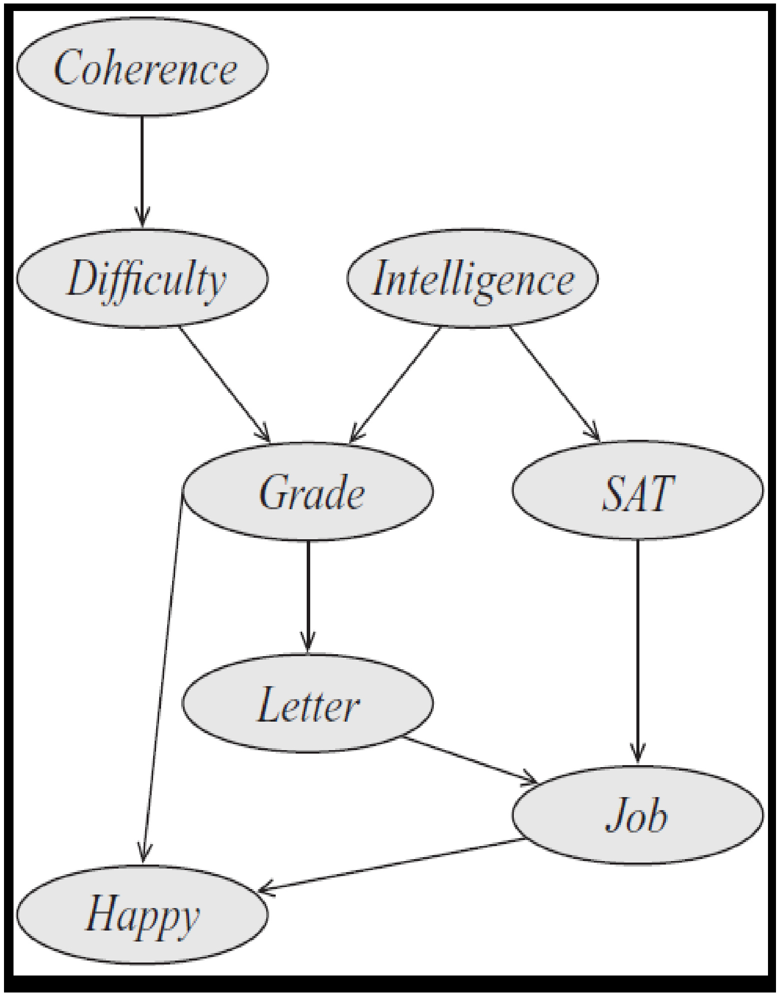
- VE Algorithm can also be applied to general graphs
  - Example to follow
- **X** is a set of variables, Φ is a set of factors whose scope is ⊆**X**
- Let **Y** ⊂ **X** be a set of query variables
- Let **Z** = **X** – **Y** (set of other variables, to be eliminated)
- For any ordering ≺ over **Z**, Sum-Product-VE (Φ, ≺, Z) returns

$$\phi^{*}(\mathbf{Y}) = \sum_{Z} \prod_{\phi \in \Phi} \phi$$

- If Φ is the set of all factors (one factor associated with each variable X_i , derived from the CPD of X_i) then above gives P(**Y**).
- Same algorithm also applies to Markov Networks
  - Initial factors are clique potentials
  - If original factors are not normalized, we also need to compute a partition function.

VE Applied to General Graph (Ex 9.1)

Goal is to compute P(J)



$$
\begin{aligned}
P(C, D, I, G, S, L, J, H) &= P(C)P(D \mid C)P(I)P(G \mid I, D)P(S \mid I) \\
&\quad P(L \mid G)P(J \mid L, S)P(H \mid G, J) \\
&= \phi_C(C)\phi_D(D, C)\phi_I(I)\phi_G(G, I, D)\phi_S(S, I) \\
&\quad \phi_L(L, G)\phi_J(J, L, S)\phi_H(H, G, J).
\end{aligned}
$$

Elimination Steps

- Use elimination ordering C, D, I, H, G, S, L
- 1. Eliminate C

$$\begin{aligned} \psi_1(C, D) &= \phi_C(C) \cdot \phi_D(D, C) \\ \tau_1(D) &= \sum_C \psi_1. \end{aligned}$$

- 2. Eliminate D: Note ϕ_D (D,C) already eliminated

$$\begin{aligned} \psi_2(G, I, D) &= \phi_G(G, I, D) \cdot \tau_1(D) \\ \tau_2(G, I) &= \sum_D \psi_2(G, I, D). \end{aligned}$$

- 3. Eliminate I:

$$\begin{aligned} \psi_3(G, I, S) &= \phi_I(I) \cdot \phi_S(S, I) \cdot \tau_2(G, I) \\ \tau_3(G, S) &= \sum_I \psi_3(G, I, S). \end{aligned}$$

Elimination Steps

- 4. Eliminate H

$$\begin{aligned}\psi_4(G,J,H) &= \phi_H(H,G,J)\\ \tau_4(G,J) &= \sum_H \psi_4(G,J,H).\end{aligned}$$

Note: that τ_4 is $\equiv 1$ as we are just computing $\sum_H P(H|G,J)$ so this step serves little purpose but complicates the next elimination

- 5. Eliminate G

$$\begin{aligned}\psi_5(G,J,L,S) &= \tau_4(G,J)\cdot\tau_3(G,S)\cdot\phi_L(L,G)\\ \tau_5(J,L,S) &= \sum_G \psi_5(G,J,L,S).\end{aligned}$$

- 6. Eliminate S

$$\begin{aligned}\psi_6(J,L,S) &= \tau_5(J,L,S)\cdot\phi_J(J,L,S)\\ \tau_6(J,L) &= \sum_S \psi_6(J,L,S).\end{aligned}$$

- 7. Eliminate L

$$\begin{aligned}\psi_7(J,L) &= \tau_6(J,L)\\ \tau_7(J) &= \sum_L \psi_7(J,L).\end{aligned}$$

Table Summarizing Elimination for P(J)

Step	Variable eliminated	Factors used	Variables involved	New factor
1	C	$\phi_C(C)$, $\phi_D(D,C)$	C, D	$\tau_1(D)$
2	D	$\phi_G(G,I,D)$, $\tau_1(D)$	G, I, D	$\tau_2(G,I)$
3	I	$\phi_I(I)$, $\phi_S(S,I)$, $\tau_2(G,I)$	G, S, I	$\tau_3(G,S)$
4	H	$\phi_H(H,G,J)$	H, G, J	$\tau_4(G,J)$
5	G	$\tau_4(G,J)$, $\tau_3(G,S)$, $\phi_L(L,G)$	G, J, L, S	$\tau_5(J,L,S)$
6	S	$\tau_5(J,L,S)$, $\phi_J(J,L,S)$	J, L, S	$\tau_6(J,L)$
7	L	$\tau_6(J,L)$	J, L	$\tau_7(J)$

Table 9.1 A run of variable elimination for the query $P(J)$

Another Elimination Order

- Note: order affects the size of factors generated, and hence the complexity of the computation.

Step	Variable eliminated	Factors used	Variables involved	New factor
1	G	$\phi_G(G,I,D)$, $\phi_L(L,G)$, $\phi_H(H,G,J)$	G,I,D,L,J,H	$\tau_1(I,D,L,J,H)$
2	I	$\phi_I(I)$, $\phi_S(S,I)$, $\tau_1(I,D,L,S,J,H)$	S,I,D,L,J,H	$\tau_2(D,L,S,J,H)$
3	S	$\phi_J(J,L,S)$, $\tau_2(D,L,S,J,H)$	D,L,S,J,H	$\tau_3(D,L,J,H)$
4	L	$\tau_3(D,L,J,H)$	D,L,J,H	$\tau_4(D,J,H)$
5	H	$\tau_4(D,J,H)$	D,J,H	$\tau_5(D,J)$
6	C	$\tau_5(D,J)$, $\phi_C(C)$, $\phi_D(D,C)$	D,J,C	$\tau_6(D,J)$
7	D	$\tau_6(D,J)$	D,J	$\tau_7(J)$

Table 9.2 **A different run of variable elimination for the query** $P(J)$

Complexity and how to choose elimination order will be discussed a bit later.

Next Class

- Read sections 9.3.2,9.4, 10.1 of the KF book

Just for formatting

- Message from $\mathbf{C}_i$ to $\mathbf{C}_j$ is given by
  - V_T vertices of T be ε_T the edges
  - If X is in $\mathbf{C}_i$ and also in $\mathbf{C}_j$ then X is also in every cluster in the path between $\mathbf{C}_i$ and $\mathbf{C}_j$.
  - Implies that $\mathbf{S}_{i,j} = \mathbf{C}_i \cap \mathbf{C}_j$.
  - **Family preserving**: Each factor φ must be associated with some cluster, say $\mathbf{C}_i$, called α (φ). scope [φ] $\subseteq \mathbf{C}_i$
  - Each edge between two nodes, say $\mathbf{C}_i$ and $\mathbf{C}_j$, is associated with a set of nodes, called a sepset, $\mathbf{S}_{i,j}$, $\mathbf{S}_{i,j} \subseteq \mathbf{C}_i \cap \mathbf{C}_j$.
- $\delta_{i \to \mathbf{j}}\ \Sigma$
- $P(D) = \sum_C \sum_B \sum_A P(A, B,C,D)$
- $D \perp J$
- ϕ