Probabilistic (Graphical) Models

and inference

Oliver Obst · Autumn 2024

WESTERN SYDNEY
UNIVERSITY

Probabilistic (Graphical) Models and Inference

(PGM: Probabilistic Graphical Models: Principles and Techniques by Daphne Koller and Nir Friedman. MIT Press)
(PMLI: Probabilistic Machine Learning: An introduction by Kevin Murphy. MIT Press)

Week	Lecture	Required reading	Assessment
1 Monday, 4 March 2024	Introduction, Probability Theory	PGM Chapter 2, PMLI Chapter 6.1	
2 Monday, 11 March 2024	Directed and undirected networks introduction	PGM Chapter 3 & 4	Quiz 1
3 Monday, 18 March 2024	Variable elimination	PGM Chapter 9	
4 Monday, 25 March 2024	Belief propagation		Quiz 2
5 Monday, 1 April 2024	public holiday		5 April 2024: census date
6 Monday, 8 April 2024	Message passing / Graph neural networks		
7 Monday, 15 April 2024	Sampling		Quiz 3
8 Monday, 22 April 2024	Mid-term break		
9 Monday, 29 April 2024	Variational inference		Intra-session exam
10 Monday, 6 May 2024	Autoregressive models		Quiz 4
11 Monday, 13 May 2024	Variational Auto-Encoders		
12 Monday, 20 May 2024	GANs		Quiz 5
13 Monday, 27 May 2024	Energy-based models		
14 Monday, 3 June 2024	Evaluating generative models		Quiz 6
Monday, 17 June 2024			Project due

Factors

P factorises over G

- Let G be a graph over $X_1, \ldots, X_n$.
- P factorises over G if

$$P(X_1, \ldots, X_n) = \prod_i P(X_i \mid \mathsf{Pa}_G(X_i))$$

Factors

- A factor $\phi(X_1, \ldots, X_k)$

  $\phi : \text{Val}(X_1, \ldots, X_k) \rightarrow \mathbb{R}$

- Scope $= \{X_1, \ldots, X_k\}$

Factors

a factor is a function

- A factor $\phi(X_1, \ldots, X_k)$

  $\phi : \text{Val}(X_1, \ldots, X_k) \rightarrow \mathbb{R}$

- Scope $= \{X_1, \ldots, X_k\}$

Factors

a factor is a function

- A factor $\phi(X_1, \ldots, X_k)$

  $\phi : \text{Val}(X_1, \ldots, X_k) \to \mathbb{R}$

the variable arguments of that function

- Scope $= \{X_1, \ldots, X_k\}$

A joint distribution is a factor

$$P(I, D, G)$$

I	D	G	Probability
i^0	d^0	g^1	0.126
i^0	d^0	g^2	0.168
i^0	d^0	g^3	0.126
i^0	d^1	g^1	0.009
i^0	d^1	g^2	0.045
i^0	d^1	g^3	0.126
i^1	d^0	g^1	0.252
i^1	d^0	g^2	0.0224
i^1	d^0	g^3	0.0056
i^1	d^1	g^1	0.06
i^1	d^1	g^2	0.036
i^1	d^1	g^3	0.024

A joint distribution is a factor

$P(I, D, G)$

I	D	G	Probability
i^0	d^0	g^1	0.126
i^0	d^0	g^2	0.168
i^0	d^0	g^3	0.126
i^0	d^1	g^1	0.009
i^0	d^1	g^2	0.045
i^0	d^1	g^3	0.126
i^1	d^0	g^1	0.252
i^1	d^0	g^2	0.0224
i^1	d^0	g^3	0.0056
i^1	d^1	g^1	0.06
i^1	d^1	g^2	0.036
i^1	d^1	g^3	0.024

Here, they sum to 1.0, but for general factors, they don't have to...

Unnormalised measure $P(I, D, g^1)$

$P(I, D, g^1)$

I	D	G	Probability
i^0	d^0	g^1	0.126
i^0	d^1	g^1	0.009
i^1	d^0	g^1	0.252
i^1	d^1	g^1	0.06

Scope $= \{I, D\}$

Unnormalised measure $P(I, D, g^1)$

$P(I, D, g^1)$

Scope

I	D	G	Probability
i^0	d^0	g^1	0.126
i^0	d^1	g^1	0.009
i^1	d^0	g^1	0.252
i^1	d^1	g^1	0.06

Scope $= \{I, D\}$

Conditional Probability Distribution (CPD)

$$P(G \mid I, D)$$

	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

General factors

A	B	ϕ
a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10

General factors

don't have to be probabilities

A	B	ϕ
a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10

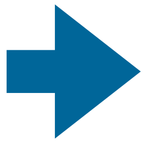
Factor product

$\phi_1(A, B)$

a^1	b^1	0.5
a^1	b^2	0.8
a^2	b^1	0.1
a^2	b^2	0
a^3	b^1	0.3
a^3	b^2	0.9

$\phi_2(B, C)$

b^1	c^1	0.5
b^1	c^2	0.7
b^2	c^1	0.1
b^2	c^2	0.2



a^1	b^1	c^1	0.5*0.5 = 0.25
a^1	b^1	c^2	0.5*0.7 = 0.35
a^1	b^2	c^1	0.8 * 0.1 = 0.08
a^1	b^2	c^2	0.8 * 0.2 = 0.16
a^2	b^1	c^1	0.1* 0.5 = 0.05
a^2	b^1	c^2	0.1 * 0.7 = 0.07
a^2	b^2	c^1	0 * 0.1 = 0
a^2	b^2	c^2	0 * 0.2 = 0
a^3	b^1	c^1	0.3 * 0.5 = 0.15
a^3	b^1	c^2	0.3 * 0.7 = 0.21
a^3	b^2	c^1	0.9 * 0.1 = 0.09
a^3	b^2	c^2	0.9 * 0.2 = 0.18

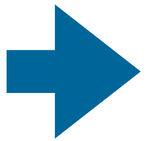
Factor product

$\phi_1(A, B)$

a^1	b^1	0.5
a^1	b^2	0.8
a^2	b^1	0.1
a^2	b^2	0
a^3	b^1	0.3
a^3	b^2	0.9

$\phi_2(B, C)$

b^1	c^1	0.5
b^1	c^2	0.7
b^2	c^1	0.1
b^2	c^2	0.2



a^1	b^1	c^1	0.5*0.5 = 0.25
a^1	b^1	c^2	0.5*0.7 = 0.35
a^1	b^2	c^1	0.8 * 0.1 = 0.08
a^1	b^2	c^2	0.8 * 0.2 = 0.16
a^2	b^1	c^1	0.1* 0.5 = 0.05
a^2	b^1	c^2	0.1 * 0.7 = 0.07
a^2	b^2	c^1	0 * 0.1 = 0
a^2	b^2	c^2	0 * 0.2 = 0
a^3	b^1	c^1	0.3 * 0.5 = 0.15
a^3	b^1	c^2	0.3 * 0.7 = 0.21
a^3	b^2	c^1	0.9 * 0.1 = 0.09
a^3	b^2	c^2	0.9 * 0.2 = 0.18

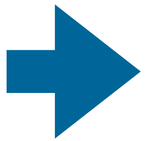
Factor product

$\phi_1(A, B)$

a^1	b^1	0.5
a^1	b^2	0.8
a^2	b^1	0.1
a^2	b^2	0
a^3	b^1	0.3
a^3	b^2	0.9

$\phi_2(B, C)$

b^1	c^1	0.5
b^1	c^2	0.7
b^2	c^1	0.1
b^2	c^2	0.2



a^1	b^1	c^1	0.5*0.5 = 0.25
a^1	b^1	c^2	0.5*0.7 = 0.35
a^1	b^2	c^1	0.8 * 0.1 = 0.08
a^1	b^2	c^2	0.8 * 0.2 = 0.16
a^2	b^1	c^1	0.1* 0.5 = 0.05
a^2	b^1	c^2	0.1 * 0.7 = 0.07
a^2	b^2	c^1	0 * 0.1 = 0
a^2	b^2	c^2	0 * 0.2 = 0
a^3	b^1	c^1	0.3 * 0.5 = 0.15
a^3	b^1	c^2	0.3 * 0.7 = 0.21
a^3	b^2	c^1	0.9 * 0.1 = 0.09
a^3	b^2	c^2	0.9 * 0.2 = 0.18

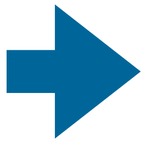
Factor product

$\phi_1(A, B)$

a^1	b^1	0.5
a^1	b^2	0.8
a^2	b^1	0.1
a^2	b^2	0
a^3	b^1	0.3
a^3	b^2	0.9

$\phi_2(B, C)$

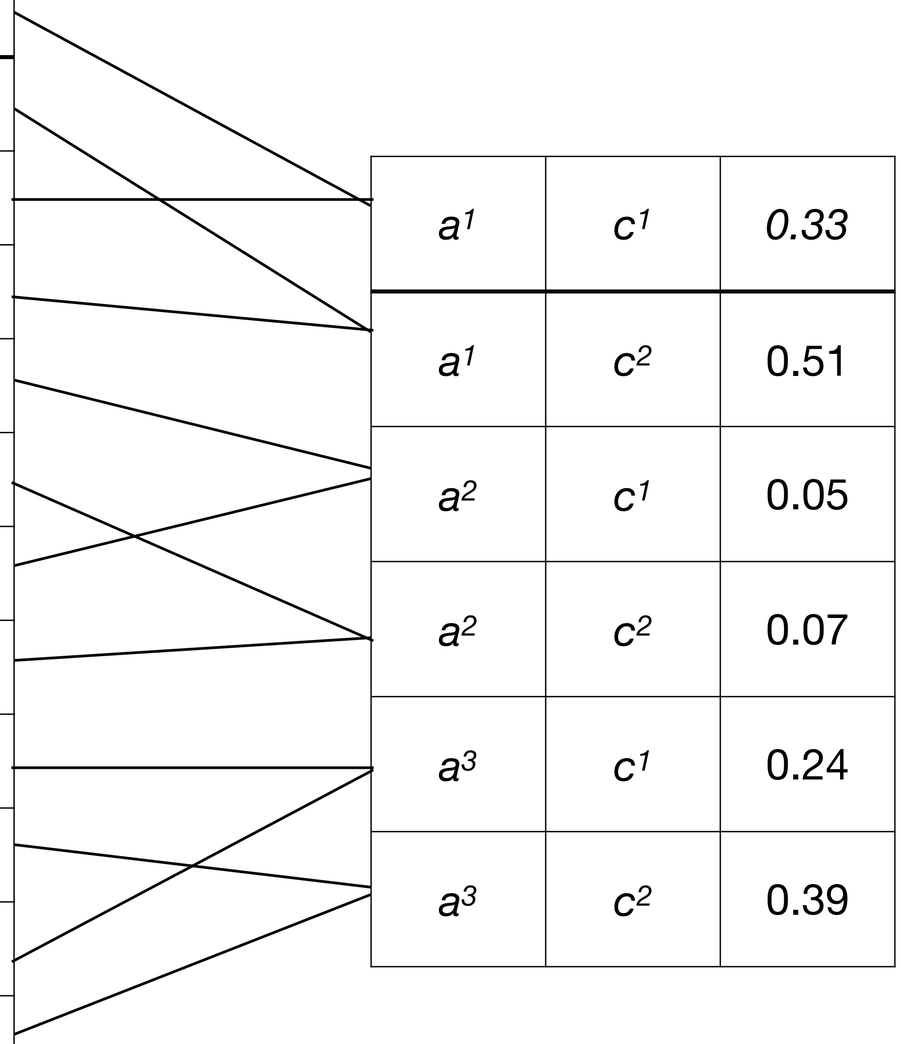
b^1	c^1	0.5
b^1	c^2	0.7
b^2	c^1	0.1
b^2	c^2	0.2



a^1	b^1	c^1	0.5*0.5 = 0.25
a^1	b^1	c^2	0.5*0.7 = 0.35
a^1	b^2	c^1	0.8 * 0.1 = 0.08
a^1	b^2	c^2	0.8 * 0.2 = 0.16
a^2	b^1	c^1	0.1* 0.5 = 0.05
a^2	b^1	c^2	0.1 * 0.7 = 0.07
a^2	b^2	c^1	0 * 0.1 = 0
a^2	b^2	c^2	0 * 0.2 = 0
a^3	b^1	c^1	0.3 * 0.5 = 0.15
a^3	b^1	c^2	0.3 * 0.7 = 0.21
a^3	b^2	c^1	0.9 * 0.1 = 0.09
a^3	b^2	c^2	0.9 * 0.2 = 0.18

Factor Marginalisation

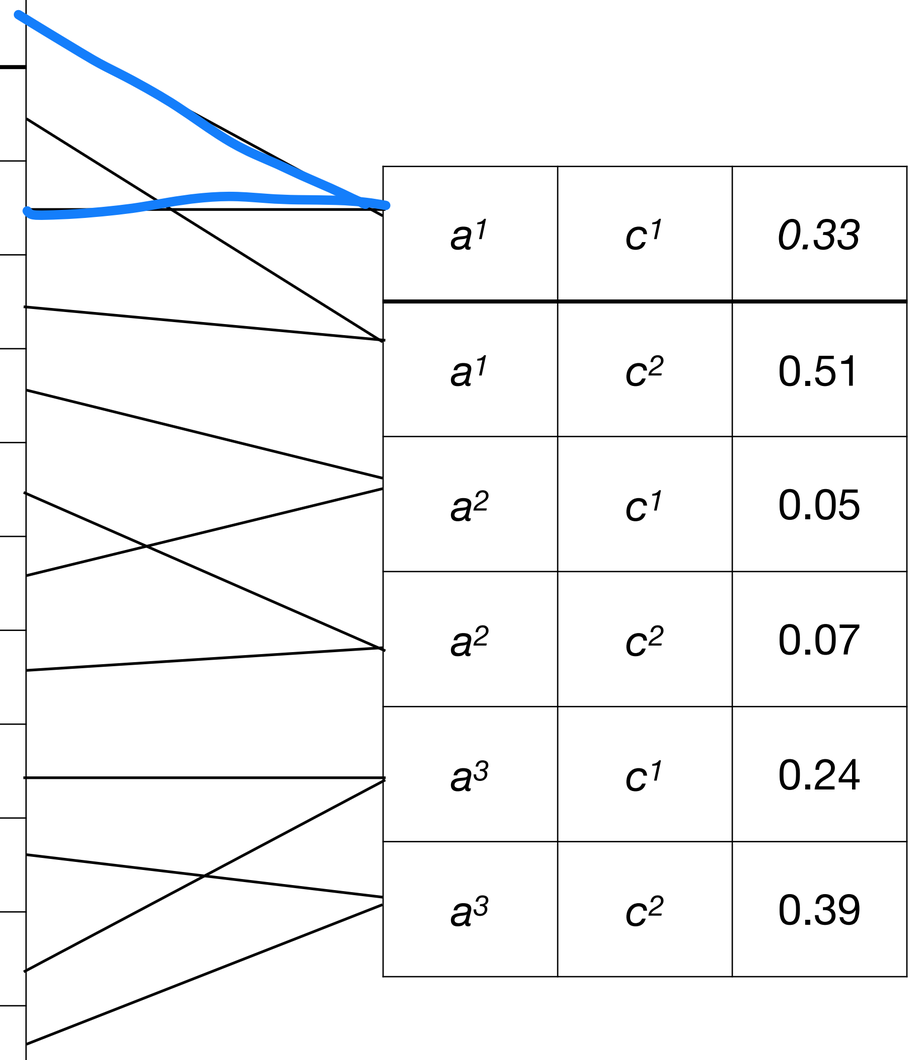
a^1	b^1	c^1	*0.25*
a^1	b^1	c^2	0.35
a^1	b^2	c^1	0.08
a^1	b^2	c^2	0.16
a^2	b^1	c^1	0.05
a^2	b^1	c^2	0.07
a^2	b^2	c^1	0
a^2	b^2	c^2	0
a^3	b^1	c^1	0.15
a^3	b^1	c^2	0.21
a^3	b^2	c^1	0.09
a^3	b^2	c^2	0.18



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 Marginalisation

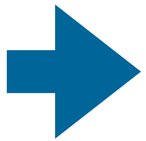
a^1	b^1	c^1	*0.25*
a^1	b^1	c^2	0.35
a^1	b^2	c^1	0.08
a^1	b^2	c^2	0.16
a^2	b^1	c^1	0.05
a^2	b^1	c^2	0.07
a^2	b^2	c^1	0
a^2	b^2	c^2	0
a^3	b^1	c^1	0.15
a^3	b^1	c^2	0.21
a^3	b^2	c^1	0.09
a^3	b^2	c^2	0.18



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 Reduction

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	b^1	c^1	*0.25*
a^1	b^2	c^1	0.08
a^2	b^1	c^1	0.05
a^2	b^2	c^1	0
a^3	b^1	c^1	0.15
a^3	b^2	c^1	0.09

Factors

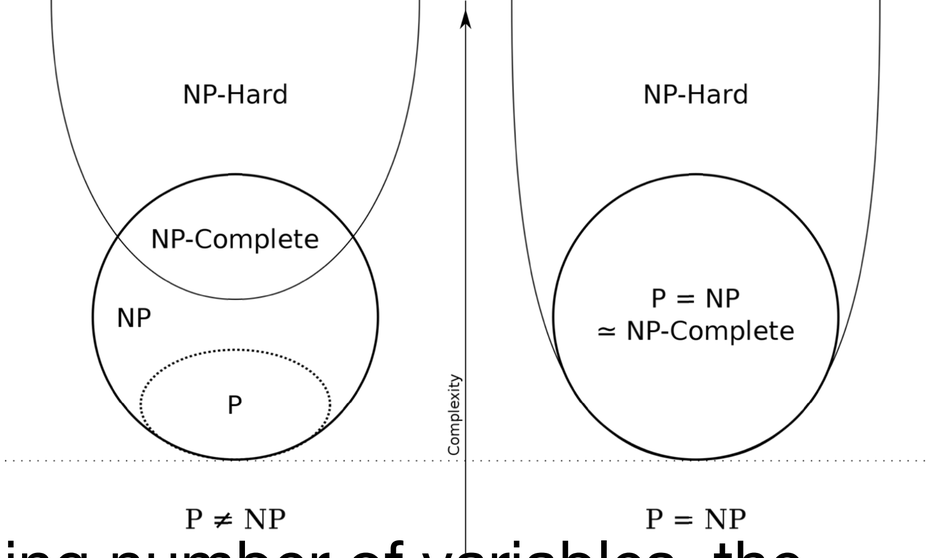
- Factors are a building block for defining distributions in high-dimensional spaces
- We have seen a set of basic operations on factors that are useful for manipulating these probability distributions

Variable elimination

Conditional probability queries

- Evidence: $\boldsymbol{E} = \boldsymbol{e}$
- Query: a subset of variables $\boldsymbol{Y}$
- Task: compute $P(\boldsymbol{Y} \mid \boldsymbol{E} = \boldsymbol{e})$

NP-hardness

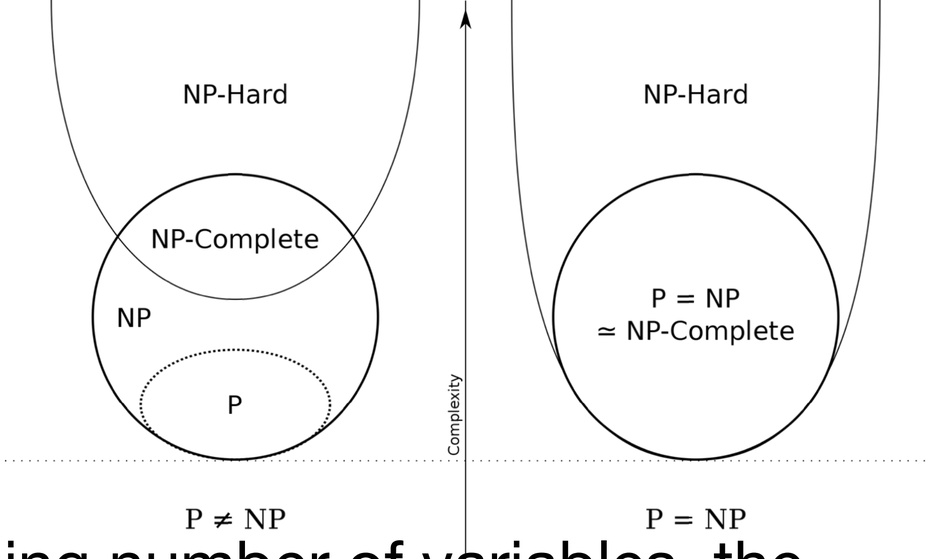


- A problem is NP-hard means:

  There is no efficient solution (that is, with increasing number of variables, the problem becomes much harder, in terms of compute time)

https://en.wikipedia.org/wiki/NP-hardness

NP-hardness



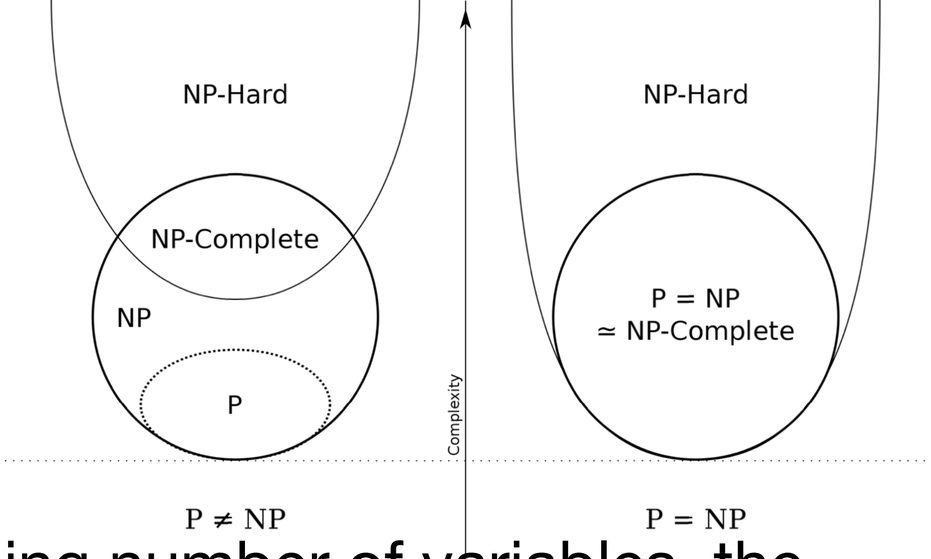
- A problem is NP-hard means:

  There is no efficient solution (that is, with increasing number of variables, the problem becomes much harder, in terms of compute time)

- A more technical definition — theory of computation (not in this class)

https://en.wikipedia.org/wiki/NP-hardness

NP-hardness



- A problem is NP-hard means:

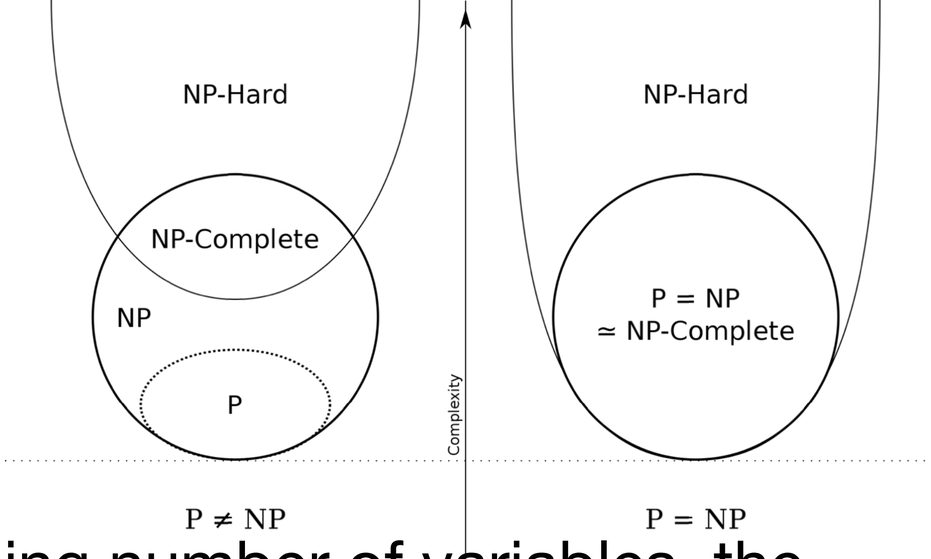
  There is no efficient solution (that is, with increasing number of variables, the problem becomes much harder, in terms of compute time)

- A more technical definition — theory of computation (not in this class)

- But one thing to know from this: in theory of computation, we are interested in the worst-case complexity. Many NP-hard problems can be solved in reasonable time, in most cases.

https://en.wikipedia.org/wiki/NP-hardness

NP-hardness



- A problem is NP-hard means:

  There is no efficient solution (that is, with increasing number of variables, the problem becomes much harder, in terms of compute time)

- A more technical definition — theory of computation (not in this class)

- But one thing to know from this: in theory of computation, we are interested in the worst-case complexity. Many NP-hard problems can be solved in reasonable time, in most cases.

- Probabilistic inference is NP-hard.

https://en.wikipedia.org/wiki/NP-hardness

Proofs in computational complexity

- In computational complexity, to show that a new problem is NP-hard:

  You start with a problem that is known to be NP-hard.

Proofs in computational complexity

- In computational complexity, to show that a new problem is NP-hard:

  You start with a problem that is known to be NP-hard.

- Then, if you can reduce that known problem to your new problem — you know your new problem is also NP-hard.

Proofs in computational complexity

- In computational complexity, to show that a new problem is NP-hard:

  You start with a problem that is known to be NP-hard.

- Then, if you can reduce that known problem to your new problem — you know your new problem is also NP-hard.

- One such problem that is frequently used for this type of proof is called the 3-SAT problem. It is about satisfiability of logical formulas that can consist of many clauses, but each clause has at most 3 logical literals (hence 3-SAT).

Proofs in computational complexity

- In computational complexity, to show that a new problem is NP-hard:

  You start with a problem that is known to be NP-hard.

- Then, if you can reduce that known problem to your new problem — you know your new problem is also NP-hard.

- One such problem that is frequently used for this type of proof is called the 3-SAT problem. It is about satisfiability of logical formulas that can consist of many clauses, but each clause has at most 3 logical literals (hence 3-SAT).

- This is here to explain what this computational complexity terms are about — you will find some of those in the PGM book. We will not go deeper into computational complexity for now.

Proofs in computational complexity

- In computational complexity, to show that a new problem is NP-hard:

  You start with a problem that is known to be NP-hard.

- Then, if you can reduce that known problem to your new problem — you know your new problem is also NP-hard.

- One such problem that is frequently used for this type of proof is called the 3-SAT problem. It is about satisfiability of logical formulas that can consist of many clauses, but each clause has at most 3 logical literals (hence 3-SAT).

- This is here to explain what this computational complexity terms are about — you will find some of those in the PGM book. We will not go deeper into computational complexity for now.

- 3-SAT can be reduced to probabilistic inference in BN.

Probabilistic inference

Today we consider exact inference in graphical models

In particular, we focus on conditional probability queries,

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

(e.g., the probability of a patient having a disease given some observed symptoms)

Probabilistic inference

Today we consider exact inference in graphical models

In particular, we focus on conditional probability queries,

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

(e.g., the probability of a patient having a disease given some observed symptoms)

Let $\mathbf{W} = \mathcal{X} - \mathbf{Y} - \mathbf{E}$ be the random variables that are neither the query nor the evidence. Each of these joint distributions can be computed by marginalizing over the other variables:

$$p(\mathbf{Y},\mathbf{e}) = \sum_{\mathbf{w}} p(\mathbf{Y},\mathbf{e},\mathbf{w}), \quad p(\mathbf{e}) = \sum_{\mathbf{y}} p(\mathbf{y},\mathbf{e})$$

Naively marginalizing over all unobserved variables requires an exponential number of computations

Probabilistic inference

Today we consider exact inference in graphical models

In particular, we focus on conditional probability queries,

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

(e.g., the probability of a patient having a disease given some observed symptoms)

Let $\mathbf{W} = \mathcal{X} - \mathbf{Y} - \mathbf{E}$ be the random variables that are neither the query nor the evidence. Each of these joint distributions can be computed by marginalizing over the other variables:

$$p(\mathbf{Y},\mathbf{e}) = \sum_{\mathbf{w}} p(\mathbf{Y},\mathbf{e},\mathbf{w}), \quad p(\mathbf{e}) = \sum_{\mathbf{y}} p(\mathbf{y},\mathbf{e})$$

Naively marginalizing over all unobserved variables requires an exponential number of computations

Does there exist a more efficient algorithm?

Complexity for approximate marginal inference

Might there exist polynomial-time algorithms that can *approximately* answer marginal queries, i.e. for some ϵ, find ρ such that

$$\rho - \epsilon \leq p(\mathbf{Y} \mid \mathbf{E} = \mathbf{e}) \leq \rho + \epsilon \quad ?$$

Suppose such an algorithm exists, for any $\epsilon \in (0, \frac{1}{2})$. Consider the following:

1. Start with $\mathbf{E} = \{ X = 1 \}$
2. For $i = 1, \ldots, n$:
3. Let $q_i = \arg\max_q \; p(Q_i = q \mid \mathbf{E})$
4. $\mathbf{E} \leftarrow \mathbf{E} \cup (Q_i = q_i)$

Complexity for approximate marginal inference

Might there exist polynomial-time algorithms that can *approximately* answer marginal queries, i.e. for some ϵ, find ρ such that

$$\rho - \epsilon \leq p(\mathbf{Y} \mid \mathbf{E} = \mathbf{e}) \leq \rho + \epsilon \quad ?$$

Suppose such an algorithm exists, for any $\epsilon \in (0, \frac{1}{2})$. Consider the following:

1. Start with $\mathbf{E} = \{ X = 1 \}$
2. For $i = 1, \ldots, n$:
3. Let $q_i = \arg\max_q \; p(Q_i = q \mid \mathbf{E})$
4. $\mathbf{E} \leftarrow \mathbf{E} \cup (Q_i = q_i)$

Complexity for approximate marginal inference

Might there exist polynomial-time algorithms that can *approximately* answer marginal queries, i.e. for some ϵ, find ρ such that

$$\rho - \epsilon \leq p(\mathbf{Y} \mid \mathbf{E} = \mathbf{e}) \leq \rho + \epsilon \quad ?$$

Suppose such an algorithm exists, for any $\epsilon \in (0, \frac{1}{2})$. Consider the following:

1. Start with $\mathbf{E} = \{ X = 1 \}$
2. For $i = 1, \ldots, n$:
3. Let $q_i = \arg\max_q \; p(Q_i = q \mid \mathbf{E})$
4. $\mathbf{E} \leftarrow \mathbf{E} \cup (Q_i = q_i)$

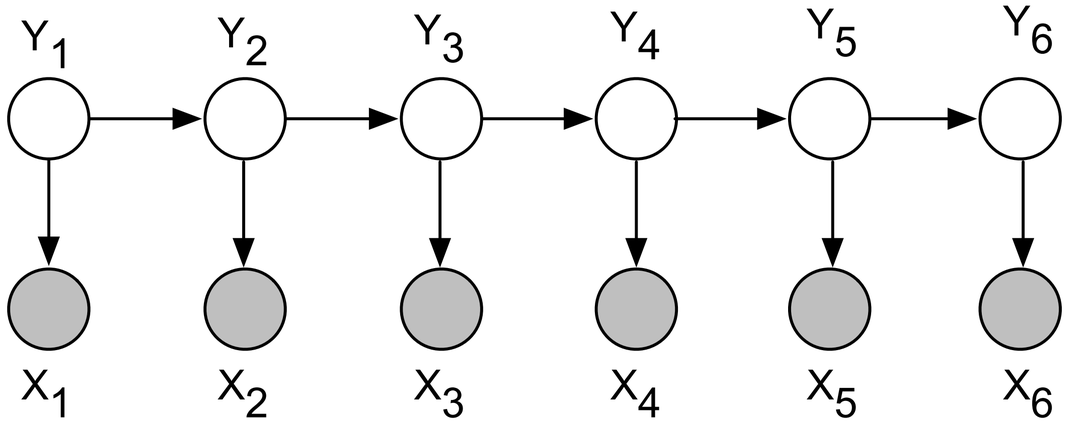
At termination, $\mathbf{E}$ is a satisfying assignment (if one exists). Pf by induction:

- In iteration i, if $\exists$ satisfying assignment extending $\mathbf{E}$ for **both** $q_i = 0$ *and* $q_i = 1$, then choice in line 3 does not matter
- Otherwise, suppose $\exists$ satisfying assignment extending $\mathbf{E}$ for $q_i = 1$ but not for $q_i = 0$. Then, $p(Q_i = 1 \mid \mathbf{E}) = 1$ and $p(Q_i = 0 \mid \mathbf{E}) = 0$
- Even if approximate inference returned $p(Q_i = 1 \mid \mathbf{E}) = 0.501$ and $p(Q_i = 0 \mid \mathbf{E}) = .499$, we would still choose $q_i = 1$

Thus, it is even NP-hard to *approximately* perform marginal inference!

Probabilistic inference in practice

- NP-hardness simply says that there exist difficult inference problems
- Real-world inference problems are not necessarily as hard as these worst-case instances
- Some graphs are in fact **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 (VE)

Exact algorithm for probabilistic inference in **any** graphical model

Running time will depend on the *graph structure*

Uses **dynamic programming** to circumvent enumerating all assignments

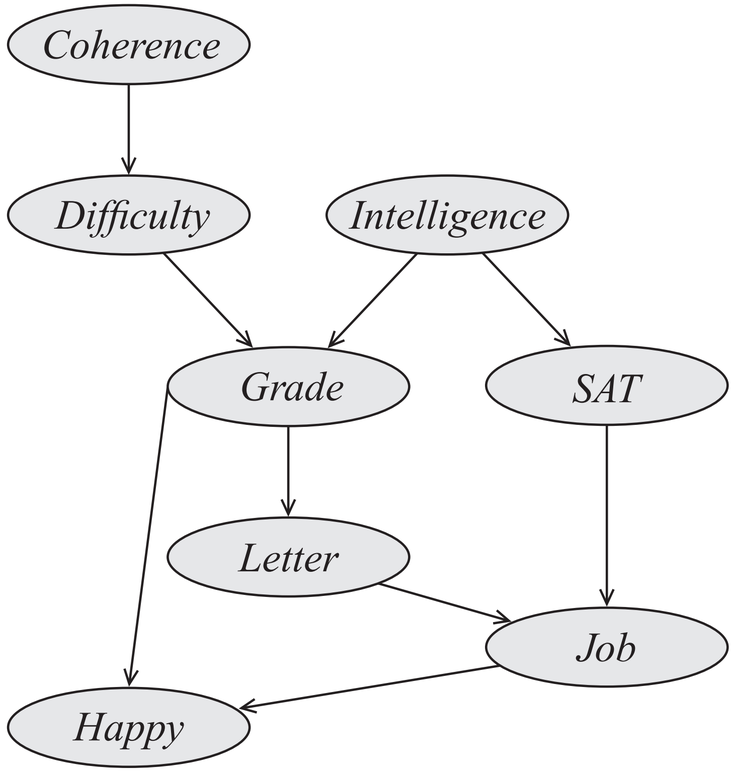
First we introduce the concept for computing marginal probabilities, $p(X_i)$, in Bayesian networks

After this, we will generalize to MRFs and conditional queries

Sum-product

$$\phi_C(C)\,\phi_D(C,D)\,\phi_I(I)\,\phi_G(G,I,D)$$

$$\phi_S(S,I)\,\phi_L(L,G)\,\phi_J(J,L,S)\,\phi_H(H,G,J)$$

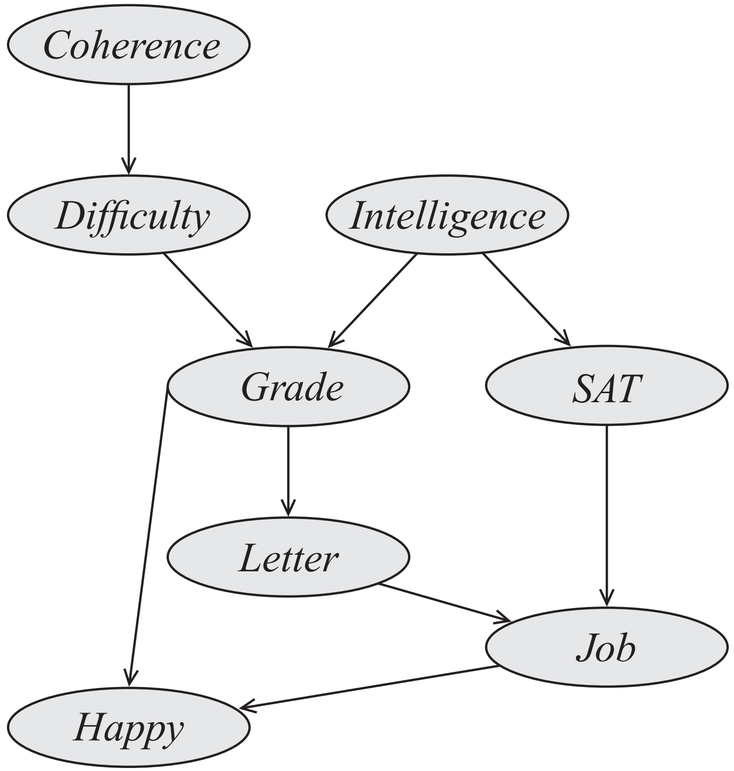


Sum-product

$$\phi_C(C)\,\phi_D(C,D)\,\phi_I(I)\,\phi_G(G,I,D)$$

$$\phi_S(S,I)\,\phi_L(L,G)\,\phi_J(J,L,S)\,\phi_H(H,G,J)$$

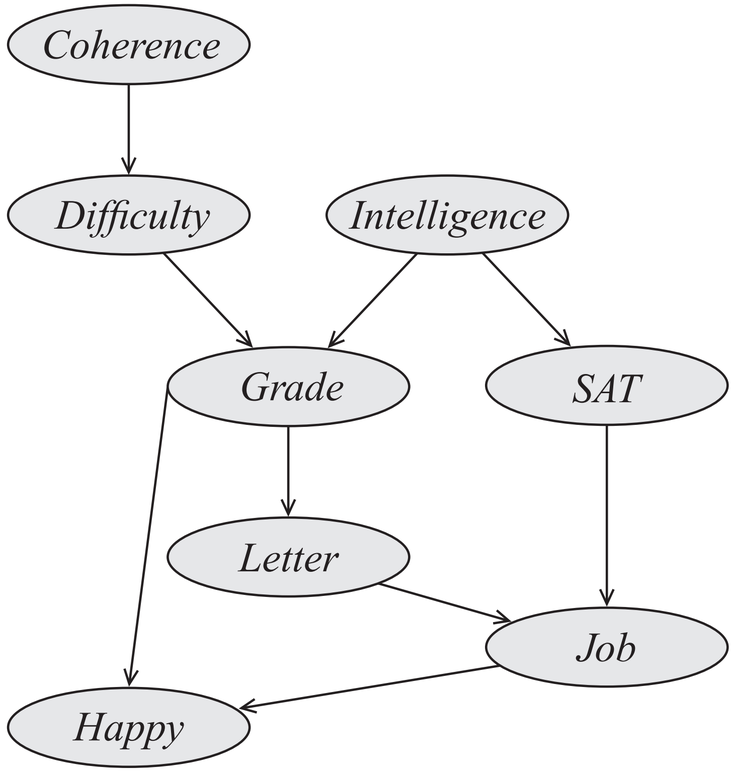
Compute $P(J)$



Sum-product

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

Compute $P(J)$



Evidence: reduced factors

Evidence: reduced factors

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

Evidence: reduced factors

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

$$W = \{X_1, \ldots, X_n\} - \boldsymbol{Y} - \boldsymbol{E}$$

Evidence: reduced factors

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

$$P(\boldsymbol{Y}, \boldsymbol{E} = \boldsymbol{e}) = \sum_{\boldsymbol{W}} P(\boldsymbol{Y}, \boldsymbol{W}, \boldsymbol{E} = \boldsymbol{e})$$

$$W = \{X_1, \ldots, X_n\} - \boldsymbol{Y} - \boldsymbol{E}$$

Evidence: reduced factors

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

$$W = \{X_1, \ldots, X_n\} - \boldsymbol{Y} - \boldsymbol{E}$$

$$P(\boldsymbol{Y},\boldsymbol{E}=\boldsymbol{e}) = \sum_{\boldsymbol{W}} P(\boldsymbol{Y},\boldsymbol{W},\boldsymbol{E}=\boldsymbol{e})$$

$$= \sum_{\boldsymbol{W}} \frac{1}{Z} \prod_{k} \phi_k(\boldsymbol{D}_k, \boldsymbol{E}=\boldsymbol{e})$$

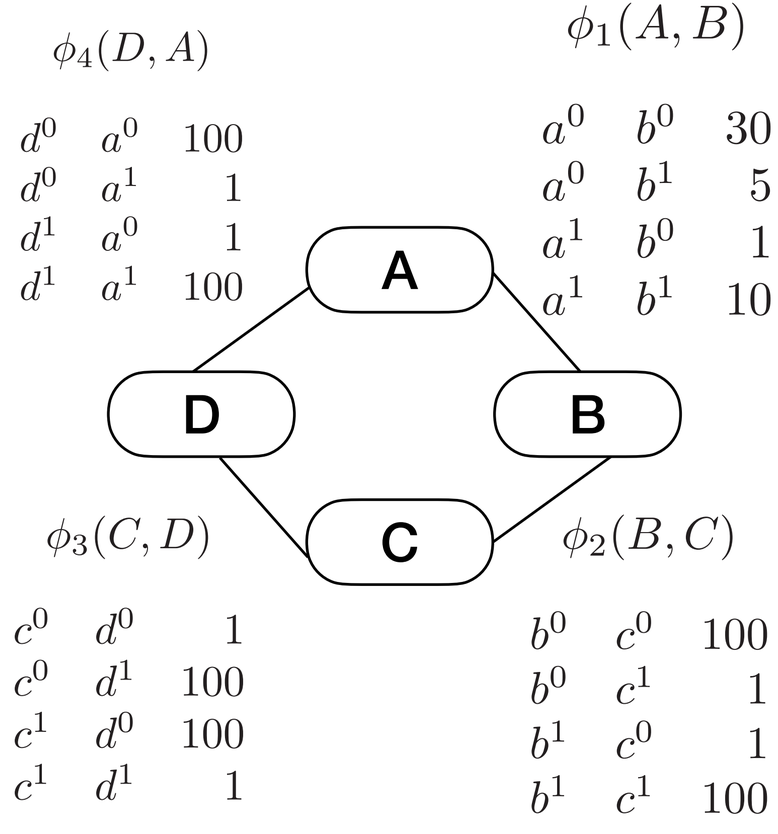
Evidence: reduced factors

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

$$P(\boldsymbol{Y},\boldsymbol{E}=\boldsymbol{e}) = \sum_{\boldsymbol{W}} P(\boldsymbol{Y},\boldsymbol{W},\boldsymbol{E}=\boldsymbol{e})$$

$$= \sum_{\boldsymbol{W}} \frac{1}{Z} \prod_{k} \phi_k(\boldsymbol{D}_k,\boldsymbol{E}=\boldsymbol{e})$$

$$W = \{X_1,\ldots,X_n\} - \boldsymbol{Y} - \boldsymbol{E}$$



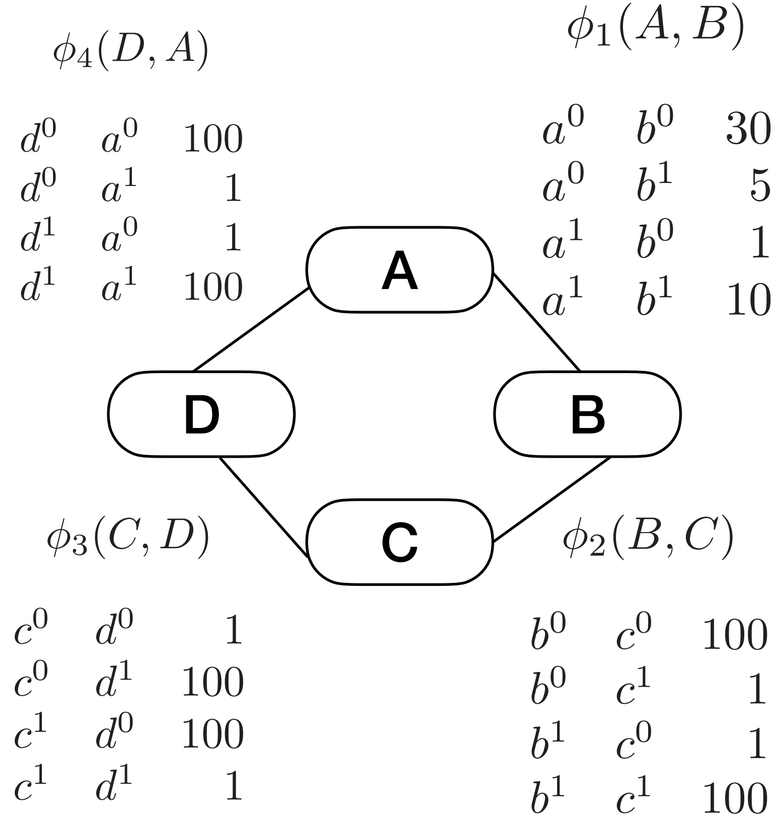
Evidence: reduced factors

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

$$P(\boldsymbol{Y}, \boldsymbol{E} = \boldsymbol{e}) = \sum_{\boldsymbol{W}} P(\boldsymbol{Y}, \boldsymbol{W}, \boldsymbol{E} = \boldsymbol{e})$$

$$= \sum_{\boldsymbol{W}} \frac{1}{Z} \prod_{k} \phi_k(\boldsymbol{D}_k, \boldsymbol{E} = \boldsymbol{e})$$

$$W = \{X_1, \ldots, X_n\} - \boldsymbol{Y} - \boldsymbol{E}$$



$$A = a^0$$

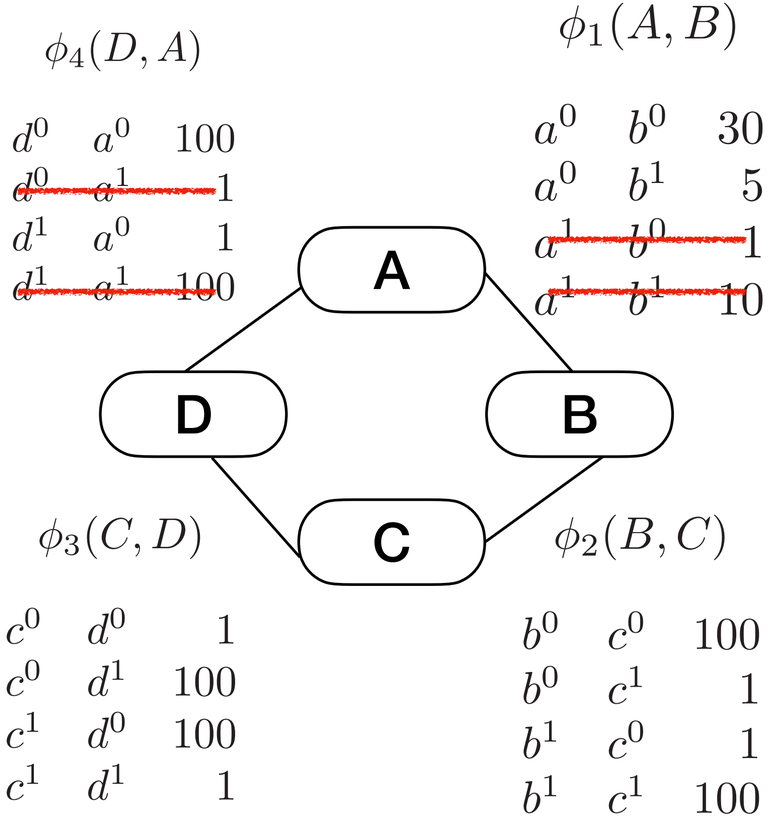
Evidence: reduced factors

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

$$P(\boldsymbol{Y},\boldsymbol{E}=\boldsymbol{e}) = \sum_{\boldsymbol{W}} P(\boldsymbol{Y},\boldsymbol{W},\boldsymbol{E}=\boldsymbol{e})$$

$$= \sum_{\boldsymbol{W}} \frac{1}{Z} \prod_{k} \phi_k(\boldsymbol{D}_k,\boldsymbol{E}=\boldsymbol{e})$$

$$W = \{X_1,\ldots,X_n\} - \boldsymbol{Y} - \boldsymbol{E}$$



$$A = a^0$$

Evidence: reduced factors

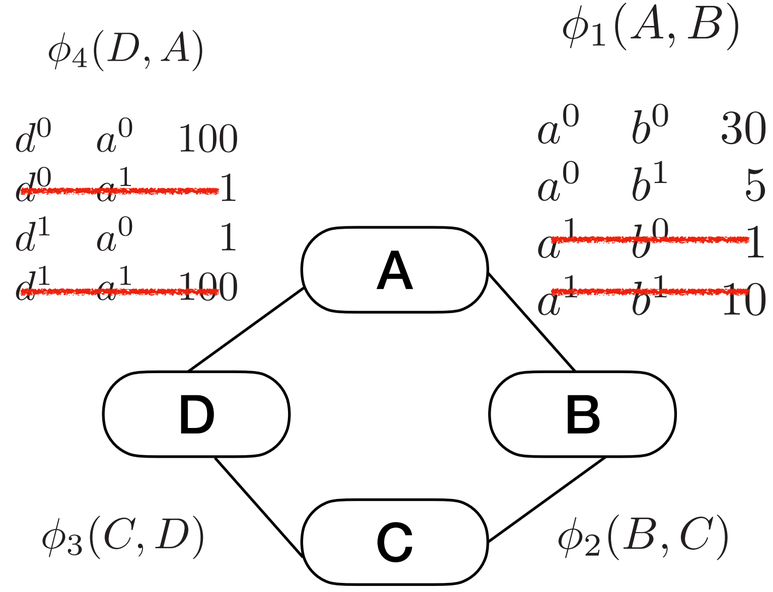
$$P(\boldsymbol{Y} \mid \boldsymbol{E} = \boldsymbol{e}) = \frac{P(\boldsymbol{Y}, \boldsymbol{E} = \boldsymbol{e})}{P(\boldsymbol{E} = \boldsymbol{e})}$$

$$P(\boldsymbol{Y}, \boldsymbol{E} = \boldsymbol{e}) = \sum_{\boldsymbol{W}} P(\boldsymbol{Y}, \boldsymbol{W}, \boldsymbol{E} = \boldsymbol{e})$$

$$= \sum_{\boldsymbol{W}} \frac{1}{Z} \prod_{k} \phi_k(\boldsymbol{D}_k, \boldsymbol{E} = \boldsymbol{e})$$

$$= \sum_{\boldsymbol{W}} \frac{1}{Z} \prod_{k} \phi'_k(\boldsymbol{D}'_k)$$

$$W = \{X_1, \ldots, X_n\} - \boldsymbol{Y} - \boldsymbol{E}$$



$$A = a^0$$

Evidence: reduced factors

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

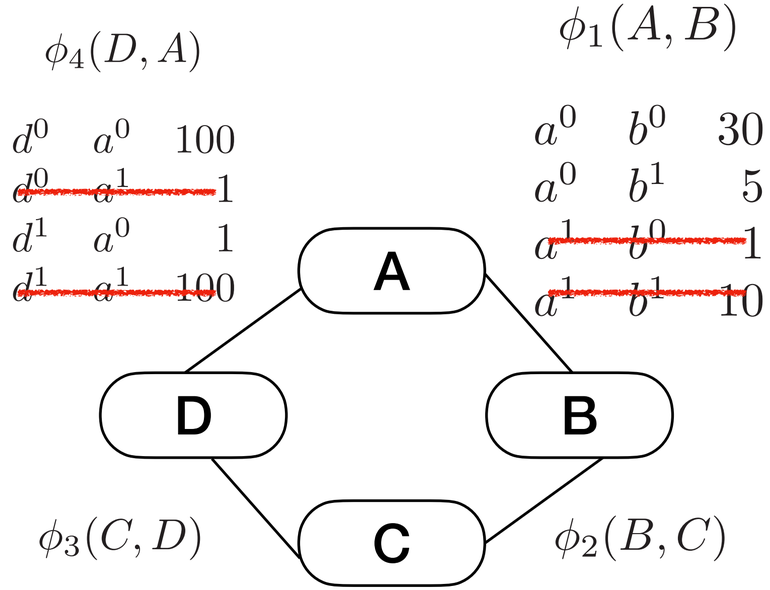
$$P(\boldsymbol{Y}, \boldsymbol{E} = \boldsymbol{e}) = \sum_{\boldsymbol{W}} P(\boldsymbol{Y}, \boldsymbol{W}, \boldsymbol{E} = \boldsymbol{e})$$

$$= \sum_{\boldsymbol{W}} \frac{1}{Z} \prod_{k} \phi_k(\boldsymbol{D}_k, \boldsymbol{E} = \boldsymbol{e})$$

$$= \sum_{\boldsymbol{W}} \frac{1}{Z} \prod_{k} \phi'_k(\boldsymbol{D}'_k)$$

Renormalising: compute the value from factors (ignoring Z), and then divide by the total (to turn into proper distribution)

$$W = \{X_1, \ldots, X_n\} - \boldsymbol{Y} - \boldsymbol{E}$$



$$A = a^0$$

Evidence: reduced factors

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

$$P(\boldsymbol{Y}, \boldsymbol{E} = \boldsymbol{e}) = \sum_{\boldsymbol{W}} P(\boldsymbol{Y}, \boldsymbol{W}, \boldsymbol{E} = \boldsymbol{e})$$

$$= \sum_{\boldsymbol{W}} \frac{1}{Z} \prod_{k} \phi_k(\boldsymbol{D}_k, \boldsymbol{E} = \boldsymbol{e})$$

$$\propto \sum_{\boldsymbol{W}} \frac{1}{Z} \prod_{k} \phi'_k(\boldsymbol{D}'_k)$$

Renormalising: compute the value from factors (ignoring Z), and then divide by the total (to turn into proper distribution)

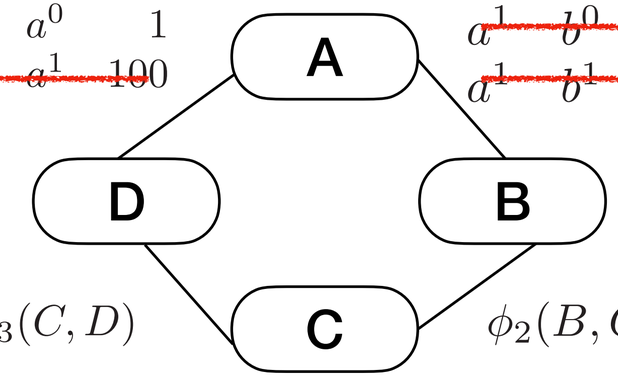
$$W = \{X_1, \ldots, X_n\} - \boldsymbol{Y} - \boldsymbol{E}$$

$\phi_4(D, A)$

d^0	a^0	100
~~d^0~~	~~a^1~~	~~1~~
d^1	a^0	1
~~d^1~~	~~a^1~~	~~100~~

$\phi_1(A, B)$

a^0	b^0	30
a^0	b^1	5
~~a^1~~	~~b^0~~	~~1~~
~~a^1~~	~~b^1~~	~~10~~



$\phi_3(C, D)$

c^0	d^0	1
c^0	d^1	100
c^1	d^0	100
c^1	d^1	1

$\phi_2(B, C)$

b^0	c^0	100
b^0	c^1	1
b^1	c^0	1
b^1	c^1	100

$$A = a^0$$

Approaches for inference in PGM

- Push summations into factor product
  - Variable elimination (today)
- Message passing over a graph
  - Belief propagation
  - Variational approximations
- Random sampling instantiations
  - Markov chain Monte Carlo
  - Importance sampling

Approaches for inference in PGM

- Push summations into factor product
  - Variable elimination (today)
- Message passing over a graph
  - Belief propagation
  - Variational approximations
- Random sampling instantiations
  - Markov chain Monte Carlo
  - Importance sampling

exact

Approaches for inference in PGM

- Push summations into factor product
  - Variable elimination (today)
- Message passing over a graph
  - Belief propagation
  - Variational approximations
- Random sampling instantiations
  - Markov chain Monte Carlo
  - Importance sampling

exact

exact & approximate methods

Approaches for inference in PGM

- Push summations into factor product
  - Variable elimination (today)
- Message passing over a graph
  - Belief propagation
  - Variational approximations
- Random sampling instantiations
  - Markov chain Monte Carlo
  - Importance sampling

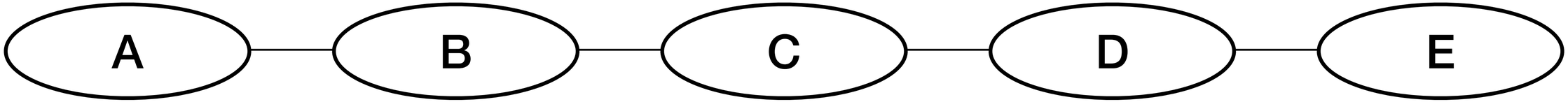
exact

exact & approximate methods

approximate

Variable Elimination

In chains

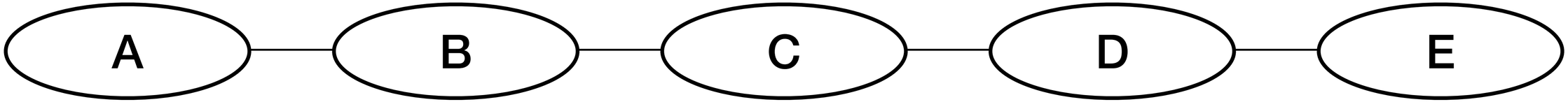


$$P(E) \propto \sum_D \sum_C \sum_B \sum_A \tilde{P}(A, B, C, D, E)$$

$$= \sum_D \sum_C \sum_B \sum_A \phi_1(A, B)\phi_2(B, C)\phi_3(C, D)\phi_4(D, E)$$

Variable Elimination

In chains



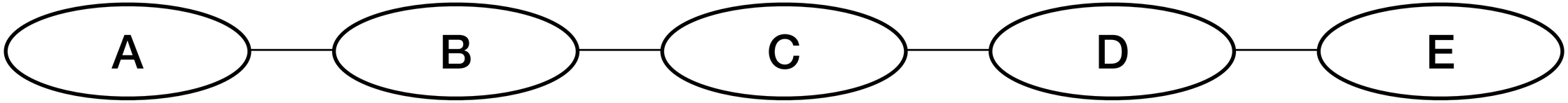
$$P(E) \propto \sum_D \sum_C \sum_B \sum_A \tilde{P}(A,B,C,D,E)$$

$$= \sum_D \sum_C \sum_B \sum_A \phi_1(A,B)\phi_2(B,C)\phi_3(C,D)\phi_4(D,E)$$

$$= \sum_D \sum_C \sum_B \phi_2(B,C)\phi_3(C,D)\phi_4(D,E) \sum_A \phi_1(A,B)$$

Variable Elimination

In chains



$$P(E) \propto \sum_D \sum_C \sum_B \sum_A \tilde{P}(A,B,C,D,E)$$

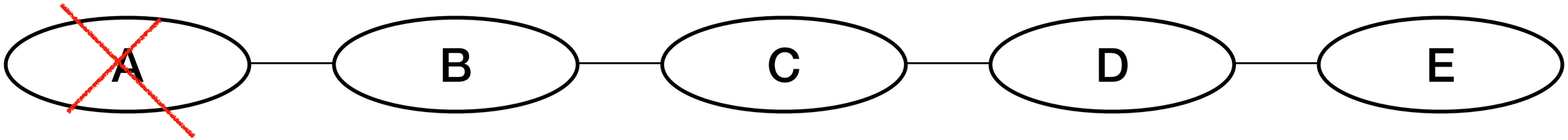
$$= \sum_D \sum_C \sum_B \sum_A \phi_1(A,B)\phi_2(B,C)\phi_3(C,D)\phi_4(D,E)$$

$$= \sum_D \sum_C \sum_B \phi_2(B,C)\phi_3(C,D)\phi_4(D,E) \sum_A \phi_1(A,B)$$

$$= \sum_D \sum_C \sum_B \phi_2(B,C)\phi_3(C,D)\phi_4(D,E)\tau_1(B)$$

Variable Elimination

In chains



$$P(E) \propto \sum_D \sum_C \sum_B \sum_A \tilde{P}(A,B,C,D,E)$$

$$= \sum_D \sum_C \sum_B \sum_A \phi_1(A,B)\phi_2(B,C)\phi_3(C,D)\phi_4(D,E)$$

$$= \sum_D \sum_C \sum_B \phi_2(B,C)\phi_3(C,D)\phi_4(D,E) \sum_A \phi_1(A,B)$$

$$= \sum_D \sum_C \sum_B \phi_2(B,C)\phi_3(C,D)\phi_4(D,E)\tau_1(B)$$

VE basic idea

Suppose we have a simple chain, $A \to B \to C \to D$, and we want to compute $p(D)$

$p(D)$ is a **set** of values, $\{p(D = d), d \in \mathrm{Val}(D)\}$. Algorithm computes sets of values at a time – an entire distribution

By the chain rule and conditional independence, the joint distribution factors 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 have to marginalize over A, B, C:

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

Explicitly

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

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

Explicitly

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

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

There is structure to the summation, e.g., repeated $P(c^1|b^1)P(d^1|c^1)$

Let's modify the computation to first compute

$$P(a^1)P(b^1|a^1) + P(a^2)P(b^1|a^2)$$

Explicitly

Computing by enumeration
(enumerating and adding all relevant cases)

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

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

There is structure to the summation, e.g., repeated $P(c^1|b^1)P(d^1|c^1)$

Let's modify the computation to first compute

$$P(a^1)P(b^1|a^1) + P(a^2)P(b^1|a^2)$$

Let's modify the computation to first compute

$$P(a^1)P(b^1|a^1) + P(a^2)P(b^1|a^2)$$

and

$$P(a^1)P(b^2|a^1) + P(a^2)P(b^2|a^2)$$

Then, we get

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

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

We define $\tau_1 : \mathrm{Val}(B) \to \Re, \quad \tau_1(b^i) = P(a^1)P(b^i|a^1) + P(a^2)P(b^i|a^2)$

We now have

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

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

We now have

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

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

We can once more reverse the order of the product and the sum and get

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

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

We now have

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

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

We can once more reverse the order of the product and the sum and get

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

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

There are still other repeated computations!

We define $\tau_2 : \mathrm{Val}(C) \to \Re$, with

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

Now we can compute the marginal $p(D)$ as

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

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

OK, what did we just do?

Our goal was to compute

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

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

OK, what did we just do?

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 the summations inside to obtain:

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

OK, what did we just do?

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 the summations inside to obtain:

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

Let's call $\psi_1(A,B) = P(A)P(B|A)$. Then, $\tau_1(B) = \sum_a \psi_1(a,B)$

Similarly, let $\psi_2(B,C) = \tau_1(B)P(C|B)$. Then, $\tau_2(C) = \sum_b \psi_1(b,C)$

This procedure is dynamic programming: computation is inside out instead of outside in

Inference in a chain

Generalizing the previous example, suppose we have a chain $X_1 \to X_2 \to \cdots \to X_n$ where each variable has k states

For $i = 1$ up to $n - 1$, compute (and cache)

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

Inference in a chain

Generalizing the previous example, suppose we have a chain $X_1 \to X_2 \to \cdots \to X_n$ where each variable has k states

For $i = 1$ up to $n - 1$, compute (and cache)

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

Each update takes k^2 time (why?)

The total running time is $\mathcal{O}(nk^2)$

Inference in a chain

Generalizing the previous example, suppose we have a chain $X_1 \to X_2 \to \cdots \to X_n$ where each variable has k states

For $i = 1$ up to $n - 1$, compute (and cache)

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

Each update takes k^2 time (why?)

The total running time is $\mathcal{O}(nk^2)$

In comparison, naively marginalizing over all latent variables has complexity $\mathcal{O}(k^n)$

We did inference over the joint without ever explicitly constructing it!

Summary so far

Worst-case analysis says that marginal inference is NP-hard

Even approximating it is NP-hard

In practice, due to the structure of the Bayesian network, we can cache computations that are otherwise computed exponentially many times

This depends on our having a good **variable elimination ordering**

Sum-product inference task

We want to give an algorithm to compute $p(\mathbf{Y})$ for BNs and MRFs

This can be reduced to the following **sum-product** inference task:

$$\text{Compute} \quad \tau(\mathbf{y}) = \sum_{\mathbf{z}} \prod_{\phi \in \Phi} \phi(\mathbf{z}_{\text{Scope}[\phi] \cap \mathbf{Z}}, \mathbf{y}_{\text{Scope}[\phi] \cap \mathbf{Y}}) \quad \forall \mathbf{y},$$

where Φ is a set of factors or potentials

For a BN, Φ is given by the conditional probability distributions for all variables,

$$\Phi = \{\phi_{X_i}\}_{i=1}^n = \{p(X_i \mid \mathbf{X}_{\text{Pa}(X_i)})\}_{i=1}^n,$$

and where we sum over the set $\mathbf{Z} = \mathcal{X} - \mathbf{Y}$

For Markov networks, the factors Φ correspond to the set of potentials which we earlier called C

- Sum-product returns an unnormalized distribution, so we divide by $\sum_{\mathbf{y}} \tau(\mathbf{y})$

Factor marginalisation

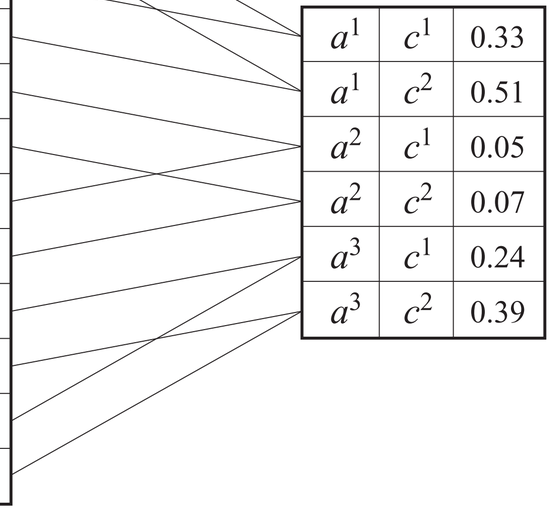
Let $\phi(\mathbf{X}, Y)$ be a factor where $\mathbf{X}$ is a set of variables and $Y \notin \mathbf{X}$

Factor marginalization of ϕ over Y (also called "summing out Y in ϕ") gives a new factor:

$$\tau(\mathbf{X}) = \sum_{Y} \phi(\mathbf{X}, Y)$$

For example,

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

Sum-product variable elimination

Order the variables $\mathbf{Z}$ (called the **elimination ordering**)

Iteratively marginalize out variable Z_i, one at a time

For each i,

1. Multiply all factors that have Z_i in their scope, generating 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, and add the new one

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 $\Phi'' \cup \{\tau\}$

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 $\Phi'' \cup \{\tau\}$

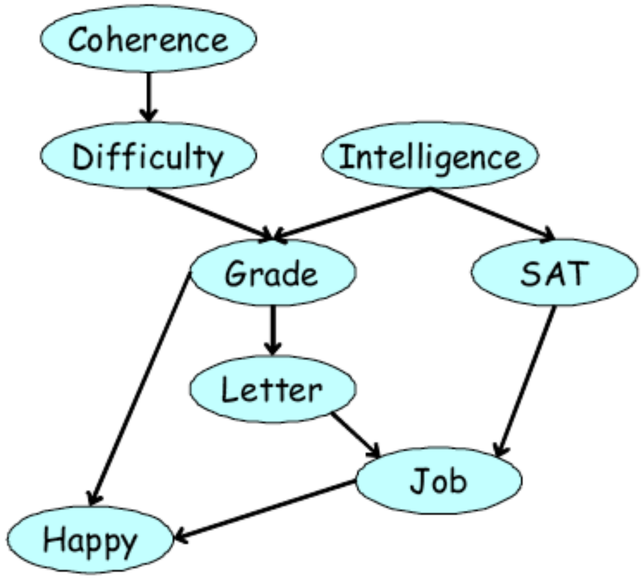
Φ: (big) phi

ϕ: phi

ψ: psi

τ: tau

Example



- What is $p(\mathrm{Job})$? Joint distribution factorizes as:

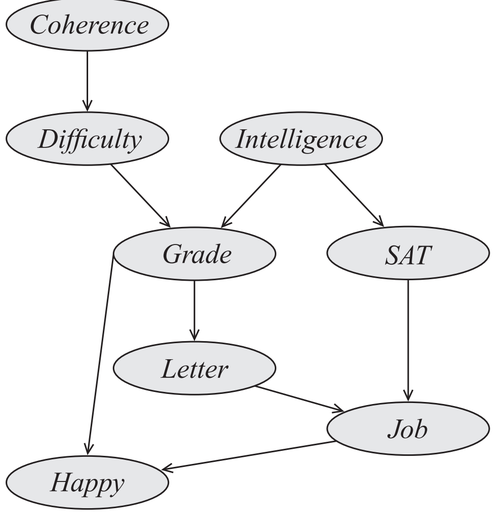
$$p(C, D, I, G, S, L, H, J) = p(C)p(D|C)p(I)p(G|D, I)p(L|G)P(S|I)P(J|S, L)p(H|J, G)$$

with factors

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

- Let's do variable elimination with ordering $\{C, D, I, H, G, S, L\}$

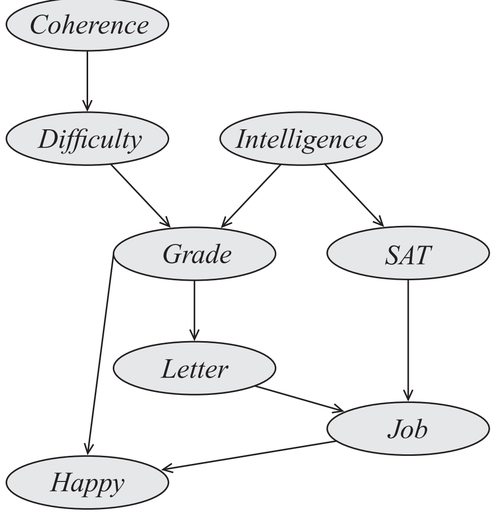
Variable elimination example (1)



Goal: $P(J)$

Eliminate: C, D, I, H, G, S, L

Variable elimination example (1)

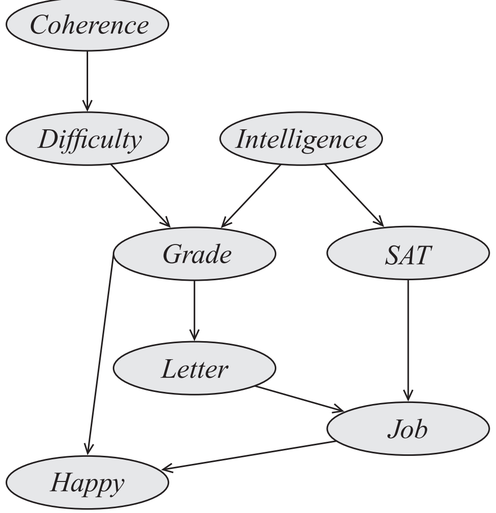


Goal: $P(J)$

Eliminate: C, D, I, H, G, S, L

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

Variable elimination example (1)



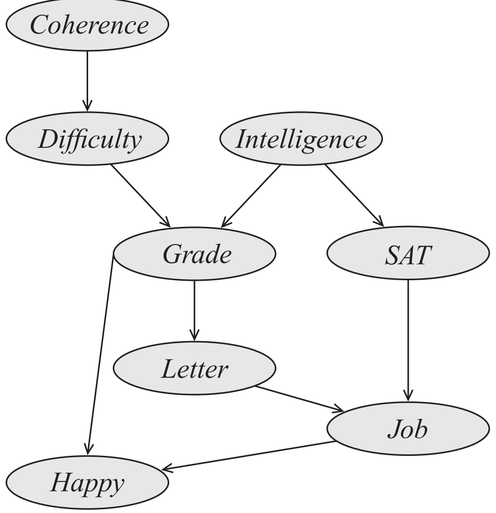
Goal: $P(J)$

Eliminate: C, D, I, H, G, S, L

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

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

Variable elimination example (1)



Goal: $P(J)$

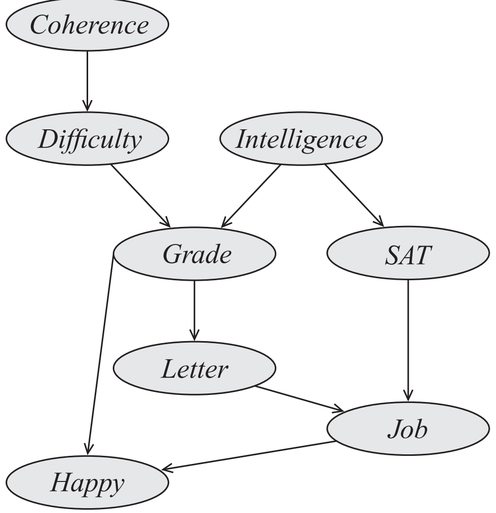
Eliminate: C, D, I, H, G, S, L

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

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

Compute $\tau_1(D) = \sum_C \phi_C(C)\,\phi_D(C,D)$

Variable elimination example (1)



Goal: $P(J)$

Eliminate: C, D, I, H, G, S, L

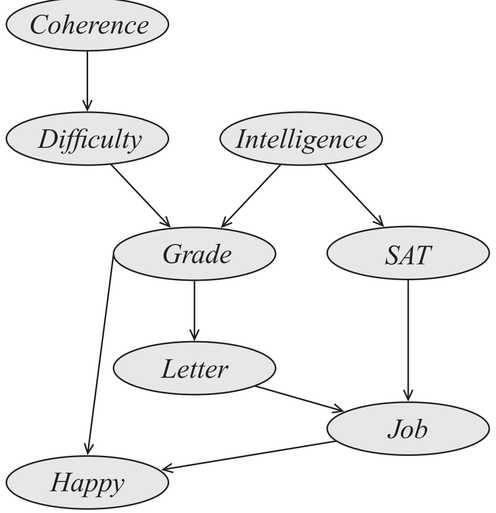
$$\sum_{L,S,G,H,I,D,C} \phi_J(J,L,S)\,\phi_L(L,G)\,\phi_S(S,I)\,\phi_G(G,I,D)\,\phi_H(H,G,J)\,\phi_I(I)\,\phi_D(C,D)\,\phi_C(C)$$

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

Compute $\tau_1(D) = \sum_C \phi_C(C)\,\phi_D(C,D)$

$$\sum_{L,S,G,H,I,D} \phi_J(J,L,S)\,\phi_L(L,G)\,\phi_S(S,I)\,\phi_G(G,I,D)\,\phi_H(H,G,J)\,\phi_I(I)\,\tau_1(D)$$

Variable elimination example (2)

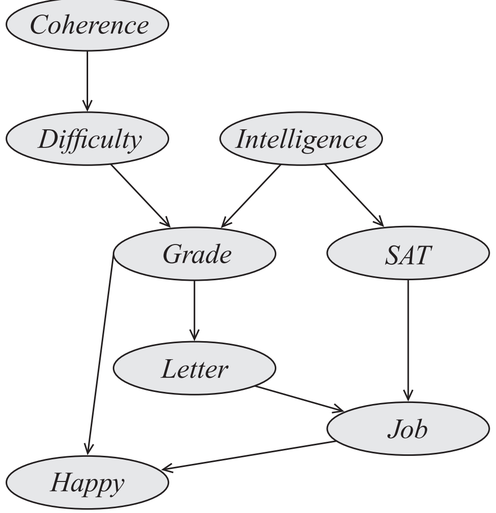


Goal: $P(J)$

Eliminate: D, I, H, G, S, L

$$\sum_{L,S,G,H,I,D} \phi_J(J,L,S)\,\phi_L(L,G)\,\phi_S(S,I)\,\phi_G(G,I,D)\,\phi_H(H,G,J)\,\phi_I(I)\,\tau_1(D)$$

Variable elimination example (2)



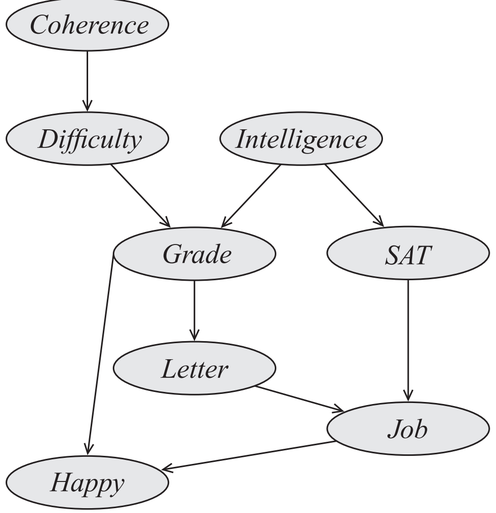
Goal: $P(J)$

Eliminate: D, I, H, G, S, L

$$\sum_{L,S,G,H,I,D} \phi_J(J,L,S)\,\phi_L(L,G)\,\phi_S(S,I)\,\phi_G(G,I,D)\,\phi_H(H,G,J)\,\phi_I(I)\,\tau_1(D)$$

$$\sum_{L,S,G,H,I} \phi_J(J,L,S)\,\phi_L(L,G)\,\phi_S(S,I)\,\phi_H(H,G,J)\,\phi_I(I) \sum_{D} \phi_G(G,I,D)\,\tau_1(D)$$

Variable elimination example (2)



Goal: $P(J)$

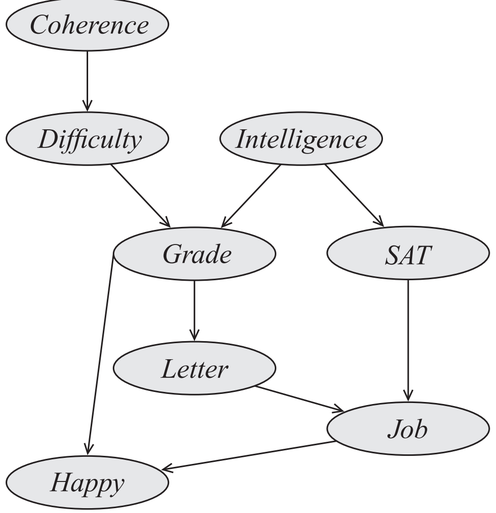
Eliminate: D, I, H, G, S, L

$$\sum_{L,S,G,H,I,D} \phi_J(J,L,S)\,\phi_L(L,G)\,\phi_S(S,I)\,\phi_G(G,I,D)\,\phi_H(H,G,J)\,\phi_I(I)\,\tau_1(D)$$

$$\sum_{L,S,G,H,I} \phi_J(J,L,S)\,\phi_L(L,G)\,\phi_S(S,I)\,\phi_H(H,G,J)\,\phi_I(I) \sum_{D} \phi_G(G,I,D)\,\tau_1(D)$$

Compute $\tau_2(G,I) = \sum_D \phi_G(G,I,D)\,\tau_1(D)$

Variable elimination example (2)



Goal: $P(J)$

Eliminate: D, I, H, G, S, L

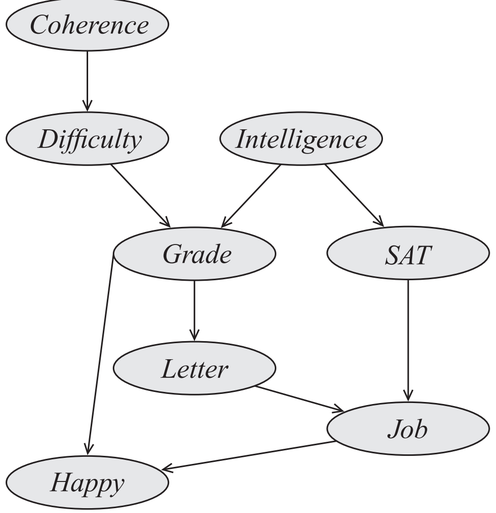
$$\sum_{L,S,G,H,I,D} \phi_J(J,L,S)\,\phi_L(L,G)\,\phi_S(S,I)\,\phi_G(G,I,D)\,\phi_H(H,G,J)\,\phi_I(I)\,\tau_1(D)$$

$$\sum_{L,S,G,H,I} \phi_J(J,L,S)\,\phi_L(L,G)\,\phi_S(S,I)\,\phi_H(H,G,J)\,\phi_I(I) \sum_{D} \phi_G(G,I,D)\,\tau_1(D)$$

Compute $\tau_2(G,I) = \sum_D \phi_G(G,I,D)\,\tau_1(D)$

$$\sum_{L,S,G,H,I} \phi_J(J,L,S)\,\phi_L(L,G)\,\phi_S(S,I)\,\phi_H(H,G,J)\,\phi_I(I)\,\tau_2(G,I)$$

Variable elimination example (3)

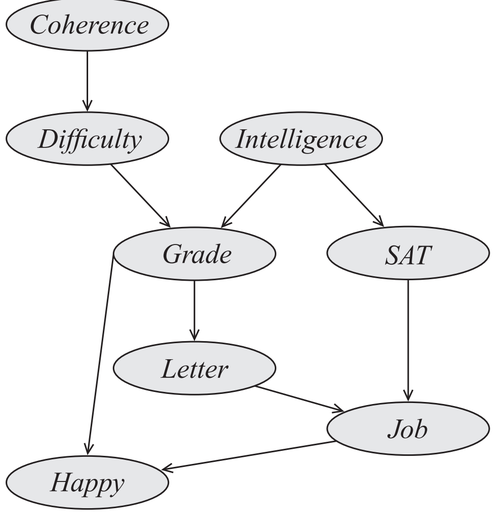


Goal: $P(J)$

Eliminate: I, H, G, S, L

$$\sum_{L,S,G,H,I} \phi_J(J,L,S)\,\phi_L(L,G)\,\phi_S(S,I)\,\phi_H(H,G,J)\,\phi_I(I)\,\tau_2(G,I)$$

Variable elimination example (3)



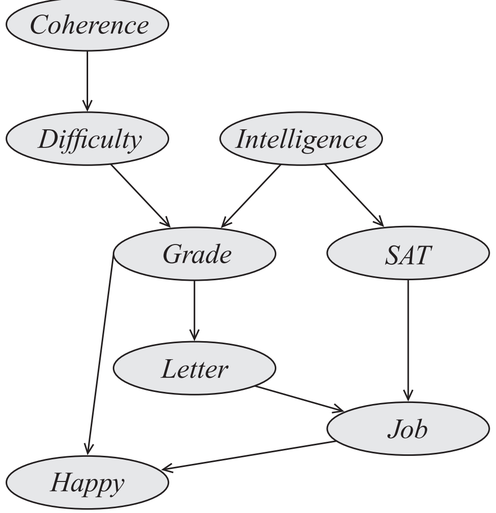
Goal: $P(J)$

Eliminate: I, H, G, S, L

$$\sum_{L,S,G,H,I} \phi_J(J,L,S)\,\phi_L(L,G)\,\phi_S(S,I)\,\phi_H(H,G,J)\,\phi_I(I)\,\tau_2(G,I)$$

$$\sum_{L,S,G,H} \phi_J(J,L,S)\,\phi_L(L,G)\,\phi_H(H,G,J) \sum_{I} \phi_S(S,I)\,\phi_I(I)\,\tau_2(G,I)$$

Variable elimination example (3)



Goal: $P(J)$

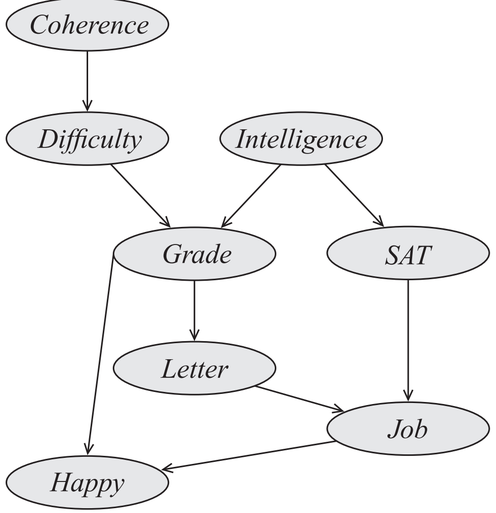
Eliminate: I, H, G, S, L

$$\sum_{L,S,G,H,I} \phi_J(J,L,S)\,\phi_L(L,G)\,\phi_S(S,I)\,\phi_H(H,G,J)\,\phi_I(I)\,\tau_2(G,I)$$

$$\sum_{L,S,G,H} \phi_J(J,L,S)\,\phi_L(L,G)\,\phi_H(H,G,J) \sum_{I} \phi_S(S,I)\,\phi_I(I)\,\tau_2(G,I)$$

Compute $\tau_3(S,G) = \sum_I \phi_S(S,I)\,\phi_I(I)\,\tau_2(G,I)$

Variable elimination example (3)



Goal: $P(J)$

Eliminate: I, H, G, S, L

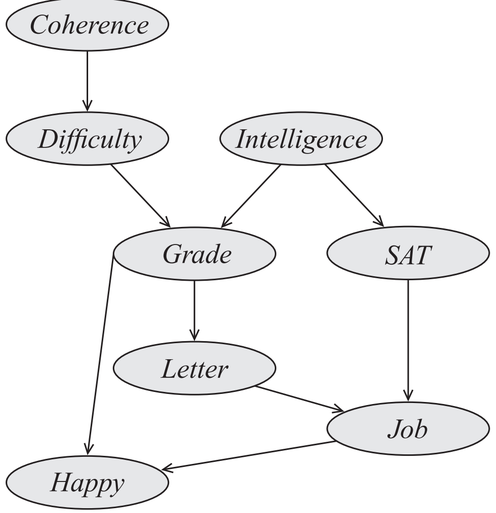
$$\sum_{L,S,G,H,I} \phi_J(J,L,S)\,\phi_L(L,G)\,\phi_S(S,I)\,\phi_H(H,G,J)\,\phi_I(I)\,\tau_2(G,I)$$

$$\sum_{L,S,G,H} \phi_J(J,L,S)\,\phi_L(L,G)\,\phi_H(H,G,J) \sum_{I} \phi_S(S,I)\,\phi_I(I)\,\tau_2(G,I)$$

Compute $\tau_3(S,G) = \sum_I \phi_S(S,I)\,\phi_I(I)\,\tau_2(G,I)$

$$\sum_{L,S,G,H} \phi_J(J,L,S)\,\phi_L(L,G)\,\phi_H(H,G,J)\,\tau_3(S,G)$$

Variable elimination example (4)

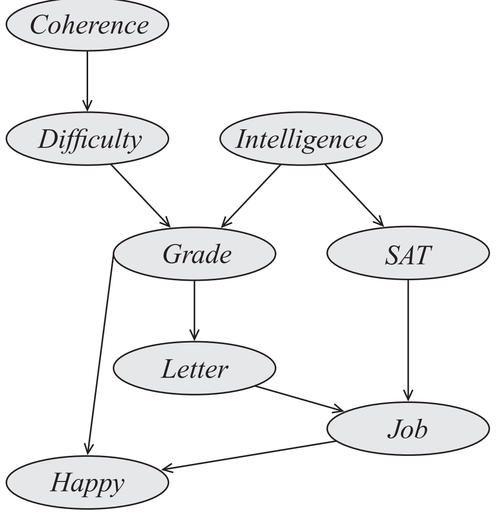


Goal: $P(J)$

Eliminate: H, G, S, L

$$\sum_{L,S,G,H} \phi_J(J, L, S)\, \phi_L(L, G)\, \phi_H(H, G, J)\, \tau_3(S, G)$$

Variable elimination example (4)



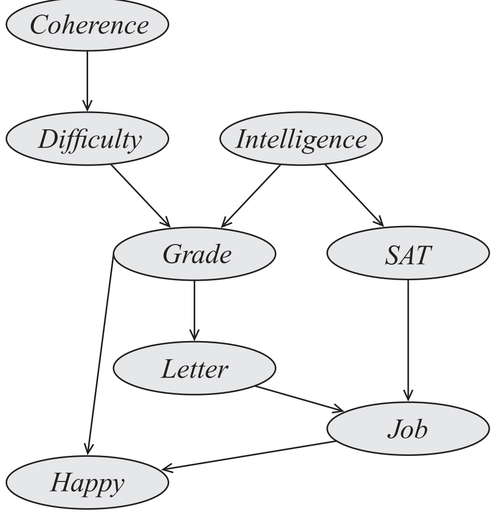
Goal: $P(J)$

Eliminate: H, G, S, L

$$\sum_{L,S,G,H} \phi_J(J, L, S)\, \phi_L(L, G)\, \phi_H(H, G, J)\, \tau_3(S, G)$$

$$\sum_{L,S,G} \phi_J(J, L, S)\, \phi_L(L, G)\, \tau_3(S, G) \sum_{H} \phi_H(H, G, J)$$

Variable elimination example (4)



Goal: $P(J)$

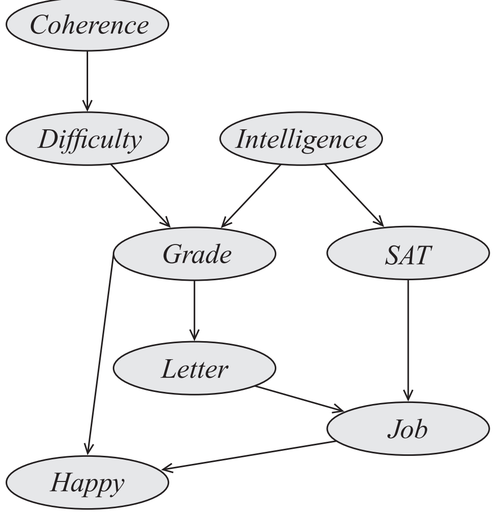
Eliminate: H, G, S, L

$$\sum_{L,S,G,H} \phi_J(J,L,S)\,\phi_L(L,G)\,\phi_H(H,G,J)\,\tau_3(S,G)$$

$$\sum_{L,S,G} \phi_J(J,L,S)\,\phi_L(L,G)\,\tau_3(S,G) \sum_{H} \phi_H(H,G,J)$$

Compute $\sum_H P(H \mid G,J) = 1$ (if our algorithm is clever enough..), simple way: $\tau_4(G,J) = \sum_H \phi_H(H,G,J)$

Variable elimination example (4)



Goal: $P(J)$

Eliminate: H, G, S, L

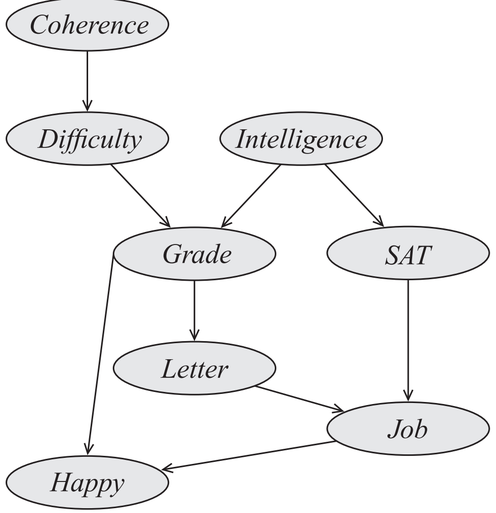
$$\sum_{L,S,G,H} \phi_J(J,L,S)\,\phi_L(L,G)\,\phi_H(H,G,J)\,\tau_3(S,G)$$

$$\sum_{L,S,G} \phi_J(J,L,S)\,\phi_L(L,G)\,\tau_3(S,G) \sum_{H} \phi_H(H,G,J)$$

Compute $\sum_H P(H \mid G,J) = 1$ (if our algorithm is clever enough..), simple way: $\tau_4(G,J) = \sum_H \phi_H(H,G,J)$

$$\sum_{L,S,G} \phi_J(J,L,S)\,\phi_L(L,G)\,\tau_3(S,G)\,\tau_4(G,J)$$

Variable elimination example (5)

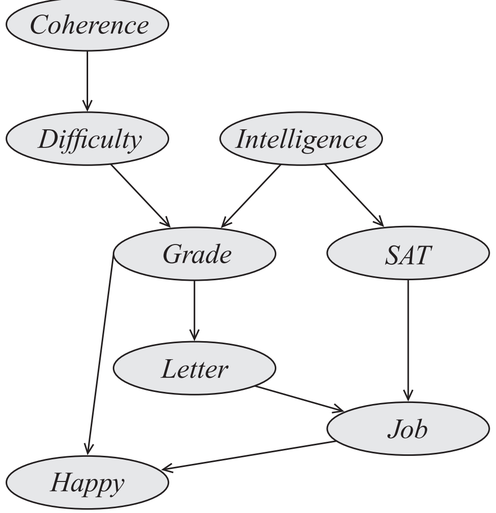


Goal: $P(J)$

Eliminate: G, S, L

$$\sum_{L,S,G} \phi_J(J, L, S)\, \phi_L(L, G)\, \tau_3(S, G)\, \tau_4(G, J)$$

Variable elimination example (5)



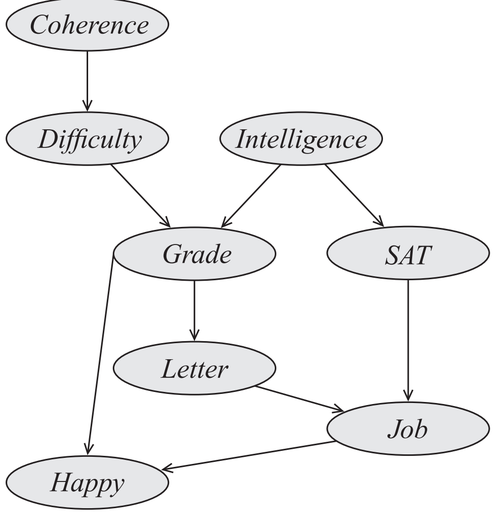
Goal: $P(J)$

Eliminate: G, S, L

$$\sum_{L,S,G} \phi_J(J, L, S)\, \phi_L(L, G)\, \tau_3(S, G)\, \tau_4(G, J)$$

$$\sum_{L,S} \phi_J(J, L, S) \sum_{G} \phi_L(L, G)\, \tau_3(S, G)\, \tau_4(G, J)$$

Variable elimination example (5)



Goal: $P(J)$

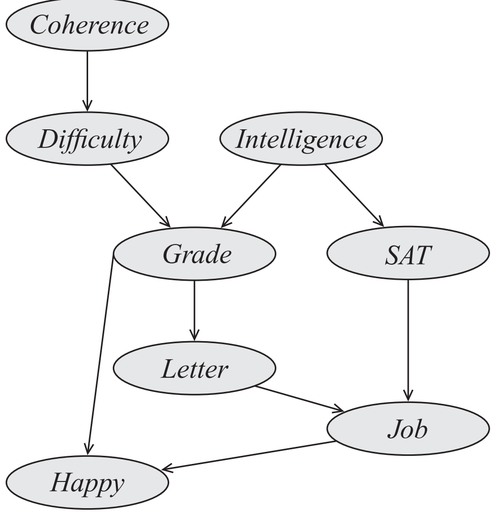
Eliminate: G, S, L

$$\sum_{L,S,G} \phi_J(J, L, S)\, \phi_L(L, G)\, \tau_3(S, G)\, \tau_4(G, J)$$

$$\sum_{L,S} \phi_J(J, L, S) \sum_{G} \phi_L(L, G)\, \tau_3(S, G)\, \tau_4(G, J)$$

Compute $\tau_5(L, J, S) = \sum_{G} \phi_L(L, G)\, \tau_3(S, G)\, \tau_4(G, J)$

Variable elimination example (5)



Goal: $P(J)$

Eliminate: G, S, L

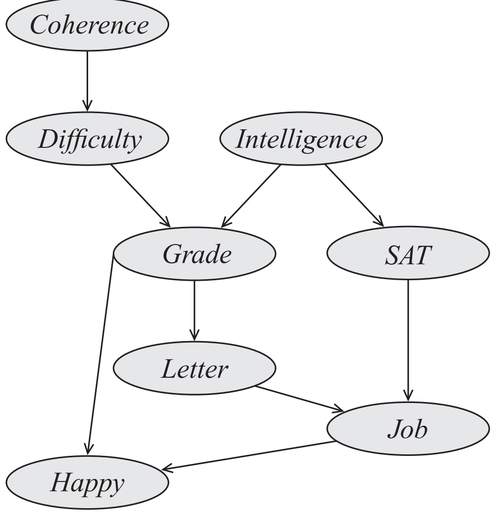
$$\sum_{L,S,G} \phi_J(J, L, S)\, \phi_L(L, G)\, \tau_3(S, G)\, \tau_4(G, J)$$

$$\sum_{L,S} \phi_J(J, L, S) \sum_{G} \phi_L(L, G)\, \tau_3(S, G)\, \tau_4(G, J)$$

Compute $\tau_5(L, J, S) = \sum_{G} \phi_L(L, G)\, \tau_3(S, G)\, \tau_4(G, J)$

$$\sum_{L,S} \phi_J(J, L, S)\, \tau_5(L, J, S)$$

Variable elimination example (6)

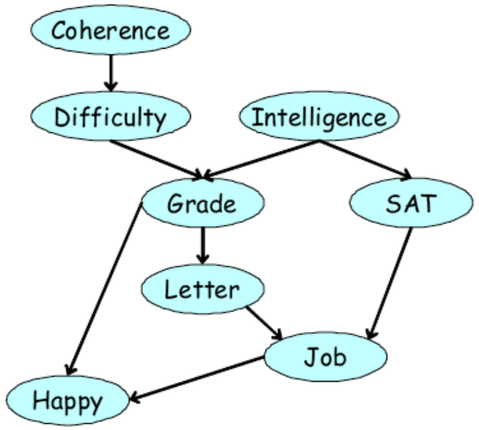


Goal: $P(J)$

Eliminate: S, L

$$\sum_{L,S} \phi_J(J, L, S)\, \tau_5(L, J, S)$$

We can pick any order we want, but some orderings introduce factors with much larger scope



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	$\tau_7(J)$

Alternative ordering...

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

Finding elimination orderings

- Finding a good elimination ordering is (yet another) hard problem (NP-hard)
- Even if you get the best elimination ordering (for free), inference can still be exponential in terms of compute time.
- As for finding the best ordering, there are some good heuristics

Choosing an ordering

Set of possible heuristics:

- **Min-neighbors**: The cost of a vertex is the number of neighbors it has in the current graph.
- **Min-weight**: the cost of a vertex is the product of weights (domain cardinality) of its neighbors.
- **Min-fill**: the cost of a vertex is the number of edges that need to be added to the graph due to its elimination.
- **Weighted-Min-Fill**: the cost of a vertex is the sum of weights of the edges that need to be added to the graph due to its elimination. Weight of an edge is the product of weights of its constituent vertices.

Which one better?

- None of these criteria is better than others.
- Often will try several.

How to introduce evidence

Recall that our original goal was to answer *conditional* probability queries,

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

Apply variable elimination algorithm to the task of computing $P(\mathbf{Y},\mathbf{e})$

Replace each factor $\phi \in \Phi$ that has $\mathbf{E} \cap \mathrm{Scope}[\phi] \neq \emptyset$ with

$$\phi'(\mathbf{x}_{\mathrm{Scope}[\phi]-\mathbf{E}}) = \phi(\mathbf{x}_{\mathrm{Scope}[\phi]-\mathbf{E}}, \mathbf{e}_{\mathbf{E}\cap\mathrm{Scope}[\phi]})$$

Then, eliminate the variables in $\mathcal{X} - \mathbf{Y} - \mathbf{E}$. The returned factor $\phi^*(\mathbf{Y})$ is $p(\mathbf{Y},\mathbf{e})$

To obtain the conditional $p(\mathbf{Y} \mid \mathbf{e})$, normalize the resulting product of factors – the normalization constant is $p(\mathbf{e})$

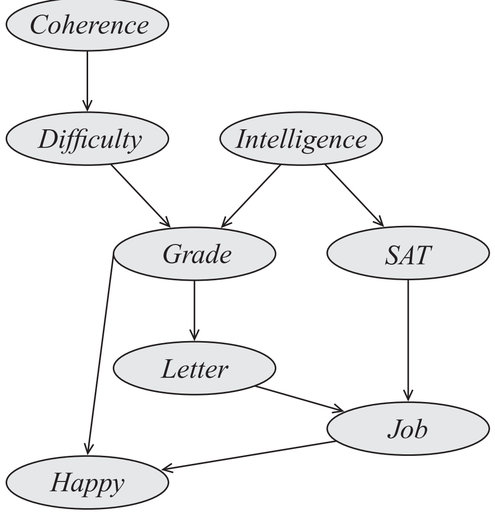
Sum-product VE for conditional distributions

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 w/ evidence



Goal: $P(J, I = i, H = h)$

Eliminate: C, D, G, S, L

$$\sum_{L,S,G,D,C} \phi_J(J, L, S)\, \phi_L(L, G)\, \phi'_S(S)\, \phi'_G(G, D)\, \phi'_H(G, J)\, \phi'_I()\, \phi_D(C, D)\, \phi_C(C)$$

Elimination as before … but just not with H, I

So how do we get $P(J \mid I = i, H = h)$?

Renormalise, that is divide $\dfrac{P(J, I = i, H = h)}{P(I = i, H = h)}$.

VE Algorithm summary

- Reduce all factors by evidence
  - Get a set of factors Φ
- For each non-query variable Z
  - Eliminate-Var Z from Φ
- Multiply all remaining factors
- Renormalise to get distribution

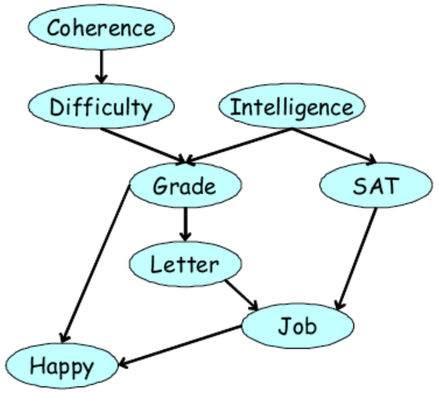
Removes Φ'

Adds τ

Summary

- Simple algorithm
- Works for both BN and MNs
- Factor product and summation steps can be done in any order subject to
  - When Z is eliminated, all factors involving Z have been multiplied in

Running time 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 and multiply all factors involving X_i, resulting in a single factor ψ_i

Let N_i be the number of variables in the factor ψ_i, and let $N_{max} = \max_i N_i$

The running time of VE is then $O(mk^{N_{max}})$, where $k = |\mathrm{Val}(X)|$. Why?

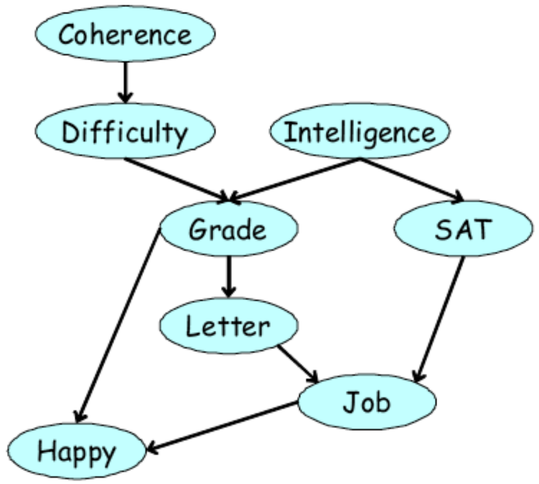
The primary concern is that N_{max} can potentially be as large as n

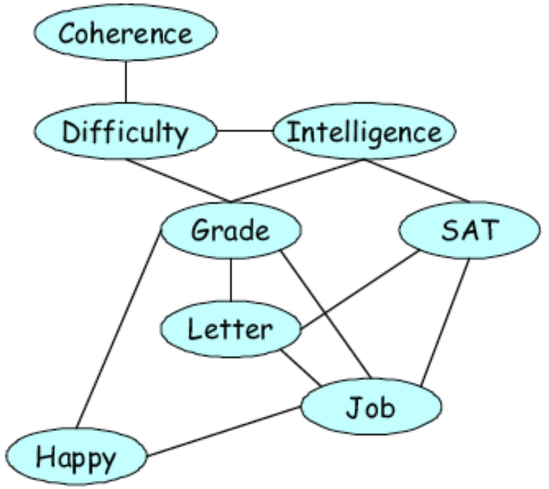
Running time in graph-theoretic concepts

Let's try to analyze the complexity in terms of the graph structure

G_Φ is the undirected graph with one node per variable, where there is an edge (X_i, X_j) if these appear together in the scope of some factor ϕ

Ignoring evidence, this is either the original MRF (for sum-product VE on MRFs) or the moralized Bayesian network:





Elimination as graph transformation

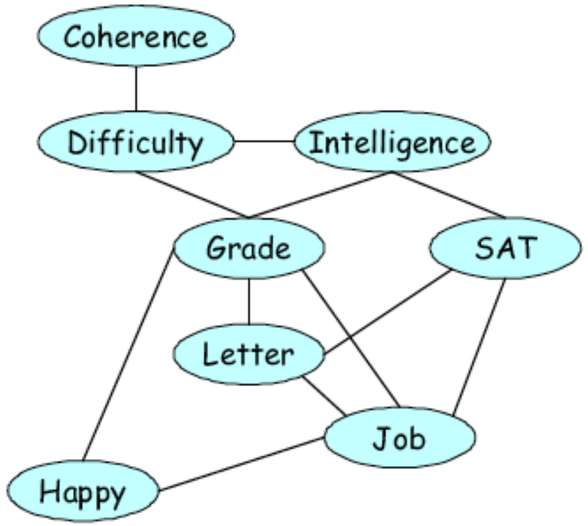
When a variable X is eliminated,

- We create a single factor ψ that contains X and all of the variables $\mathbf{Y}$ with which it appears in factors
- We eliminate X from ψ, replacing it with a new factor τ that contains all of the variables $\mathbf{Y}$, but not X. Let's call the new set of factors Φ_X

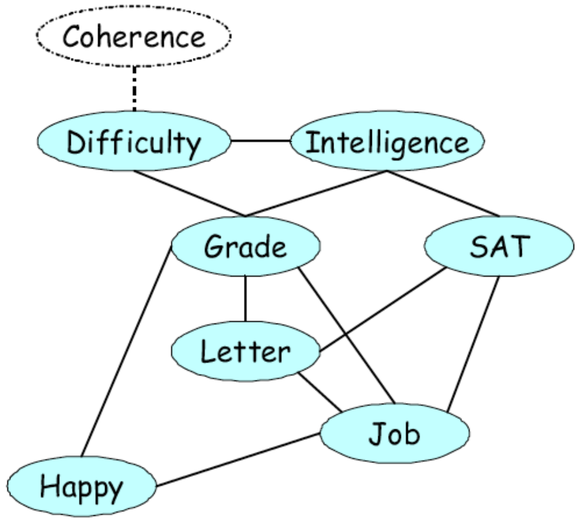
How does this modify the graph, going from G_Φ to G_{Φ_X}?

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

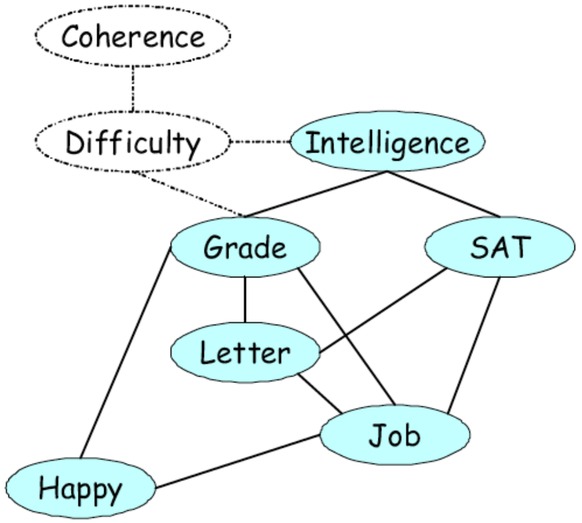
Example



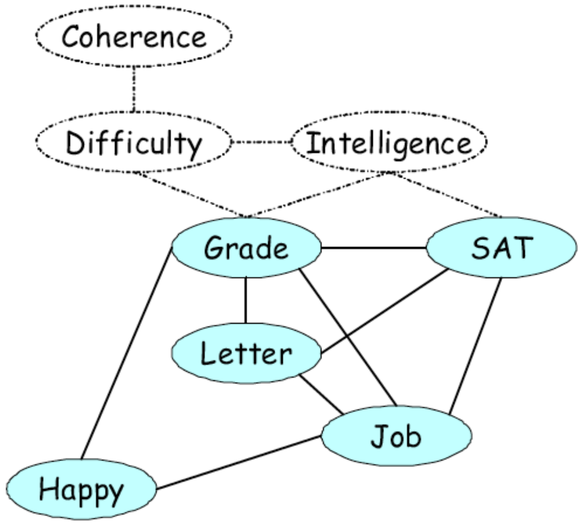
(Graph)



(Elim. C)



(Elim. D)



(Elim. I)

Induced graph

We can summarize the computation cost using a single graph that is the union of all the graphs resulting from each step of the elimination

We call this the **induced graph** $\mathcal{I}_{\Phi,\prec}$, where $\prec$ is the elimination ordering