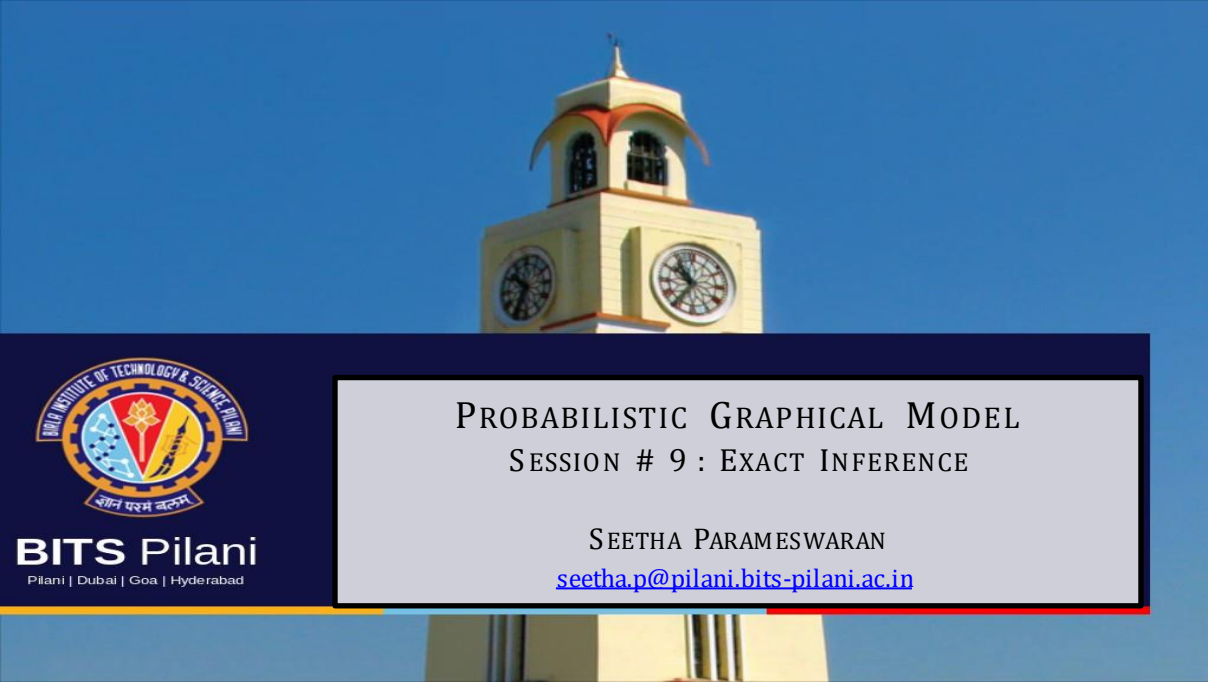


BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Probabilistic Graphical Model

Session # 9 : Exact Inference

Seetha Parameswaran
seetha.p@pilani.bits-pilani.ac.in

Table of Contents

Inference Queries

Queries about the distribution P_Φ include queries about marginal probabilities of variables and queries about the partition function Z.

1. Conditional Probability Query
   - Compute $P(Y|E=e)$
   - These queries are useful for reasoning patterns, including explanation, prediction, intercausal reasoning.
   - Applications include medical diagnosis and pedigree analysis.
2. Maximum a Posteriori (MAP) Queries

Table of Contents

Conditional Probability Query

- Query variable = Y
- Evidence = e
- Task is compute $P(Y|E = e)$

$$P(Y|E = e) = \frac{P(Y, e)}{P(e)}$$

$$W = X - Y - E$$

main distribution

$$P(Y, e) = \sum_{w} P(y, e, w)$$

$$P(e) = \sum_{y} P(y, e)$$

Inference Queries and NP hardness

The following queries are NP hard

Definition

1. Given a P_Φ, a variable X, and a value $x \in Val(X)$; compute $P_\Phi(X = x)$. (Exact inference)
2. Given a P_Φ, a variable X, a value $x \in Val(X)$, an observation $e \in Val(E)$ and an absolute error $\epsilon = 0.5$; find an estimate p such that $|P_\Phi(X = x|E = e) - p| \leq \epsilon$. (Approximate inference) NP-hard

- Both are worst-case queries.
- Both are solved in exponential time and are intractable.
- For general-cases, exact or approximate inference algorithms for graphical models do better.

3-SAT $\xrightarrow{\text{polynomial}}$ Exact Inference

Inference $\xrightarrow{\text{polynomial time}}$ 3-SAT

Proof Sketch

Why is Exact Inference NP-hard? In decision form

Reduction from 3-SAT $F = (x_1' + x_2 + x_3')(x_1' + x_4 + x_5)(x_5' + x_6 \ldots)$

A 3-SAT formula can be encoded as a Bayesian network such that the 3-SAT formula is satisfiable if and only if we can solve the exact inference question.

3-SAT $\xrightarrow{\text{polynomial time}}$ Bayesian inference

Bayesian inference $\xrightarrow{\text{polynomial time}}$ 3-SAT

Proof Sketch

Given a Bayesian network B over X , a variable X ∈ X , and a value x ∈ Val(X), decide whether $P_B(X = x) > 0$.

This is the BN-Pr-DP problem. We have the result below.

Theorem 9.1 The decision problem BN-Pr-DP is N P-complete.

Proof Sketch

$$\prod P\left(X_i / \text{parent}(X_i)\right) = P(\xi)$$

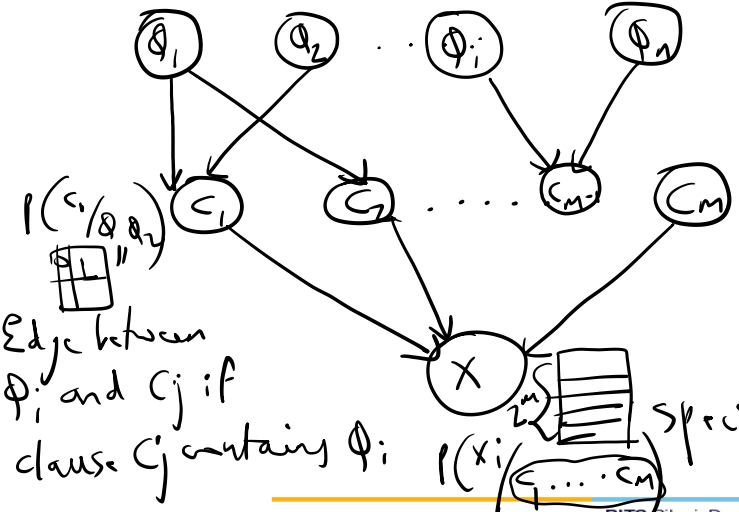
$$P(X = x) = \sum P(\xi_i)$$

Why is BN-Pr-DP in NP?

- We guess an assignment ξ to the variables in the Bayesian network
- We check if $X = x$ in ξ and whether $P(\xi) > 0$
- Computing $P(\xi)$ can be done using the Bayesian Network JPDF formula in linear time
- One of the guesses ξ may $P(\xi) > 0 \iff P(X = x) > 0$

$M \rightarrow (\phi_1 + \phi_2 + \phi_3')(\phi_1' + \phi_2 + \phi_3)$

Main Reduction



[variables in propositional formula]

[clause variables]

$P(C_i / \phi_i \phi_j \phi_k) = 1$ or 0

Because when we specify an assignment, $C_i = 0$ or $C_i = 1$

Edge between ϕ_i and C_j if clause C_j contains ϕ_i

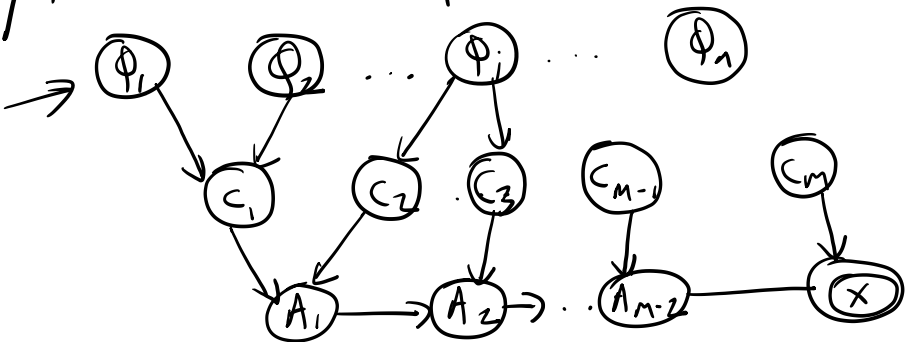
Main Reduction

The reduction in the previous slide appears to work in that $P(X = x') > 0 \Leftrightarrow$ the original 3-SAT formula is satisfiable.

However $P(X / c_1, c_2, \ldots c_m)$ is exponentially large and we can therefore not build a Bayesian Network in polynomial time from ϕ, the 3-SAT formula

Main Reduction

To get around the problem, modify the diagram



A_1 = AND of C_1 and C_2, ... X is 'AND' of A_{M-2} and C_M

Main Reduction

N – a each variable has atmost 3 parents,

S – B_ϕ is polynomial in the size of ϕ.

X has value 1 if and only if all clauses are satisfied

N – w $P_B(x' | q_1, q_2 \cdots q_n) = 1$ if and only if $q_1, q_2, \cdots q_n$ is a satisfying assignment for ϕ.

Main Reduction

We take $P(q_i') = \frac{1}{2}$ for each q_i $\therefore P(q_1 q_2 \cdots q_n)$

$= \frac{1}{2^n}$

$P_{B_\phi}(x') = \frac{\#\text{ satisfying assignments to } \phi}{2^n}$

$\therefore P_{B_\phi}(x') > 0 \Leftrightarrow \phi$ is satisfiable

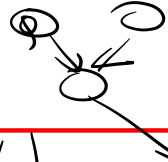
Why is Approximate Inference Hard?

An estimate ρ has absolute error ϵ for $P(y|e)$ if $|P(y|e) - \rho| < \epsilon$

The following problem is NP-hard for any $\epsilon \in (0, 1/2)$: Given a Bayesian network B over X , a variable $X \in X$, a value $x \in Val(X)$, and an observation $E = e$ for $E \subset X$ and $e \in Val(E)$, find a number ρ that has absolute error ϵ for $P_B(X = x \mid e)$.

Reduction

- Again from 3-SAT
- We construct a Bayesian Network as before
- Assume that we have the ability to approximate any conditional probability to within 0.5
- We will show that we can use this ability to decide if an arbitrary formula is satisfiable.



Reduction

We begin by computing $P(\phi'/x')$ → we can compute these probabilities to within 0.5 of their real values.

We pick the value v_1 for ϕ' that is most likely given x' ie we instantiate $\phi' = v_1$

Similarly we pick values for the other variables

Reduction

Finally we have an assignment $v_1, v_2, \ldots v_n$ to all the ϕ_i's.

We claim that that this is a satisfying assignment to the 3-SAT formula ϕ if and only if it is satisfiable.

Reduction

- If ϕ is not satisfiable it is obvious that $v_1, v_2 \ldots v_n$ is not a satisfying assignment
- If ϕ is satisfiable with both $\phi_1 = q'$ and $\phi_1 = q^0$ then obviously we have a satisfying assignment with $\phi_1 = v'$

Interesting case: We select $\phi_1 = v_1$, but all satisfying assignments have $\phi_1 = \neg v_1$

Reduction

We show that this interesting case is not possible

If it is possible it must be true that

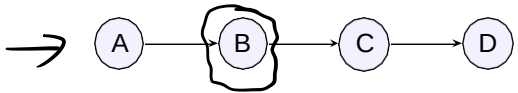
$P(Q_1 = v_1 / x') = 0$ but then we must have determined this probability to be < 0.5 (since 0.5 is maximum absolute error), and that the probability of $P(Q_1 = \neg v_1 / x') > 0.5$

Reduction

Since we take the max $\left(P(\phi_1 = v_1 / x'), P(\phi_1 = \neg v_1 / x')\right)$ we would have selected the right assignment for ϕ_1

We can continue on in this fashion for other variables and finally pick a satisfying assignment for ϕ.

Conditional Queries



- Compute $P(B)$

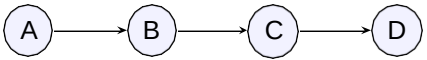
$$P(B) = \sum_a P(a)P(B|a)$$ compute the distribution

$P(A)$ = CPD for A and A has k values

$P(b|a)$ = CPD for B and B have m values

- To compute $P(b)$, multiply $P(b|a)$ with $P(a)$ for each of k values of A and then add them up. i.e. k multiplication and $k-1$ additions.
- Repeat this process for each of m values of b.
- Complexity is $O(km)$.

Conditional Queries



- Compute $P(C)$

$$P(C) = \sum_{b} P(b)P(C|b)$$

$P(c|b)$ = CPD for C

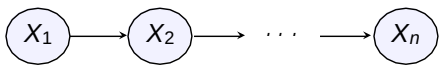
$P(B)$ = Use the distribution computed in the previous step.

- Similarly compute $P(D)$.

$$P(D) = \sum_{c} P(c)P(D|c)$$

$P(d|c)$ = CPD for D

Conditional Queries



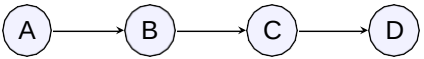
- In general, the given chain with n variables, each variable X_i taking k values,

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

- Computation of each distribution $P(X_{i+1})$ from $P(X_i)$ takes $O(k^2)$.
- For n variables in the chain, the total cost = $O(nk^2)$.
- For an exponential size $O(n^k)$ of joint distribution, perform inference in linear time $O(nk^2)$.

Table of Contents

Sum Product



$$P(A, B, C, D) = P(A) \cdot P(B|A) \cdot P(C|B) \cdot P(D|C)$$

Compute $P(D)$. Let each variable be binary.

$$P(D) = \sum_{a,b,c} P(A)\ P(B|A)\ P(C|B)\ P(D|C)$$

Conditional probability inferences are called sum product, as we sum over product of factors.

$P(D = d^1) =$

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

8

$P(D = d^2) =$

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

8

D = d'
24 multiplications, 7 additions

Sum Product

$$P(D) = P(a^1)\ P(b^1|a^1)\ P(c^1|b^1)\ P(d^1|c^1)$$
$$+\ P(a^2)\ P(b^1|a^2)\ P(c^1|b^1)\ P(d^1|c^1)$$
$$+\ P(a^1)\ P(b^2|a^1)\ P(c^1|b^2)\ P(d^1|c^1)$$
$$+\ P(a^2)\ P(b^2|a^2)\ P(c^1|b^2)\ P(d^1|c^1)$$
$$+\ P(a^1)\ P(b^1|a^1)\ P(c^2|b^1)\ P(d^1|c^2)$$
$$+\ P(a^2)\ P(b^1|a^2)\ P(c^2|b^1)\ P(d^1|c^2)$$
$$+\ P(a^1)\ P(b^2|a^1)\ P(c^2|b^2)\ P(d^1|c^2)$$
$$+\ P(a^2)\ P(b^2|a^2)\ P(c^2|b^2)\ P(d^1|c^2)$$
$$+\ P(a^1)\ P(b^1|a^1)\ P(c^1|b^1)\ P(d^2|c^1)$$
$$+\ P(a^2)\ P(b^1|a^2)\ P(c^1|b^1)\ P(d^2|c^1)$$
$$+\ P(a^1)\ P(b^2|a^1)\ P(c^1|b^2)\ P(d^2|c^1)$$
$$+\ P(a^2)\ P(b^2|a^2)\ P(c^1|b^2)\ P(d^2|c^1)$$
$$+\ P(a^1)\ P(b^1|a^1)\ P(c^2|b^1)\ P(d^2|c^1)$$
$$+\ P(a^2)\ P(b^1|a^2)\ P(c^2|b^1)\ P(d^2|c^1)$$
$$+\ P(a^1)\ P(b^2|a^1)\ P(c^2|b^2)\ P(d^2|c^1)$$
$$+\ P(a^2)\ P(b^2|a^2)\ P(c^2|b^2)\ P(d^2|c^1)$$

Transformation 1

$$P(D) \equiv [P(a^1)\ P(b^1|a^1) + P(a^2)\ P(b^1|a^2)]\ P(c^1|b^1)\ P(d^1|c^1)$$
$$+\ [P(a^1)\ P(b^2|a^1) + P(a^2)\ P(b^2|a^2)]\ P(c^1|b^2)\ P(d^1|c^1)$$
$$+\ [P(a^1)\ P(b^1|a^1) + P(a^2)\ P(b^1|a^2)]\ P(c^2|b^1)\ P(d^1|c^2)$$
$$+\ [P(a^1)\ P(b^2|a^1) + P(a^2)\ P(b^2|a^2)]\ P(c^2|b^2)\ P(d^1|c^2)$$

$$P(\theta = d^2) = [P(a^1)\ P(b^1|a^1) + P(a^2)\ P(b^1|a^2)]\ P(c^1|b^1)\ P(d^2|c^1)$$
$$+\ [P(a^1)\ P(b^2|a^1) + P(a^2)\ P(b^2|a^2)]\ P(c^1|b^2)\ P(d^2|c^1)$$
$$+\ [P(a^1)\ P(b^1|a^1) + P(a^2)\ P(b^1|a^2)]\ P(c^2|b^1)\ P(d^2|c^2)$$
$$+\ [P(a^1)\ P(b^2|a^1) + P(a^2)\ P(b^2|a^2)]\ P(c^2|b^2)\ P(d^2|c^2)$$

Sum Product

Transformation 1

$$P(D=d^1) = [P(a^1)\ P(b^1|a^1) + P(a^2)\ P(b^1|a^2)]\ P(c^1|b^1)\ P(d^1|c^1)$$
$$+ [P(a^1)\ P(b^2|a^1) + P(a^2)\ P(b^2|a^2)]\ P(c^1|b^2)\ P(d^1|c^1)$$
$$+ [P(a^1)\ P(b^1|a^1) + P(a^2)\ P(b^1|a^2)]\ P(c^2|b^1)\ P(d^1|c^2)$$
$$+ [P(a^1)\ P(b^2|a^1) + P(a^2)\ P(b^2|a^2)]\ P(c^2|b^2)\ P(d^1|c^2)$$

$$P(D=d^2) = [P(a^1)\ P(b^1|a^1) + P(a^2)\ P(b^1|a^2)]\ P(c^1|b^1)\ P(d^2|c^1)$$
$$+ [P(a^1)\ P(b^2|a^1) + P(a^2)\ P(b^2|a^2)]\ P(c^1|b^2)\ P(d^2|c^1)$$
$$+ [P(a^1)\ P(b^1|a^1) + P(a^2)\ P(b^1|a^2)]\ P(c^2|b^1)\ P(d^2|c^2)$$
$$+ [P(a^1)\ P(b^2|a^1) + P(a^2)\ P(b^2|a^2)]\ P(c^2|b^2)\ P(d^2|c^2)$$

Transformation 2 Let

$$\tau_1(b^1) = [P(a^1)\ P(b^1|a^1) + P(a^2)\ P(b^1|a^2)]$$
$$\tau_1(b^2) = P(a^1)\ P(b^2|a^1) + P(a^2)\ P(b^2|a^2)$$

$$P(D=d^1) = \tau_1(b^1)\ P(c^1|b^1)\ P(d^1|c^1)$$
$$+ \tau_1(b^2)\ P(c^1|b^2)\ P(d^1|c^1)$$
$$+ \tau_1(b^1)\ P(c^2|b^1)\ P(d^1|c^2)$$
$$+ \tau_1(b^2)\ P(c^2|b^2)\ P(d^1|c^2)$$

$$P(D=d^2) = \tau_1(b^1)\ P(c^1|b^1)\ P(d^2|c^1)$$
$$+ \tau_1(b^2)\ P(c^1|b^2)\ P(d^2|c^1)$$
$$+ \tau_1(b^1)\ P(c^2|b^1)\ P(d^2|c^2)$$
$$+ \tau_1(b^2)\ P(c^2|b^2)\ P(d^2|c^2)$$

Sum Product

Transformation 3

$$P(D) = \tau_1(b^1)\ P(c^1|b^1)\ P(d^1|c^1) + \tau_1(b^2)\ P(c^1|b^2)\ P(d^1|c^1) + \tau_1(b^1)\ P(c^2|b^1)\ P(d^1|c^2) + \tau_1(b^2)\ P(c^2|b^2)\ P(d^1|c^2)$$

$$P(D = d^2) = \tau_1(b^1)\ P(c^1|b^1)\ P(d^2|c^1) + \tau_1(b^2)\ P(c^1|b^2)\ P(d^2|c^1) + \tau_1(b^1)\ P(c^2|b^1)\ P(d^2|c^2) + \tau_1(b^2)\ P(c^2|b^2)\ P(d^2|c^2)$$

Transformation 4

Let

$$\tau_2(c^1) = \tau_1(b^1)P(c^1|b^1) + \tau_1(b^2)\ P(c^1|b^2)$$

$$\tau_2(c^2) = \tau_1(b^1)P(c^2|b^1) + \tau_1(b^2)\ P(c^2|b^2)$$

$$P(D) = \tau_2(c^1)\ P(d^1|c^1) + \tau_2(c^2)\ P(d^1|c^2)$$

$$P(D = d^2) = \tau_2(c^1)\ P(d^2|c^1) + \tau_2(c^2)\ P(d^2|c^2)$$

Sum Product

Computations:

- Joint distribution

  4 Binary varaibles * 3 variables eliminated
  $= 2^4 * 3 = 16 * 3 = 48$ multiplications
  +14 additions (7 for each value in D)

- Transformations

  $\tau_1(B) \Longrightarrow$ 4 multiplications and 2 additions

  $\tau_2(C) \Longrightarrow$ 4 multiplications and 2 additions

  $P(D) \Longrightarrow$ 4 multiplications and 2 additions

  Total = 4 + 2 + 4 + 2 + 4 + 2 = 18

Dynamic Programming:

- Because of the structure of the Bayesian network, some subexpressions in the joint depend only on a small number of variables.
- By computing these expressions once and caching the results, we can avoid generating them exponentially many times.
- Dynamic programming “inverts” the order of computation; performing it inside out instead of outside in.

Sum Product

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

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

$$= \sum_C P(D|C) \sum_B \tau_1(B)\,P(C|B)$$

$$= \sum_C \tau_2(C)\,P(D|C)$$

$$\psi_1(A, B) = P(A)P(B|A)$$

$$\tau_1(B) = \sum_A \psi_1(A, B)$$

$$\psi_2(B, C) = \tau_1(B)P(C|B)$$

$$\tau_2(C) = \sum_B \psi_2(B, C)$$

Product of factors. Then sum of the new factors.

Table of Contents

Variable Elimination Algorithm

$$P(A, B, C, D) = \phi_A \cdot \phi_B \cdot \phi_C \cdot \phi_D$$

$$P(D) = \sum_C \sum_B \sum_A \phi_A \cdot \phi_B \cdot \phi_C \cdot \phi_D$$

$$= \sum_C \sum_B \phi_C \cdot \phi_D \cdot \left(\sum_A \phi_A \cdot \phi_B \right)$$

$$= \sum_C \phi_D \cdot \left(\sum_B \phi_C \cdot \left(\sum_A \phi_A \cdot \phi_B \right) \right)$$

Variable Elimination given Evidence

$$P(Y|E=e) = \frac{P(Y,e)}{P(e)} \qquad W = X - Y - E$$

$$P(Y,e) = \sum_W P(Y,W,e)$$

$$= \sum_W \frac{1}{Z} \prod_k \phi_k(D_k, e)$$

$$= \sum_W \frac{1}{Z} \prod_k \phi_k^0(D_k^0)$$

$$P(e) = \sum_y P(y,e)$$

$$= \sum_Y \sum_W \frac{1}{Z} \prod_k \phi_k^0(D_k^0)$$

Variable Elimination Algorithm

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 $\boldsymbol{Z}$
)

1 Let $Z_1, \ldots, Z_k$ be an ordering of $\boldsymbol{Z}$ such that
2 $Z_i \prec Z_j$ if and only if $i < j$
3 **for** $i = 1, \ldots, k$
4 $\Phi \leftarrow$ Sum-Product-Eliminate-Var(Φ, Z_i)
5 $\phi^* \leftarrow \prod_{\phi \in \Phi} \phi$
6 **return** ϕ^*

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

1 $\Phi' \leftarrow \{\phi \in \Phi : Z \in Scope[\phi]\}$
2 $\Phi'' \leftarrow \Phi - \Phi'$
3 $\psi \leftarrow \prod_{\phi \in \Phi'} \phi$
4 $\tau \leftarrow \sum_Z \psi$
5 **return** $\Phi'' \cup \{\tau\}$

[contain Z]

factors that don't contain Z

Variable Elimination Algorithm given Evidence

Algorithm 9.2 Using Sum-Product-VE **for computing conditional probabilities**

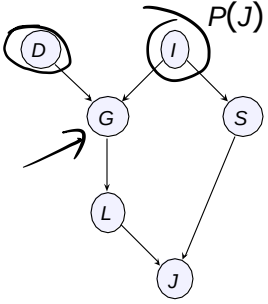
Procedure Cond-Prob-VE (
  $\mathcal{K}$, // A network over $\mathcal{X}$
  $\boldsymbol{Y}$, // Set of query variables
  $\boldsymbol{E} = \boldsymbol{e}$ // Evidence
)

1 $\Phi \leftarrow$ Factors parameterizing $\mathcal{K}$
2 Replace each $\phi \in \Phi$ by $\phi[\boldsymbol{E} = \boldsymbol{e}]$
3 Select an elimination ordering $\prec$
4 $\boldsymbol{Z} \leftarrow = \mathcal{X} - \boldsymbol{Y} - \boldsymbol{E}$
5 $\phi^* \leftarrow$ Sum-Product-VE($\Phi, \prec, \boldsymbol{Z}$)
6 $\alpha \leftarrow \sum_{\boldsymbol{y} \in Val(\boldsymbol{Y})} \phi^*(\boldsymbol{y})$
7 **return** α, ϕ^*

Variable Elimination in BN

$$P(D, I, G, S, L, J) = \phi_D(D) \cdot \phi_I(I) \cdot \phi_G(G, I, D)\phi_S(S, I) \cdot \phi_L(L, G) \cdot \phi_J(J, L, S)$$

$$P(J) = \sum_{D,I,G,S,L} P(D, I, G, S, L, J)$$

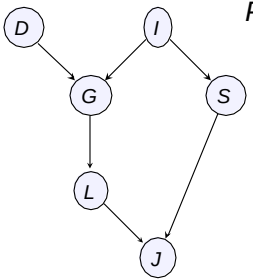


$$= \sum_{D,I,G,S,L} \phi_J(J, L, S) \cdot \phi_L(L, G) \cdot \phi_S(S, I) \cdot \phi_I(I) \cdot \phi_G(G, I, D) \cdot \phi_D(D)$$

$$= \sum_{I,G,S,L} \phi_J(J, L, S) \cdot \phi_L(L, G) \cdot \phi_S(S, I) \cdot \phi_I(I) \cdot \sum_{D} \phi_G(G, I, D) \cdot \phi_D(D)$$

$$= \sum_{I,G,S,L} \phi_J(J, L, S) \cdot \phi_L(L, G) \cdot \phi_S(S, I) \cdot \phi_I(I) \cdot \tau_1(G, I)$$

Variable Elimination in BN

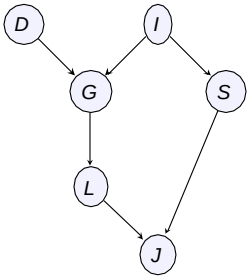


$$P(J) = \sum_{I,G,S,L} \phi_J(J, L, S) \cdot \phi_L(L, G) \cdot \phi_S(S, I) \cdot \phi_I(I) \cdot \tau_1(G, I)$$

$$= \sum_{G,S,L} \phi_J(J, L, S) \cdot \phi_L(L, G) \cdot \sum_{I} \phi_S(S, I) \cdot \phi_I(I) \cdot \tau_1(G, I)$$

$$= \sum_{G,S,L} \phi_J(J, L, S) \cdot \phi_L(L, G) \cdot \tau_2(G, S)$$

Variable Elimination in BN



$$P(J) = \sum_{S,L} \phi_J(J, L, S) \cdot \sum_{G} \phi_L(L, G) \cdot \tau_2(G, S)$$

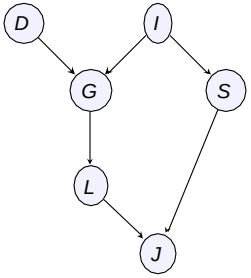
$$= \sum_{G,S,L} \phi_J(J, L, S) \cdot \tau_3(L, S)$$

$$= \sum_{S,L} \phi_J(J, L, S) \cdot \tau_3(L, S)$$

$$= \sum \tau_4(J, L)$$

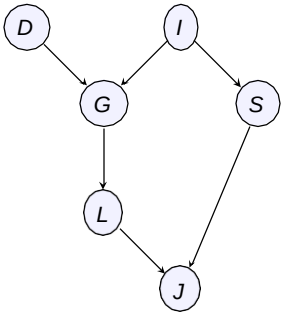
$$= \tau_5(J)$$

Variable Elimination in BN



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

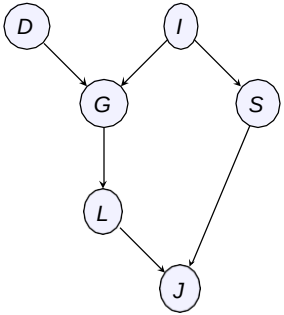
Variable Elimination in BN given Evidence



$$P(J) = \sum_{D,I,G,S,L} \phi_D(D)\,\phi_I(I)\,\phi_G(G, I, D)\,\phi_S(S, I)\,\phi_L(L, G)\,\phi_J(J, L, S)$$

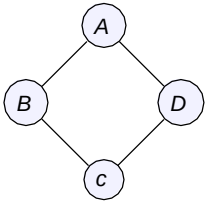
$$P(J, I = i) = \sum_{D,I,G,S,L} \phi_D(D)\,\phi_I(I = i)\,\phi_G(G, I = i, D)\,\phi_S(S, I = i)\,\phi_L(L, G)\,\phi_J(J, L, S)$$

Variable Elimination in BN given Evidence



Step	Variable eliminated	Factors used	Variables involved	New factor
1	D	$\phi_D(D) \cdot \phi_G(G, I = i, D)$	G, D	$\tau_1^q(G)$
2	G	$\tau_1^q(G) \cdot \phi_L(L, G)$	G, L	$\tau_2^q(L)$
3	S	$\phi_J(J, L, S) \cdot \phi_S(S, I = i)$	J, L, S	$\tau_3^q(J, L)$
4	L	$\tau_2^q(L) \cdot \tau_3^q(J, L)$	J, L	$\tau_4^q(J)$

Variable Elimination in MN



$$\tilde{P} = \phi_1(A, B) \cdot \phi_2(B, C) \cdot \phi_3(C, D) \cdot \phi_4(D, A)$$

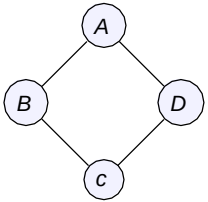
$$\tilde{P}(D) = \sum_{A,B,C} \tilde{P}$$

$$= \sum_{A,B,C} \phi_2(B, C) \cdot \phi_3(C, D) \cdot \phi_1(A, B) \cdot \phi_4(D, A)$$

$$= \sum_{B,C} \phi_2(B, C) \cdot \phi_3(C, D) \cdot \sum_{A} \phi_1(A, B) \cdot \phi_4(D, A)$$

$$= \sum_{B,C} \phi_2(B, C) \cdot \phi_3(C, D) \cdot \tau_1(B, D)$$

Variable Elimination in MN



$$\tilde{P}(D) = \sum_{B,C} \phi_2(B, C) \cdot \phi_3(C, D) \cdot \tau_1(B, D)$$

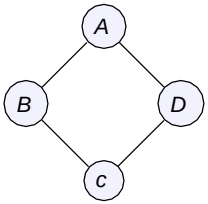
$$= \sum_{C} \phi_3(C, D) \sum_{B} \phi_2(B, C) \cdot \tau_1(B, D)$$

$$= \sum_{C} \phi_3(C, D)\, \tau_2(C, D)$$

$$= \tau_3(D)$$

Renormalize using partition function.

Variable Elimination in MN



Step	Variable eliminated	Factors used	Variables involved	New factor
1	A	$\phi_1(A, B) \cdot \phi_4(D, A)$	A, B, D	$\tau_1(B, D)$
2	B	$\phi_2(B, C) \cdot \tau_1(B, D)$	B, C, D	$\tau_2(C, D)$
3	C	$\phi_3(C, D) \cdot \tau_2(C, D)$	C,D	$\tau_4(D)$

Complexity of Variable Elimination Algorithm

- Number of random variables $= n$
- Number of factors $= m$
- For Bayesian Network; $m \leq n$ and larger for Markov Network.
- Number of entries (rows) in factor $\psi_i = N_i$.
- $N_{max} = \max(N_i)$
- VE Algo Step 1: Factor product

$$\psi_k(X_k) = \prod_{i=1}^{m_k} \phi_i \qquad Cost = N_k(m_k - 1) = O(mN_{max})$$

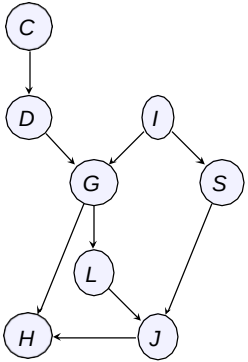
- VE Algo Step 2: Factor marginalization

$$\tau_k(X_k - Z) = \sum_{Z} \psi_k(X_k) \qquad Cost = N_k = O(nN_{max})$$

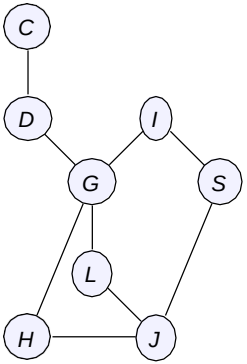
- Cost depends on N_{max}, size of intermediate facotrs τ_k; giving exponential growth in the number of variables in a factor.

Graph Analysis of VE Algorithm

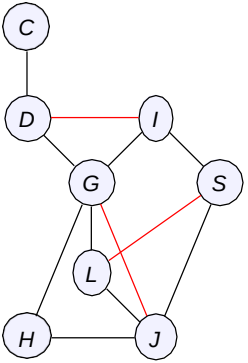
Bayesian Network



Directed edged converted into Undirected edges.

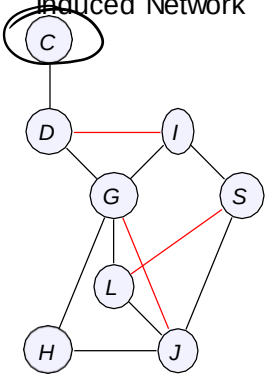


Apply Moralization = Induced Network

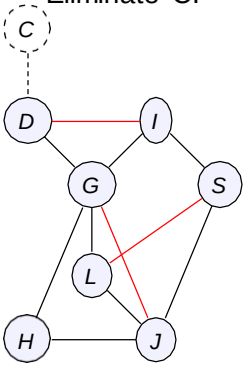


Graph Analysis of VE Algorithm

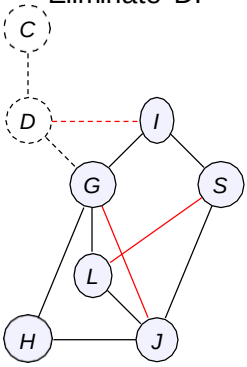
Induced Network



Eliminate C.



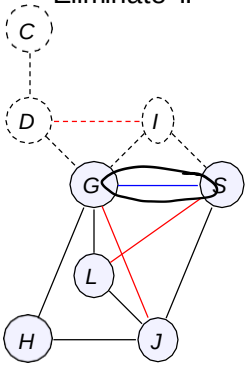
Eliminate D.



$$\tau_1(D) = \sum_C \phi(C)\phi(C, D) \qquad \tau_2(G, I) = \sum_D \phi(G, I, D)\tau_1(D)$$

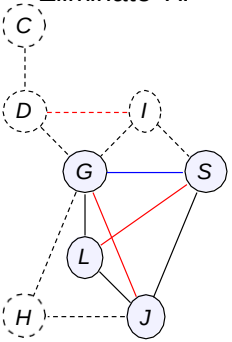
Graph Analysis of VE Algorithm

Eliminate I.



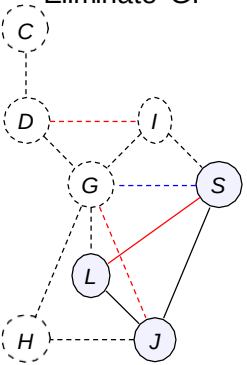
$\tau_3(G,S) = \sum_I \varphi(S,I)\varphi(I)\tau_2(G,I)$

Eliminate H.



$\tau_4(G,J) = \sum_H \varphi(H,G,J)$

Eliminate G.



$\tau_5(J,L,S) = \sum_G \varphi(L,G)\tau_3(G,S)\tau_4(G,J)$

Induced Graph

- An induced graph $I_{\varphi,\alpha}$ over factors ϕ and ordering α is
  - an undirected graph
  - X_i and X_j are connected if they appear in the same factor in an iteration of variable elimination using α as ordering.
- Induced graphs considers the moralizations and fill edges.
- Fill edges are edges that are introduced between variables, that become connected directly after an elimination step.
- Eg of fill edge - between G and S after I was eliminated.

Induced Graph

Theorem

1. *Every factor produced during variable elimination is a clique in the induced graph.*

- Clique is a maximal fully connected subgraph.

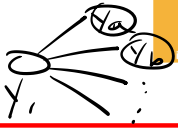
2. Every maximal clique in $I_{\phi,\prec}$ is the scope of some intermediate factor in the computation

Proof Sketch

Consider some maximal clique $Y = \{Y_1, Y_2, \ldots Y_k\}$

Let Y_1 be the first vertex in the clique to be eliminated

Since Y is a clique there is an edge between Y_1 and each Y_i in the clique



Proof Sketch

- After Y_i is eliminated it does not appear in any other factors which means that there is no chance of any edges being added to Y_i after it is eliminated
- So all edges between Y_i and Y_j are already present prior to elimination of Y_i which means there is already a factor containing Y_i and Y_j

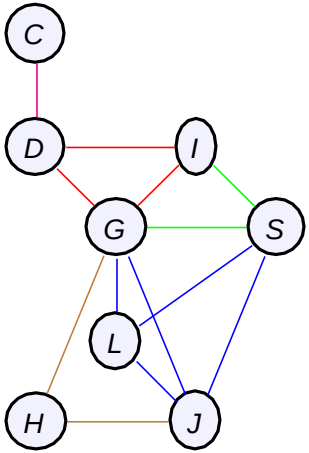
Proof Sketch

Eliminating Y_1 means that all factors containing Y_1 have to be multiplied together

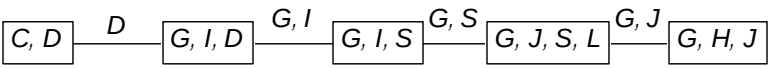
This product step results in a factor ψ that contains $Y_1, Y_{n_1} \cdots Y_k$ and no other variables

because otherwise we would have a larger maximal clique

Induced Graph and Clique Tree



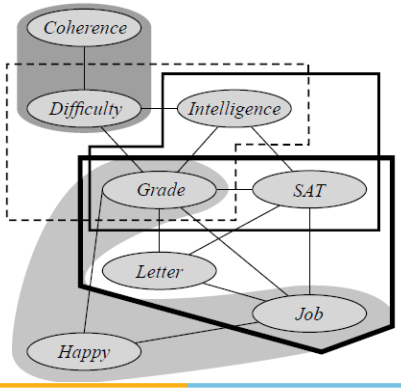
- $\tau(G, I) = \sum_C \phi(C)\phi(C, D)$
- $\tau_2(G, I) = \sum_D \phi(G, I, D)\tau_1(D)$
- $\tau_3(G, S) = \sum_I \phi(S, I)\phi(I)\tau_2(G, I)$
- $\tau_4(G, J) = \sum_H \phi(H, G, J)$
- $\tau_5(J, L, S) = \sum_G \phi(L, G)\tau_3(G, S)\tau_4(G, J)$



Induced Graph

Theorem

Every maximal clique in the induced graph is a factor produced during variable elimination.



Width of Induced Graph

Definition

The width of induced graph is the one less than the number of nodes in the largest clique in the graph.

- Minimal induced width $= \min_{\alpha}(width(I_{\varphi,\alpha}))$ provides a lower bound of the performance of the variable elimination algorithm.

Variable Elimination Ordering

- Greedy search using a heuristic cost function. At each step, eliminate node with smallest cost. The possible cost functions:
  - min-neighbours – number of neighbours in current graph.
  - min-weight – weight or total number of values of the factor formed.
  - min-fill – number of new fill edges. – oftern used.
- Find low-width triangulation of original graph.
  - Triangulation means there are no loops for size > 3 that does not have a bridge.
  - If size > 3, add a fill edge.

Questions

1. Given a BN or MN, apply Variable Elimination algorithm. Show the intermediate steps.
2. Draw the induced graph of BN or MN and compute its width.
3. Find the best VE ordering.