Probabilistic Graphical Models

Lecture 6: Exact Inference: Variable Elimination

Matthew Walter

TTI-Chicago

April 23, 2020

Probabilistic Inference

- We have described two different ways to represent distributions over random variables, and to identify (conditional) independencies
  1. Bayesian networks
  2. Markov networks

- The next few lectures explore the following topics related to inference
  1. Conditional and MAP queries
  2. Computational cost of inference
  3. Exact inference methods
  4. Approximate inference methods

Probabilistic Inference

- For now, we will focus on conditional probability queries

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

(e.g., probability of having the flu given you have a runny nose)

Probabilistic Inference

- For now, we will focus on conditional probability queries

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

  (e.g., probability of having the flu given you have a runny nose)
- Let $\boldsymbol{W} = \mathcal{X} - \boldsymbol{Y} - \boldsymbol{E}$ be the random variables that are neither in the query nor evidence

Probabilistic Inference

- For now, we will focus on conditional probability queries

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

  (e.g., probability of having the flu given you have a runny nose)
- Let $\boldsymbol{W} = \mathcal{X} - \boldsymbol{Y} - \boldsymbol{E}$ be the random variables that are neither in the query nor evidence
- We can compute these joint distributions by marginalizing over $\boldsymbol{W}$

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

Probabilistic Inference

- For now, we will focus on conditional probability queries

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

(e.g., probability of having the flu given you have a runny nose)

- Let $\boldsymbol{W} = \mathcal{X} - \boldsymbol{Y} - \boldsymbol{E}$ be the random variables that are neither in the query nor evidence
- We can compute these joint distributions by marginalizing over $\boldsymbol{W}$

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

- Naive marginalization over unobserved variables requires an exponential number of computations

Probabilistic Inference

- For now, we will focus on conditional probability queries

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

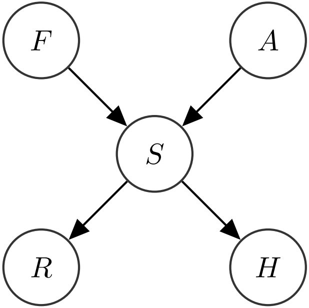
(e.g., probability of having the flu given you have a runny nose)

- Let $\boldsymbol{W} = \mathcal{X} - \boldsymbol{Y} - \boldsymbol{E}$ be the random variables that are neither in the query nor evidence
- We can compute these joint distributions by marginalizing over $\boldsymbol{W}$

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

- Naive marginalization over unobserved variables requires an exponential number of computations
- Are there techniques for doing inference more efficiently?

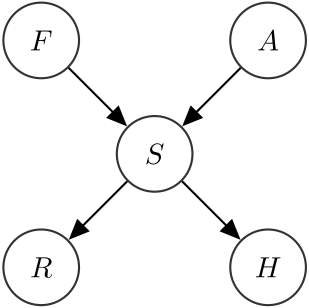
Example: Simple Medical Diagnosis



- Query: $P(F \mid R)$
- Requires accounting for all other variables A, S, and H

$$
\begin{aligned}
P(F \mid R) &= \frac{P(F, R)}{P(R)} \\
&= \frac{\sum_{A,S,H} P(F, A, S, R, H)}{\sum_{F,A,S,H} P(F, A, S, R, H)}
\end{aligned}
$$

Example: Simple Medical Diagnosis



- Query: $P(F \mid R)$
- Requires accounting for all other variables A, S, and H

$$
\begin{aligned}
P(F \mid R) &= \frac{P(F, R)}{P(R)} \\
&= \frac{\sum_{A,S,H} P(F, A, S, R, H)}{\sum_{F,A,S,H} P(F, A, S, R, H)}
\end{aligned}
$$

- Nominally, requires 2^4 summations (recall QMR-DT with $\sim 2^{4600}$ instantiations)

- A standard way of showing that a problem is is difficult to solve is to show that it is NP-hard
- 3-SAT considers the *satisfiability* of a logical formula over n Boolean variables $q_1, q_2, \ldots, q_n$ defined as a conjunction $\phi = C_1 \wedge \cdots \wedge C_m$, where each clause $C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$, where $l_{i,j} = q_k$ or $\neg q_k$

  e.g., is the following satisfiable?

$$\phi = (q_1 \vee \neg q_2 \vee q_3) \wedge (q_2 \vee q_3 \vee \neg q_4) \wedge (\neg q_1 \vee q_2 \vee \neg q_4) \wedge (q_1 \vee \neg q_3 \vee q_4)$$

Computational Complexity of Probabilistic Inference

- A standard way of showing that a problem is is difficult to solve is to show that it is NP-hard
- 3-SAT considers the *satisfiability* of a logical formula over n Boolean variables $q_1, q_2, \ldots, q_n$ defined as a conjunction $\phi = C_1 \wedge \cdots \wedge C_m$, where each clause $C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$, where $l_{i,j} = q_k$ or $\neg q_k$

  e.g., is the following satisfiable? Yes, e.g., $q_1 = q_2 = q_3 = q_4 = 1$

  $$\phi = (q_1 \vee \neg q_2 \vee q_3) \wedge (q_2 \vee q_3 \vee \neg q_4) \wedge (\neg q_1 \vee q_2 \vee \neg q_4) \wedge (q_1 \vee \neg q_3 \vee q_4)$$

Computational Complexity of Probabilistic Inference

- A standard way of showing that a problem is is difficult to solve is to show that it is NP-hard
- 3-SAT considers the *satisfiability* of a logical formula over n Boolean variables $q_1, q_2, \dots, q_n$ defined as a conjunction $\phi = C_1 \wedge \cdots \wedge C_m$, where each clause $C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$, where $l_{i,j} = q_k$ or $\neg q_k$

  e.g., is the following satisfiable? Yes, e.g., $q_1 = q_2 = q_3 = q_4 = 1$

  $$\phi = (q_1 \vee \neg q_2 \vee q_3) \wedge (q_2 \vee q_3 \vee \neg q_4) \wedge (\neg q_1 \vee q_2 \vee \neg q_4) \wedge (q_1 \vee \neg q_3 \vee q_4)$$

- 3-SAT is NP-complete (i.e., there exists a polynomial-time reduction for any NP problem (NP-hardness) and it is in NP)

Computational Complexity of Probabilistic Inference

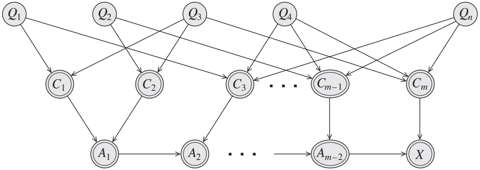
Definition (BN-Pr-DP Decision Problem)

Given a Bayesian network B over $\mathcal{X}$ and a variable $X \in \mathcal{X}$ and value $x \in \mathrm{Val}(X)$, decide whether $P(X = x) > 0$

- Formulate inference as the BN-Pr-DP decision problem
- **Theorem**: The BN-Pr-DP decision problem is NP-complete
- **Proof** (sketch):
  1. BN-Pr-DP is in NP
  2. Any NP problem can be reduced to BN-Pr-DP in polynomial-time

Computational Complexity of Probabilistic Inference

- Consider the reduction from 3-SAT (NP-complete) with formula ϕ
- Create a Bayesian network B_ϕ with variable X such that $P(X = x^1) > 0$ iff ϕ is satisfiable



Q_i : Boolean variable, $P(q_i) = 0.5$

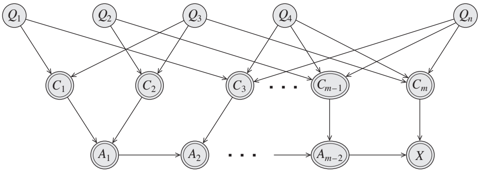
C_j : Boolean variable for each clause

$A_1 : C_1 \wedge C_2, \quad A_k = A_{k-1} \wedge C_{k+1}$

$X : A_{m-2} \wedge C_m$

Computational Complexity of Probabilistic Inference

- Consider the reduction from 3-SAT (NP-complete) with formula ϕ
- Create a Bayesian network B_ϕ with variable X such that $P(X = x^1) > 0$ iff ϕ is satisfiable



Q_i : Boolean variable, $P(q_i) = 0.5$

C_j : Boolean variable for each clause

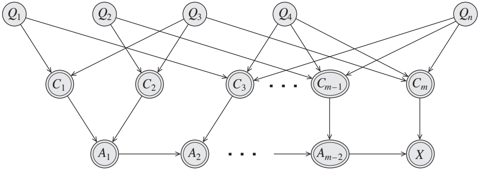
$A_1 : C_1 \wedge C_2, \quad A_k = A_{k-1} \wedge C_{k+1}$

$X : A_{m-2} \wedge C_m$

- $P(x^1)$ is the total number of satisfying assignments divided by 2^n (due to uniform prior)
- Checking if $P(x^1) > 0$ is sufficient to determine if ϕ is satisfiable

Computational Complexity of Probabilistic Inference

- Consider the reduction from 3-SAT (NP-complete) with formula ϕ
- Create a Bayesian network B_ϕ with variable X such that $P(X = x^1) > 0$ iff ϕ is satisfiable



Q_i : Boolean variable, $P(q_i) = 0.5$

C_j : Boolean variable for each clause

$A_1 : C_1 \wedge C_2, \quad A_k = A_{k-1} \wedge C_{k+1}$

$X : A_{m-2} \wedge C_m$

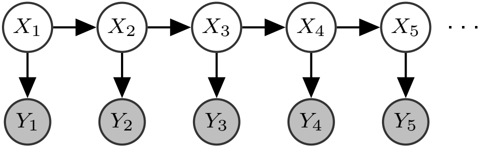
- $P(x^1)$ is the total number of satisfying assignments divided by 2^n (due to uniform prior)
- Checking if $P(x^1) > 0$ is sufficient to determine if ϕ is satisfiable
- Thus, BN-Pr-DP is NP-complete (since BN-Pr-DP is also in NP)

Probabilistic Inference in Practice

- NP-hardness simply means that difficult (i.e., exponential) inference problems **exist**
- Real-world inference problems are not necessarily as hard as these worst-case instances

Probabilistic Inference in Practice

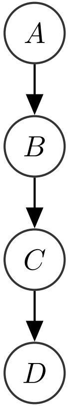
- NP-hardness simply means that difficult (i.e., exponential) inference problems **exist**
- Real-world inference problems are not necessarily as hard as these worst-case instances
- Some graphs are **easy** to do inference in



For example, inference in hidden Markov models (and other tree-structured graphs) can be performed in linear time

Variable Elimination: Markov Chain

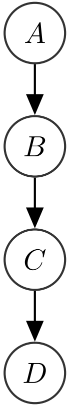
- Consider a simple Markov chain $A \rightarrow B \rightarrow C \rightarrow D$
- Let's calculate $P(B)$



Variable Elimination: Markov Chain

- Consider a simple Markov chain $A \to B \to C \to D$
- Let's calculate $P(B)$

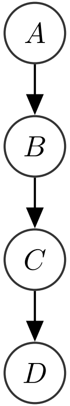
$$P(B) = \sum_{a \in \mathsf{Val}(A)} P(B \mid a) P(a)$$



- Consider a simple Markov chain $A \to B \to C \to D$
- Let's calculate $P(B)$

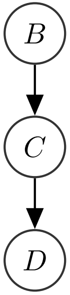
$$P(B) = \sum_{a \in \mathsf{Val}(A)} P(B \,|\, a) P(a)$$

- Note that C and D don't matter



Variable Elimination: Markov Chain

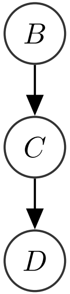
- Now, we have a Bayesian network for the marginal distribution $P(B, C, D)$



Variable Elimination: Markov Chain

- Now, we have a Bayesian network for the marginal distribution $P(B, C, D)$
- We can repeat the same process to calculate $P(C)$

$$P(C) = \sum_{b \in \mathsf{Val}(B)} P(C \mid b) P(b)$$

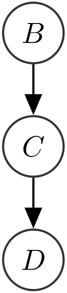


Variable Elimination: Markov Chain

- Now, we have a Bayesian network for the marginal distribution $P(B, C, D)$
- We can repeat the same process to calculate $P(C)$

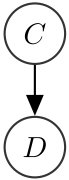
$$P(C) = \sum_{b \in \mathsf{Val}(B)} P(C \mid b) P(b)$$

- We can reuse our calculation of $P(B)$



Variable Elimination: Markov Chain

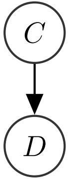
- Now, we have a Bayesian network for the marginal distribution $P(C, D)$
- By exploiting the structure of the Bayesian network, marginalizing out A and B happened in two easy steps



Variable Elimination: Markov Chain

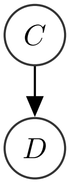
- Now, we have a Bayesian network for the marginal distribution $P(C, D)$
- By exploiting the structure of the Bayesian network, marginalizing out A and B happened in two easy steps
- We can repeat the same process to calculate $P(D)$

$$P(D) = \sum_{c \in \mathsf{Val}(C)} P(D \mid c) P(c)$$



- Now, we have a Bayesian network for the marginal distribution $P(C, D)$
- By exploiting the structure of the Bayesian network, marginalizing out A and B happened in two easy steps
- We can repeat the same process to calculate $P(D)$

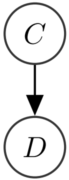
$$P(D) = \sum_{c \in \mathsf{Val}(C)} P(D \mid c) P(c)$$



- We can reuse our calculation of $P(C)$

Variable Elimination: Markov Chain

- Now, we have a Bayesian network for the marginal distribution $P(C, D)$
- By exploiting the structure of the Bayesian network, marginalizing out A and B happened in two easy steps
- We can repeat the same process to calculate $P(D)$



$$P(D) = \sum_{c \in \mathsf{Val}(C)} P(D \mid c) P(c)$$

- We can reuse our calculation of $P(C)$
- That was **variable elimination**
- Each marginalization involves k^2 multiplications and $k \times (k-1)$ additions, where $k = |\mathsf{Val}()|$

Variable Elimination

- Exploits the Bayesian network structure to avoid having to enumerate all assignments (via dynamic programming)
- Reuses computation from previous steps, avoiding the need to do work more than once by inverting order of computations
- Except for worst-case scenarios, avoids exponential blowup
- Running time will depend on the *graph structure*
- Exact algorithm for probabilistic inference in **any** graphical model

Variable Elimination: Basic Idea

- Let's again consider the Markov chain $A \rightarrow B \rightarrow C \rightarrow D$
- Suppose that we want to compute $P(D)$
- Per the chain rule and implied conditional independencies, the joint distribution factorizes as

$$P(A, B, C, D) = P(A)P(B \mid A)P(C \mid B)P(D \mid C)$$

- In order to compute $P(D)$, we nominally need to marginalize over A, B, and C:

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

Variable Elimination: Basic Idea

- If A, B, C, and D are binary, this becomes

$$
\begin{array}{lllll}
 & P(a^1) & P(b^1 \mid a^1) & P(c^1 \mid b^1) & P(d^1 \mid c^1) \\
+ & P(a^2) & P(b^1 \mid a^2) & P(c^1 \mid b^1) & P(d^1 \mid c^1) \\
+ & P(a^1) & P(b^2 \mid a^1) & P(c^1 \mid b^2) & P(d^1 \mid c^1) \\
+ & P(a^2) & P(b^2 \mid a^2) & P(c^1 \mid b^2) & P(d^1 \mid c^1) \\
+ & P(a^1) & P(b^1 \mid a^1) & P(c^2 \mid b^1) & P(d^1 \mid c^2) \\
+ & P(a^2) & P(b^1 \mid a^2) & P(c^2 \mid b^1) & P(d^1 \mid c^2) \\
+ & P(a^1) & P(b^2 \mid a^1) & P(c^2 \mid b^2) & P(d^1 \mid c^2) \\
+ & P(a^2) & P(b^2 \mid a^2) & P(c^2 \mid b^2) & P(d^1 \mid c^2)
\end{array}
$$

$$
\begin{array}{lllll}
 & P(a^1) & P(b^1 \mid a^1) & P(c^1 \mid b^1) & P(d^2 \mid c^1) \\
+ & P(a^2) & P(b^1 \mid a^2) & P(c^1 \mid b^1) & P(d^2 \mid c^1) \\
+ & P(a^1) & P(b^2 \mid a^1) & P(c^1 \mid b^2) & P(d^2 \mid c^1) \\
+ & P(a^2) & P(b^2 \mid a^2) & P(c^1 \mid b^2) & P(d^2 \mid c^1) \\
+ & P(a^1) & P(b^1 \mid a^1) & P(c^2 \mid b^1) & P(d^2 \mid c^2) \\
+ & P(a^2) & P(b^1 \mid a^2) & P(c^2 \mid b^1) & P(d^2 \mid c^2) \\
+ & P(a^1) & P(b^2 \mid a^1) & P(c^2 \mid b^2) & P(d^2 \mid c^2) \\
+ & P(a^2) & P(b^2 \mid a^2) & P(c^2 \mid b^2) & P(d^2 \mid c^2)
\end{array}
$$

Variable Elimination: Basic Idea

- If A, B, C, and D are binary, this becomes

$$\begin{array}{rllll}
 & P(a^1) & P(b^1 \mid a^1) & P(c^1 \mid b^1) & P(d^1 \mid c^1) \\
+ & P(a^2) & P(b^1 \mid a^2) & P(c^1 \mid b^1) & P(d^1 \mid c^1) \\
+ & P(a^1) & P(b^2 \mid a^1) & P(c^1 \mid b^2) & P(d^1 \mid c^1) \\
+ & P(a^2) & P(b^2 \mid a^2) & P(c^1 \mid b^2) & P(d^1 \mid c^1) \\
+ & P(a^1) & P(b^1 \mid a^1) & P(c^2 \mid b^1) & P(d^1 \mid c^2) \\
+ & P(a^2) & P(b^1 \mid a^2) & P(c^2 \mid b^1) & P(d^1 \mid c^2) \\
+ & P(a^1) & P(b^2 \mid a^1) & P(c^2 \mid b^2) & P(d^1 \mid c^2) \\
+ & P(a^2) & P(b^2 \mid a^2) & P(c^2 \mid b^2) & P(d^1 \mid c^2)
\end{array}$$

$$\begin{array}{rllll}
 & P(a^1) & P(b^1 \mid a^1) & P(c^1 \mid b^1) & P(d^2 \mid c^1) \\
+ & P(a^2) & P(b^1 \mid a^2) & P(c^1 \mid b^1) & P(d^2 \mid c^1) \\
+ & P(a^1) & P(b^2 \mid a^1) & P(c^1 \mid b^2) & P(d^2 \mid c^1) \\
+ & P(a^2) & P(b^2 \mid a^2) & P(c^1 \mid b^2) & P(d^2 \mid c^1) \\
+ & P(a^1) & P(b^1 \mid a^1) & P(c^2 \mid b^1) & P(d^2 \mid c^2) \\
+ & P(a^2) & P(b^1 \mid a^2) & P(c^2 \mid b^1) & P(d^2 \mid c^2) \\
+ & P(a^1) & P(b^2 \mid a^1) & P(c^2 \mid b^2) & P(d^2 \mid c^2) \\
+ & P(a^2) & P(b^2 \mid a^2) & P(c^2 \mid b^2) & P(d^2 \mid c^2)
\end{array}$$

- Requires $3 \times 16 = 48$ multiplications and $2 \times 7 = 14$ additions

Variable Elimination: Basic Idea

- If A, B, C, and D are binary, this becomes

$$
\begin{array}{rllll}
 & P(a^1) & P(b^1 \mid a^1) & P(c^1 \mid b^1) & P(d^1 \mid c^1) \\
+ & P(a^2) & P(b^1 \mid a^2) & P(c^1 \mid b^1) & P(d^1 \mid c^1) \\
+ & P(a^1) & P(b^2 \mid a^1) & P(c^1 \mid b^2) & P(d^1 \mid c^1) \\
+ & P(a^2) & P(b^2 \mid a^2) & P(c^1 \mid b^2) & P(d^1 \mid c^1) \\
+ & P(a^1) & P(b^1 \mid a^1) & P(c^2 \mid b^1) & P(d^1 \mid c^2) \\
+ & P(a^2) & P(b^1 \mid a^2) & P(c^2 \mid b^1) & P(d^1 \mid c^2) \\
+ & P(a^1) & P(b^2 \mid a^1) & P(c^2 \mid b^2) & P(d^1 \mid c^2) \\
+ & P(a^2) & P(b^2 \mid a^2) & P(c^2 \mid b^2) & P(d^1 \mid c^2)
\end{array}
$$

$$
\begin{array}{rllll}
 & P(a^1) & P(b^1 \mid a^1) & P(c^1 \mid b^1) & P(d^2 \mid c^1) \\
+ & P(a^2) & P(b^1 \mid a^2) & P(c^1 \mid b^1) & P(d^2 \mid c^1) \\
+ & P(a^1) & P(b^2 \mid a^1) & P(c^1 \mid b^2) & P(d^2 \mid c^1) \\
+ & P(a^2) & P(b^2 \mid a^2) & P(c^1 \mid b^2) & P(d^2 \mid c^1) \\
+ & P(a^1) & P(b^1 \mid a^1) & P(c^2 \mid b^1) & P(d^2 \mid c^2) \\
+ & P(a^2) & P(b^1 \mid a^2) & P(c^2 \mid b^1) & P(d^2 \mid c^2) \\
+ & P(a^1) & P(b^2 \mid a^1) & P(c^2 \mid b^2) & P(d^2 \mid c^2) \\
+ & P(a^2) & P(b^2 \mid a^2) & P(c^2 \mid b^2) & P(d^2 \mid c^2)
\end{array}
$$

- Requires $3 \times 16 = 48$ multiplications and $2 \times 7 = 14$ additions
- But, certain terms are repeated several times, e.g.,

$$P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)$$

Variable Elimination: Basic Idea

- We can modify the computation to first compute

$$P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)$$

and

$$P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)$$

Variable Elimination: Basic Idea

- We can modify the computation to first compute

$$P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)$$

and

$$P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)$$

- This yields the modified form

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

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

Variable Elimination: Basic Idea

- We can modify the computation to first compute

$$P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)$$

and

$$P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)$$

- This yields the modified form

$$\begin{array}{lccc}
 & (P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)) & P(c^1 \mid b^1) & P(d^1 \mid c^1) \\
+ & (P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)) & P(c^1 \mid b^2) & P(d^1 \mid c^1) \\
+ & (P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)) & P(c^2 \mid b^1) & P(d^1 \mid c^2) \\
+ & (P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)) & P(c^2 \mid b^2) & P(d^1 \mid c^2)
\end{array}$$

$$\begin{array}{lccc}
 & (P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)) & P(c^1 \mid b^1) & P(d^2 \mid c^1) \\
+ & (P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)) & P(c^1 \mid b^2) & P(d^2 \mid c^1) \\
+ & (P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)) & P(c^2 \mid b^1) & P(d^2 \mid c^2) \\
+ & (P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)) & P(c^2 \mid b^2) & P(d^2 \mid c^2)
\end{array}$$

- Define $\tau_1 : \mathrm{Val}(B) \rightarrow \mathbb{R}$, $\tau_1(b^i) = P(a^1)P(b^i \mid a^1) + P(a^2)P(b^i \mid a^2)$ (These *messages* are also denoted by m)

Variable Elimination: Basic Idea

- We can modify the computation to first compute

$$P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)$$

and

$$P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)$$

Variable Elimination: Basic Idea

- We can modify the computation to first compute

$$P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)$$

and

$$P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)$$

- This yields the modified form

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

Variable Elimination: Basic Idea

- We can modify the computation to first compute

$$P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)$$

and

$$P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)$$

- This yields the modified form

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

- Define $\tau_1 : \mathrm{Val}(B) \rightarrow \mathbb{R}$, $\tau_1(b^i) = P(a^1)P(b^i \mid a^1) + P(a^2)P(b^i \mid a^2)$ (These *messages* are also denoted by m)

Variable Elimination: Basic Idea

- We now have

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

Variable Elimination: Basic Idea

- We now have

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

- We can reshuffle terms

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

Variable Elimination: Basic Idea

- We now have

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

- We can reshuffle terms

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

- There are more repeated computations

Variable Elimination: Basic Idea

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

- Define $\tau_2 : \text{Val}(C) \to \mathbb{R}$ as

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

Variable Elimination: Basic Idea

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

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

- Define $\tau_2 : \text{Val}(C) \rightarrow \mathbb{R}$ as

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

- The marginal $P(D)$ then reduces to

$$
\begin{aligned}
& \tau_2(c^1) \quad P(d^1 \mid c^1) \\
+ \; & \tau_2(c^2) \quad P(d^1 \mid c^2)
\end{aligned}
$$

$$
\begin{aligned}
& \tau_2(c^1) \quad P(d^2 \mid c^1) \\
+ \; & \tau_2(c^2) \quad P(d^2 \mid c^2)
\end{aligned}
$$

Variable Elimination: Basic Idea

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

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

- Define $\tau_2 : \text{Val}(C) \rightarrow \mathbb{R}$ as

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

- The marginal $P(D)$ then reduces to

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

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

- Requires 4 multiplications and 2 additions to compute each of the 3 messages, resulting in 18 computations (vs. $48 + 14 = 62$ originally)

Variable Elimination: Basic Idea

- Our goal was to compute

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

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

Variable Elimination: Basic Idea

- Our goal was to compute

$$P(D) = \sum_{a,b,c} P(a,b,c,D) = \sum_{a,b,c} P(a)P(b \mid a)P(c \mid b)P(D \mid c)$$

$$= \sum_{c} \sum_{b} \sum_{a} P(D \mid c)P(c \mid b)P(b \mid a)P(a)$$

- We can push summations inside to obtain

$$P(D) = \sum_{c} P(D \mid c) \sum_{b} P(c \mid b) \underbrace{\sum_{a} \underbrace{P(b \mid a)P(a)}_{\psi_1(a,b)}}_{\tau_1(b)}$$

Variable Elimination: Basic Idea

- Our goal was to compute

$$P(D) = \sum_{a,b,c} P(a,b,c,D) = \sum_{a,b,c} P(a)P(b \mid a)P(c \mid b)P(D \mid c)$$
$$= \sum_{c} \sum_{b} \sum_{a} P(D \mid c)P(c \mid b)P(b \mid a)P(a)$$

- We can push summations inside to obtain

$$P(D) = \sum_{c} P(D \mid c) \sum_{b} P(c \mid b) \underbrace{\sum_{a} \underbrace{P(b \mid a)P(a)}_{\psi_1(a,b)}}_{\tau_1(b)}$$

- Denote $\psi_1(A,B) = P(A)P(B \mid A)$. Then $\tau_1(B) = \sum_a \psi_1(a,B)$
- Similarly, let $\psi_2(B,C) = \tau_1(B)P(C \mid B)$. Then $\tau_2(C) = \sum_b \psi_2(b,C)$
- This procedure is dynamic programming (flip the computation order)

Inference in a Chain

- Generalizing previous example, suppose that we have a chain $X_1 \rightarrow X_2 \rightarrow \cdots \rightarrow X_n$ where $k = |\mathsf{Val}(X_i)|$
- For $i = 1$ to $i = n - 1$ compute (and cache)

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

Inference in a Chain

- Generalizing previous example, suppose that we have a chain $X_1 \to X_2 \to \cdots \to X_n$ where $k = |\mathsf{Val}(X_i)|$
- For $i = 1$ to $i = n - 1$ compute (and cache)

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

- Each update requires k^2 multiplications and $k \times (k - 1)$ additions

Inference in a Chain

- Generalizing previous example, suppose that we have a chain $X_1 \to X_2 \to \cdots \to X_n$ where $k = |\mathsf{Val}(X_i)|$
- For $i = 1$ to $i = n - 1$ compute (and cache)

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

- Each update requires k^2 multiplications and $k \times (k-1)$ additions
- Total running time is $\mathcal{O}(nk^2)$

Inference in a Chain

- Generalizing previous example, suppose that we have a chain $X_1 \to X_2 \to \cdots \to X_n$ where $k = |\text{Val}(X_i)|$
- For $i = 1$ to $i = n - 1$ compute (and cache)

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

- Each update requires k^2 multiplications and $k \times (k-1)$ additions
- Total running time is $\mathcal{O}(nk^2)$
- By comparison, naive marginalization is $\mathcal{O}(k^n)$ (i.e., exponential)
- We performed inference over the joint without ever explicitly constructing it

Summary so Far

- Worst-case analysis shows that marginal inference is NP-hard!
- Even approximate inference is NP-hard!
- In practice, we can perform inference in a tractable manner by
  1. Exploiting the structure of the Bayesian network to identify subexpressions that depend only on a small number of variables
  2. Cache computations that are otherwise computed exponentially many times
- Efficient reduction in computations depends on having a good **variable elimination ordering**

Factor Marginalization

- Let $\phi(\boldsymbol{X}, Y)$ be a factor where $\boldsymbol{X}$ is a set of random variables and $Y \notin \boldsymbol{X}$ is a distinct random variable
- **Factor marginalization** of Y in $\phi(\boldsymbol{X}, Y)$ ("summing out Y in ϕ") results in a new factor over $\boldsymbol{X}$

$$\tau(\boldsymbol{X}) = \sum_{y \in \mathsf{Val}(Y)} \phi(\boldsymbol{X}, y)$$

- For example, $\tau(A, C) = \sum_B \phi(A, B, C)$

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

Factor Marginalization

- We want an algorithm that computes $P(X)$ for BNs and MRFs
- This can be reduced to a **sum-product** inference task

$$\tau(\boldsymbol{X}) = \sum_{\boldsymbol{Z}} \prod_{\phi \in \Phi} \phi(\boldsymbol{Z}_{\mathsf{Scope}[\phi] \cap \boldsymbol{Z}}, \boldsymbol{X}_{\mathsf{Scope}[\phi] \cap \boldsymbol{X}})$$

  where Φ is a set of factors (CPDs for BNs and potentials for MRFs)
- Factor products and summations are commutative, products are associative
- Thus, if $X \notin$ Scope(ϕ_1), then $\sum_X (\phi_1 \cdot \phi_2) = \phi_1 \cdot \sum_X \phi_2$ ("push in" summations)

Sum-Product Variable Elimination

Definition (Elimination Ordering)

The **elimination ordering** $\prec$ is the order in which the variables $\boldsymbol{Z}$ will be marginalized (i.e., "eliminated")

- Given an elimination ordering $\prec$) of $\boldsymbol{Z}$
- Iteratively marginalize out each variable $Z_i \in \boldsymbol{Z}$
- For each $Z_i \in \boldsymbol{Z}$ according to $\prec$:
  1. Multiply all factors with Z_i in their scope, creating a new product factor
  2. Marginalize this product factor over Z_i, generating a smaller factor
  3. Remove the old factors from the set of all factors, add the new one

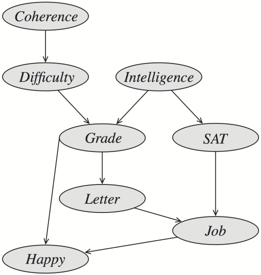
Sum-Product Variable Elimination

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

Example: Elimination Ordering



- What is $P(\text{Job})$?

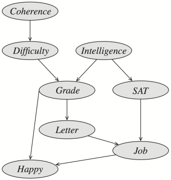
$$P(C, D, I, G, S, L, H, J) = P(C)P(D \mid C)P(I)P(G \mid D, I)P(L \mid G) \\ P(S \mid I)P(J \mid S, L)P(H \mid J, G)$$

- The corresponding factors are

$$\Phi = \{\phi_C(C), \phi_D(C, D), \phi_I(I), \phi_G(G, D, I), \phi_L(L, G), \\ \phi_S(S, I), \phi_J(J, S, L), \phi_H(H, J, G)\}$$

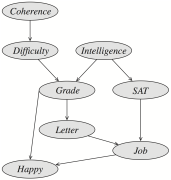
Example: Elimination Ordering

- We are free to choose any ordering, but some orderings introduce factors with larger scope. Consider $\prec = C, D, I, H, G, S, L$



Example: Elimination Ordering

- We are free to choose any ordering, but some orderings introduce factors with larger scope. Consider $\prec = C, D, I, H, G, S, L$

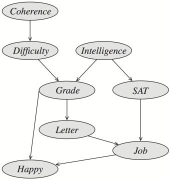


Step	Variable eliminated	Factors used	Variables involved	New factor
1	C	$\phi_C(C)$, $\phi_D(D,C)$	C, D	$\tau_1(D)$
2	D	$\phi_G(G,I,D)$, $\tau_1(D)$	G, I, D	$\tau_2(G,I)$
3	I	$\phi_I(I)$, $\phi_S(S,I)$, $\tau_2(G,I)$	G, S, I	$\tau_3(G,S)$
4	H	$\phi_H(H,G,J)$	H, G, J	$\tau_4(G,J)$
5	G	$\tau_4(G,J)$, $\tau_3(G,S)$,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

The largest induced scope is 3

Example: Elimination Ordering

- We are free to choose any ordering, but some orderings introduce factors with larger scope. Consider $\prec = C, D, I, H, G, S, L$



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	$\tau_7(J)$

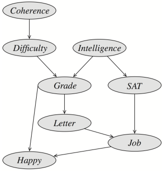
The largest induced scope is 3

- Alternatively, $\prec = G, I, S, L, H, C, D \ \ldots$

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	$\phi_C(C)$, $\phi_D(D,C)$	D, J, C	$\tau_6(D)$
7	D	$\tau_5(D,J)$, $\tau_6(D)$	D, J	$\tau_7(J)$

Induces factors with scope as large as 5

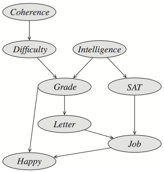
Complexity of Variable Elimination



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	$\tau_7(J)$

- Let n be the number of variables and m the number of initial factors
- At each step, we pick a variable X_i, multiply all factors involving X_i, resulting in a single factor ψ_i, and then sum out X_i to get a new factor τ_i, for a total of n *new* factors and $m + n$ total factors Φ
- Let N_i be the number of *entries* in factor $\phi_i \in \Phi$ and $N_{\text{max}} = \max_i N_i$

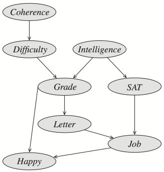
Complexity of Variable Elimination



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	$\tau_7(J)$

- Let n be the number of variables and m the number of initial factors
- At each step, we pick a variable X_i, multiply all factors involving X_i, resulting in a single factor ψ_i, and then sum out X_i to get a new factor τ_i, for a total of n *new* factors and $m + n$ total factors Φ
- Let N_i be the number of *entries* in factor $\phi_i \in \Phi$ and $N_{\text{max}} = \max_i N_i$
- Each factor is multiplied *once* to compute ψ_i at a cost of N_i
- Results in $(n + m)N_i \leq (n + m)N_{\text{max}} = \mathcal{O}(mN_{\text{max}})$ multiplications

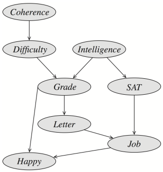
Complexity of Variable Elimination



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	$\tau_7(J)$

- Let n be the number of variables and m the number of initial factors
- At each step, we pick a variable X_i, multiply all factors involving X_i, resulting in a single factor ψ_i, and then sum out X_i to get a new factor τ_i, for a total of n *new* factors and $m + n$ total factors Φ
- Let N_i be the number of *entries* in factor $\phi_i \in \Phi$ and $N_{\text{max}} = \max_i N_i$
- Each factor is multiplied *once* to compute ψ_i at a cost of N_i
- Results in $(n+m)N_i \leq (n+m)N_{\text{max}} = \mathcal{O}(mN_{\text{max}})$ multiplications
- Marginalization involves N_i additions per factor (each entry in ψ_i is added once), for a total of no more than nN_{max}

Complexity of Variable Elimination

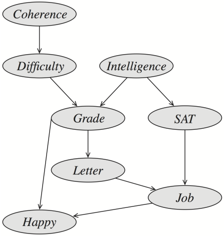


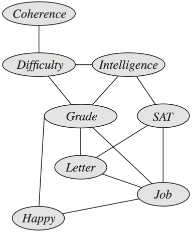
Step	Variable eliminated	Factors used	Variables involved	New factor
1	C	$\phi_C(C)$, $\phi_D(D,C)$	C, D	$\tau_1(D)$
2	D	$\phi_G(G,I,D)$, $\tau_1(D)$	G, I, D	$\tau_2(G,I)$
3	I	$\phi_I(I)$, $\phi_S(S,I)$, $\tau_2(G,I)$	G, S, I	$\tau_3(G,S)$
4	H	$\phi_H(H,G,J)$	H, G, J	$\tau_4(G,J)$
5	G	$\tau_4(G,J)$, $\tau_3(G,S)$,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

- Let n be the number of variables and m the number of initial factors
- At each step, we pick a variable X_i, multiply all factors involving X_i, resulting in a single factor ψ_i, and then sum out X_i to get a new factor τ_i, for a total of n *new* factors and $m + n$ total factors Φ
- Let N_i be the number of *entries* in factor $\phi_i \in \Phi$ and $N_{\text{max}} = \max_i N_i$
- Each factor is multiplied *once* to compute ψ_i at a cost of N_i
- Results in $(n+m)N_i \leq (n+m)N_{\text{max}} = \mathcal{O}(mN_{\text{max}})$ multiplications
- Marginalization involves N_i additions per factor (each entry in ψ_i is added once), for a total of no more than nN_{max}
- Running time of VE is $\mathcal{O}((m+n)N_{\text{max}})$
- Computational cost dominated by the size of intermediate factors

Complexity of Variable Elimination: Graph-Theoretic

- We can also analyze the complexity in terms of graph structure
- Let H_Φ be an undirected graph with one node per variable and an edge (X_i, X_j) for all X_i and X_j in the scope of a factor ϕ
- H_Φ corresponds to either a Markov random field or a moralized Bayesian network





Complexity of Variable Elimination: Graph-Theoretic

When a variable X_i is eliminated, we

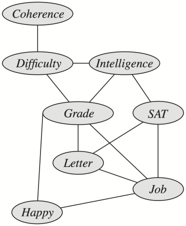
1. Create a single factor ψ that contains X_i and all of the variables $\boldsymbol{Y}$ with which X_i appears in factors
2. Eliminate X_i from ψ, replacing ψ with a new factor τ that contains all of the variables $\boldsymbol{Y}$, but not X_i. Denote the new set of factors as Φ_{X_i}

How does this modify the graph going from H_Φ to $H_{\Phi_{X_i}}$?

- Constructing ψ generates edges between all of the variables in $Y \in \boldsymbol{Y}$
- Some of these edges were already in H_Φ, some are new
- The new edges are called **fill edges**
- The step of removing X_i from Φ to construct Φ_{X_i} removes X_i and all of its incident edges from the graph

Example: Graph-Theoretic Complexity Analysis

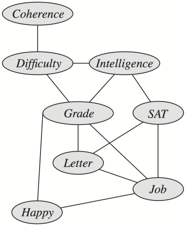
$$\prec = C, D, I$$



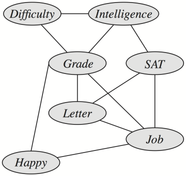
Original

Example: Graph-Theoretic Complexity Analysis

$$\prec = C, D, I$$



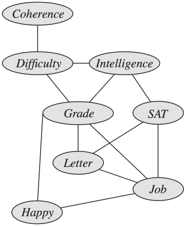
Original



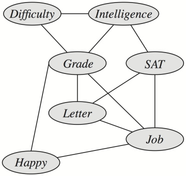
Eliminate C

Example: Graph-Theoretic Complexity Analysis

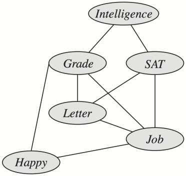
$$\prec = C, D, I$$



Original



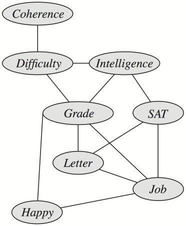
Eliminate C



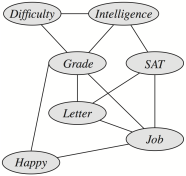
Eliminate D

Example: Graph-Theoretic Complexity Analysis

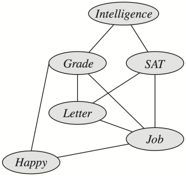
$$\prec = C, D, I$$



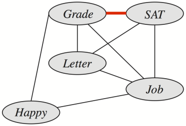
Original



Eliminate C



Eliminate D



Eliminate I

Induced Graph

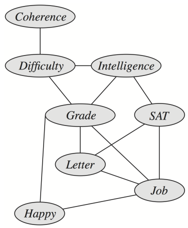
Definition (Induced Graph)

Let Φ be a set of factors over $X_1, \ldots, X_n$ and $\prec$ be an elimination ordering

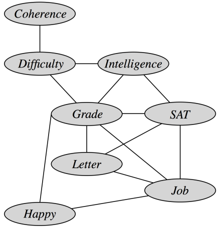
The **induced graph** $\mathcal{I}_{\phi,\prec}$ is an undirected graph where there is an edge (X_i, X_j) for all X_i and X_j that appear in some intermediate factor ψ generated by the VE algorithm

We can use this graph to evaluate the computational cost of variable elimination

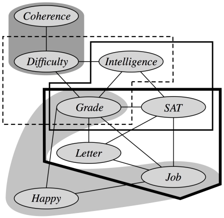
Example



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	$\tau_7(J)$



Induced Graph $(G-S)$



Maximal Cliques

Induced Graph

Theorem

Let $\mathcal{I}_{\phi,\prec}$ be the induced graph for a set of factors Φ and ordering $\prec$, then

1. *The scope for every generated factor ψ is a clique in $\mathcal{I}_{\phi,\prec}$*
2. *Every maximal clique in $\mathcal{I}_{\phi,\prec}$ is the scope of some intermediate factor*

Induced Graph

Theorem

Let $\mathcal{I}_{\phi,\prec}$ be the induced graph for a set of factors Φ and ordering $\prec$, then

1. *The scope for every generated factor ψ is a clique in $\mathcal{I}_{\phi,\prec}$*
2. *Every maximal clique in $\mathcal{I}_{\phi,\prec}$ is the scope of some intermediate factor*

Proof.

1. This follows directly from the definition of an induced graph
2. Let $\boldsymbol{Y} = \{Y_1, Y_2, \ldots, Y_m\}$ be a maximal clique
   - Let Y_1 be the first variable from $\boldsymbol{Y}$ in $\prec$
   - Every edge between Y_1 and $Y_i \in \boldsymbol{Y}$ existed before eliminating Y_1
   - There are factors involving Y_1 and each Y_i
   - Eliminating Y_1 involves multiplying these factors, creating a factor ψ over $Y_1, Y_2, \ldots, Y_m$ that involves no other variables

$\square$

Induced Graph

Definition (Induced Width)

The **induced width** $w_{K,\prec}$ for a graph K with elimination ordering $\prec$ is the number of nodes in the *largest* clique in the induced graph minus one

Induced Graph

Definition (Induced Width)

The **induced width** $w_{K,\prec}$ for a graph K with elimination ordering $\prec$ is the number of nodes in the *largest* clique in the induced graph minus one

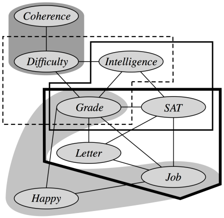
- Running time $\mathcal{O}(mk^{w_{K,\prec}})$ is exponential in the size of the largest clique of the induced graph, where $k = |\mathsf{Val}(X_i)|$
- Referring to the previous analysis, $N_{\mathsf{max}} = k^{w_{K,\prec}}$

Definition (Tree-Width)

The **tree-width** ("minimal induced width") of a graph K is the minimal induced width

$$w^*_{K,\prec} = \min_{\prec} w_{K,\prec}$$

Induced Graph: Example



Maximal Cliques

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	$\tau_7(J)$

The maximal cliques in $\mathcal{I}_{\phi,\prec}$ are

$$C_1 = \{C, D\}$$
$$C_2 = \{D, I, G\}$$
$$C_3 = \{G, L, S, J\}$$
$$C_4 = \{G, J, H\}$$

Induced Width

- The tree-width of a graph K is the minimal induced width

$$w^*_{K,\prec} = \min_{\prec} w_{K,\prec}$$

- Tree-width provides a bound on the best running time achievable with VE on a distribution that factorizes over K: $\mathcal{O}(mk^{w^*_{K,\prec}})$
- Unfortunately, finding the tree width (and, equivalently, the best elimination ordering) for a graph is NP-hard
- In practice, heuristics provide a good elimination ordering

Chordal Graphs

Definition (Chordal Graph)

A graph is **chordal** ("triangulated") if every cycle of length ≥ 3 has a shortcut (a "chord")

Theorem

Every induced graph is chordal

Proof.

- Assume chordless cycle $X_1 - X_2 - X_3 - X_4 - X_1$ in the induced graph
- Suppose X_1 was the first variable that we eliminated of the 4
- After a node is eliminated, no fill edges can be added. Thus $X_1 - X_2$ and $X_1 - X_4$ must have already existed
- Eliminating X_1 induces edge $X_2 - X_4$, contradicting our assumption

□

Chordal Graphs

Theorem

Every induced graph is chordal

Theorem

Any chordal graph has an elimination ordering that does not induce any fill edges

- Proof relies on concepts that we will see in next lecture
- Thus, finding a good elimination ordering is equivalent to making a graph chordal with minimal width

Choosing an Elimination Ordering

- Unfortunately, finding the optimal elimination ordering is NP-hard
- Several heuristics exist for finding *good* elimination orderings via greedy cost minimization:
  - **Min-neighbors**: Cost of a vertex is the number of its neighbors in the current graph
  - **Min-weight**: Cost of a vertex is the product of weights of its neighbors
  - **Min-fill**: Cost of a vertex is the number of edges that need to be added to the graph due to elimination
  - **Weighed-Min-Fill**: Cost of a vertex is the sum of weights of the edges that need to be added to the graph due to its elimination. The weight of an edge is the product of weights of its constituent vertices
- None is particularly better than any others
- Can be used deterministically or stochastically (e.g., sampling according to cost)