Probabilistic graphical models
CPSC 532c (Topics in AI)
Stat 521a (Topics in multivariate analysis)

Lecture 7

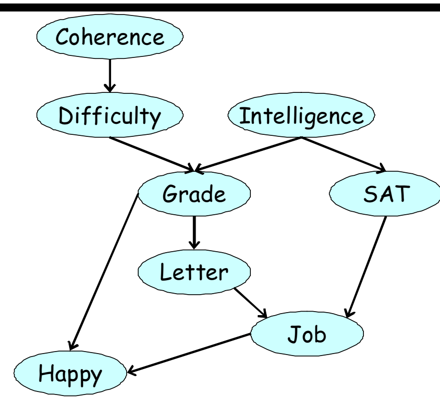
Kevin Murphy

Monday 4 October, 2004

Administrivia

- Homework 3 due Wednesday, 9.30am;
  send by email to crowley@cs.ubc.ca.

Variable elimination algorithm



- Key idea 1: push sum inside products.
- Key idea 2: use (non-serial) dynamic programming to cache shared subexpressions.

$$
\begin{aligned}
P(J) &= \sum_L \sum_S \sum_G \sum_H \sum_I \sum_D \sum_C P(C, D, I, G, S, L, J, H) \\
&= \sum_L \sum_S \sum_G \sum_H \sum_I \sum_D \sum_C P(C)P(D|C)P(I)P(G|I, D)P(S|I)P(L|G)P(J|L, S)P(H|G, J) \\
&= \sum_L \sum_S \sum_G \sum_H \sum_I \sum_D \sum_C \phi_C(C)\phi_D(D, C)\phi_I(I)\phi_G(G, I, D)\phi_S(S, I)\phi_L(L, G)\phi_J(J, L, S)\phi_H(H, G, J) \\
&= \sum_L \sum_S \phi_J(J, L, S) \sum_G \phi_L(L, G) \sum_H \phi_H(H, G, J) \sum_I \phi_S(S, I)\phi_I(I) \sum_D \phi_(G, I, D) \sum_C \phi_C(C)\phi_D(D, C)
\end{aligned}
$$

Working right to left (peeling)

$$\begin{aligned}
P(J) &= \sum_L \sum_S \phi_J(J,L,S) \sum_G \phi_L(L,G) \sum_H \phi_H(H,G,J) \sum_I \phi_S(S,I)\phi_I(I) \sum_D \phi_(G,I,D) \underbrace{\sum_C \phi_C(C)\phi_D(D,C)}_{\tau_1(D)} \\
&= \sum_L \sum_S \phi_J(J,L,S) \sum_G \phi_L(L,G) \sum_H \phi_H(H,G,J) \sum_I \phi_S(S,I)\phi_I(I) \underbrace{\sum_D \phi_(G,I,D)\tau_1(D)}_{\tau_2(G,I)} \\
&= \sum_L \sum_S \phi_J(J,L,S) \sum_G \phi_L(L,G) \sum_H \phi_H(H,G,J) \underbrace{\sum_I \phi_S(S,I)\phi_I(I)\tau_2(G,I)}_{\tau_3(G,S)} \\
&= \sum_L \sum_S \phi_J(J,L,S) \sum_G \phi_L(L,G) \underbrace{\sum_H \phi_H(H,G,J)}_{\tau_4(G,J)} \tau_3(G,S) \\
&= \sum_L \sum_S \phi_J(J,L,S) \underbrace{\sum_G \phi_L(L,G)\tau_4(G,J)\tau_3(G,S)}_{\tau_5(J,L,S)} \\
&= \sum_L \underbrace{\sum_S \phi_J(J,L,S)\tau_5(J,L,S)}_{\tau_6(J,L)} \\
&= \underbrace{\sum_L \tau_6(J,L)}_{\tau_7(J)}
\end{aligned}$$

Bucket elimination

- We first multiply together all factors that mention C to create $\psi_1(C, D)$, and store the result in C's bucket:

$$P(J) = \sum_L \sum_S \phi_J(J, L, S) \sum_G \phi_L(L, G) \sum_H \phi_H(H, G, J) \sum_I \phi_S(S, I)\phi_I(I) \sum_D \phi_(G, I, D) \sum_C \underbrace{\phi_C(C)\phi_D(D, C)}_{\psi_1(C,D)}$$

- Then we sum out C to make $\tau_1(D)$:

$$P(J) = \sum_L \sum_S \phi_J(J, L, S) \sum_G \phi_L(L, G) \sum_H \phi_H(H, G, J) \sum_I \phi_S(S, I)\phi_I(I) \sum_D \phi_(G, I, D) \underbrace{\sum_C \psi_1(C, D)}_{\tau_1(D)}$$

- and multiply into D's bucket to make $\psi_2(G, I, D)$:

$$P(J) = \sum_L \sum_S \phi_J(J, L, S) \sum_G \phi_L(L, G) \sum_H \phi_H(H, G, J) \sum_I \phi_S(S, I)\phi_I(I) \sum_D \underbrace{\phi_(G, I, D)\tau_1(D)}_{\psi_2(G,I,D)}$$

- Then we sum out D to make $\tau_2(G, I)$:

$$P(J) = \sum_L \sum_S \phi_J(J, L, S) \sum_G \phi_L(L, G) \sum_H \phi_H(H, G, J) \sum_I \phi_S(S, I)\phi_I(I) \underbrace{\sum_D \psi_2(G, I, D)}_{\tau_2(G,I)}$$

- and multiply into I's bucket to make $\psi_3(G, S, I)$, etc.

Computing the partition function

- Let

$$P(X_{1:n}) = \frac{1}{Z} P'(X_{1:n}) = \frac{1}{Z} \prod_c \phi_c(X_c)$$

- For Bayes nets, $Z = 1$ (since each ϕ_c is a CPD).
- If we marginalize out all variables except Q, the result is

$$F(Q) = \sum_{X_{1:n} \setminus Q} \prod_c \phi_c(X_c)$$

- Hence if $Q = \emptyset$, we get

$$F(\emptyset) = \sum_{X_{1:n}} \prod_c \phi_c(X_c) = Z$$

Dealing with evidence

- Method 1: we instantiate observed variables to their observed values, by taking the appropriate "slices" of the factors
- e.g., evidence $I = 1, H = 0$:

$$P(J, I=1, H=0) =$$
$$\sum_L \sum_S \phi_J(J,L,S) \sum_G \phi_L(L,G)\phi_H(H=0,G,J)\phi_S(S,I=1)\phi_I(I=1) \sum_D \phi_(G,I=1,D) \sum_C \phi_C(C)\phi_D(D,C)$$

- Method 2: we multiply in local evidence factors $\phi_i(X_i)$ for each node. If X_i is observed to have value x_i^*, we set $\phi_i(X_i) = \delta(X_i, x_i^*)$.

$$P(J, I=1, H=0) =$$
$$\sum_L \sum_S \phi_J(J,L,S) \sum_G \phi_L(L,G) \sum_H \phi_H(H,G,J)\delta(H,0) \sum_I \phi_S(S,I)\phi_I(I)\delta(I,1) \sum_D \phi_(G,I,D) \sum_C \phi_C(C)\phi_D(D,C)$$

Dealing with evidence

- Once we instantiate evidence, the final factor is

$$F(Q,e) = P'(Q,e)$$

- Hence

$$\begin{aligned} P(Q|e) &= \frac{P(Q,e)}{P(e)} = \frac{P(Q,e)}{\sum_{q'} P(q',e)} \\ &= \frac{(1/Z)P'(Q,e)}{(1/Z)\sum_{q'} P'(q',e)} \\ &= \frac{F(Q,e)}{\sum_{q'} F(q',e)} \end{aligned}$$

- and

$$P(e) = \sum_{q'} P(q',e) = (1/Z)\sum_{q'} F(q',e)$$

Ordering 1

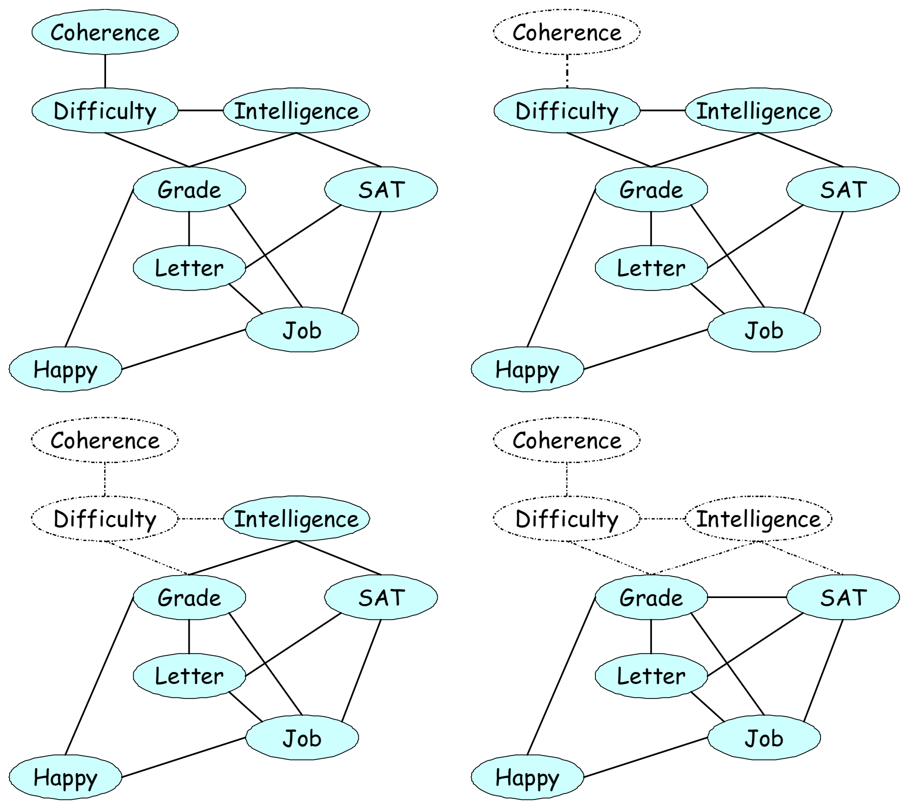
$$
\begin{aligned}
P(J) &= \sum_L \sum_S \phi_J(J,L,S) \sum_G \phi_L(L,G) \sum_H \phi_H(H,G,J) \sum_I \phi_S(S,I)\phi_I(I) \sum_D \phi_(G,I,D) \underbrace{\sum_C \phi_C(C)\phi_D(D,C)}_{\tau_1(D)} \\
&= \sum_L \sum_S \phi_J(J,L,S) \sum_G \phi_L(L,G) \sum_H \phi_H(H,G,J) \sum_I \phi_S(S,I)\phi_I(I) \underbrace{\sum_D \phi_(G,I,D)\tau_1(D)}_{\tau_2(G,I)} \\
&= \sum_L \sum_S \phi_J(J,L,S) \sum_G \phi_L(L,G) \sum_H \phi_H(H,G,J) \underbrace{\sum_I \phi_S(S,I)\phi_I(I)\tau_2(G,I)}_{\tau_3(G,S)} \\
&= \sum_L \sum_S \phi_J(J,L,S) \sum_G \phi_L(L,G) \underbrace{\sum_H \phi_H(H,G,J)}_{\tau_4(G,J)} \tau_3(G,S) \\
&= \sum_L \sum_S \phi_J(J,L,S) \underbrace{\sum_G \phi_L(L,G)\tau_4(G,J)\tau_3(G,S)}_{\tau_5(J,L,S)} \\
&= \sum_L \underbrace{\sum_S \phi_J(J,L,S)\tau_5(J,L,S)}_{\tau_6(J,L)} \\
&= \underbrace{\sum_L \tau_6(J,L)}_{\tau_7(J)}
\end{aligned}
$$

Different ordering

$$
\begin{aligned}
P(J) &= \sum_D \sum_C \phi_D(D,C) \sum_H \sum_L \sum_S \phi_J(J,L,S) \sum_I \phi_I(I)\phi_S(S,I) \underbrace{\sum_G \phi_G(G,I,D)\phi_L(L,)\phi_H(H,G,J)}_{\tau_1(I,D,L,J,H)} \\
&= \sum_D \sum_C \phi_D(D,C) \sum_H \sum_L \sum_S \phi_J(J,L,S) \underbrace{\sum_I \phi_I(I)\phi_S(S,I)\tau_1(I,D,L,J,H)}_{\tau_2(D,L,S,J,H)} \\
&= \sum_D \sum_C \phi_D(D,C) \sum_H \sum_L \underbrace{\sum_S \phi_J(J,L,S)\tau_2(D,L,S,J,H)}_{\tau_3(D,L,J,H)} \\
&= \sum_D \sum_C \phi_D(D,C) \sum_H \underbrace{\sum_L \tau_3(D,L,J,H)}_{\tau_4(D,J,H)} \\
&= \sum_D \sum_C \phi_D(D,C) \underbrace{\sum_H \tau_4(D,J,H)}_{\tau_5(D,J)} \\
&= \sum_D \underbrace{\sum_C \phi_D(D,C)\tau_5(D,J)}_{\tau_6(D,J)} \\
&= \underbrace{\sum_D \tau_6(D,J)}_{\tau_7(J)}
\end{aligned}
$$

Elimination as graph transformation

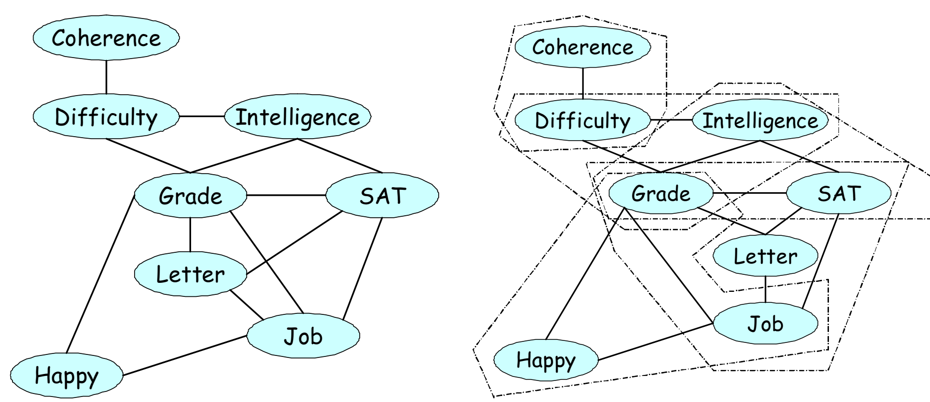
- Start by moralizing the graph (if necessary), so all terms in each factor form a (sub)clique.
- When we eliminate a variable X_i, we connect it to all variables that share a factor with X_i (to reflect new factor τ_i). Such edges are called "fill-in edges" (e.g., $\sum_I$ induces $G - S$).



Cliques and factors

- Let $I_{G,\prec}$ be the (undirected) graph induced by applying variable elimination to G using ordering $\prec$.
- Thm 7.3.4: Every factor generating by VE is a subclique of $I_{G,\prec}$.
- Thm 7.3.4: Every maximal clique of $I_{G,\prec}$ corresponds to an intermediate term created by VE.
- e.g., $\prec = (C, D, I, H, G, S, L)$, max cliques $=$

$$\{C, D\}, \{D, I, G\}, \{G, L, S, J\}, \{G, J, H\}, \{G, I, S\}$$



Complexity of variable elimination

- Consider an ordering $\prec$.
- Define the induced width of the graph as the size of the largest factor (induced clique) minus 1:

$$W_{G,\prec} = \max_i |\psi_i| - 1$$

- Define the width of the graph as the minimal induced width:

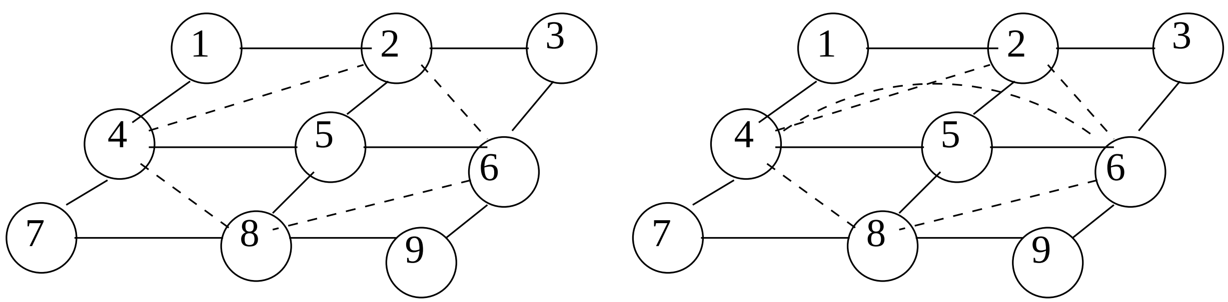
$$W_G = \min_{\prec} W_{G,\prec}$$

- e.g., width of an undirected tree is 1 (cliques $=$ edges).
- Thm: the complexity of VarElim is $O(NV^{W_G+1})$.

Chordal (triangulated) graphs

- An undirected graph is **chordal** is every loop $X_1 - X_2 - \cdots - X_k - X_1$ for $k \geq 4$ has a *chord*, i.e., an edge $X_i - X_j$ for non-adjacent i, j.
- Thm 7.3.6: every induced graph is chordal.
- The left graph is *not* chordal, because the cycle $2 - 6 - 8 - 4 - 2$ does not have any of the chords $2 - 8$ or $6 - 4$.
- The right graph *is* chordal; the max cliques are

$$\{1,2,4\}, \{2,3,6\}, \{4,7,8\}, \{6,8,9\}, \{2,4,5,6\}, \{4,5,6,8\}$$



Max cardinality search

- Thm 7.3.9: $X - Y$ is a fill-in edge iff there is a path $X - Z_1 - \cdots Z_k - Y$ s.t. $Z_i \prec X$ and $Z_i \prec Y$ for all $i = 1, \ldots, k$.
- Hence should try to find nodes X where many of their neightbors Z are already ordered, so $X \prec Z$

```
function pi = max-cardinality-search(H)
mark all nodes as unmarked
for i=N downto 1
  X = the unmarked variable with the largest
         number of marked neighbors
  pi(X) = i
  mark X
end
```

- Thm 7.3.10: if G is chordal, and $\prec =$ max cardinality ordering, then $I_{G,\prec}$ has no fill-in edges.

Triangulation

- Thm 7.3.8: finding the ordering $\prec$ which minimizes the max induced clique size, $W_{G,\prec}$, is NP-hard.
- Max cardinality ordering is only optimal if G is already triangulated.
- In practice, people use greedy (one-step-lookahead) algorithms:

```
function pi = find-elim-order-greedy(H, score-fn)
for i=1:N
  X = the node that minimizes score-fn(H, X)
  pi(X) = i
  Add edges between all neighbors of X
  Remove X from H
end
```

Triangulation: heuristic cost functions $score(H, X)$

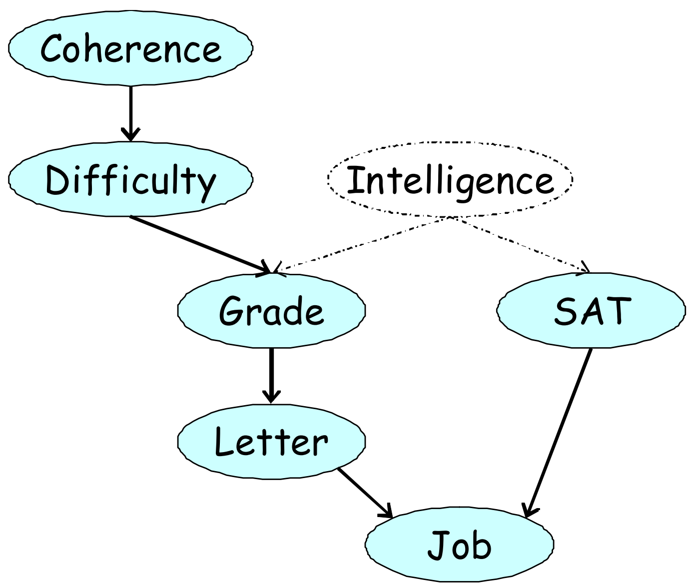
- Min-fill (min discrepancy): minimize number of fill-ins.
- Min-size: minimize size of induced clique, $|C_t|$.
- Min-weight: minimize number of states of induced clique, $\prod_{j \in C_t} |v_j|$.
- Min-weight works best in practice: a 3-clique of binary nodes is better than a 2-clique of ternary nodes, since $2^3 < 3^2$.

Conditioning

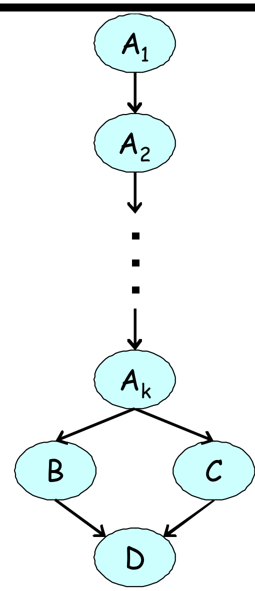
- We can instantiate some hidden variables, perform VarElim on the rest, and then repeat for each possible value, e.g.,

$$P(J) = \sum_i P(J|I = i)P(I = i)$$

- If the resulting subgraph is a tree, this is called cutset conditioning.



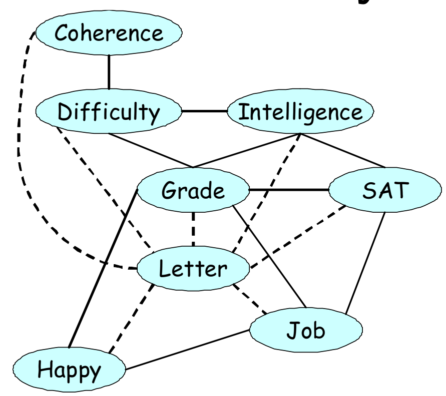
Inefficiencies of cutset conditioning



- If we condition on U, we repeatedly call VarElim once for each value of $|U|$.
- This may involved redundant work.
- Left: if we condition on A_k, we repeatedly eliminate $A_1 \to \cdots \to A_{k-1}$.
- Right: if we condition on $A_2, A_4, \ldots, A_k$, we break all the loops, but the cutset has size $V^{k/2}$, whereas VarElim would take $O(kV^3)$.

Conditioning vs VarElim is space-time tradeoff

- Thm 7.5.6: Conditioning on L takes the same amount of time as it would to do VarElim on a modified graph, in which we connect L to all other nodes (i.e., add L to every factor).



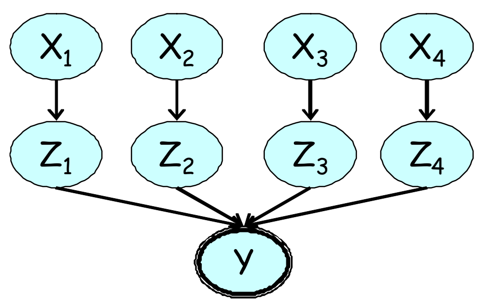
- Thm 7.5.7: The space required is that needed to store the induced cliques in the subgraph created by removing all links from L (i.e., remove L from every factor).
- Hence conditioning takes less space but more time.

Exploiting local structure

- VarElim exploits the factorization properties implied by the graph to push sums inside products.
- Hence VarElim works for any kind of factor.
- However, some factors have local structure which can be exploited to further speed up inference.
- Two main methods:
  1. Make local structure graphically explicit (by adding extra nodes), then run stand VarElim on expanded graph; **or**
  2. Implement the $\sum$ and $\times$ operators for structured factors in a special way.
- We will focus on the first method, since it can be used to speed up any graph-based inference engine.
- David Poole has focused on the second method (structured VarElim).

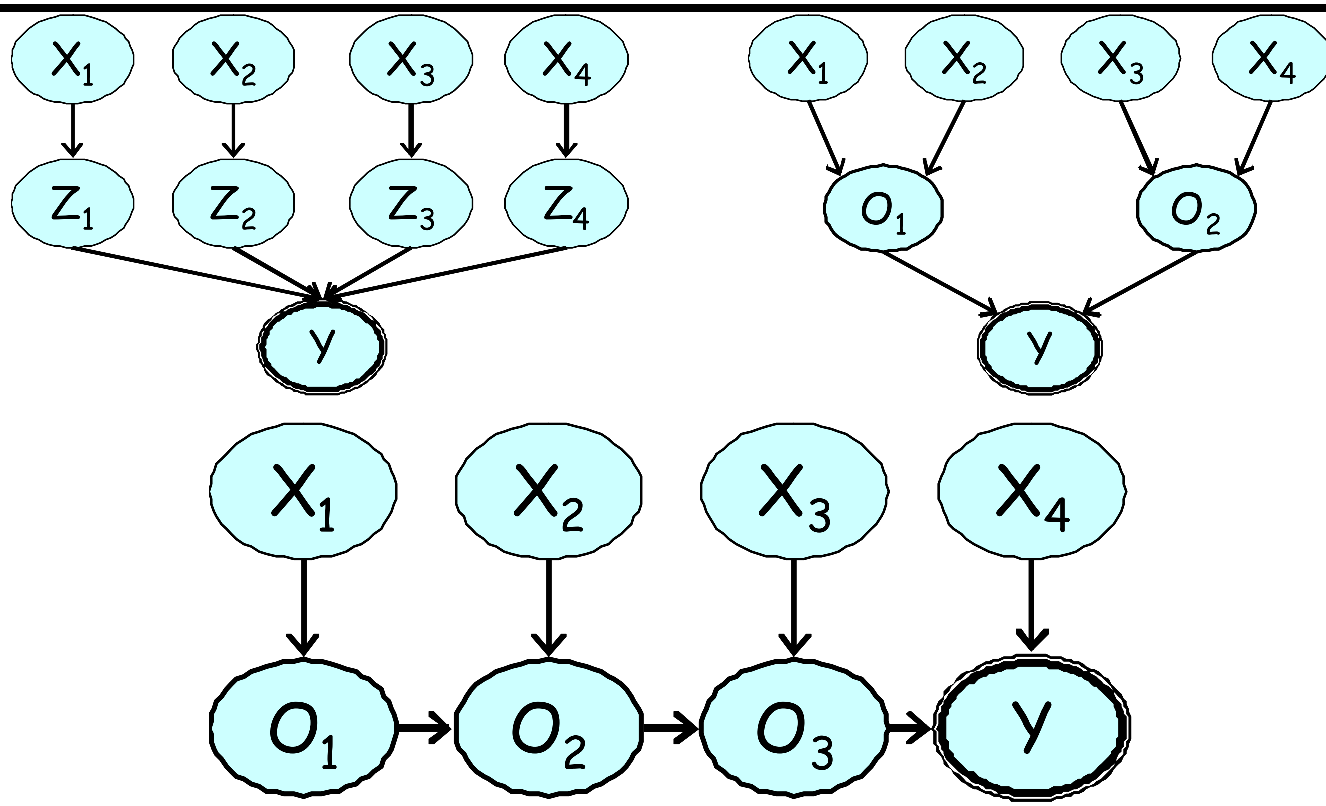
Independence of causal influence (ICI)

- In general, a node with k parents creates a factor of size V^{k+1} to represent its CPD $P(Y|X_{1:k})$.
- Hence it takes $O(V^{k+1})$ time to eliminate this clique, and there are $O(V^{k+1})$ parameters to learn.
- If the parents X_i do not interact with each other (only with the child), the family can be eliminated in $O(k)$ time, and there are only $O(k)$ parameters to learn.
- e.g., noisy-or, generalized linear model



$$P(Y = 0|X_{1:4}) = q_0 \prod_{i=1}^{4} q_i^{X_i}$$

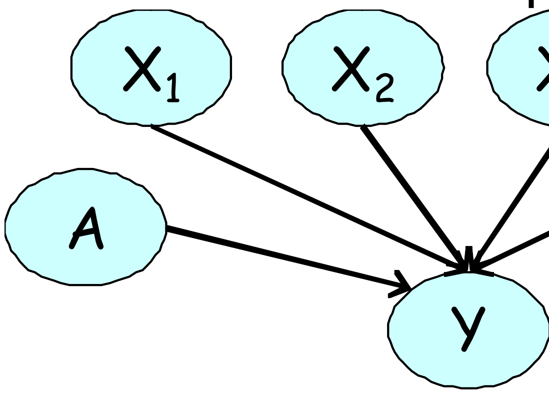
Exploiting Independence of Causal Influence (ICI)

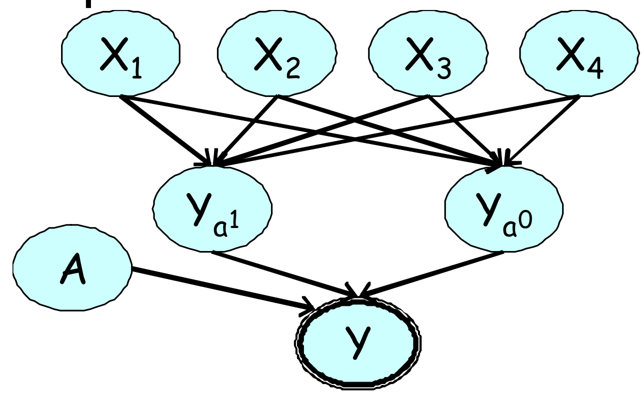


- Assumes deterministic function can be represented by $f(x_{1:k}) = x_1 \oplus x_2 \oplus \cdots \oplus x_k$ where $\oplus$ is commutative and asssociative.
- State-space of tree is $O(|Z|^3)$, chain $O(|Z|^2|X|)$.

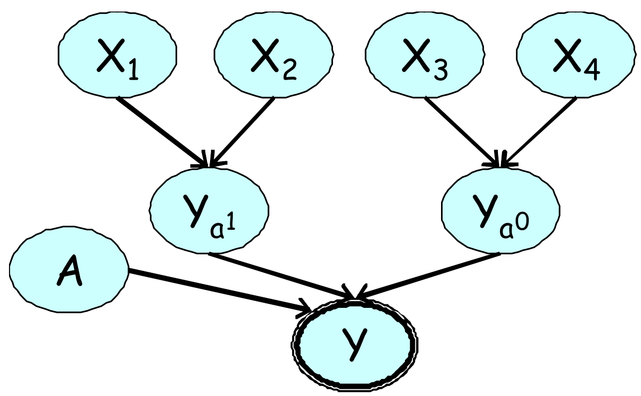
Exploiting Context specific independence (CSI)

- Suppose $P(Y|A, X_{1:4})$ is represented as a decision tree. Then we can make the structure explicit using multiplexer nodes.

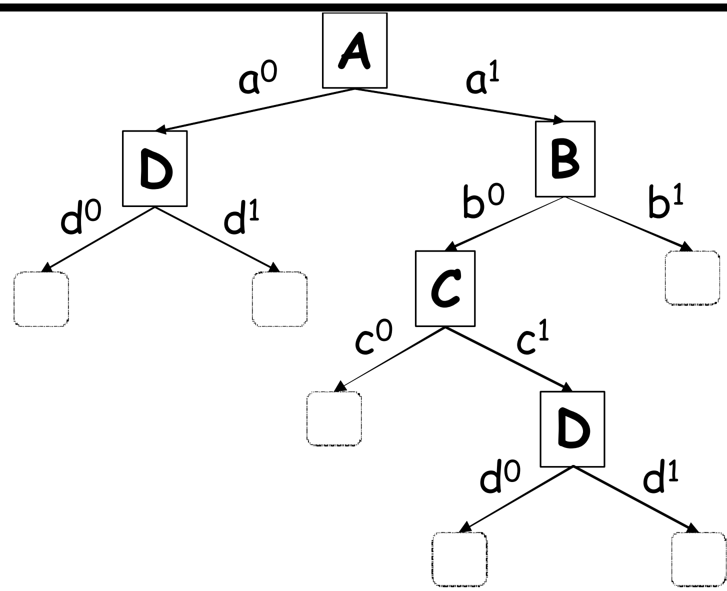


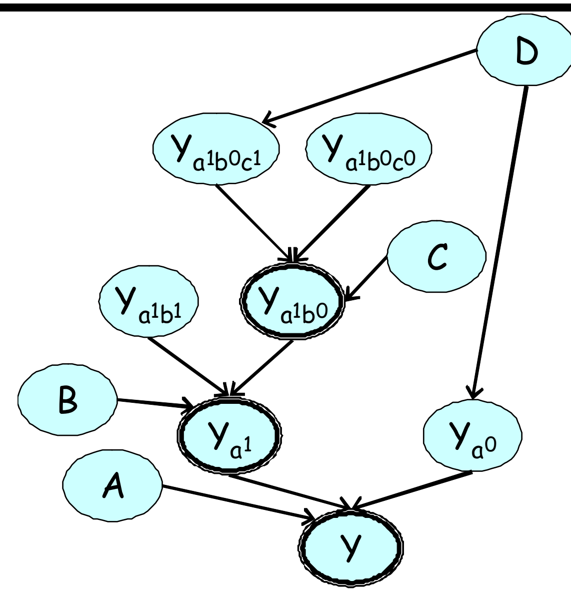


- If $Y \perp X_3, X_4 | A = 1$ and $Y \perp X_1, X_2 | A = 0$, then



More complex example





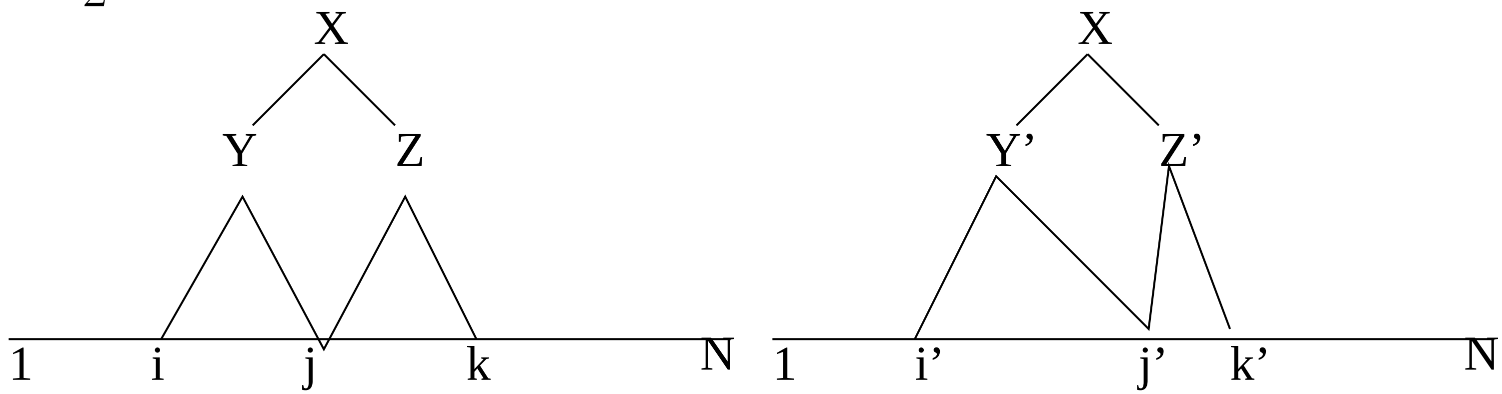
- (Recursive) conditioning provides a simpler method of exploiting CSI.
- Project idea: implement both methods and compare.

Stochastic context free grammars (SCFGs)

- If you construct a graphical model given a grammar and a sentence of length N, the treewidth is $O(N)$, suggesting inference takes $O(2^N)$.
- However, we can do exact inference using the inside-outside algorithm in $O(N^3)$ time.
- The reason is that there is a lot of CSI.

Stochastic context free grammars (SCFGs)

- Represent production rule $X \to YZ$ by a binary variable R_1, and $X \to Y'Z'$ by R_2. If $R_1 = 1$, the structure of the graph is different than if $R_2 = 1$.



- See "Case-factor diagrams for structured probabilistic modeling", McAllester, Collins, Pereira, UAI 2004.

- Project idea: implement this algorithm and compare to inside-outside algorithm.