PROBABILISTIC MODELS – PART 4a: EXACT INFERENCE IN BAYESIAN NETWORKS

Gerhard Widmer

Institute of Computational Perception
Johannes Kepler University
Linz, Austria

gerhard.widmer@jku.at
www.cp.jku.at/people/widmer

October 13, 2025

Presentation partly based on and inspired by [Koller & Friedman, 2009] and [Russell & Norvig, 2021], including the use of some figures from their books and/or lecture slides.

Many thanks to Daphne Koller, Nir Friedman, Stuart Russell, and Peter Norvig for making these available (`pgm.stanford.edu`; `aima.cs.berkeley.edu`).

Goals of this Lecture

- Review the probabilistic query
- Describe the general Inference-by-Enumeration Algorithm for BNs
- Analyse the computational complexity of exact inference
- Optimisation: Introduce the basic idea of Variable Elimination
- The Variable Elimination Algorithm for Bayesian Networks
- Revisit the computational complexity of exact inference

Outline

1. Probabilistic Queries
2. Inference by Enumeration in BNs
   - The General Algorithm
   - Computational Complexity
3. Variable Elimination
   - A Simple Example
   - Deriving Insight from the Simple Example
   - The General Algorithm
   - Complexity

The Probabilistic Query

Remember:
We will be interested in answering questions of the following form

Definition

A **Probabilistic Query** involves computing the **conditional probability distribution**

$$P(\boldsymbol{X} \mid \boldsymbol{E} = \boldsymbol{e})$$

or $P(\boldsymbol{X} \mid \boldsymbol{e})$ for short

for some sets of variables $\boldsymbol{X}, \boldsymbol{E} \subseteq \mathcal{X}$ and a specific value assignment $\boldsymbol{E} = \boldsymbol{e}$.

IN WORDS:

Given that we know or have observed the specific values $\boldsymbol{e}$ of the variables $\boldsymbol{E}$ (the '**evidence variables**'), what are the probabilities for the different value combinations of the $\boldsymbol{X}$ (the '**query variables**')?

The Probabilistic Query: Special Cases

We might also be interested in computing

- the **probability of a specific value combination** $\boldsymbol{x}$ of the $\boldsymbol{X}$:

$$P(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{e})$$

- or the **most likely value combination** of the $\boldsymbol{X}$, given the evidence $\boldsymbol{e}$:[a]

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}} P(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{e})$$

[a] This is called a **MAP** (**M**aximal **A** **P**osteriori) Query.

Of course, these follow naturally once we have computed the full conditional distribution $P(\boldsymbol{X} \mid \boldsymbol{e})$.

Computing a Conditional Distribution: Inference by Enumeration

Remember from last section:

By the definition of conditional probability, we have that

$$P(\boldsymbol{X} \mid \boldsymbol{e}) = \frac{P(\boldsymbol{X}, \boldsymbol{e})}{P(\boldsymbol{e})} = \frac{1}{Z} P(\boldsymbol{X}, \boldsymbol{e})$$

IN WORDS:

The conditional distribution $P(\boldsymbol{X} \mid \boldsymbol{e})$ is equal to the *joint probabilities* $P(\boldsymbol{X}, \boldsymbol{e})$ (the probabilities of conjunctions $P(\boldsymbol{x}, \boldsymbol{e})$), normalised by the *probability of the evidence* $P(\boldsymbol{e})$.

- ☞ Compute this for each possible assignment of values $\boldsymbol{x}$ of the query variables $\boldsymbol{X}$, and you have the full conditional distribution $P(\boldsymbol{X} \mid \boldsymbol{E} = \boldsymbol{e})$.
- ☞ $P(\boldsymbol{e})$ simply acts as a normalisation constant Z – compute only once.

Inference by Enumeration for Probabilistic Queries

Goal: Compute $P(\boldsymbol{X} \mid \boldsymbol{e})$ for some sets of variables $\boldsymbol{X}, \boldsymbol{E} \subseteq \mathcal{X}$ and some specific value assignment $\boldsymbol{E} = \boldsymbol{e}$.

Algorithm

1. Compute $P(\boldsymbol{X}, \boldsymbol{e})$ via *Inference by Enumeration*, $\boldsymbol{Z} = \mathcal{X} - \boldsymbol{X} - \boldsymbol{E}$

$$P(\boldsymbol{X}, \boldsymbol{e}) = \sum_{\boldsymbol{z} \in Val(\boldsymbol{Z})} P(\boldsymbol{X}, \boldsymbol{e}, \boldsymbol{z})$$

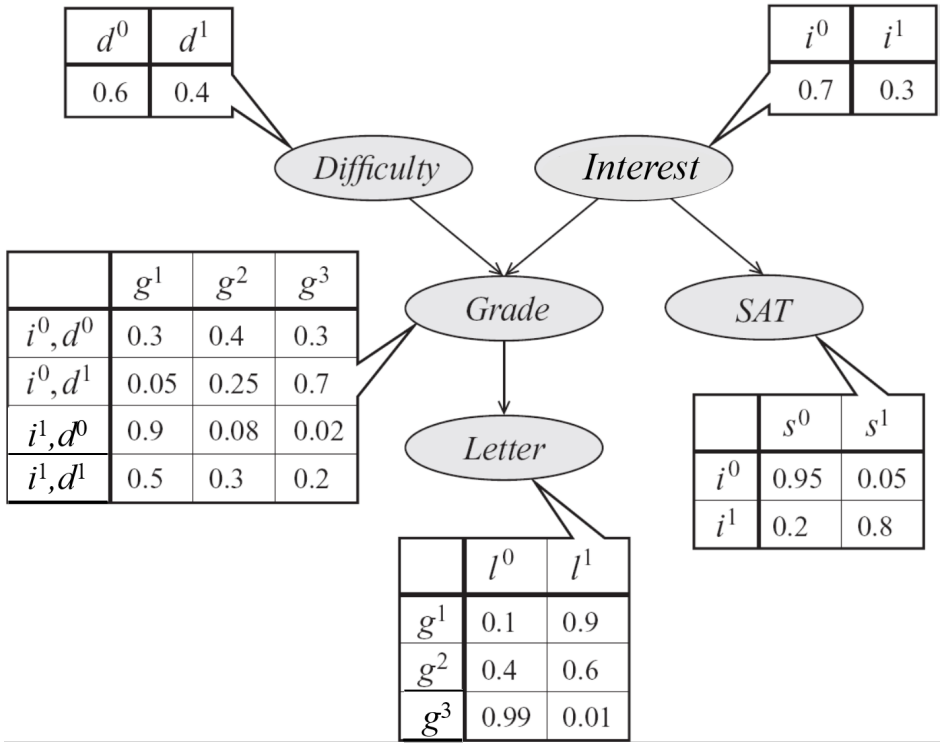
2. Compute normalisation constant Z from the above as

$$Z = P(\boldsymbol{e}) = \sum_{\boldsymbol{x}} P(\boldsymbol{x}, \boldsymbol{e})$$

3. Condition via renormalisation:

$$P(\boldsymbol{X} \mid \boldsymbol{e}) = \frac{1}{Z} P(\boldsymbol{X}, \boldsymbol{e})$$

Example: The Student Network



d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

	g^1	g^2	g^3
i^0, d^0	0.3	0.4	0.3
i^0, d^1	0.05	0.25	0.7
i^1, d^0	0.9	0.08	0.02
i^1, d^1	0.5	0.3	0.2

	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

	l^0	l^1
g^1	0.1	0.9
g^2	0.4	0.6
g^3	0.99	0.01

Example query: $P(D \mid g^3, s^1) = ?$

Solving the Example Query (1)

$$
\begin{aligned}
\underline{P(d^0, g^3, s^1)} &= \sum_{i=\{i^0,i^1\}, l=\{l^0,l^1\}} P(i, d^0, g^3, l, s^1) \\
&= \sum_{i=\{i^0,i^1\}, l=\{l^0,l^1\}} P(i)P(d^0)P(g^3|d^0, i)P(l|g^3)P(s^1|i) \\
&= P(i^0)P(d^0)P(g^3|d^0, i^0)P(l^0|g^3)P(s^1|i^0) + \\
&\quad P(i^0)P(d^0)P(g^3|d^0, i^0)P(l^1|g^3)P(s^1|i^0) + \\
&\quad P(i^1)P(d^0)P(g^3|d^0, i^1)P(l^0|g^3)P(s^1|i^1) + \\
&\quad P(i^1)P(d^0)P(g^3|d^0, i^1)P(l^1|g^3)P(s^1|i^1) \\
&= 0.7 \cdot 0.6 \cdot 0.3 \cdot 0.99 \cdot 0.05 + \\
&\quad 0.7 \cdot 0.6 \cdot 0.3 \cdot 0.01 \cdot 0.05 + \\
&\quad 0.3 \cdot 0.6 \cdot 0.02 \cdot 0.99 \cdot 0.8 + \\
&\quad 0.3 \cdot 0.6 \cdot 0.02 \cdot 0.01 \cdot 0.8 \\
&= \underline{0.00918}
\end{aligned}
$$

Solving the Example Query (2)

$$
\begin{aligned}
\underline{P(d^1, g^3, s^1)} &= \sum_{i=\{i^0,i^1\}, l=\{l^0,l^1\}} P(i, d^1, g^3, l, s^1) \\
&= \sum_{i=\{i^0,i^1\}, l=\{l^0,l^1\}} P(i)P(d^1)P(g^3|d^1, i)P(l|g^3)P(s^1|i) \\
&= P(i^0)P(d^1)P(g^3|d^1, i^0)P(l^0|g^3)P(s^1|i^0) + \\
&\quad P(i^0)P(d^1)P(g^3|d^1, i^0)P(l^1|g^3)P(s^1|i^0) + \\
&\quad P(i^1)P(d^1)P(g^3|d^1, i^1)P(l^0|g^3)P(s^1|i^1) + \\
&\quad P(i^1)P(d^1)P(g^3|d^1, i^1)P(l^1|g^3)P(s^1|i^1) \\
&= 0.7 \cdot 0.4 \cdot 0.7 \cdot 0.99 \cdot 0.05 + \\
&\quad 0.7 \cdot 0.4 \cdot 0.7 \cdot 0.01 \cdot 0.05 + \\
&\quad 0.3 \cdot 0.4 \cdot 0.2 \cdot 0.99 \cdot 0.8 + \\
&\quad 0.3 \cdot 0.4 \cdot 0.2 \cdot 0.01 \cdot 0.8 \\
&= \underline{0.029}
\end{aligned}
$$

Solving the Example Query (3)

$$
\begin{aligned}
P(d^0, g^3, s^1) &= 0.00918 \\
P(d^1, g^3, s^1) &= 0.029
\end{aligned}
$$

$$
Z = P(g^3, s^1) = 0.00918 + 0.029 = 0.03818
$$

$$
\boxed{P(D \mid g^3, s^1)} = \frac{1}{Z} \times \begin{bmatrix} 0.00918 \\ 0.029 \end{bmatrix} = \begin{bmatrix} 0.24044 \\ 0.75956 \end{bmatrix}
$$

$$
\Rightarrow \quad P(d^1 \mid g^3, s^1) = 75.96\%
$$

The probability that the AI class was difficult, given that Joe got a C grade but has a high SAT score, is $\approx 76\%$.

(see last chapter, “Reasoning Patterns”)

The Complexity of Exact Inference

Consider

- a Bayesian Network $\mathcal{B}$ over binary random variables $\mathcal{X} = \boldsymbol{X} \cup \boldsymbol{E} \cup \boldsymbol{Z}$
- a query $P(\boldsymbol{X} \mid \boldsymbol{E} = \boldsymbol{e})$

To compute

$$P(\boldsymbol{x}, \boldsymbol{e}) = \sum_{\boldsymbol{z}} P(\boldsymbol{x}, \boldsymbol{e}, \boldsymbol{z}) = \sum_{\boldsymbol{z}} \prod_{x_i \in \boldsymbol{x} \cup \boldsymbol{e} \cup \boldsymbol{z}} P(x_i \mid \mathrm{pa}(X_i))$$

for some specific assignment of values $\boldsymbol{x}$ to $\boldsymbol{X}$,

- we need to compute $P(\boldsymbol{x}, \boldsymbol{e}, \boldsymbol{z})$ for every combination of values $\boldsymbol{z}$ of the variables $\boldsymbol{Z}$
- If $|\boldsymbol{Z}| = k$, there are 2^k such assignments $\boldsymbol{z}$
- If we only have few evidence variables $\boldsymbol{E}$ and few query variables $\boldsymbol{X}$, then $k \approx N$ and we must sum out over $O(2^N)$ terms $P(\boldsymbol{x}, \boldsymbol{e}, \boldsymbol{z})$!
- Not feasible for networks of reasonable complexity ...

The Complexity of Exact Inference

The Bad News

- It can be shown that the problem of inference in graphical models is $\mathcal{NP}$**-hard**, and therefore requires **exponential time**, in the worst case.

The Good News:

- Many real-world problems seem to be far from the worst case ...
- ... and for many sparsely connected networks (i.e., networks where variables tend to have few parents), inference can be done quite efficiently with optimised algorithms (e.g., $\rightarrow$ *Variable Elimination*)

Making Inference More Efficient (in the Average Case)

Problem with the Standard Approach:

- Computations are highly redundant
- Many terms are computed many (an exponential number of!) times ...

Consider example from above:

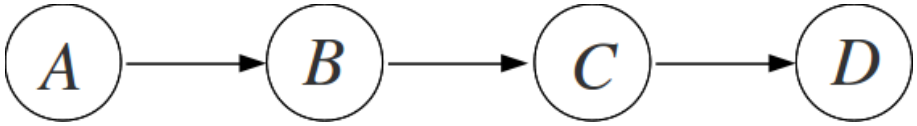
$$\begin{aligned} P(d^0, g^3, s^1) \quad = \quad & 0.7 \times 0.6 \times 0.3 \times 0.99 \times 0.05 + \\ & 0.7 \times 0.6 \times 0.3 \times 0.01 \times 0.05 + \\ & 0.3 \times 0.6 \times 0.02 \times 0.99 \times 0.8 + \\ & 0.3 \times 0.6 \times 0.02 \times 0.01 \times 0.8 \end{aligned}$$

In the following:

- Demonstrate the problem on a simple linear example network
- Explain basic idea of **Dynamic Programming**
- Show that inference on this simple network can be done in linear time
- Then: Generalise this process to general networks.

A Simple Example

Consider a simple linearly structured Bayesian network:

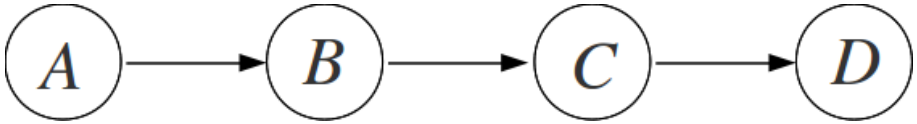


where each variable has k values

Task:

Compute $P(D)$.

A Simple Example



Standard Inference by Enumeration:

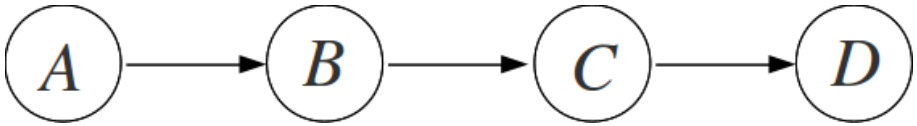
$$P(D) = \sum_c \sum_b \sum_a P(a,b,c,D) = \sum_c \sum_b \sum_a P(a)P(b|a)P(c|b)P(D|c)$$

- would repeatedly compute $P(a)$ for each b, c and d
- would repeatedly compute $\sum_a P(a)P(b|a) = P(b)$ for each c, d
- would repeatedly compute $\sum_b \sum_a P(a)P(b|a)P(c|b) = \sum_b P(b)P(c|b) = P(c)$ for each d.

☞ **Wasteful**

☞ **Idea:** Pre-compute these parts and store them for repeated use.

A Simple Example



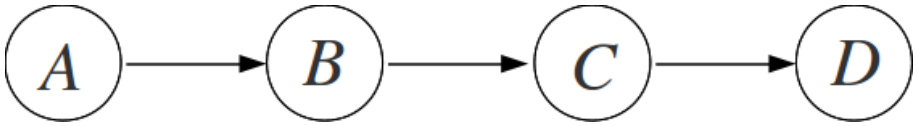
Step 1: Let's first compute $P(A)$:

$\Rightarrow$ Trivial. The numbers $P(a^0), P(a^1), \ldots$ are given by A's CPD.

Computational complexity of this first step:

$O(1)$ (constant).

A Simple Example



Step 2: Let's compute $P(B)$:

$$P(B) = \sum_a P(a)P(B \mid a)$$

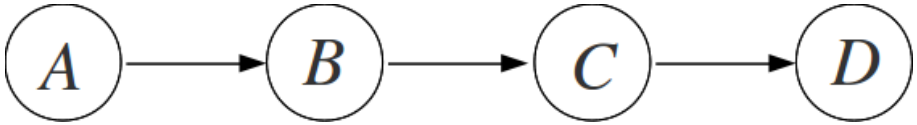
The numbers for $P(B \mid a)$ are in B's CPD; $P(A)$ **has been computed before.**

Computational complexity (if both A and B have k values) is

$$O(k \times k) = O(k^2)$$

- k multiplications $P(a)P(b|a)$ (one for each value of A)
- $k - 1$ additions (the sum)
- Must compute this for all k values of B to get probability distribution $P(B)$.

A Simple Example



Step 3: Let's compute $P(C)$:

$$P(C) = \sum_b P(b)P(C \mid b)$$

The numbers for $P(C \mid b)$ are in C's CPD; $P(B)$ **has been computed before.**

Complexity:

$$O(k \times k) = O(k^2).$$

Step 4: Compute $P(D)$: analogous to above, **using cached results for $P(C)$ ...**

Complexity of the Whole Process

Assume:

- A Bayesian network in the form of a strict chain $X_1 \rightarrow ... \rightarrow X_N$
- where each variable X_i has k values.
- An algorithm that recursively computes $P(X_{i+1})$ from $P(X_i)$:

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

and **caches the result** $P(X_{i+1})$ as a table (list of values).

Complexity of the i^{th} step – computing $P(X_{i+1})$:

$$O(k \times k) = O(k^2)$$

Cost of the entire process:

$$O(N \times k^2)$$

☞ **For fixed k, this is linear in N!**

Complexity of the Whole Process

Notes:

- Example where inference can be done in **linear time**, despite the exponential size $O(k^N)$ of the joint distribution!
- This extreme reduction of complexity was made possible by the specific linear structure of the network
- Linear network is extremely **sparse**: every variable has only one parent
- Computed not single values, but *sets of values* (distributions over X_i)
- These are stored and then used to compute $P(X_{i+1})$.

☞ Algorithm needs to store **tables** as intermediate results (see *factors* τ below).

A somewhat more general view of what we just did

- **Want to compute**

$$P(D) = \sum_c \sum_b \sum_a P(a,b,c,D) = \sum_c \sum_b \sum_a P(a)P(b|a)P(c|b)P(D|c)$$

- **Step 1:** Push in summation over A:

$$P(D) = \sum_c \sum_b P(c|b)P(D|c) \sum_a P(a)P(b|a)$$

- **Step 2:** Compute innermost sum ("sum out A") for each value b^i of B:

$$\sum_a P(a)P(b^i|a) = \tau_A(b^i)$$

Store results as a function (table) $\boxed{\tau_A(B)}$ (holds a value $\tau_A(b^i)$ for each b^i)

- **Replace summation by table lookup:**

$$P(D) = \sum_c \sum_b P(c|b)P(D|c)\,\tau_A(b)$$

- **Now repeat the story: want to compute**

$$P(D) \quad = \quad \sum_c \sum_b P(c|b)P(D|c)\,\tau_A(b)$$

- **Step 1:** Push in the next summation (over B):

$$P(D) \quad = \quad \sum_c P(D|c) \sum_b P(c|b)\tau_A(b)$$

- **Step 2:** Compute innermost sum ("sum out B") for each value c^i of C:

$$\sum_b P(c^i|b)\tau_A(b) \quad = \quad \tau_B(c^i)$$

and store the results as table $\boxed{\tau_B(C)}$

- **Replace summation by table lookup:**

$$P(D) \quad = \quad \sum_c P(D|c)\,\tau_B(c)$$

which can be easily computed for each value d^i of D. □

Insights

Central Insights

1. **Sparse Connectivity:** Due to the structure of the Bayesian network, some subexpressions in the joint distribution depend only on a small number of variables.
2. **Caching:** By computing these expressions once and caching the results, we avoid computing them exponentially many times.

Notes:

- This is a form of **Dynamic Programming** – doing summations inside out, storing intermediate results for re-use
- Computing $P(D) = \sum_c \sum_b \sum_a P(a)P(b|a)P(c|b)P(D|c)$ in the naive way would have us compute every $\sum_a P(a)P(b|a) = P(b)$ many times (once for every value of C and D)
- In general: in a chain of length N, we would compute this internal sum *exponentially many times*!

The Variable Elimination Algorithm for Bayesian Networks

Comments:

- The linear network above was *extremely* simple and 'nice'
- Generally, a Bayesian network does not have a simple linear structure
- A variable usually depends on *several* other variables
- Intermediate factors to be cached become multi-dimensional
- Not clear what the optimal order of variables is for summing out.

The Variable Elimination Algorithm for Bayesian Networks

Variable Elimination Algorithm for Bayesian Networks

Given:

- A Bayesian network $\mathcal{B}$ over variables $\mathcal{X} = \boldsymbol{X} \cup \boldsymbol{E} \cup \boldsymbol{Z}$
- A query $P(\boldsymbol{X} \mid \boldsymbol{E} = \boldsymbol{e})$
- Some ordering $Z_1, ..., Z_k$ of the non-query variables $\boldsymbol{Z}$

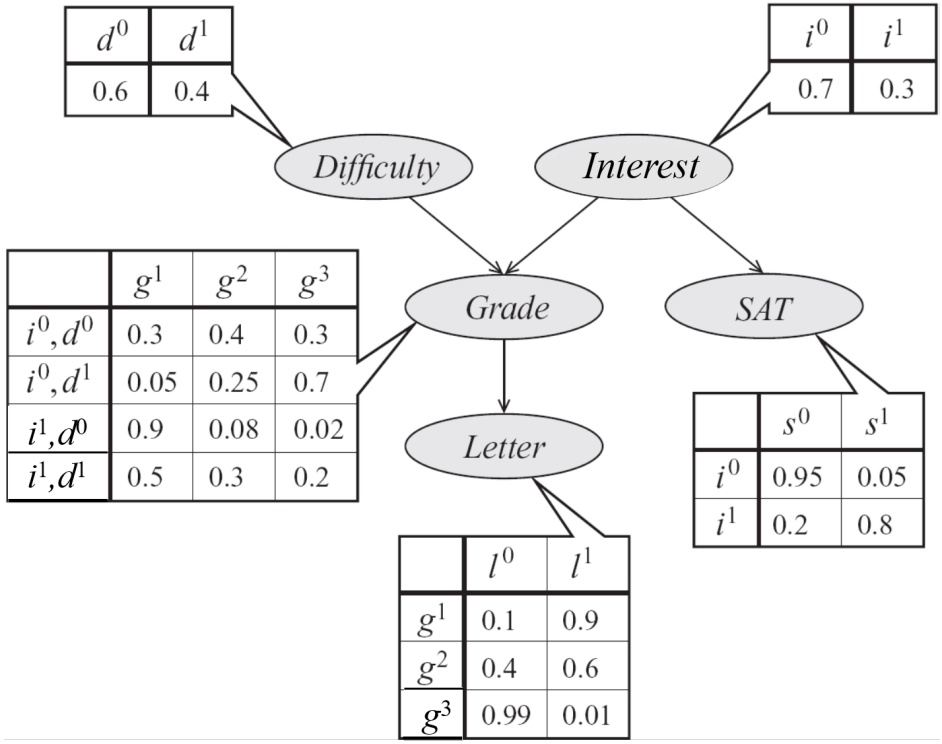
Eliminate Variables $\boldsymbol{Z}$: for $i = 1, ..., k$ do

1. Push in summation over Z_i as far as possible.
   Let $\boldsymbol{U} =$ all variables that appear in this inner sum, except Z_i
2. Compute the product of terms in this inner sum, for all possible combinations of values of $Z_i \cup \boldsymbol{U} \Rightarrow$ temporary table $\mathcal{T}$
3. ... and sum out over Z_i (i.e., for each value combination $\boldsymbol{u}$, sum over all z_i)
4. Store resulting table (of dimensionality $|\boldsymbol{U}|$) as intermediate "factor" $\tau_{Z_i}(\boldsymbol{U})$

Simplify: Multiply out whatever factors are left; the result is a table $P(\boldsymbol{X}, \boldsymbol{e})$

Renormalise $P(\boldsymbol{X}, \boldsymbol{e})$ to obtain $P(\boldsymbol{X}|\boldsymbol{e})$.

An Example



Example query:

$$P(I \mid g^3) = ?$$

Solving the Query via Variable Elimination

Query:

$$P(I \mid g^3) = ?$$

Variable subsets (see algorithm above):

$$\begin{aligned} \boldsymbol{X} &= \{I\} \\ \boldsymbol{E} &= \{G\} \quad (\boldsymbol{e} = \{g^3\}) \\ \boldsymbol{Z} &= \{D, L, S\} \end{aligned}$$

Assume some variable ordering over $\boldsymbol{Z}$
(variables to be eliminated in this order):

$$D, L, S$$

Solving the Query via Variable Elimination

Starting Point:

$$P(I, g^3) = \sum_{D=d^0,d^1} \sum_{L=l^0,l^1} \sum_{S=s^0,s^1} P(D)P(I)P(g^3|D, I)P(L|g^3)P(S|I)$$

1. Eliminate D ($Z_1 = D$):

- Push in summation over D:

$$P(I, g^3) = \sum_L \sum_S P(I)P(L|g^3)P(S|I) \sum_D \underbrace{P(D)P(g^3|D, I)}_{\mathcal{T}}$$

- Compute $\mathcal{T} = P(D)P(g^3|D, I)$ for all values of D, I
  (see next slide)

- $\boldsymbol{U} = \{I\}$ (variables in the inner expression other than D)

Solving the Query via Variable Elimination

- Compute $\mathcal{T} = P(D)P(g^3|D, I)$ for all values of D, I:

D	I	$P(D)P(g^3\|D, I)$
d^0	i^0	$0.6 \times 0.3 = 0.18$
d^0	i^1	$0.6 \times 0.02 = 0.012$
d^1	i^0	$0.4 \times 0.7 = 0.28$
d^1	i^1	$0.4 \times 0.2 = 0.08$

- Sum out D:

I	$\sum_D P(D)P(g^3\|D, I)$
i^0	$0.18 + 0.28 = 0.46$
i^1	$0.012 + 0.08 = 0.092$

- Store result as table $\boxed{\tau_D(I)}$:

$$\tau_D(I) = \begin{bmatrix} 0.46 \\ 0.092 \end{bmatrix}$$

Solving the Query via Variable Elimination

New Target:

$$P(I, g^3) \quad = \quad \sum_L \sum_S P(I)P(L|g^3)P(S|I)\,\tau_D(I)$$

2. Eliminate L ($Z_2 = L$):

- Push in summation over L:

$$P(I, g^3) \quad = \quad \sum_S P(I)P(S|I)\,\tau_D(I) \sum_L \underbrace{P(L|g^3)}_{\mathcal{T}}$$

- Compute $\mathcal{T} = P(L|g^3)$ for all values of L
  (see next slide)

- $\boldsymbol{U} = \{\}$ (variables in the inner expression other than L)

Solving the Query via Variable Elimination

- Compute $\mathcal{T} = P(L|g^3)$ for all values of L:

L	$P(L\|g^3)$
l^0	0.99
l^1	0.01

(from L's CPD)

- Sum out L:

$$\tau_L = 0.99 + 0.01 = 1.0$$

(the function τ_L has no arguments because $\boldsymbol{U} = \{\}$)

$\Rightarrow$ is a **constant**.

Solving the Query via Variable Elimination

New Target:

$$P(I, g^3) \quad = \quad \sum_S P(I)P(S|I)\,\tau_D(I)\,\tau_L$$

3. Eliminate S ($Z_3 = S$):

- Push in summation over S:

$$P(I, g^3) \quad = \quad P(I)\,\tau_D(I)\,\tau_L \sum_S \underbrace{P(S|I)}_{\mathcal{T}}$$

- Compute $\mathcal{T} = P(S|I)$ for all values of S, I
  (see next slide)

- $\boldsymbol{U} = \{I\}$ (variables in the inner expression other than S)

Solving the Query via Variable Elimination

- Compute $\mathcal{T} = P(S|I)$ for all values of S, I:

S	I	$P(S\|I)$
s^0	i^0	0.95
s^0	i^1	0.2
s^1	i^0	0.05
s^1	i^1	0.8

(= CPD of S)

- Sum out S:

I	$\sum_S P(S\|I)$
i^0	$0.95 + 0.05 = 1.0$
i^1	$0.2 + 0.8 = 1.0$

- Store result as table $\boxed{\tau_S(I)}$:

$$\tau_S(I) = \begin{bmatrix} 1.0 \\ 1.0 \end{bmatrix}$$

Solving the Query via Variable Elimination

New Target:

$$
\begin{aligned}
P(I, g^3) &= P(I)\ \tau_D(I)\ \tau_L\ \tau_S(I) \\
&= \begin{bmatrix}0.7\\0.3\end{bmatrix} \times \begin{bmatrix}0.46\\0.092\end{bmatrix} \times 1 \times \begin{bmatrix}1.0\\1.0\end{bmatrix} \\
&= \begin{bmatrix}0.7 \times 0.46 \times 1.0\\0.3 \times 0.092 \times 1.0\end{bmatrix} \\
&= \begin{bmatrix}0.322\\0.0276\end{bmatrix} \\
Z &= 0.322 + 0.0276 = 0.3496 \\
\boxed{P(I \mid g^3)} &= \frac{1}{Z} \cdot \begin{bmatrix}0.322\\0.0276\end{bmatrix} \approx \begin{bmatrix}0.92105\\0.07895\end{bmatrix}
\end{aligned}
$$

The probability that Joe is highly motivated, given that he got a C grade (and given that we know nothing else about him) is $\approx 7.9\%$.

Computational Complexity of Variable Elimination

Consider

- a Bayesian network with N variables $\mathcal{X} = \{X_1, ..., X_N\}$
- let $M =$ maximum number of entries in any of the intermediate factors $\tau_{Z_i}(\boldsymbol{U})$ that are generated during variable elimination.

Then

- the number of arithmetic operations performed by the variable elimination algorithm is $O(N \cdot M)$

  (easy to show; proof omitted)

The Bad News:

Have not gotten rid of the problem of exponential complexity (of course):
The intermediate factors $\tau_{Z_i}(\boldsymbol{U})$ (and thus M) can grow to **exponential size** (the number of entries is exponential in the number of the variables $\boldsymbol{U}$).

Computational Complexity of Variable Elimination

More detailed analysis:

- Real complexity depends on specific structure of the given network graph, and the ordering of the variables in the elimination process
- ... but systematic analysis of this is extremely complex.
- Not treated in this class.

General Conclusion:

- Exact inference can be feasible in **very sparse** networks

  (if we happen to guess a good variable ordering)
- but is still **exponential** (and thus **intractable**) in general!

Advanced Topics (not treated here)

A lot ...

Enhancements of variable elimination algorithms:

- Methods for finding good variable elimination orderings; Conditioning; ...

Alternative classes of algorithms for efficient inference:

- Clique Trees, Factor Graphs, Message Passing and Belief Update Algorithms, ...

What you should remember of this section

- The General Inference-by-Enumeration Algorithm for BNs
- The complexity of exact inference
- The basic strategy in variable elimination
- That even with such optimisations, exact inference is generally intractable
- That this is only a small part of a large field of inference algorithms for Bayesian Networks.

Literature

Koller, Daphne and Friedman, Nir (2009).
Probabilistic Graphical Models: Principles and Techniques. Cambridge, MA: MIT Press.