



cmsc726: Graphical Models: Inference

material from: Michael Jordan, Nir Friedman and Daphne Koller

Inference

- We now have compact representations of probability distributions: Bayesian Networks
- Network describes a unique probability distribution \mathcal{P}
- How do we answer queries about \mathcal{P} ?
- We use **inference** as a name for the process of computing answers to such queries

Queries: Likelihood

- There are many types of queries we might ask.
- Most of these involve **evidence**
 - Evidence \mathbf{e} is an assignment of values to a set \mathbf{E} variables in the domain
 - Without loss of generality $\mathbf{E} = \{X_{k+1}, \dots, X_n\}$
- Simplest query: compute probability of evidence

$$P(\mathbf{e}) = \sum_{x_1} \dots \sum_{x_k} P(x_1, \dots, x_k, \mathbf{e})$$

- This is often referred to as computing the **likelihood** of the evidence

Queries

- There are many types of queries we might ask.
- Most of these involve **evidence**
 - Evidence \mathbf{e} is an assignment of values to a set \mathbf{E} variables in the domain
 - Without loss of generality $\mathbf{E} = \{X_{k+1}, \dots, X_n\}$

Queries: Conditional Probability

- Often we are interested in the conditional probability of a variable given the evidence

$$P(X = x | \mathbf{e}) = \frac{P(X = x, \mathbf{e})}{P(\mathbf{e})}$$

- This is the **a posteriori belief** in X , given evidence \mathbf{e}
- A related task is computing the term $P(X, \mathbf{e})$
 - i.e., the likelihood of \mathbf{e} and $X = x$ for values of X
 - we can recover the a posteriori belief by

$$P(X = x | \mathbf{e}) = \frac{P(X = x, \mathbf{e})}{\sum_{x'} P(X = x', \mathbf{e})}$$

A posteriori belief

This query is useful in many cases:

- **Prediction**: what is the probability of an outcome given the starting condition
 - Target is a descendent of the evidence
- **Diagnosis**: what is the probability of disease/fault given symptoms
 - Target is an ancestor of the evidence
- As we shall see, the direction between variables does not restrict the directions of the queries
 - Probabilistic inference can combine evidence from all parts of the network

Queries: Most Probable Assignment

- In this query we want to find the **most probable joint assignment** for some variables of interest
- Two flavors:
 - MPE – most probable explanation

$$MPE(X | e) = \arg \max_x P(x, e)$$

- MAP - maximum a posterior query (combination of conditional probability and MPE)

$$MAP(Y | e) = \arg \max_y \sum_z P(y, z | e)$$

Queries: MPA

We can use MAP for:

- Classification
 - find most likely label, given the evidence
- Explanation
 - What is the most likely scenario, given the evidence

Queries: MPA

Cautionary note:

- The MPA depends on the set of variables
- Example:
 - MPA of X
 - MPA of (X, Y)

x	y	$P(x, y)$
0	0	0.35
0	1	0.05
1	0	0.3
1	1	0.3

Complexity of Inference

Thm:

Computing $P(X = x)$ in a Bayesian network is NP-hard

Not surprising, since we can simulate Boolean gates.

Hardness

- Hardness does not mean we cannot solve inference
 - It implies that we cannot find a general procedure that works efficiently for all networks
 - For particular families of networks, we can have provably efficient procedures

Approaches to inference

- Exact inference
 - Inference in Simple Chains
 - Variable elimination
 - Clustering / join tree algorithms
- Approximate inference
 - Stochastic simulation / sampling methods
 - Markov chain Monte Carlo methods
 - Variational Methods

Inference in Simple Chains



How do we compute $P(X_2)$?

$$P(x_2) = \sum_{x_1} P(x_1, x_2) = \sum_{x_1} P(x_1)P(x_2 | x_1)$$

Inference in Simple Chains (cont.)



How do we compute $P(X_3)$?

$$P(x_3) = \sum_{x_2} P(x_2, x_3) = \sum_{x_2} P(x_2)P(x_3 | x_2)$$

- we already know how to compute $P(X_2)$...

$$P(x_2) = \sum_{x_1} P(x_1, x_2) = \sum_{x_1} P(x_1)P(x_2 | x_1)$$

Inference in Simple Chains (cont.)



How do we compute $P(X_n)$?

- Compute $P(X_1), P(X_2), P(X_3), \dots$
- We compute each term by using the previous one

$$P(x_{i+1}) = \sum_{x_i} P(x_i)P(x_{i+1} | x_i)$$

Complexity:

- Each step costs $O(|Val(X_i)| * |Val(X_{i+1})|)$ operations
- Compare to naive evaluation, that requires summing over joint values of $n-1$ variables

Elimination in Chains

- We now try to understand the simple chain example using first-order principles



- Using definition of probability, we have

$$P(e) = \sum_a \sum_c \sum_b \sum_a P(a, b, c, d, e)$$

Elimination in Chains



- By chain decomposition, we get

$$\begin{aligned} P(e) &= \sum_a \sum_c \sum_b \sum_a P(a, b, c, d, e) \\ &= \sum_a \sum_c \sum_b \sum_a P(a)P(b | a)P(c | b)P(d | c)P(e | d) \end{aligned}$$

Elimination in Chains



- Rearranging terms ...

$$\begin{aligned} P(e) &= \sum_a \sum_c \sum_b \sum_a P(a)P(b | a)P(c | b)P(d | c)P(e | d) \\ &= \sum_a \sum_c \sum_b P(c | b)P(d | c)P(e | d) \sum_a P(a)P(b | a) \end{aligned}$$

Elimination in Chains



- Now we can perform innermost summation

$$P(e) = \sum_d \sum_c \sum_b P(c|b)P(d|c)P(e|d) \underbrace{\sum_a P(a)P(b|a)}_{p(b)}$$

$$= \sum_d \sum_c \sum_b P(c|b)P(d|c)P(e|d)p(b)$$

- This summation, is exactly the first step in the forward iteration we describe before

Elimination in Chains



- Rearranging and then summing again, we get

$$P(e) = \sum_d \sum_c \underbrace{\sum_b P(c|b)P(d|c)P(e|d)}_{p(d|c)P(e|d)} p(b)$$

$$= \sum_d \sum_c P(d|c)P(e|d) \underbrace{\sum_b P(c|b)p(b)}_{p(c)}$$

$$= \sum_d \sum_c P(d|c)P(e|d)p(c)$$

- Let's move to more complex BNs....

Variable Elimination

General idea:

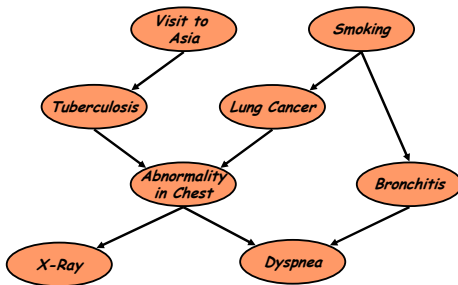
- Write query in the form

$$P(X_n, \mathbf{e}) = \sum_{x_k} \dots \sum_{x_3} \sum_{x_2} \prod_i P(x_i | pa_i)$$

- Iteratively
 - Move all irrelevant terms outside of innermost sum
 - Perform innermost sum, getting a new term
 - Insert the new term into the product

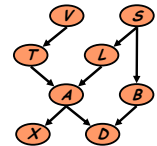
A More Complex Example

- "Asia" network:



- We want to compute $P(d)$
- Need to eliminate: v, s, x, t, l, a, b

Initial factors



$$P(v)P(s)P(t|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

- We want to compute $P(d)$
- Need to eliminate: v, s, x, t, l, a, b

Initial factors

$$P(v)P(s)P(t|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

Eliminate: v

Compute: $f_v(t) = \sum_v P(v)P(t|v)$

$$\Rightarrow f_v(t)P(s)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

Note: $f_v(t) = P(t)$
In general, result of elimination is not necessarily a probability term

- We want to compute $P(d)$
- Need to eliminate: s, x, t, l, a, b

Initial factors

$$P(v)P(s)P(t|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)P(s)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

Eliminate: s

Compute: $f_s(b,l) = \sum_s P(s)P(b|s)P(l|s)$

$$\Rightarrow f_v(t)f_s(b,l)P(a|t,l)P(x|a)P(d|a,b)$$

Summing on s results in a factor with two arguments $f_s(b,l)$
In general, result of elimination may be a function of several variables

- We want to compute $P(d)$
- Need to eliminate: x, t, l, a, b

Initial factors

$$P(v)P(s)P(t|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)P(s)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)f_s(b,l)P(a|t,l)P(x|a)P(d|a,b)$$

Eliminate: x

Compute: $f_x(a) = \sum_x P(x|a)$

$$\Rightarrow f_v(t)f_s(b,l)f_x(a)P(a|t,l)P(d|a,b)$$

Note: $f_x(a) = 1$ for all values of a !!

- We want to compute $P(d)$
- Need to eliminate: t, l, a, b

Initial factors

$$P(v)P(s)P(t|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)P(s)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)f_s(b,l)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)f_s(b,l)f_x(a)P(a|t,l)P(d|a,b)$$

Eliminate: t

Compute: $f_t(a,l) = \sum_t f_v(t)P(a|t,l)$

$$\Rightarrow f_s(b,l)f_x(a)f_t(a,l)P(d|a,b)$$

- We want to compute $P(d)$
- Need to eliminate: l, a, b

Initial factors

$$P(v)P(s)P(t|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)P(s)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)f_s(b,l)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)f_s(b,l)f_x(a)P(a|t,l)P(d|a,b)$$

$$\Rightarrow f_s(b,l)f_x(a)f_t(a,l)P(d|a,b)$$

Eliminate: l

Compute: $f_l(a,b) = \sum_l f_s(b,l)f_t(a,l)$

$$\Rightarrow f_l(a,b)f_x(a)P(d|a,b)$$

- We want to compute $P(d)$
- Need to eliminate: b

Initial factors

$$P(v)P(s)P(t|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)P(s)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)f_s(b,l)P(a|t,l)P(x|a)P(d|a,b)$$

$$\Rightarrow f_v(t)f_s(b,l)f_x(a)P(a|t,l)P(d|a,b)$$

$$\Rightarrow f_s(b,l)f_x(a)f_t(a,l)P(d|a,b)$$

$$\Rightarrow f_l(a,b)f_x(a)P(d|a,b) \Rightarrow f_b(b,d) \Rightarrow f_b(d)$$

Eliminate: a, b

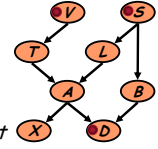
Compute: $f_b(b,d) = \sum_a f_l(a,b)f_x(a)P(d|a,b)$ $f_b(d) = \sum_b f_b(b,d)$

Variable Elimination

- We now understand variable elimination as a sequence of **rewriting** operations
- Actual computation is done in elimination step
- Exactly the same computation procedure applies to Markov networks
- Computation depends on order of elimination

Dealing with evidence

- How do we deal with evidence?
- Suppose get evidence $V = t, S = f, D = t$
- We want to compute $P(L, V = t, S = f, D = t)$

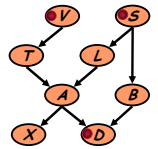


Dealing with Evidence

- We start by writing the factors:

$$P(v)P(s)P(t|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

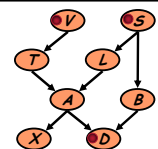
- Since we know that $V = t$, we don't need to eliminate V
- Instead, we can replace the factors $P(V)$ and $P(T|V)$ with $f_{P(V)} = P(V = t)$ and $f_{P(T|V)}(T) = P(T | V = t)$
- These "select" the appropriate parts of the original factors given the evidence
- Note that $f_{P(V)}$ is a constant, and thus does not appear in elimination of other variables



Dealing with Evidence

- Given evidence $V = t, S = f, D = t$
- Compute $P(L, V = t, S = f, D = t)$
- Initial factors, after setting evidence:

$$f_{P(V)} f_{P(S)} f_{P(T|V)}(t) f_{P(L|S)}(l) f_{P(B|S)}(b) P(a|t,l) P(x|a) f_{P(D|A,B)}(a,b)$$



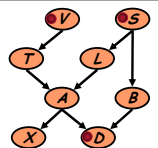
Dealing with Evidence

- Given evidence $V = t, S = f, D = t$
- Compute $P(L, V = t, S = f, D = t)$
- Initial factors, after setting evidence:

$$f_{P(V)} f_{P(S)} f_{P(T|V)}(t) f_{P(L|S)}(l) f_{P(B|S)}(b) P(a|t,l) P(x|a) f_{P(D|A,B)}(a,b)$$

- Eliminating x , we get

$$f_{P(V)} f_{P(S)} f_{P(T|V)}(t) f_{P(L|S)}(l) f_{P(B|S)}(b) P(a|t,l) f_x(a) f_{P(D|A,B)}(a,b)$$



Dealing with Evidence

- Given evidence $V = t, S = f, D = t$
- Compute $P(L, V = t, S = f, D = t)$
- Initial factors, after setting evidence:

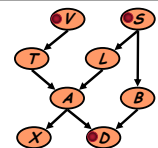
$$f_{P(V)} f_{P(S)} f_{P(T|V)}(t) f_{P(L|S)}(l) f_{P(B|S)}(b) P(a|t,l) P(x|a) f_{P(D|A,B)}(a,b)$$

- Eliminating x , we get

$$f_{P(V)} f_{P(S)} f_{P(T|V)}(t) f_{P(L|S)}(l) f_{P(B|S)}(b) P(a|t,l) f_x(a) f_{P(D|A,B)}(a,b)$$

- Eliminating t , we get

$$f_{P(V)} f_{P(S)} f_{P(L|S)}(l) f_{P(B|S)}(b) f_t(a,l) f_x(a) f_{P(D|A,B)}(a,b)$$



Dealing with Evidence

- Given evidence $V = t, S = f, D = t$
- Compute $P(L, V = t, S = f, D = t)$
- Initial factors, after setting evidence:

$$f_{P(V)} f_{P(S)} f_{P(T|V)}(t) f_{P(L|S)}(l) f_{P(B|S)}(b) P(a|t, l) P(x|a) f_{P(D|a, B)}(a, b)$$
- Eliminating x , we get

$$f_{P(V)} f_{P(S)} f_{P(T|V)}(t) f_{P(L|S)}(l) f_{P(B|S)}(b) P(a|t, l) f_x(a) f_{P(D|a, B)}(a, b)$$
- Eliminating t , we get

$$f_{P(V)} f_{P(S)} f_{P(L|S)}(l) f_{P(B|S)}(b) f_t(a, l) f_x(a) f_{P(D|a, B)}(a, b)$$
- Eliminating a , we get

$$f_{P(V)} f_{P(S)} f_{P(L|S)}(l) f_{P(B|S)}(b) f_a(b, l)$$

Dealing with Evidence

- Given evidence $V = t, S = f, D = t$
- Compute $P(L, V = t, S = f, D = t)$
- Initial factors, after setting evidence:

$$f_{P(V)} f_{P(S)} f_{P(T|V)}(t) f_{P(L|S)}(l) f_{P(B|S)}(b) P(a|t, l) P(x|a) f_{P(D|a, B)}(a, b)$$
- Eliminating x , we get

$$f_{P(V)} f_{P(S)} f_{P(T|V)}(t) f_{P(L|S)}(l) f_{P(B|S)}(b) P(a|t, l) f_x(a) f_{P(D|a, B)}(a, b)$$
- Eliminating t , we get

$$f_{P(V)} f_{P(S)} f_{P(L|S)}(l) f_{P(B|S)}(b) f_t(a, l) f_x(a) f_{P(D|a, B)}(a, b)$$
- Eliminating a , we get

$$f_{P(V)} f_{P(S)} f_{P(L|S)}(l) f_{P(B|S)}(b) f_a(b, l)$$
- Eliminating b , we get

$$f_{P(V)} f_{P(S)} f_{P(L|S)}(l) f_b(b, l)$$
- Final result:

$$f_{P(V)} f_{P(S)} f_{P(L|S)}(l) f_b(b, l)$$

Complexity of variable elimination

- Suppose in one elimination step we compute

$$f_x(y_1, \dots, y_k) = \sum_x f_x(x, y_1, \dots, y_k)$$

$$f_x(x, y_1, \dots, y_k) = \prod_{i=1}^m f_i(x, y_{1,1}, \dots, y_{1,i})$$
- This requires
 - $m \cdot |\text{Val}(X)| \cdot \prod_i |\text{Val}(Y_i)|$ multiplications
 - For each value for x, y_1, \dots, y_k , we do m multiplications
 - $|\text{Val}(X)| \cdot \prod_i |\text{Val}(Y_i)|$ additions
 - For each value of y_1, \dots, y_k , we do $|\text{Val}(X)|$ additions
- Complexity is exponential in number of variables in the intermediate factor

Understanding Variable Elimination

- We want to select "good" elimination orderings that reduce complexity
- We start by attempting to understand variable elimination via the graph we are working with
- This will reduce the problem of finding good ordering to graph-theoretic operation that is well-understood

Undirected graph representation

- At each stage of the procedure, we have an algebraic term that we need to evaluate
- In general this term is of the form:

$$P(x_1, \dots, x_k) = \sum_{y_1} \dots \sum_{y_r} \prod_i f_i(\mathbf{Z}_i)$$
 where \mathbf{Z}_i are sets of variables
- We now plot a graph where there is undirected edge $X-Y$ if X, Y are arguments of some factor
 - that is, if X, Y are in some \mathbf{Z}_i
- Note: this is the Markov network that describes the probability on the variables we did not eliminate yet

Chordal Graphs

- elimination ordering \Rightarrow undirected chordal graph

Graph:

- Maximal cliques are factors in elimination
- Factors in elimination are cliques in the graph
- Complexity is exponential in size of the largest clique in graph

Induced Width

- The size of the largest clique in the induced graph is thus an indicator for the complexity of variable elimination
- This quantity is called the **induced width** of a graph according to the specified ordering
- Finding a good ordering for a graph is equivalent to finding the minimal induced width of the graph

General Networks

- From graph theory:

Thm:

- Finding an ordering that minimizes the induced width is NP-Hard

However,

- There are reasonable heuristic for finding “relatively” good ordering
- There are provable approximations to the best induced width
- If the graph has a small induced width, there are algorithms that find it in polynomial time

Elimination on Trees

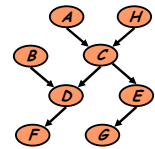
- Formally, for any tree, there is an elimination ordering with induced width = 1

Thm

- Inference on trees is linear in number of variables

PolyTrees

- A polytree is a network where there is at most one path from one variable to another



Thm:

- Inference in a polytree is linear in the representation size of the network
 - This assumes tabular CPT representation

Approaches to inference

- **Exact inference**
 - Inference in Simple Chains
 - Variable elimination
 - Clustering / join tree algorithms
- **Approximate inference**
 - Stochastic simulation / sampling methods
 - Markov chain Monte Carlo methods
 - Mean field theory

Stochastic simulation

- Suppose you are given values for some subset of the variables, G, and want to infer values for unknown variables, U
- Randomly generate a very large number of instantiations from the BN
 - Generate instantiations for **all** variables – start at root variables and work your way “forward”
- Only keep those instantiations that are consistent with the values for G
- Use the frequency of values for U to get estimated probabilities
- Accuracy of the results depends on the size of the sample (asymptotically approaches exact results)

Markov chain Monte Carlo methods

- So called because
 - Markov chain – each instance generated in the sample is dependent on the previous instance
 - Monte Carlo – statistical sampling method
- Perform a random walk through variable assignment space, collecting statistics as you go
 - Start with a random instantiation, consistent with evidence variables
 - At each step, for some nonevidence variable, randomly sample its value, consistent with the other current assignments
- Given enough samples, MCMC gives an accurate estimate of the true distribution of values