10-701 Recitation (11/16) Jayant Krishnamurthy

Topics:

I. Undirected Graphical Models

II. Inference in BNs

   A. Variable Elimination

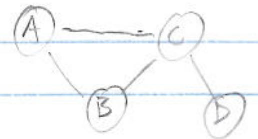
   B. Complexity of Variable Elimination

III. Learning in BNs

   A. Learning CPDs given a BN structure

   B. Learning BN structure & CPDs.

Undirected Graphical Models (Markov Random Fields)



$$P(A,B,C,D) = \frac{1}{Z} \psi_{ABC}(A,B,C)\, \psi_{CD}(C,D)$$

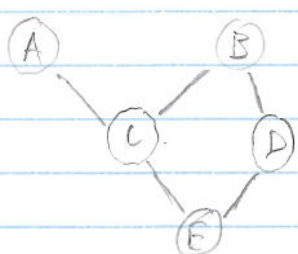
potentials ψ also called factors.

- Distribution is written as a product of → potentials ψ, each one associated with a maximal clique in the graph.
- Potentials divided by normalizing constant Z (also called the partition function) to ensure the probabilities sum to 1.

$$Z = \sum_{a,b,c,d} \psi_{ABC}(a,b,c)\, \psi_{CD}(c,d)$$ ← Typically hard to compute.

- Conditional independence in MRFs: two variables x_i, x_j are conditionally independent given set of variables X if there is no path between x_i and x_j composed entirely of unobserved variables.

Ex:



$A \perp B \mid C$

$B \perp E \mid C, D$ (but not given only one of them)

$C \perp D \mid B, E$

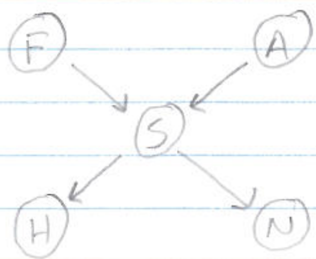
Not true:

$A \perp D \mid B$

Recall:

- BN is concise way of encoding a distribution

Ex:



$P(F, A, S, H, N)$
$= P(F)\,P(A)\,P(S \mid A, H)\,P(H \mid S)\,P(N \mid S)$

- D-separation lets us read the conditional independence assumptions in the BN

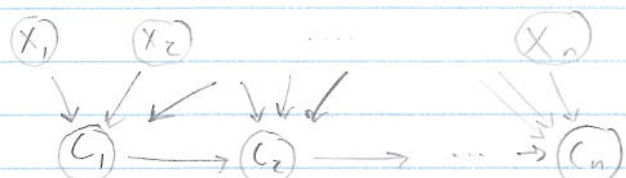
Today: Inference in BNs.

Given a BN like the one above, and its CPTs (Conditional Probability Tables), how can we answer queries like:

- $P(H = \text{true} \mid N = \text{false})$
- $P(S = \text{true})$

$\boxed{\text{Requires \underline{marginalization}, or summing out the other variables}}$

Bad News: Inference is NP-hard in general.
(Reduction from 3-SAT)



← variables in formula

← clauses in formula

$P(x_i = \text{true}) = 1/2$, $\quad C_i = (X_{i1} \vee X_{i2} \vee X_{i3}) \wedge C_{i-1}$

deterministic function of inputs, which is fine in BN

$P(C_n = \text{true}) > 0 \iff$ original 3-SAT formula is satisfiable.

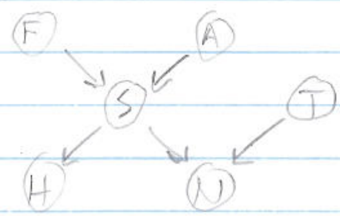
Good News: Inference frequently possible, depending on the graph structure of the BN.

Variable Elimination

- Algorithm for marginalizing out variables in a BN to answer queries like the two previous ones
- Basic idea: iteratively marginalize out variables by summing up the terms containing the variable.

Requires us to → choose an order to sum out each variable

Ex:



$P(F,A,S,H,N,T)$
$= P(F)P(A)P(T)P(S|F,A)P(H|S)P(N|S,T)$

Compute $P(N, A=1)$ by variable elimination:

$$P(N, A=1) = \sum_{f,s,h,t} P(f, A=1, s, h, N, t)$$

Say I choose → to sum s out first

$$= \sum_{f,h,t} P(f)P(A=1)P(t) \sum_{s} P(s|f,A=1)P(h|s)P(N|s,t)$$

$$= \sum_{f,h,t} P(f)P(A=1)P(t) \cdot \underbrace{g(f,h,n,t)}$$

g is some function of the variables, called a factor

g may be a CPD, but it also may not be.

$$= P(A=1)\sum_{f} P(f) \sum_{t} P(t) \sum_{h} g(f,h,n,t)$$

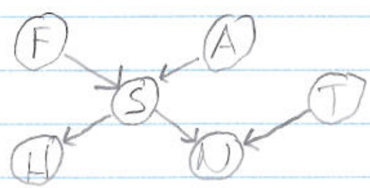
... Just keep doing the sums and creating factors to get your answer.

- Complexity of Variable Elimination is exponential in the size of the largest factor. Why? Because factors contain an exponential number of table entries. (E.g. $g(f,h,n,t)$ from the previous example has 2^4 values, assuming the variables are binary.)
- Order of eliminating variables matters! Different orders lead to differently sized factors.

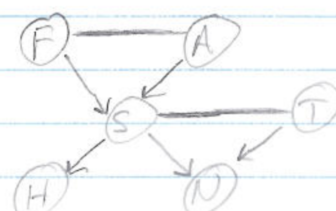
Can we determine the complexity of variable elimination?
- Yes, using Treewidth of the graph.

Use this process:

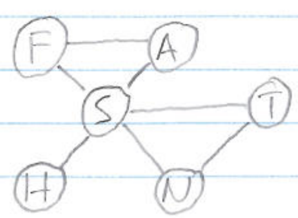
1. Moralize graph: Draw an undirected edge between parents who share a child.



=>



2. Remove edge directions.

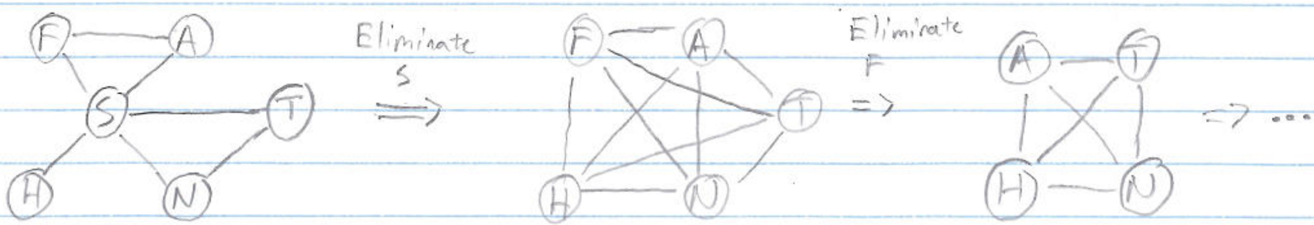


Aside: This undirected graph is now a Markov Network which represents the same distribution as the original BN. (assuming the factors are chosen appropriately.)

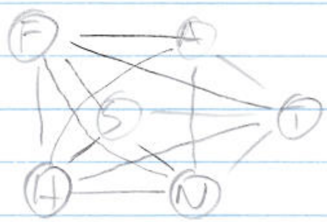
3. Run Variable Elimination. Each time a variable is eliminated, draw a clique between all of its neighbors, then remove the variable from the graph.

(Answer query P(N))

Ex. Elimination order: S, F, H, T, A



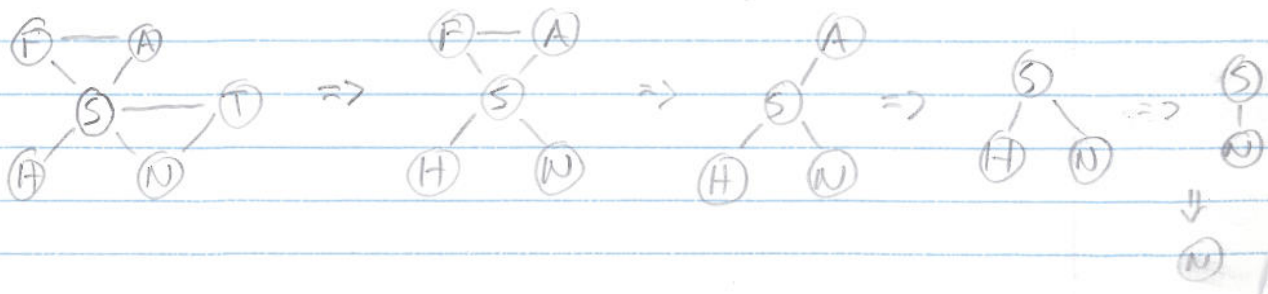
<u>Induced Graph</u>: Take union of edges from the sequence of graphs you create.
(In this case, we get the complete graph on all 6 variables.)



Note: Induced graph depends on elimination order! See example below.

<u>Induced width:</u> Size of largest clique in the induced graph, minus one.
Corresponds to the size of the largest factor created during variable elimination (after marginalizing.)
(In this case, $6-1=5$, for factor $g(f,a,t,h,n)$)

<u>Treewidth</u>: Minimum induced width over all possible elimination orders. Provides a lower bound on complexity of variable elimination.
(In this case, treewidth is 2; Try the elimination order T, F, A, H, S)

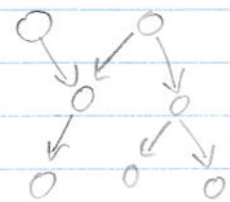


- Induced graph same as original graph!

Conclusion: when is variable elimination fast?

- Poly trees – a graph with no undirected cycles. (I.e. remove edge directions, then test for cycles.)

Polytree:



Not Polytree:

Undirected Cycle.

- Important special case is chain structures (like HMMs!)

Learning in BNs:

Two settings:

1. Given graph, learn CPDs (easy)
2. Learn graph structure and CPDs from data (harder)

Learning CPDs:

Just use MLE or MAP estimate:

$$\hat{\Theta}_{MLE} = \arg\max_{\theta} \sum_{i=1}^{n} \log P(\mathbf{x}^i ; \theta) \quad (\text{Data } \mathbf{x}^1, \ldots, \mathbf{x}^n)$$

$$= \arg\max_{\theta} \sum_{i=1}^{n} \sum_{j=1}^{d} \log P(x_j^i \mid \text{Parents}(x_j) ; \theta_j)$$

$$= \arg\max_{\theta} \underbrace{\sum_{j=1}^{d} \sum_{i=1}^{n}}_{\text{sums switched}} \log P(x_j^i \mid \text{Parents}(x_j) ; \theta_j)$$

MLE estimate for graph is simply MLE for each conditional distribution independently!

Learning Graph Structure & CPDs:

MLE estimate for graph and parameters given data $D = \{x^1, \ldots, x^n\}$:

$$\log P(D \mid \theta, G) = \sum_{j=1}^{d} \sum_{i=1}^{n} \log P\left(X_j = x_j^{(i)} \mid X_{Pa(j)} = x_{Pa(j)}^{(i)}\right)$$

Let $\hat{P}(X_j = x) = \frac{1}{n} \cdot \sum_{i=1}^{n} \mathbb{1}(x_j^i = x)$

$\boxed{\mathbb{1} \text{ is the indicator function}}$

$$= \frac{1}{n} \text{Count}(X_j = x)$$

($\hat{P}$ is the empirical distribution of the data)

Then we can write:

$$\log P(D \mid \theta, G) = \sum_{j=1}^{d} \sum_{x_j} \sum_{x_{Pa(j)}} \text{Count}\left(X_j = x_j, X_{Pa(j)} = x_{Pa(j)}\right) \times \log P\left(X_j = x_j \mid X_{Pa(j)} = x_{Pa(j)}\right)$$

Just re-write sum to be over the values of each variable.

$$= n \cdot \sum_{j=1}^{d} \sum_{x_j} \sum_{x_{Pa(j)}} \hat{P}(x_j, x_{Pa(j)}) \log P(x_j \mid x_{Pa(j)})$$

Observe that given a graph G, we know the MLE estimate for θ sets each CPD to:

$$P(x_j \mid x_{Pa(j)}) = \frac{\text{Count}(x_j, x_{Pa(j)})}{\text{Count}(x_{Pa(j)})}$$

depends on G because of parents function $Pa(\cdot)$

$$= \hat{P}(x_j \mid x_{Pa(j)})$$

Plugging this estimate in to the likelihood, we get

$$\log P(D \mid \hat{\theta}, G) = n \sum_{j=1}^{d} \sum_{x_i} \sum_{x_{Pa(i)}} \hat{P}(x_j, x_{Pa(j)}) \log \hat{P}(x_j \mid x_{Pa(j)})$$

Note this only depends on the graph structure G!

Simplifying a bit, we get

$$\log P(D \mid \Theta, G) = n \sum_{j=1}^{d} \sum_{x_j} \sum_{x_{pa(j)}} \hat{P}(x_j, x_{pa(j)}) \log \left(\frac{\hat{P}(x_j, x_{pa(j)})}{\hat{P}(x_{pa(j)})} \right)$$

$$= n \sum_{j=1}^{d} \sum_{x_j} \sum_{x_{pa(j)}} \hat{P}(x_j, x_{pa(j)}) \log \left(\frac{\hat{P}(x_j, x_{pa(j)})}{\hat{P}(x_j) \hat{P}(x_{pa(j)})} \right)$$

$$+ \hat{P}(x_j, x_{pa(j)}) \log \hat{P}(x_j)$$

$$= n \sum_{j=1}^{d} \hat{I}(x_j, x_{pa(j)}) - \hat{H}(x_j)$$

Where I is the mutual information

$$I(X,Y) = \sum_x \sum_y P(x,y) \log \left(\frac{P(x,y)}{P(x) P(y)} \right)$$

and H is the entropy

$$H(X) = - \sum_x p(x) \log p(x)$$

Note that $H(x_j)$ doesn't depend on the graph. So to find the best graph G, we simply maximize the mutual information

$$\hat{G}_{MLE} = \arg\max_G \log P(D \mid \hat{\Theta}_G, G)$$

$$= \arg\max_G \sum_{j=1}^{d} \hat{I}(x_j, X_{pa(j)})$$

In general, this doesn't work! $\hat{G}_{MLE}$ is the complete graph, as adding more parents to a variable never decreases mutual information. Possible solutions are to penalize complexity of graph (e.g. using a MAP estimate), or limit graph type.

Chow-Liu Algorithm

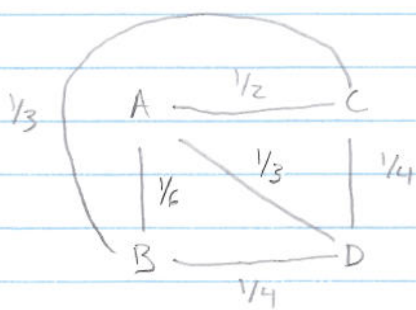
- Find the best graph G that is a tree

1. Note that edge directions in the tree don't matter. Each node has only 1 parent, so V-structures are impossible.

2. Create undirected graph on vertices $X_1, \ldots, X_j$ where edge (X_i, X_j) has weight $\hat{I}(X_i, X_j)$

3. Compute maximum spanning tree of graph (Can use Prim's or Kruskal's algorithm.)

4. Choose any vertex as root; point all edges away from the root.



Edges chosen in order are:
(A, C)
(A, D)
(C, B)

Say we choose A as the root. We get the graph: