NUS National University of Singapore | School of Computing

CS5340
Uncertainty Modeling in AI

Lecture 5:
Variable Elimination and Belief Propagation

or "An Introduction to How to do Inference"

Asst. Prof. Harold Soh

AY 2022/23

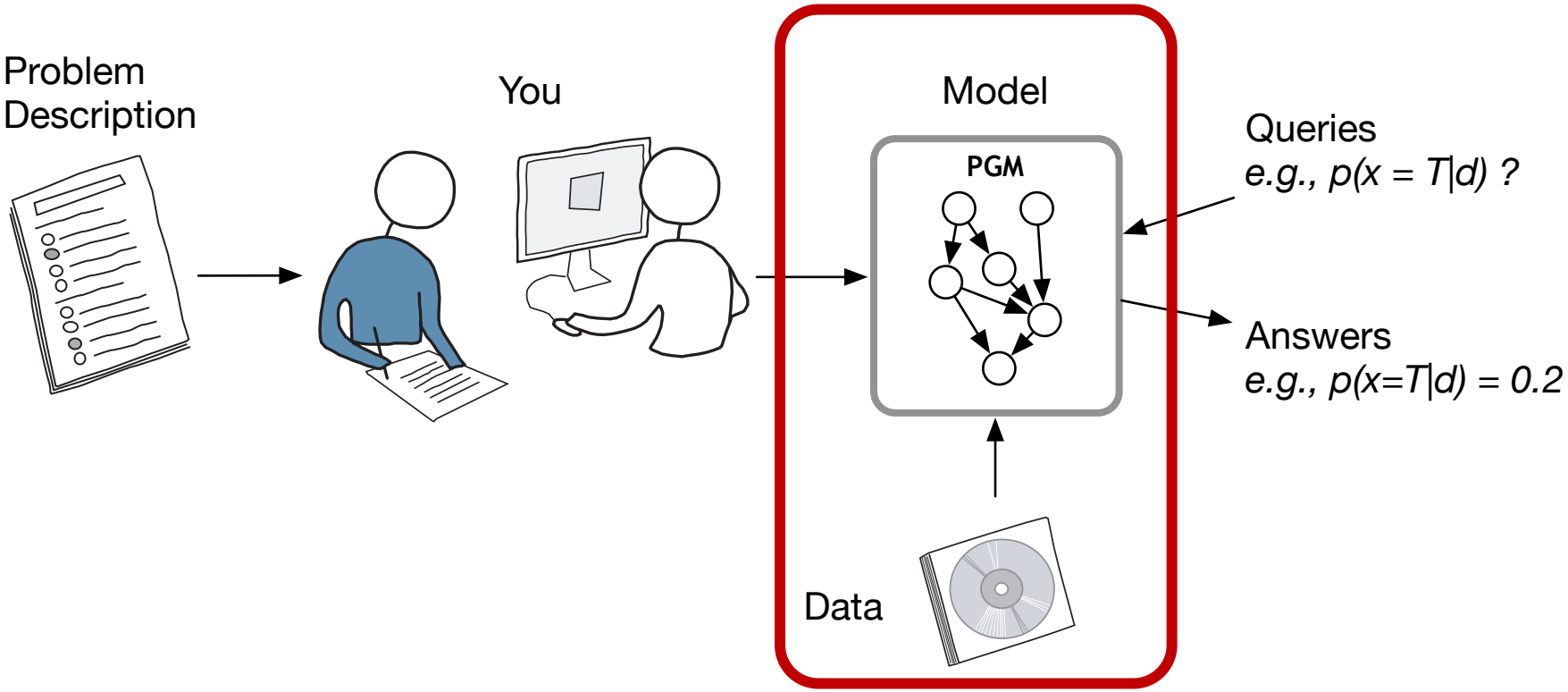
Semester 2

Quick Recap

Directed & Undirected Graphical Models

CS5340 in a nutshell

CS5340 is about how to "**represent**" and "**reason**" with uncertainty in a computer.



Bayes Nets (BN) and I-maps

- **Definition** (*Bayesian Network*) A Bayesian network is a tuple $B = (G, P)$ where P factorizes according to G and where P is specified as a set of conditional probability distributions (CPDs) associated with G's nodes.

$$p(x_1, \ldots, x_N) = \prod_{i=1}^{N} p(x_i | x_{\pi_i})$$

Undirected Graphical Models:
In a nutshell

- An alternative to DGMs is to use an Undirected Graphical model (UGM), also called a Markov Random Field (MRF) or Markov network.

- Formally, an UGM is a tuple (G,P) with graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where:
  - $\mathcal{V}$ is a set of nodes that are in one-to-one correspondence with a set of random variables.
  - $\mathcal{E}$ is a set of undirected edges.

Undirected Graphical Models:
In a nutshell

- Parameterization is achieved via factors.
- A factor $\varphi(\mathcal{C})$ is a function that maps a set of random variables $\mathcal{C} = \{X, \dots, Z\}$ to a real number.
  - Restrict: non-negative factors only ($\varphi(\mathcal{C})$ only maps to non-negative numbers)
- The factorization is:

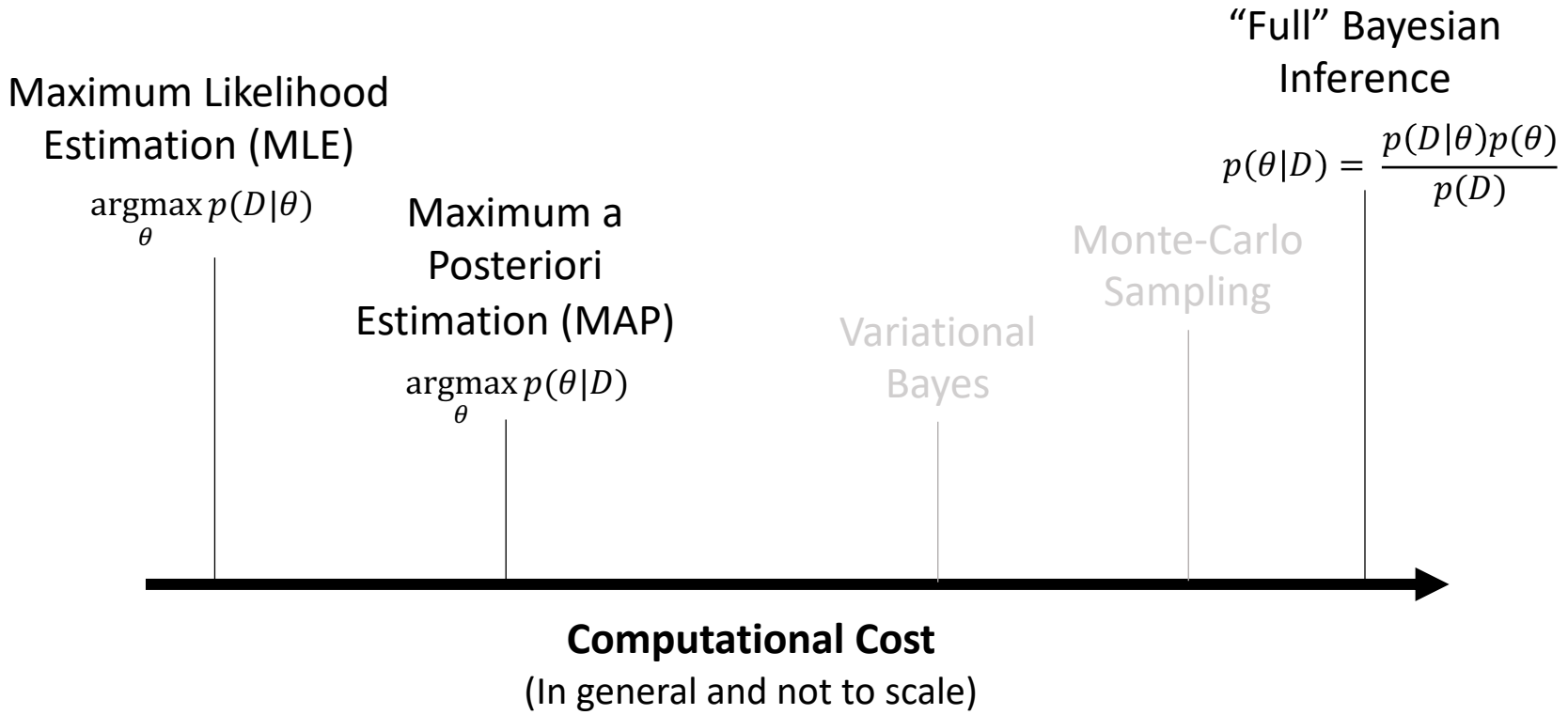
$$p(x_1, \dots, x_N) = \frac{1}{Z} \prod_{j=1}^{M} \varphi_j(\mathcal{C}_j)$$

Compare to DGM:

$$p(x_1, \dots, x_N) = \prod_{i=1}^{N} p(x_i | x_{\pi_i})$$

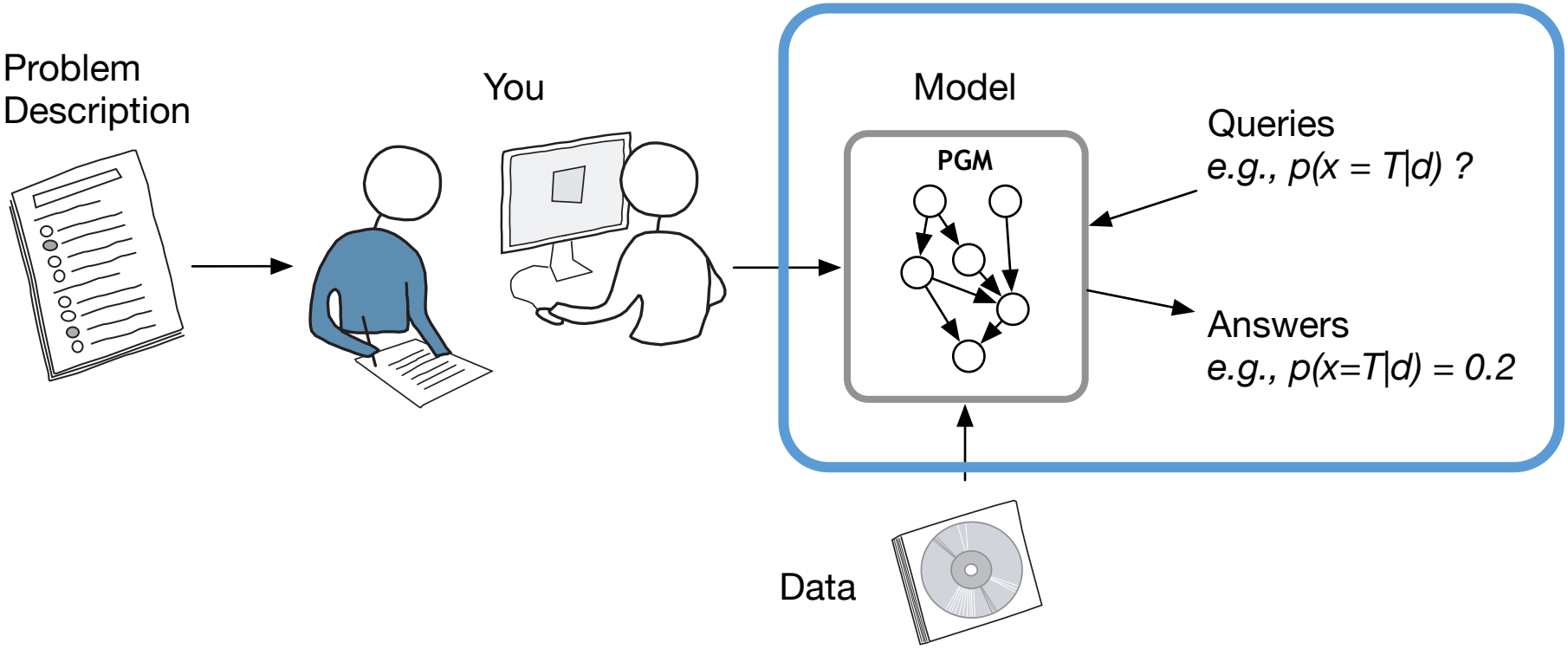
Learning Parameters

- Common approaches to learn the unknown parameters θ from a set of given data $\mathcal{D} = \{x[1], \ldots, x[N]\}$:



CS5340 in a nutshell

CS5340 is about how to "**represent**" and "**reason**" with uncertainty in a computer.



Course Schedule

Week	Date	Lecture Topic	Tutorial Topic
1	12 Jan	Introduction to Uncertainty Modeling + Probability Basics	Introduction
2	19 Jan	Simple Probabilistic Models	Probability Basics
3	26 Jan	Bayesian networks (Directed graphical models)	More Basic Probability
4	2 Feb	Markov random Fields (Undirected graphical models)	DGM modelling and d-separation
5	9 Feb	Variable elimination and belief propagation	MRF + Sum/Max Product
6	16 Feb	Factor graph and the junction tree algorithm	**Quiz 1**
-	-	RECESS WEEK	
7	2 Mar	Mixture Models and Expectation Maximization (EM)	Linear Gaussian Models
8	9 Mar	Hidden Markov Models (HMM)	Probabilistic PCA
9	**16** Mar	Monte-Carlo Inference (Sampling)	Linear Gaussian Dynamical System
10	23 Mar	Variational Inference	MCMC + Sequential VAE
11	30 Mar	Inference and Decision-Making (Special Topic)	**Quiz 2**
12	6 Apr	Gaussian Processes (Special Topic)	Wellness Day
13	13 Apr	**Project Presentations**	Closing

Acknowledgements

- A lot of slides and content of this lecture are adapted from:

1. Michael I. Jordan "An introduction to probabilistic graphical models", 2002
   Chapters 3 and 4.1
   http://people.eecs.berkeley.edu/~jordan/prelims/chapter3.pdf
   http://people.eecs.berkeley.edu/~jordan/prelims/chapter4.pdf (Section 4.1)

2. Kevin Murphy, "Machine learning: a probabilistic approach"
   Chapter 20.1, 20.2, 20.3

3. Daphne Koller and Nir Friedman, "Probabilistic graphical models"
   Chapter 9

4. David Barber, "Bayesian reasoning and machine learning"
   Chapter 5

5. Christopher Bishop "Machine learning and pattern recognition"
   Chapter 8.4

6. Slides from Dr. Lee Gim Hee

Learning Outcomes

- Students should be able to:

1. Use the Variable Elimination algorithm to compute the conditional probability of a single random variable X_f, i.e. $p(x_f | x_E)$.

2. Explain the computational complexity of variable elimination using the reconstituted graph.

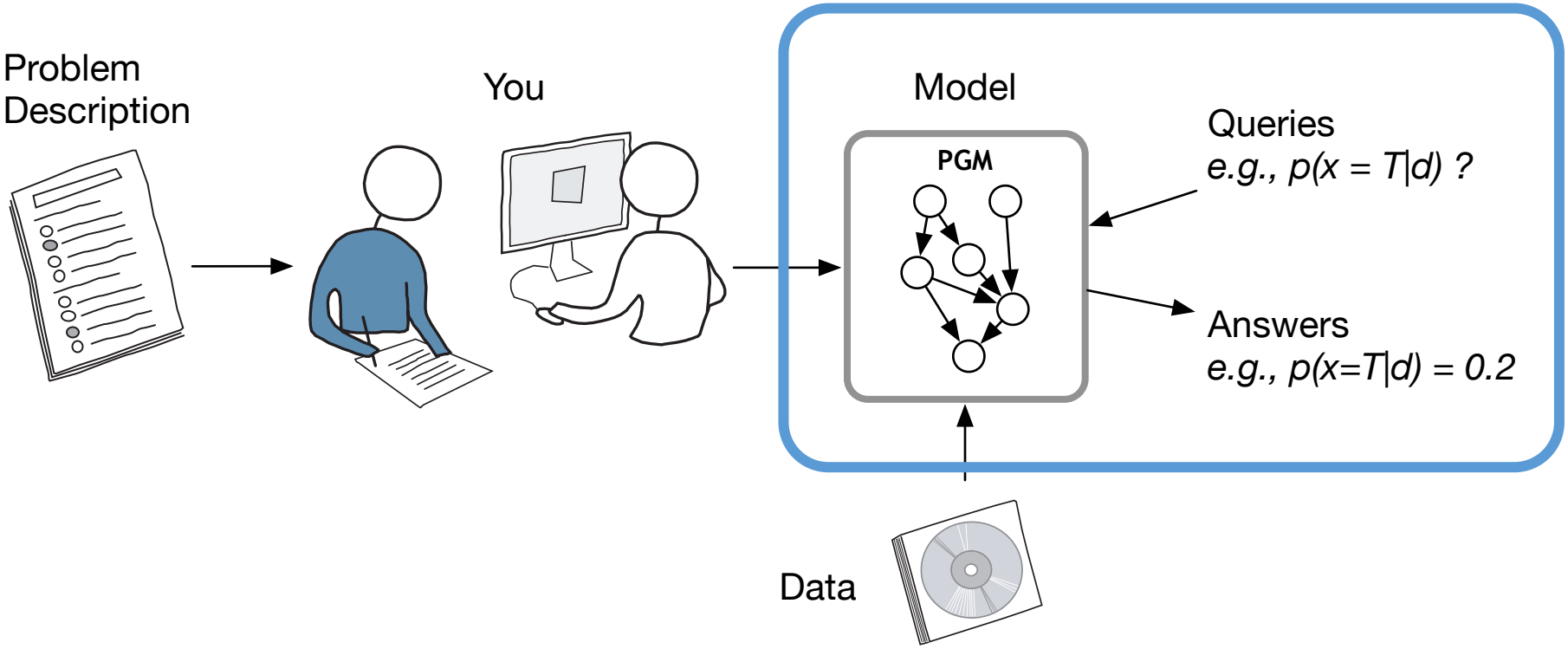
3. Use the sum-product algorithm to compute all single-node marginals for "tree-like" graphical models.

4. Use the max-product algorithm to find the maximal probability and its configurations.

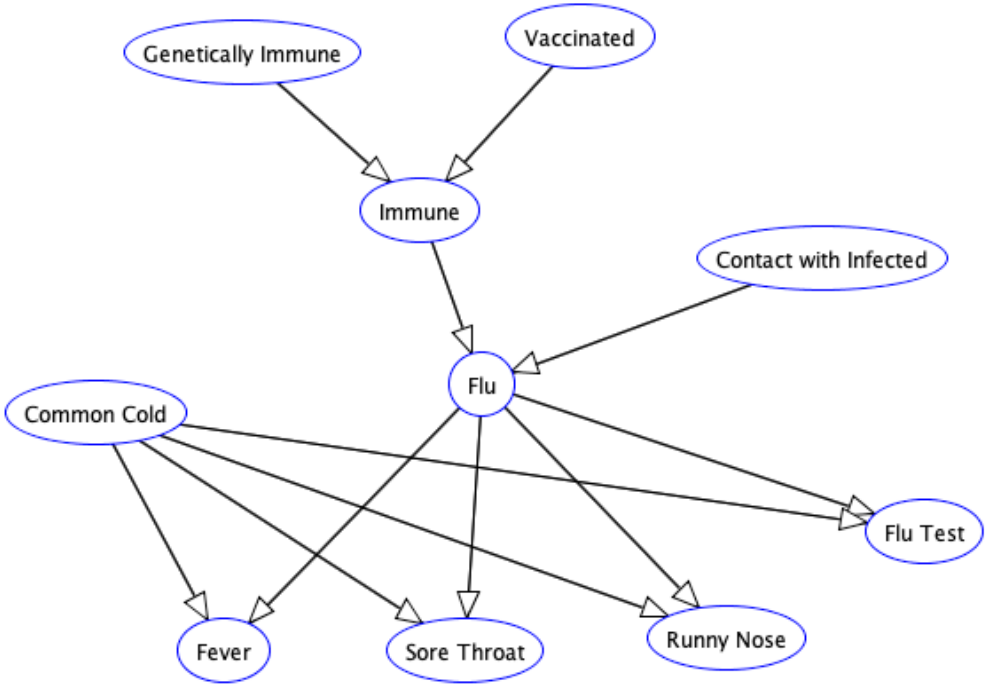
The Probabilistic Inference Problem

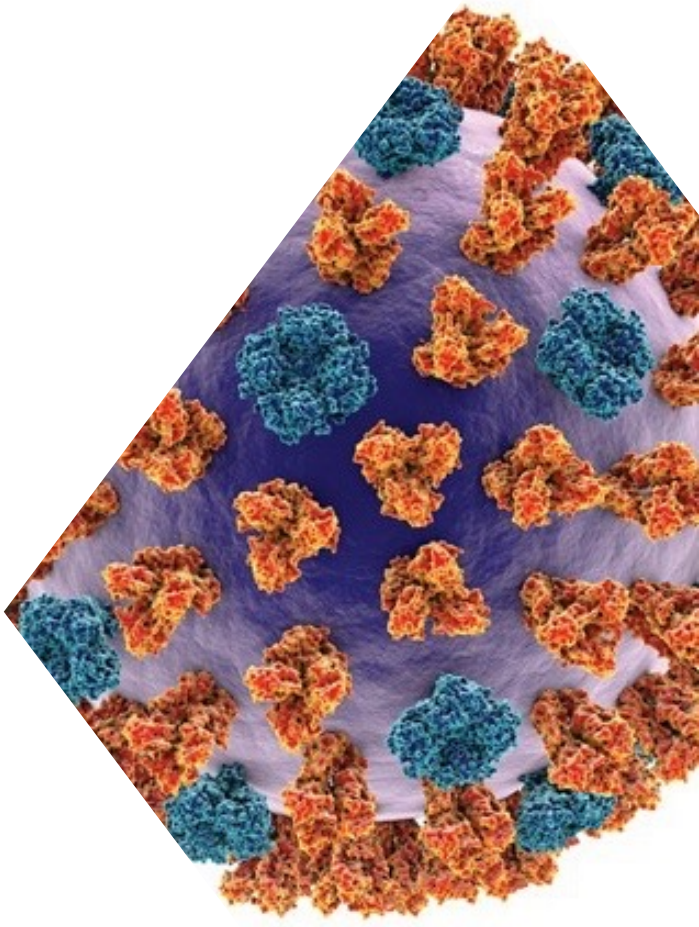
CS5340 in a nutshell

CS5340 is about how to "**represent**" and "**reason**" with uncertainty in a computer.



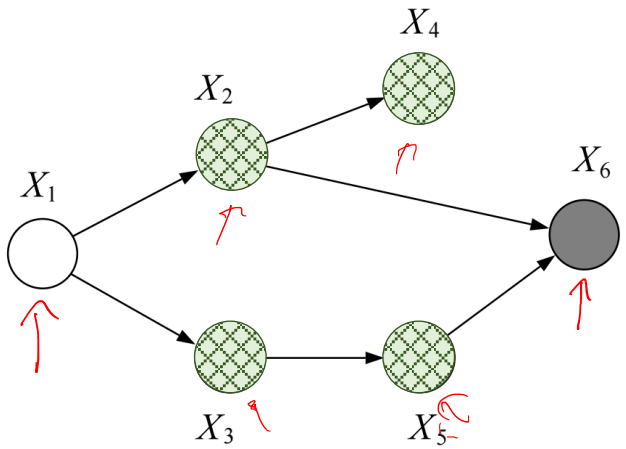
Back to our Flu Example…





Probabilistic Inference Problem

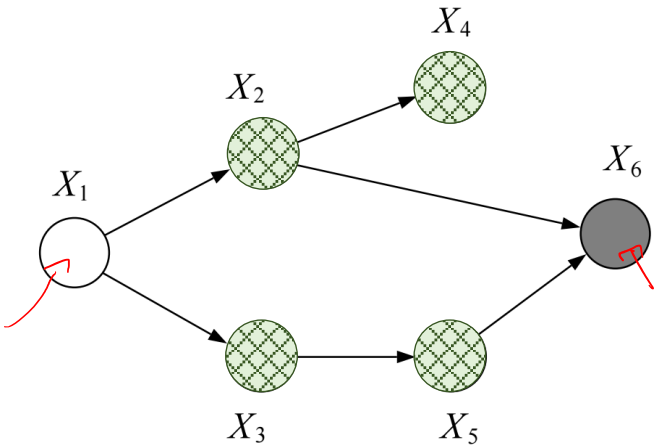
- Let E and F be disjoint subsets of the node indices of a graphical model.
- X_E and X_F are disjoint subsets of the random variables in the domain.
- Our goal is to calculate $p(X_F|X_E)$ for arbitrary subsets E and F.
  - This is the general probabilistic inference problem for graphical models (directed or undirected).



$X_F = \{X_1\}$ and $X_E = \{X_6\}$

Other "nuisance" variables $\{X_2, X_3, X_4, X_5\}$

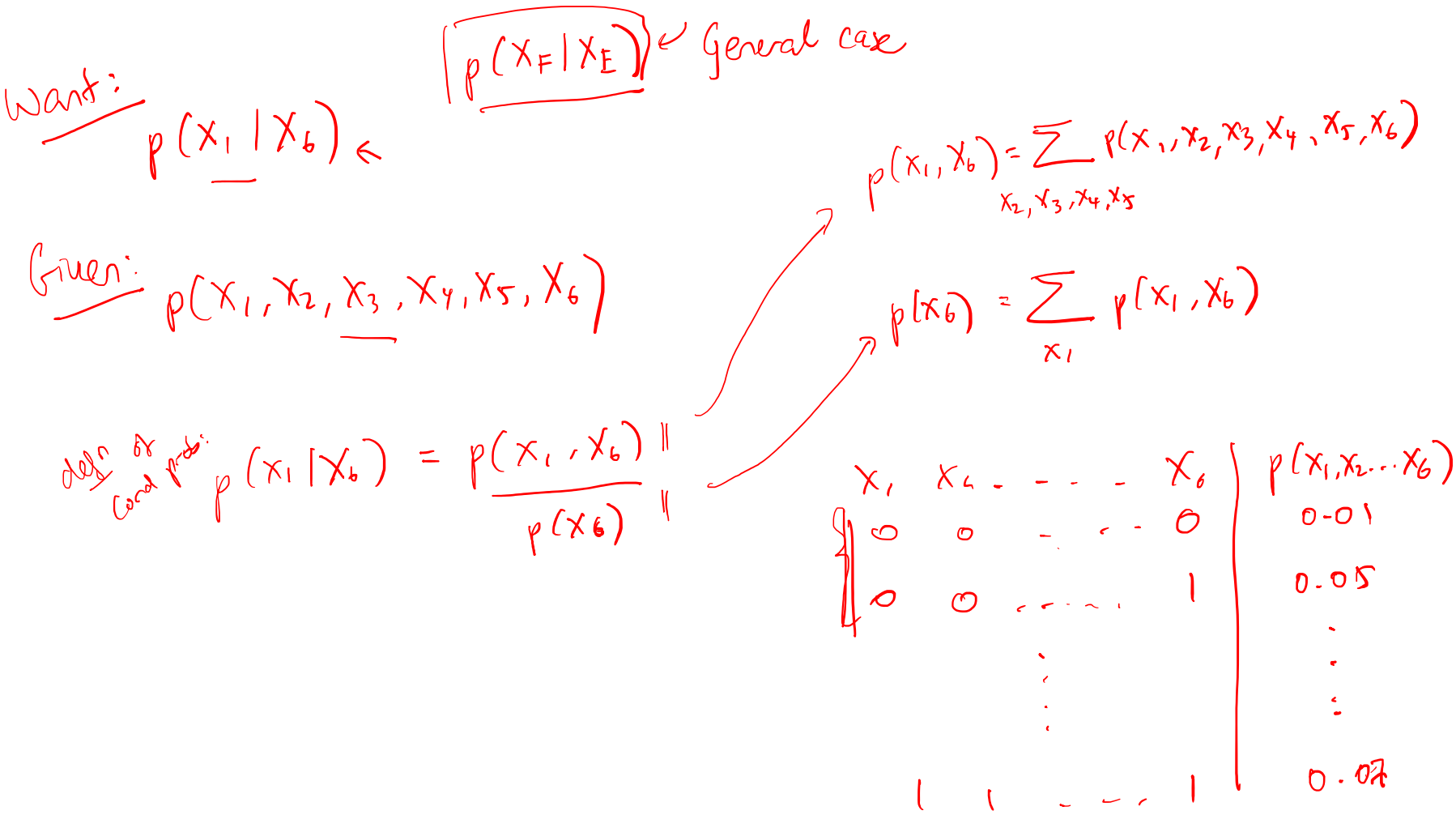
Probabilistic Inference Problem



- Dark shading indicates the "evidence nodes" X_E on which we condition.
- Unshaded node is the "query node" X_F for which we wish to compute conditional probabilities.
- Patterned nodes $X_R = X_V \setminus (X_E, X_F)$ are the nodes that must be marginalized out of the joint probability.

Image source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

The Straight-forward Approach



Recall from Lecture 1: Sum and Product Rules

- Sum rule:

$$p(x) = \int p(x, y)\, dy$$

$$p(x) = \sum_{y} p(x, y)$$

- Product/Chain rule:

$$p(x, y) = p(x|y)p(y)$$

Probabilistic Inference Problem

Conditional probability:

$$p(x_F \mid x_E) = \frac{p(x_E, x_F)}{p(x_E)}$$

Marginals from joint probability:

$$p(x_E, x_F) = \sum_{x_R} p(x_E, x_F, x_R),$$

$$p(x_E) = \sum_{x_F} p(x_E, x_F)$$

Probabilistic Inference Problem

Marginals from joint probability:

$$p(x_E, x_F) = \sum_{x_R} p(x_E, x_F, x_R), \qquad p(x_E) = \sum_{x_F} p(x_E, x_F)$$

- Σ_{x_R} expands into a sequence of summations, one for each of the random variables indexed by R.
- A naïve summation over the joint distribution of n variables that takes k states will incur a computational complexity of $O(k^n)$!

NUS National University of Singapore | School of Computing

Variable Elimination

"how to marginalize more cleverly"

Variable Elimination Algorithm

- We will first look at how to calculate the conditional probability of a single node X_F given an arbitrary set of nodes X_E.
- Variable elimination algorithm: an algorithm based on marginalization and conditional independence of the graphical model.

Motivation & Intuition with a Simple Example

x_1	x_2	x_3	x_4	$p(x_1, x_2, x_3, x_4)$
0	0	0	0	0.08
0	0	0	1	0.00
0	0	1	0	0.03
0	0	1	1	0.01
0	1	0	0	0.04
0	1	0	1	0.08
0	1	1	0	0.15
0	1	1	1	0.10
1	0	0	0	0.11
1	0	0	1	0.09
1	0	1	0	0.01
1	0	1	1	0.04
1	1	0	0	0.04
1	1	0	1	0.01
1	1	1	0	0.06
1	1	1	1	0.15

goal

$X_F = \{X_4\}$

$X_E = \{X_1\}$

$$p(x_4 \mid x_1) = \frac{p(x_4, x_1)}{p(x_1)}$$

$$p(x_4, x_1) = \sum_{x_2} \sum_{x_3} p(x_1, x_2, x_3, x_4)$$

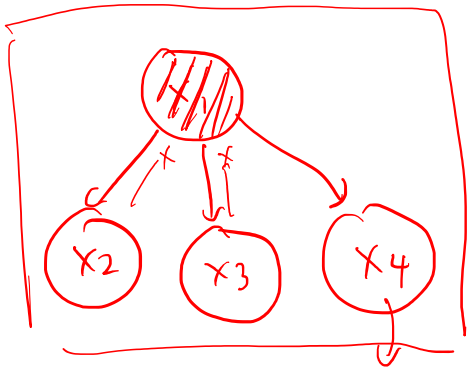
x_1	x_4	$p(x_1, x_4)$
0	0	0.3
0	1	0.19
1	0	0.22
1	1	0.29

x_1	$p(x_1)$
0	0.49
1	0.51

$$p(x_1) = \sum_{x_4} p(x_1, x_4)$$

x_1	x_4	$p(x_4 \mid x_1)$
0	0	0.3/0.49 = 0.61
0	1	0.38
1	0	0.43
1	1	0.56

Motivation & Intuition with a Simple Example



$p(x_4 | \bar{x}_1)$

$p(x_1, x_2, x_3, x_4) = \prod p(x_i | x_{\pi_i})$

$= p(\bar{x}_1)\, p(x_2|\bar{x}_1)\, p(x_3|\bar{x}_1)\, p(x_4|\bar{x}_1)$

x_1	x_4	$p(x_4 \mid x_1)$
0	0	0.2
0	1	0.8
1	0	0.1
1	1	0.9

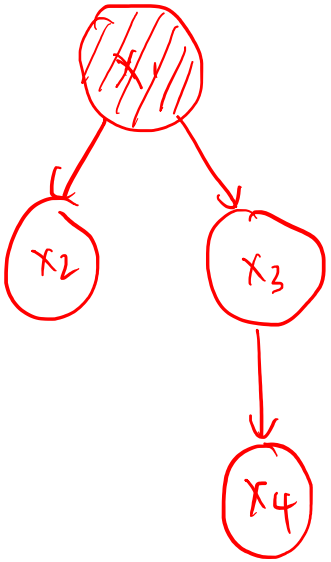
$\bar{x}_1 = 1$

$$p(x_1, x_4) = \sum_{x_2} \sum_{x_3} p(\bar{x}_1)\, p(x_2|\bar{x}_1)\, p(x_3|\bar{x}_1)\, p(x_4|\bar{x}_1)$$

$$= p(\bar{x}_1) \Big[\sum_{x_2} p(x_2|\bar{x}_1)\Big] \Big[\sum_{x_3} p(x_3|\bar{x}_1)\Big] p(x_4|\bar{x}_1)$$

$$= p(\bar{x}_1)\, p(x_4|\bar{x}_1)$$

Motivation & Intuition with a Simple Example



$$p(x_4 \mid \bar{x}_1) = \frac{p(x_4, \bar{x}_1)}{p(\bar{x}_1)}$$

$$p(x_1, x_2, x_3, x_4) = p(\bar{x}_1)\, p(x_2 \mid x_1)\, p(x_3 \mid x_1)\, p(x_4 \mid x_3)$$

$$p(x_4, \bar{x}_1) = \sum_{x_2} \sum_{x_3} p(\bar{x}_1)\, p(x_2 \mid x_1)\, p(x_3 \mid x_1)\, p(x_4 \mid x_3)$$

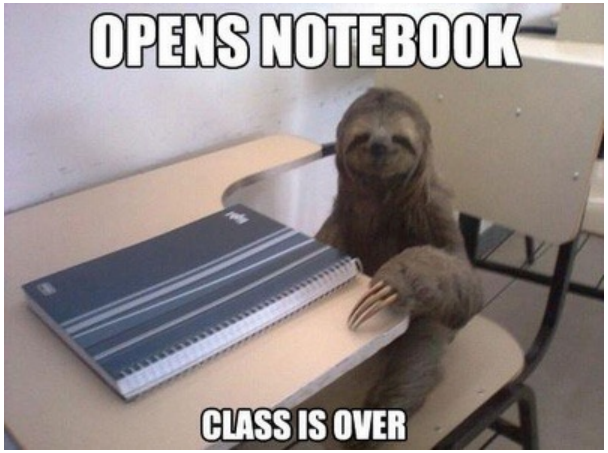
$$= p(\bar{x}_1) \left[\sum_{x_2} p(x_2 \mid x_1) \right] \left[\sum_{x_3} p(x_3 \mid \bar{x}_1)\, p(x_4 \mid x_3) \right]$$

$$= p(\bar{x}_1)\, p(x_4 \mid \bar{x}_1)$$

$$p(x_4 \mid \bar{x}_1) = \frac{p(\bar{x}_1)\, p(x_4 \mid \bar{x}_1)}{p(\bar{x}_1)}$$

Naive Summation

Naïve summation is sloowwwww!



Consider:

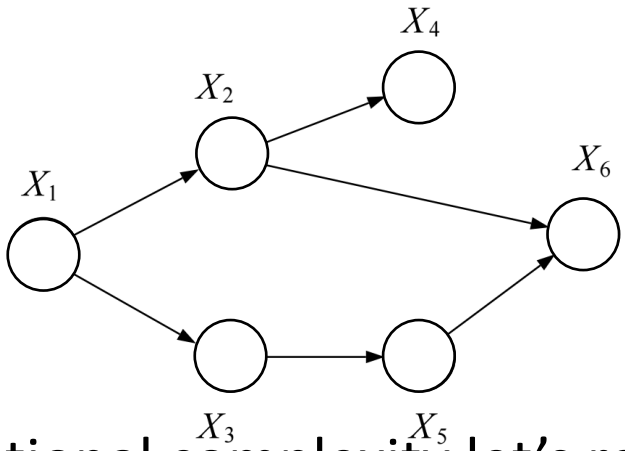
Joint probability table size is k^6

$$p(x_1, x_2, x_3, x_4, x_5) = \sum_{x_6} p(x_1, x_2, x_3, x_4, x_5, x_6)$$

$o(k^6)$ operations to do a single sum $\Rightarrow O(k^n)$ complexity

Image source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Idea: Use Factorization and Distributive Law



- To reduce computational complexity let's represent the joint probability in its factored form and exploit the distributive law:

$$p(x_1, x_2, \ldots, x_5) = \sum_{x_6} p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)p(x_5 \mid x_3)p(x_6 \mid x_2, x_5)$$

$$= p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)p(x_5 \mid x_3) \underbrace{\sum_{x_6} p(x_6 \mid x_2, x_5)}_{\text{Table size of } k^3}$$

$O(k^6)$ to $O(k^3)$ operations to do a single sum!

$\Rightarrow O(k^r)$ instead of $O(k^n)$ complexity, where $r \ll n$

Image source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Variable Elimination

Consider:

$$p(x_1|x_6) = \frac{p(x_1, x_6)}{p(x_6)}$$

where

X_6 : evidence node
X_1 : query node

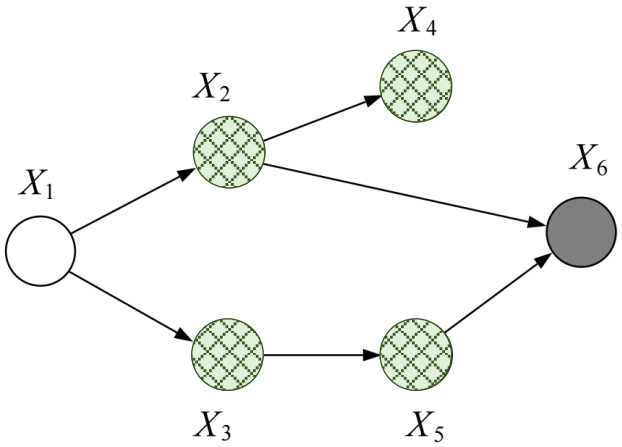


Image source: “An introduction to probabilistic graphical models”, Michael I. Jordan, 2002.

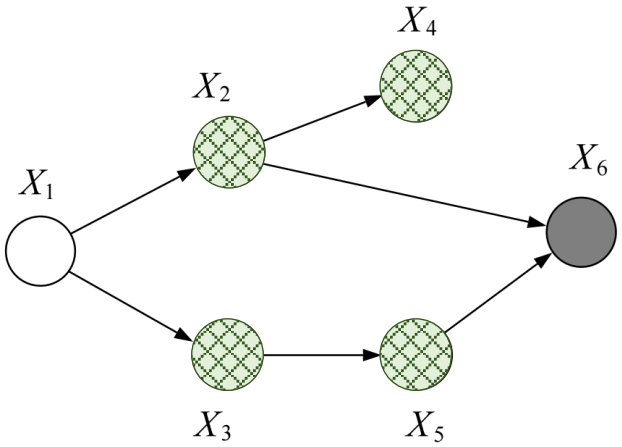
Evidence Node

Consider:

$$p(x_1|x_6) = \frac{p(x_1, x_6)}{p(x_6)}$$

where

X_6 : evidence node
X_1 : query node



- Evidence node X_6 is observed, hence a fixed constant that does not contribute to the computational complexity.
- Let us denote an observed evidence node as $\bar{X}_i$:

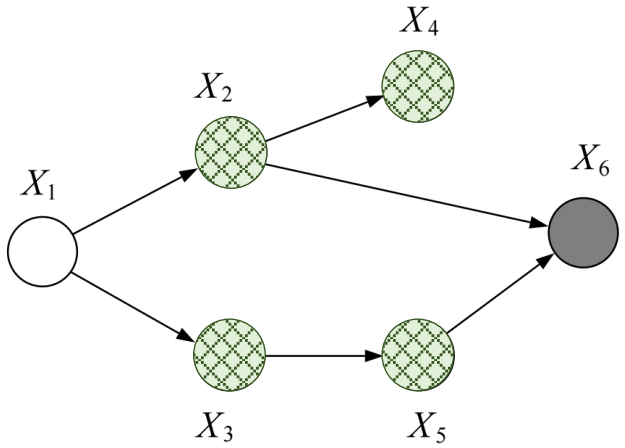
$$p(x_1|\bar{x}_6) = \frac{p(x_1, \bar{x}_6)}{p(\bar{x}_6)} \qquad \bar{x}_6 = 1$$

Image source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Evidence Node

Example: $x_i \in \{0,1\}$

We observed that $\bar{X}_6 = 1$



X_2	X_5	X_6	$p(x_6 \mid x_2, x_5)$
0	0	0	v_0
0	0	1	v_1
0	1	0	v_2
0	1	1	v_3
1	0	0	v_4
1	0	1	v_5
1	1	0	v_6
1	1	1	v_7

$\bar{X}_6 = 1$

X_2	X_5	$p(\bar{x}_6 = 1 \mid x_2, x_5)$
0	0	v_1
0	1	v_3
1	0	v_5
1	1	v_7

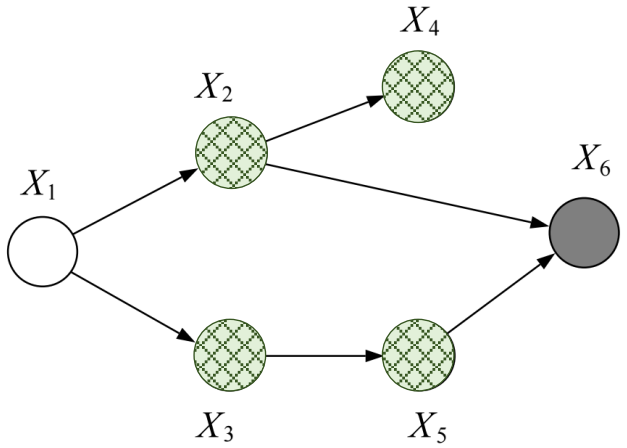
We are taking a 2d slice of the 3d probabilities or potentials

Image source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Variable Elimination

Conditional probability:

$$p(x_1|\bar{x}_6) = \frac{p(x_1, \bar{x}_6)}{p(\bar{x}_6)}$$



Marginal probability:

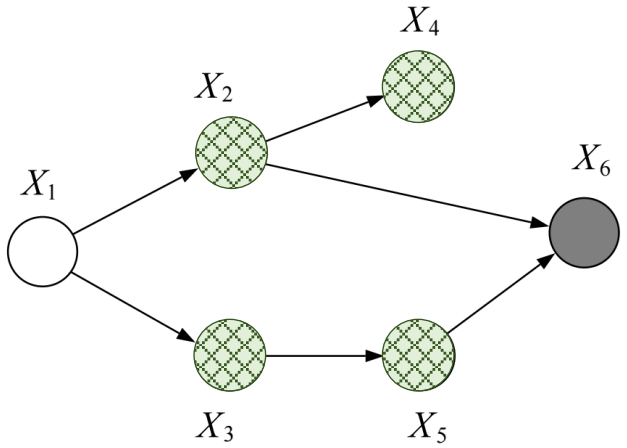
$$p(x_1, \bar{x}_6) = \sum_{x_2} \sum_{x_3} \sum_{x_4} \sum_{x_5} p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)p(x_5 \mid x_3)p(\bar{x}_6 \mid x_2, x_5)$$

- Summands can be pushed in due to the distributive law.
- $m_i(x_{S_i})$ denote the expression from performing Σ_{x_i}, where X_{S_i} are the variables, other than X_i, that appear in the summand.

Variable Elimination

Conditional probability:

$$p(x_1|\bar{x}_6) = \frac{p(x_1, \bar{x}_6)}{p(\bar{x}_6)}$$



Marginal probability:

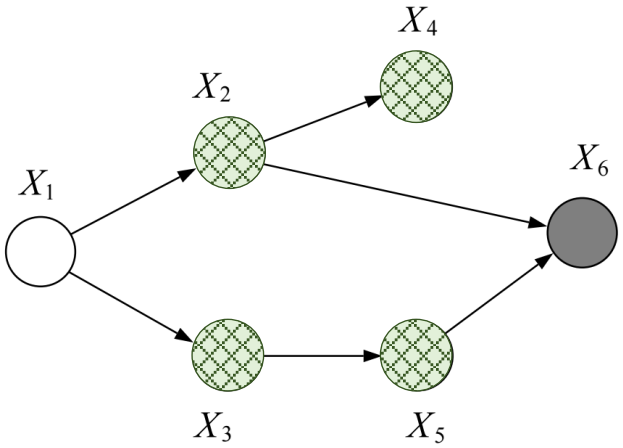
$$\begin{aligned} p(x_1, \bar{x}_6) &= \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)p(x_5 \mid x_3)p(\bar{x}_6 \mid x_2, x_5) \\ &= p(x_1)\sum_{x_2} p(x_2 \mid x_1)\sum_{x_3} p(x_3 \mid x_1)\sum_{x_4} p(x_4 \mid x_2)\sum_{x_5} p(x_5 \mid x_3)p(\bar{x}_6 \mid x_2, x_5) \end{aligned}$$

- Summands can be pushed in due to the distributive law.
- $m_i(x_{S_i})$ denote the expression from performing Σ_{x_i}, where X_{S_i} are the variables, other than X_i, that appear in the summand.

Variable Elimination

Conditional probability:

$$p(x_1|\bar{x}_6) = \frac{p(x_1, \bar{x}_6)}{p(\bar{x}_6)}$$

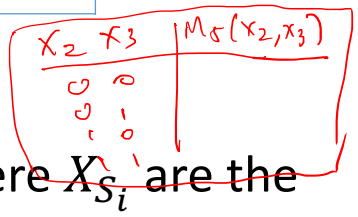


Marginal probability:

$$p(x_1, \bar{x}_6) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)p(x_5 \mid x_3)p(\bar{x}_6 \mid x_2, x_5)$$

$$= p(x_1)\sum_{x_2} p(x_2 \mid x_1)\sum_{x_3} p(x_3 \mid x_1)\sum_{x_4} p(x_4 \mid x_2)\underbrace{\sum_{x_5} p(x_5 \mid x_3)p(\bar{x}_6 \mid x_2, x_5)}_{m_5(x_2, x_3)}$$

eliminate X_5

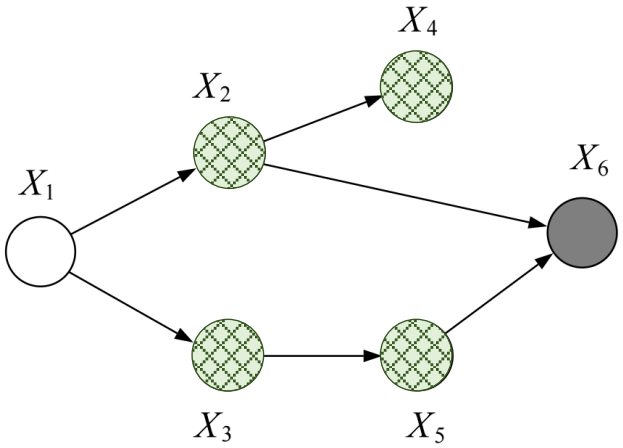


- Summands can be pushed in due to the distributive law.
- $m_i(x_{S_i})$ denote the expression from performing Σ_{x_i}, where X_{S_i} are the variables, other than X_i, that appear in the summand.

Variable Elimination

Conditional probability:

$$p(x_1|\bar{x}_6) = \frac{p(x_1, \bar{x}_6)}{p(\bar{x}_6)}$$



Marginal probability:

$$\begin{aligned}
p(x_1, \bar{x}_6) &= \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)p(x_5 \mid x_3)p(\bar{x}_6 \mid x_2, x_5) \\
&= p(x_1)\sum_{x_2} p(x_2 \mid x_1)\sum_{x_3} p(x_3 \mid x_1)\sum_{x_4} p(x_4 \mid x_2)\sum_{x_5} p(x_5 \mid x_3)p(\bar{x}_6 \mid x_2, x_5) \\
&= p(x_1)\sum_{x_2} p(x_2 \mid x_1)\sum_{x_3} p(x_3 \mid x_1)\sum_{x_4} p(x_4 \mid x_2)m_5(x_2, x_3)
\end{aligned}$$

eliminate X_5

- Summands can be pushed in due to the distributive law.
- $m_i(x_{S_i})$ denote the expression from performing Σ_{x_i}, where X_{S_i} are the variables, other than X_i, that appear in the summand.

Variable Elimination

Marginal probability:

$$p(x_1, \bar{x}_6) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)p(x_5 \mid x_3)p(\bar{x}_6 \mid x_2, x_5)$$

$$= p(x_1)\sum_{x_2} p(x_2 \mid x_1)\sum_{x_3} p(x_3 \mid x_1)\sum_{x_4} p(x_4 \mid x_2)\sum_{x_5} p(x_5 \mid x_3)p(\bar{x}_6 \mid x_2, x_5)$$

$$m_5(x_2, x_3)$$

eliminate X_5, independent of X_4

Marginalization Table

Example: $x_i \in \{0,1\}$

We observed that $\bar{X}_6 = 1$

X_2	X_5	$p(\bar{x}_6 = 1 \mid x_2, x_5)$
0	0	a_1
0	1	a_2
1	0	a_3
1	1	a_4

X_3	X_5	$p(x_5 \mid x_3)$
0	0	b_1
0	1	b_2
1	0	b_3
1	1	b_4

X_2	X_3	$\sum_{x_5} p(x_5 \mid x_3) p(\bar{x}_6 = 1 \mid x_2, x_5)$
0	0	$p(x_5 = 0 \mid x_3 = 0) p(\bar{x}_6 = 1 \mid x_2 = 0, x_5 = 0) +$ $p(x_5 = 1 \mid x_3 = 0) p(\bar{x}_6 = 1 \mid x_2 = 0, x_5 = 1) =$ $(b_1)(a_1) + (b_2)(a_2)$
0	1	$p(x_5 = 0 \mid x_3 = 1) p(\bar{x}_6 = 1 \mid x_2 = 0, x_5 = 0) +$ $p(x_5 = 1 \mid x_3 = 1) p(\bar{x}_6 = 1 \mid x_2 = 0, x_5 = 1) =$ $(b_3)(a_1) + (b_4)(a_2)$
1	0	$p(x_5 = 0 \mid x_3 = 0) p(\bar{x}_6 = 1 \mid x_2 = 1, x_5 = 0) +$ $p(x_5 = 1 \mid x_3 = 0) p(\bar{x}_6 = 1 \mid x_2 = 1, x_5 = 1) =$ $(b_1)(a_3) + (b_2)(a_4)$
1	1	$p(x_5 = 0 \mid x_3 = 1) p(\bar{x}_6 = 1 \mid x_2 = 1, x_5 = 0) +$ $p(x_5 = 1 \mid x_3 = 1) p(\bar{x}_6 = 1 \mid x_2 = 1, x_5 = 1) =$ $(b_3)(a_3) + (b_4)(a_4)$

Variable Elimination

Marginal probability:

$$p(x_1, \bar{x}_6) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)p(x_5 \mid x_3)p(\bar{x}_6 \mid x_2, x_5)$$

$$= p(x_1)\sum_{x_2} p(x_2 \mid x_1)\sum_{x_3} p(x_3 \mid x_1)\sum_{x_4} p(x_4 \mid x_2)\sum_{x_5} p(x_5 \mid x_3)p(\bar{x}_6 \mid x_2, x_5)$$

eliminate X_5, independent of X_4

$$= p(x_1)\sum_{x_2} p(x_2 \mid x_1)\sum_{x_3} p(x_3 \mid x_1)m_5(x_2, x_3)\sum_{x_4} p(x_4 \mid x_2)$$

eliminate X_4, Independent of X_3

Variable Elimination

Marginal probability:

$$p(x_1, \bar{x}_6) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)p(x_5 \mid x_3)p(\bar{x}_6 \mid x_2, x_5)$$

$$= p(x_1)\sum_{x_2} p(x_2 \mid x_1)\sum_{x_3} p(x_3 \mid x_1)\sum_{x_4} p(x_4 \mid x_2)\sum_{x_5} p(x_5 \mid x_3)p(\bar{x}_6 \mid x_2, x_5)$$

eliminate X_5, independent of X_4

$$= p(x_1)\sum_{x_2} p(x_2 \mid x_1)\sum_{x_3} p(x_3 \mid x_1)m_5(x_2, x_3)\sum_{x_4} p(x_4 \mid x_2)$$

eliminate X_4, Independent of X_3

$$= p(x_1)\sum_{x_2} p(x_2 \mid x_1)m_4(x_2)\sum_{x_3} p(x_3 \mid x_1)m_5(x_2, x_3)$$

eliminate X_3

Variable Elimination

Marginal probability:

$$p(x_1, \bar{x}_6) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)p(x_5 \mid x_3)p(\bar{x}_6 \mid x_2, x_5)$$

$$= p(x_1)\sum_{x_2} p(x_2 \mid x_1)\sum_{x_3} p(x_3 \mid x_1)\sum_{x_4} p(x_4 \mid x_2)\sum_{x_5} p(x_5 \mid x_3)p(\bar{x}_6 \mid x_2, x_5)$$

eliminate X_5, independent of X_4

$$= p(x_1)\sum_{x_2} p(x_2 \mid x_1)\sum_{x_3} p(x_3 \mid x_1)m_5(x_2, x_3)\sum_{x_4} p(x_4 \mid x_2)$$

eliminate X_4, Independent of X_3

$$= p(x_1)\sum_{x_2} p(x_2 \mid x_1)m_4(x_2)\sum_{x_3} p(x_3 \mid x_1)m_5(x_2, x_3)$$

eliminate X_3

$$= p(x_1)\sum_{x_2} p(x_2 \mid x_1)m_4(x_2)m_3(x_1, x_2)$$

$$= p(x_1)m_2(x_1)$$

eliminate X_2

Variable Elimination

Marginal probability:

$$p(x_1, \bar{x}_6) = p(x_1) m_2(x_1)$$

From this result we can obtain the probability $p(\bar{x}_6)$ by taking an additional sum over X_1:

$$p(\bar{x}_6) = \sum_{x_1} p(x_1) m_2(x_1)$$

The desired conditional is obtained by:

$$p(x_1 \mid \bar{x}_6) = \frac{p(x_1) m_2(x_1)}{\sum_{x_1} p(x_1) m_2(x_1)}$$

Towards an General Algorithm

- Conditioning to Marginalization Trick

$$g(\bar{x}_i) = \sum_{x_i} g(x_i)\delta(x_i, \bar{x}_i)$$

- The Variable Elimination Algorithm

ELIMINATE($\mathcal{G}, E, F$)
  INITIALIZE($\mathcal{G}, F$)
  EVIDENCE(E)
  UPDATE($\mathcal{G}$)
  NORMALIZE(F)

INITIALIZE($\mathcal{G}, F$)
  choose an ordering I such that F appears last
  for each node X_i in $\mathcal{V}$
    place $p(x_i \mid x_{\pi_i})$ on the active list
  end

EVIDENCE(E)
  for each i in E
    place $\delta(x_i, \bar{x}_i)$ on the active list
  end

UPDATE($\mathcal{G}$)
  for each i in I
    find all potentials from the active list that reference x_i and remove them from the active list
    let $\phi_i(x_{T_i})$ denote the product of these potentials
    let $m_i(x_{S_i}) = \sum_{x_i} \phi_i(x_{T_i})$
    place $m_i(x_{S_i})$ on the active list
  end

NORMALIZE(F)
  $p(x_F \mid \bar{x}_E) \leftarrow \phi_F(x_F) / \sum_{x_F} \phi_F(x_F)$

Conditioning to Marginalization Trick

- Notational trick in which conditioning is viewed as a summation.
- This trick will allow us to treat marginalization and conditioning as formally equivalent.
- Make it easier to bring the key operations of the inference algorithms into focus.

Conditioning to Marginalization Trick

- To capture the fact that X_i is fixed at the value $\bar{X}_i$, we define an evidence potential:

$$\delta(x_i, \bar{x}_i) = \begin{cases} 1 & if \quad x_i = \bar{x}_i \\ 0 & otherwise \end{cases}$$

- The evidence potential allows us to turn evaluations into sums:

$$g(\bar{x}_i) = \sum_{x_i} g(x_i)\delta(x_i, \bar{x}_i)$$

- A trick that also extends to multivariate functions with X_i as one of the arguments.

Conditioning to Marginalization Trick

$$g(\bar{x}_i) = \sum_{x_i} g(x_i)\delta(x_i, \bar{x}_i)$$

$$\sum_{x_i} g(x_i)\delta(x_i, \bar{x}_i)$$

$$= g(x_i = 0)\underbrace{\delta(x_i = 0)}_{0} + \cdots + g(x_i = \bar{x}_i)\underbrace{\delta(x_i = \bar{x}_i)}_{1} + \cdots + g(x_i = k)\underbrace{\delta(x_i = k)}_{0}$$

$$= g(x_i = \bar{x}_i)$$

Conditioning to Marginalization Trick

Example:

(Directed Graph)

$p(\bar{x}_6|x_2, x_5)$ from the previous example can be written as:

$$m_6(x_2, x_5) = \sum_{x_6} p(x_6 \mid x_2, x_5)\delta(x_6, \bar{x}_6)$$

$$= p(\bar{x}_6|x_2, x_5)$$

(Undirected Graph)

$\psi(x_2, x_5, \bar{x}_6)$ can be written as:

$$m_6(x_2, x_5) = \sum_{x_6} \psi(x_2, x_5, x_6)\delta(x_6, \bar{x}_6)$$

$$= \psi(x_2, x_5, \bar{x}_6)$$

We have turned conditioning into marginalization

Conditioning to Marginalization Trick

- We further define the total evidence potential on a set of nodes X_E to be conditioned on:

$$\delta(x_E, \bar{x}_E) = \prod_{i \in E} \delta(x_i, \bar{x}_i) = \begin{cases} 1 & if \quad x_E = \bar{x}_E \\ 0 & otherwise \end{cases}$$

- The numerator and the denominator of the conditional probability $p(x_F|\bar{x}_E)$ can be obtained by summation:

$$p(x_F|\bar{x}_E) = \frac{p(x_F, \bar{x}_E)}{p(\bar{x}_E)} = \frac{\sum_{x_E} p(x_F, x_E)\delta(x_E, \bar{x}_E)}{\sum_{x_F}\sum_{x_E} p(x_F, x_E)\delta(x_E, \bar{x}_E)}$$

Again, we have turned conditioning into marginalization

Conditioning to Marginalization Trick

- **Note:** evidence potentials is merely a piece of formal trickery to simplifies our description of various inference algorithms.
- In practice we would not perform the sum over a function that we know to be zero over most of the sample space.
- But rather we would take "slices" of the appropriate probabilities or potentials.

Variable Elimination Algorithm: Directed Graphs

ELIMINATE($\mathcal{G}, E, F$) // main steps of the "Variable Elimination Algorithm"
  INITIALIZE($\mathcal{G}, F$)
  EVIDENCE(E)
  UPDATE($\mathcal{G}$)
  NORMALIZE(F)

1: INITIALIZE($\mathcal{G}, F$) // choose elimination ordering, and add local condition probabilities in **active list**
  choose an ordering I such that F appears last
  for each node X_i in $\mathcal{V}$
    place $p(x_i \mid x_{\pi_i})$ on the active list
  end

2: EVIDENCE(E) // add evidence potentials in **active list**
  for each i in E
    place $\delta(x_i, \bar{x}_i)$ on the active list
  end

3: UPDATE($\mathcal{G}$) // **marginalization**, and update active list
  for each i in I
    find all potentials from the active list that reference x_i and remove them from the active list
    let $\phi_i(x_{T_i})$ denote the product of these potentials
    let $m_i(x_{S_i}) = \sum_{x_i} \phi_i(x_{T_i})$
    place $m_i(x_{S_i})$ on the active list
  end

4: NORMALIZE(F) // compute the desired **conditional probability**
  $p(x_F \mid \bar{x}_E) \leftarrow \phi_F(x_F) / \sum_{x_F} \phi_F(x_F)$

Source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Variable Elimination Algorithm: Directed Graphs

// choose elimination ordering, and add local condition probabilities in **active list**

1: INITIALIZE($\mathcal{G}, F$)

choose an ordering I such that F appears last

for each node X_i in $\mathcal{V}$

place $p(x_i \mid x_{\pi_i})$ on the active list

end

Example:

Evidence node is X_6 and query node is X_1.
We choose the elimination ordering:
$I = \{6, 5, 4, 3, 2, 1\}$,
in which the query node appears last.

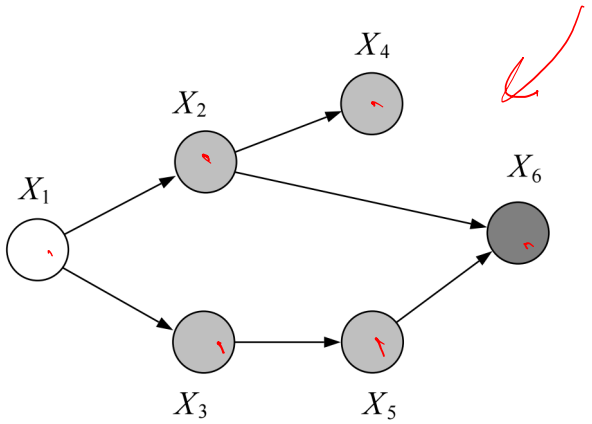


Image source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Variable Elimination Algorithm: Directed Graphs

// choose elimination ordering, and add local condition probabilities in **active list**

1: INITIALIZE($\mathcal{G}, F$)

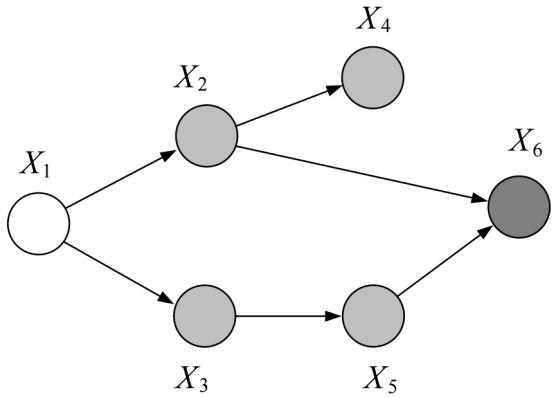
choose an ordering I such that F appears last

for each node X_i in $\mathcal{V}$

place $p(x_i \mid x_{\pi_i})$ on the active list

end

Example:



Active list:

$\{ p(x_1), p(x_2|x_1), p(x_3|x_1), p(x_4|x_2), p(x_5|x_3), p(x_6|x_2, x_5) \}$

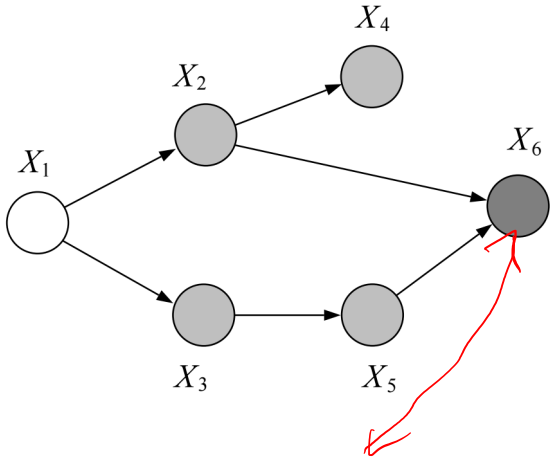
Image source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Variable Elimination Algorithm: Directed Graphs

// add evidence potentials in **active list**

2: EVIDENCE(E)

```
for each i in E
    place δ(x_i, x̄_i) on the active list
end
```



Example:

Active list:

$$\{p(x_1), p(x_2|x_1), p(x_3|x_1), p(x_4|x_2), p(x_5|x_3), p(x_6|x_2, x_5), \delta(x_6, \bar{x}_6)\}$$

Image source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Variable Elimination Algorithm: Directed Graphs

// **marginalization**, and update active list

3: UPDATE($\mathcal{G}$)

for each i in I ✓

find all potentials from the active list that reference x_i and remove them from the active list
let $\phi_i(x_{T_i})$ denote the product of these potentials
let $m_i(x_{S_i}) = \sum_{x_i} \phi_i(x_{T_i})$
place $m_i(x_{S_i})$ on the active list

end

Example: $I = \{6, 5, 4, 3, 2, 1\}$

$\boldsymbol{i = 6}$: $\{p(x_1), p(x_2|x_1), p(x_3|x_1), p(x_4|x_2), p(x_5|x_3), \cancel{p(x_6|x_2, x_5), \delta(x_6, \bar{x}_6)}\}$

$$\phi_6(x_2, x_5, x_6) = p(x_6|x_2, x_5)\delta(x_6, \bar{x}_6)$$

$$m_6(x_2, x_5) = \sum_{x_6} \phi_6(x_2, x_5, x_6) = \sum_{x_6} p(x_6|x_2, x_5)\delta(x_6, \bar{x}_6)$$

$$\{p(x_1), p(x_2|x_1), p(x_3|x_1), p(x_4|x_2), p(x_5|x_3), m_6(x_2, x_5)\}$$

Variable Elimination Algorithm: Directed Graphs

// **marginalization**, and update active list

3: UPDATE($\mathcal{G}$)

for each i in I

find all potentials from the active list that reference x_i and remove them from the active list
let $\phi_i(x_{T_i})$ denote the product of these potentials
let $m_i(x_{S_i}) = \sum_{x_i} \phi_i(x_{T_i})$
place $m_i(x_{S_i})$ on the active list

end

Example: $I = \{6, 5, 4, 3, 2, 1\}$

$\boldsymbol{i = 5}$: $\{p(x_1), p(x_2|x_1), p(x_3|x_1), p(x_4|x_2), p(x_5|x_3), m_6(x_2, x_5)\}$

$$\phi_5(x_2, x_3) = p(x_5|x_3)m_6(x_2, x_5)$$

$$m_5(x_2, x_3) = \sum_{x_5} \phi_5(x_2, x_3) = \sum_{x_5} p(x_5|x_3)m_6(x_2, x_5)$$

$$\{p(x_1), p(x_2|x_1), p(x_3|x_1), p(x_4|x_2), m_5(x_2, x_3)\}$$

Variable Elimination Algorithm: Directed Graphs

// **marginalization**, and update active list

3: UPDATE($\mathcal{G}$)

for each i in I

find all potentials from the active list that reference x_i and remove them from the active list
let $\phi_i(x_{T_i})$ denote the product of these potentials
let $m_i(x_{S_i}) = \sum_{x_i} \phi_i(x_{T_i})$
place $m_i(x_{S_i})$ on the active list

end

Example: $I = \{6, 5, 4, 3, 2, 1\}$

$\boldsymbol{i = 4}$: $\{p(x_1), p(x_2|x_1), p(x_3|x_1), \cancel{p(x_4|x_2)}, m_5(x_2, x_3)\}$

$$\phi_4(x_2) = p(x_4|x_2)$$

$$m_4(x_2) = \sum_{x_4} \phi_4(x_2) = \sum_{x_4} p(x_4|x_2) = 1$$

$\{p(x_1), p(x_2|x_1), p(x_3|x_1), m_5(x_2, x_3)\}$ Ignore $m_4(x_2)$ since its 1!

Variable Elimination Algorithm: Directed Graphs

// **marginalization**, and update active list

3: UPDATE($\mathcal{G}$)

for each i in I

find all potentials from the active list that reference x_i and remove them from the active list
let $\phi_i(x_{T_i})$ denote the product of these potentials
let $m_i(x_{S_i}) = \sum_{x_i} \phi_i(x_{T_i})$
place $m_i(x_{S_i})$ on the active list

end

Example: $I = \{6, 5, 4, 3, 2, 1\}$

$\boldsymbol{i = 3}$:

$$\{p(x_1), p(x_2|x_1), \cancel{p(x_3|x_1), m_5(x_2, x_3)}\}$$

$$\phi_3(x_1, x_2) = p(x_3|x_1) m_5(x_2, x_3)$$

$$m_3(x_1, x_2) = \sum_{x_3} \phi_3(x_1, x_2) = \sum_{x_3} p(x_3|x_1) m_5(x_2, x_3)$$

$$\{p(x_1), p(x_2|x_1), m_3(x_1, x_2)\}$$

Variable Elimination Algorithm: Directed Graphs

// **marginalization**, and update active list

3: UPDATE($\mathcal{G}$)

for each i in I

find all potentials from the active list that reference x_i and remove them from the active list
let $\phi_i(x_{T_i})$ denote the product of these potentials
let $m_i(x_{S_i}) = \sum_{x_i} \phi_i(x_{T_i})$
place $m_i(x_{S_i})$ on the active list

end

Example: $I = \{6, 5, 4, 3, 2, 1\}$

$\boldsymbol{i = 2}$:

$$\{p(x_1), \cancel{p(x_2|x_1), m_3(x_1, x_2)}\}$$

$$\phi_2(x_1) = p(x_2|x_1)m_3(x_1, x_2)$$

$$m_2(x_1) = \sum_{x_2} \phi_2(x_1) = \sum_{x_2} p(x_2|x_1)m_3(x_1, x_2)$$

$$\{p(x_1), m_2(x_1)\}$$

Variable Elimination Algorithm: Directed Graphs

// **marginalization**, and update active list

3: UPDATE($\mathcal{G}$)

for each i in I

find all potentials from the active list that reference x_i and remove them from the active list
let $\phi_i(x_{T_i})$ denote the product of these potentials
let $m_i(x_{S_i}) = \sum_{x_i} \phi_i(x_{T_i})$
place $m_i(x_{S_i})$ on the active list

end

Example: $I = \{6, 5, 4, 3, 2, 1\}$

$\boldsymbol{i = 1}$:

$$\{p(x_1), m_2(x_1)\}$$

$$\phi_1(x_1) = p(x_1)m_2(x_1)$$

Unnormalized conditional probability, $p(x_1, \bar{x}_6)$

$$m_1(x_1) = \sum_{x_1} \phi_1(x_1) = \sum_{x_1} p(x_1)m_2(x_1)$$

Normalization factor, $p(\bar{x}_6)$

Variable Elimination Algorithm: Directed Graphs

// compute the desired **conditional probability**

4: NORMALIZE(F)

$p(x_F \mid \bar{x}_E) \leftarrow \phi_F(x_F) / \sum_{x_F} \phi_F(x_F)$

Example:

From the previous step, we have $\phi_1(x_1)$ and $m_1(x_1) = \sum_{x_1} \phi_1(x_1)$, which we use to compute the desired conditional probability:

$$p(x_1|x_6) = \frac{\phi_1(x_1)}{\sum_{x_1} \phi_1(x_1)}$$

Variable Elimination Algorithm: Directed Graphs

Elimination order: $I = \{6, 5, 4, 3, 2, 1\}$

$$p(\bar{x}_6) = \sum_{x1} \sum_{x_2} \sum_{x_3} \sum_{x_4} \sum_{x_5} \sum_{x_6} p(x_1)p(x_2|x_1)$$
$$p(x_3|x_1)p(x_4|x_2)p(x_5|x_3)p(x_6|x_2, x_5)\delta(x_6, \bar{x}_6)$$

$i = 6$:

$$p(\bar{x}_6) = \sum_{x_1} \sum_{x_2} \sum_{x_3} \sum_{x_4} \sum_{x_5} p(x_1)p(x_2|x_1)\, p(x_3|x_1)p(x_4|x_2)p(x_5|x_3)$$
$$\underbrace{\sum_{x_6} p(x_6|x_2, x_5)\delta(x_6, \bar{x}_6)}_{m_6(x_2, x_5)}$$
$$= \sum_{x_1} \sum_{x_2} \sum_{x_3} \sum_{x_4} \sum_{x_5} p(x_1)p(x_2|x_1)\, p(x_3|x_1)p(x_4|x_2)p(x_5|x_3) m_6(x_2, x_5)$$

$i = 5$:

$$p(\bar{x}_6) = \sum_{x_1} \sum_{x_2} \sum_{x_3} \sum_{x_4} p(x_1)p(x_2|x_1)\, p(x_3|x_1)p(x_4|x_2) \underbrace{\sum_{x_5} p(x_5|x_3)m_6(x_2, x_5)}_{m_5(x_2, x_3)}$$
$$= \sum_{x_1} \sum_{x_2} \sum_{x_3} \sum_{x_4} p(x_1)p(x_2|x_1)\, p(x_3|x_1)p(x_4|x_2)\, m_5(x_2, x_3)$$

Variable Elimination Algorithm: Directed Graphs

Elimination order: $I = \{6, 5, 4, 3, 2, 1\}$

$i = 4$:

$$p(\bar{x}_6) = \sum_{x_1} \sum_{x_2} \sum_{x_3} p(x_1)p(x_2|x_1)p(x_3|x_1)m_5(x_2, x_3) \underbrace{\sum_{x_4} p(x_4|x_2)}_{m_4(x_2) = 1}$$

$i = 3$:

$$p(\bar{x}_6) = \sum_{x_1} \sum_{x_2} p(x_1)p(x_2|x_1) \underbrace{\sum_{x_3} p(x_3|x_1)m_5(x_2, x_3)}_{m_3(x_1, x_2)}$$

$i = 2$:

$$p(\bar{x}_6) = \sum_{x_1} p(x_1) \underbrace{\sum_{x_2} p(x_2|x_1)m_3(x_1, x_2)}_{m_2(x_1)}$$

$i = 1$:

$$p(\bar{x}_6) = \sum_{x_1} p(x_1)m_2(x_1)$$

Normalization factor

Unnormalized conditional probability,
$p(x_1, \bar{x}_6) = p(x_1)m_2(x_1)$

Variable Elimination Algorithm: Undirected Graphs

- Entire variable eliminate algorithm for directed graph goes through without essential change to the undirected case.
- Only change needed in the initialize procedure.
- Instead of using local conditional probabilities we initialize the active list to contain the potentials of $\{\psi_{x_c}(x_C)\}$.

Variable Elimination Algorithm: Undirected Graphs

Example:

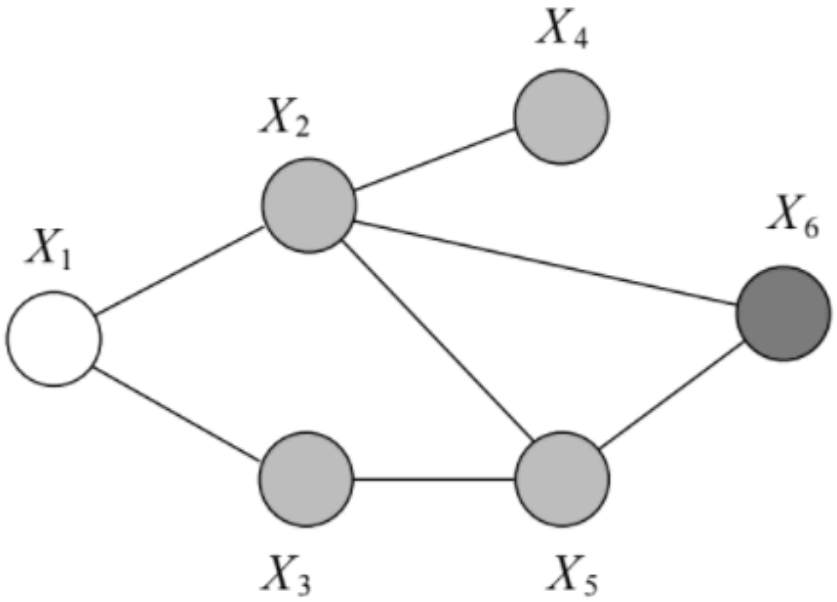


Image source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Variable Elimination Algorithm: Undirected Graphs

$$
\begin{aligned}
p(x_1, \bar{x}_6) &= \frac{1}{Z} \sum_{x_2} \sum_{x_3} \sum_{x_4} \sum_{x_5} \sum_{x_6} \psi(x_1, x_2)\psi(x_1, x_3)\psi(x_2, x_4)\psi(x_3, x_5)\psi(x_2, x_5, x_6)\delta(x_6, \bar{x}_6) \\
&= \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) \sum_{x_3} \psi(x_1, x_3) \sum_{x_4} \psi(x_2, x_4) \sum_{x_5} \psi(x_3, x_5) \sum_{x_6} \psi(x_2, x_5, x_6)\delta(x_6, \bar{x}_6) \\
&= \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) \sum_{x_3} \psi(x_1, x_3) \sum_{x_4} \psi(x_2, x_4) \sum_{x_5} \psi(x_3, x_5) m_6(x_2, x_5) \\
&= \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) \sum_{x_3} \psi(x_1, x_3) m_5(x_2, x_3) \sum_{x_4} \psi(x_2, x_4) \\
&= \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) m_4(x_2) \sum_{x_3} \psi(x_1, x_3) m_5(x_2, x_3) \\
&= \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) m_4(x_2) m_3(x_1, x_2) \\
&= \frac{1}{Z} m_2(x_1).
\end{aligned}
$$

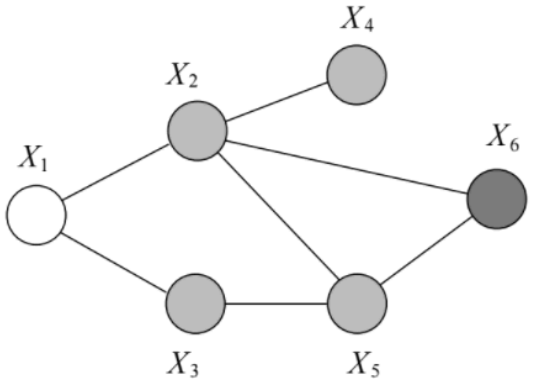


Image source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Variable Elimination Algorithm: Undirected Graphs

- Marginalizing further over X_1 yields:

$$p(\bar{x}_6) = \frac{1}{Z} \sum_{x_1} m_2(x_1),$$

- We calculate the desired conditional as:

$$p(x_1 \mid \bar{x}_6) = \frac{m_2(x_1)}{\sum_{x_1} m_2(x_1)}$$

where the normalization factor Z cancels.

- **Important note**: For a marginal probability the normalization factor Z does not cancel, and must be calculated explicitly.

NUS National University of Singapore | School of Computing

Variable Elimination: Computational Complexity

Reconstituted Graph, Elimination Clique, and Treewidth

Variable Elimination Algorithm: Directed Graphs

ELIMINATE($\mathcal{G}, E, F$) // main steps of the "Variable Elimination Algorithm"
  INITIALIZE($\mathcal{G}, F$)
  EVIDENCE(E)
  UPDATE($\mathcal{G}$)
  NORMALIZE(F)

1: INITIALIZE($\mathcal{G}, F$) // choose elimination ordering, and add local condition probabilities in **active list**
  choose an ordering I such that F appears last
  for each node X_i in $\mathcal{V}$
    place $p(x_i \mid x_{\pi_i})$ on the active list
  end

2: EVIDENCE(E) // add evidence potentials in **active list**
  f
    place $\delta(x_i, \bar{x}_i)$ on the active list
  end

How much does this cost?

3: UPDATE($\mathcal{G}$) // **marginalization**, and update active list
  for each i in I
    find all potentials from the active list that reference x_i and remove them from the active list
    let $\phi_i(x_{T_i})$ denote the product of these potentials
    let $m_i(x_{S_i}) = \sum_{x_i} \phi_i(x_{T_i})$
    place $m_i(x_{S_i})$ on the active list
  end

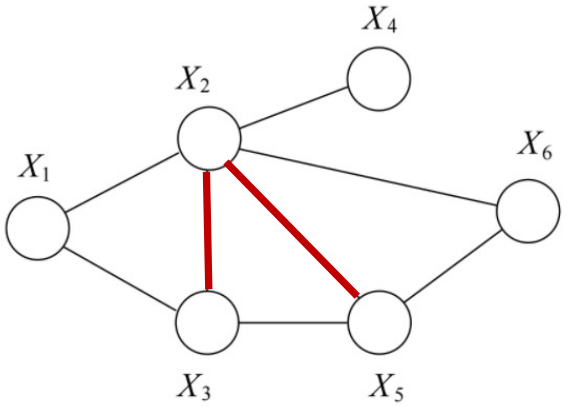
4: NORMALIZE(F) // compute the desired **conditional probability**
  $p(x_F \mid \bar{x}_E) \leftarrow \phi_F(x_F) / \sum_{x_F} \phi_F(x_F)$

Source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

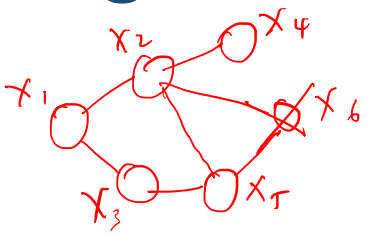
Reconstituted Graph

- The variable elimination algorithm successively eliminates the nodes of $\mathcal{G}$ in the ordering I.
- "Eliminate" means removing the node from the graph and connecting the (remaining) neighbors of the node.
- The original and newly created edges created during the elimination process are recorded in the reconstituted graph $\tilde{\mathcal{G}} = (\mathcal{V}, \tilde{\mathcal{E}})$.

ELIMINATE$(\mathcal{G}, E, F)$
  INITIALIZE$(\mathcal{G}, F)$
  EVIDENCE(E)
  UPDATE$(\mathcal{G})$
  NORMALIZE(F)



Recall: Variable Elimination Algorithm



// **marginalization**, and update active list

3: UPDATE($\mathcal{G}$)

for each i in I

find all potentials from the active list that reference x_i and remove them from the active list
let $\phi_i(x_{T_i})$ denote the product of these potentials
let $m_i(x_{S_i}) = \sum_{x_i} \phi_i(x_{T_i})$
place $m_i(x_{S_i})$ on the active list

end

Example: $I = \{6, 5, 4, 3, 2, 1\}$

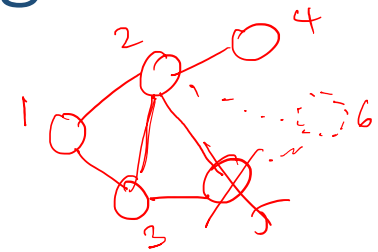
$\boldsymbol{i = 6}$: $\{p(x_1), p(x_2|x_1), p(x_3|x_1), p(x_4|x_2), p(x_5|x_3), p(x_6|x_2, x_5), \delta(x_6, \bar{x}_6)\}$

$$\phi_6(x_2, x_5, x_6) = p(x_6|x_2, x_5)\delta(x_6, \bar{x}_6)$$

$$m_6(x_2, x_5) = \sum_{x_6} \phi_6(x_2, x_5, x_6) = \sum_{x_6} p(x_6|x_2, x_5)\delta(x_6, \bar{x}_6)$$

$$\{p(x_1), p(x_2|x_1), p(x_3|x_1), p(x_4|x_2), p(x_5|x_3), m_6(x_2, x_5)\}$$

Recall: Variable Elimination Algorithm



// **marginalization**, and update active list

3: UPDATE($\mathcal{G}$)

for each i in I

find all potentials from the active list that reference x_i and remove them from the active list
let $\phi_i(x_{T_i})$ denote the product of these potentials
let $m_i(x_{S_i}) = \sum_{x_i} \phi_i(x_{T_i})$
place $m_i(x_{S_i})$ on the active list

end

Example: $I = \{6, 5, 4, 3, 2, 1\}$

$i = 5$: $\{p(x_1), p(x_2|x_1), p(x_3|x_1), p(x_4|x_2), p(x_5|x_3), m_6(x_2, x_5)\}$

$$\phi_5(x_2, x_3) = p(x_5|x_3) m_6(x_2, x_5)$$

$$m_5(x_2, x_3) = \sum_{x_5} \phi_5(x_2, x_3) = \sum_{x_5} p(x_5|x_3) m_6(x_2, x_5)$$

$$\{p(x_1), p(x_2|x_1), p(x_3|x_1), p(x_4|x_2), m_5(x_2, x_3)\}$$

Recall: Variable Elimination Algorithm

// **marginalization**, and update active list

3: UPDATE($\mathcal{G}$)

for each i in I

find all potentials from the active list that reference x_i and remove them from the active list
let $\phi_i(x_{T_i})$ denote the product of these potentials
let $m_i(x_{S_i}) = \sum_{x_i} \phi_i(x_{T_i})$
place $m_i(x_{S_i})$ on the active list

end

Example: $I = \{6, 5, 4, 3, 2, 1\}$

$\boldsymbol{i = 4}$: $\{p(x_1), p(x_2|x_1), p(x_3|x_1), \cancel{p(x_4|x_2)}, m_5(x_2, x_3)\}$

$$\phi_4(x_2) = p(x_4|x_2)$$

$$m_4(x_2) = \sum_{x_4} \phi_4(x_2) = \sum_{x_4} p(x_4|x_2) = 1$$

$\{p(x_1), p(x_2|x_1), p(x_3|x_1), m_5(x_2, x_3)\}$ Ignore $m_4(x_2)$ since its 1!

Recall: Variable Elimination Algorithm

// **marginalization**, and update active list

3: UPDATE($\mathcal{G}$)

for each i in I

find all potentials from the active list that reference x_i and remove them from the active list
let $\phi_i(x_{T_i})$ denote the product of these potentials
let $m_i(x_{S_i}) = \sum_{x_i} \phi_i(x_{T_i})$
place $m_i(x_{S_i})$ on the active list

end

Example: $I = \{6, 5, 4, 3, 2, 1\}$

$\boldsymbol{i = 3}$: $\{p(x_1), p(x_2|x_1), \cancel{p(x_3|x_1), m_5(x_2, x_3)}\}$

$$\phi_3(x_1, x_2) = p(x_3|x_1) m_5(x_2, x_3)$$

$$m_3(x_1, x_2) = \sum_{x_3} \phi_3(x_1, x_2) = \sum_{x_3} p(x_3|x_1) m_5(x_2, x_3)$$

$$\{p(x_1), p(x_2|x_1), m_3(x_1, x_2)\}$$

Recall: Variable Elimination Algorithm

// **marginalization**, and update active list

3: UPDATE($\mathcal{G}$)

for each i in I

find all potentials from the active list that reference x_i and remove them from the active list
let $\phi_i(x_{T_i})$ denote the product of these potentials
let $m_i(x_{S_i}) = \sum_{x_i} \phi_i(x_{T_i})$
place $m_i(x_{S_i})$ on the active list

end

Example: $I = \{6, 5, 4, 3, 2, 1\}$

$\boldsymbol{i = 2}$:

$$\{p(x_1), \cancel{p(x_2|x_1), m_3(x_1, x_2)}\}$$

$$\phi_2(x_1) = p(x_2|x_1)m_3(x_1, x_2)$$

$$m_2(x_1) = \sum_{x_2} \phi_2(x_1) = \sum_{x_2} p(x_2|x_1)m_3(x_1, x_2)$$

$$\{p(x_1), m_2(x_1)\}$$

Recall: Variable Elimination Algorithm

// **marginalization**, and update active list

3: UPDATE($\mathcal{G}$)

for each i in I

find all potentials from the active list that reference x_i and remove them from the active list
let $\phi_i(x_{T_i})$ denote the product of these potentials
let $m_i(x_{S_i}) = \sum_{x_i} \phi_i(x_{T_i})$
place $m_i(x_{S_i})$ on the active list

end

Example: $I = \{6, 5, 4, 3, 2, 1\}$

$\boldsymbol{i = 1}$:

$$\{p(x_1), m_2(x_1)\}$$

$$\phi_1(x_1) = p(x_1)m_2(x_1)$$

Unnormalized conditional probability, $p(x_1, \bar{x}_6)$

$$m_1(x_1) = \sum_{x_1} \phi_1(x_1) = \sum_{x_1} p(x_1)m_2(x_1)$$

Normalization factor, $p(\bar{x}_6)$

MRFs: Reconstituted Graph

Example: Elimination ordering (6; 5; 4; 3; 2; 1)

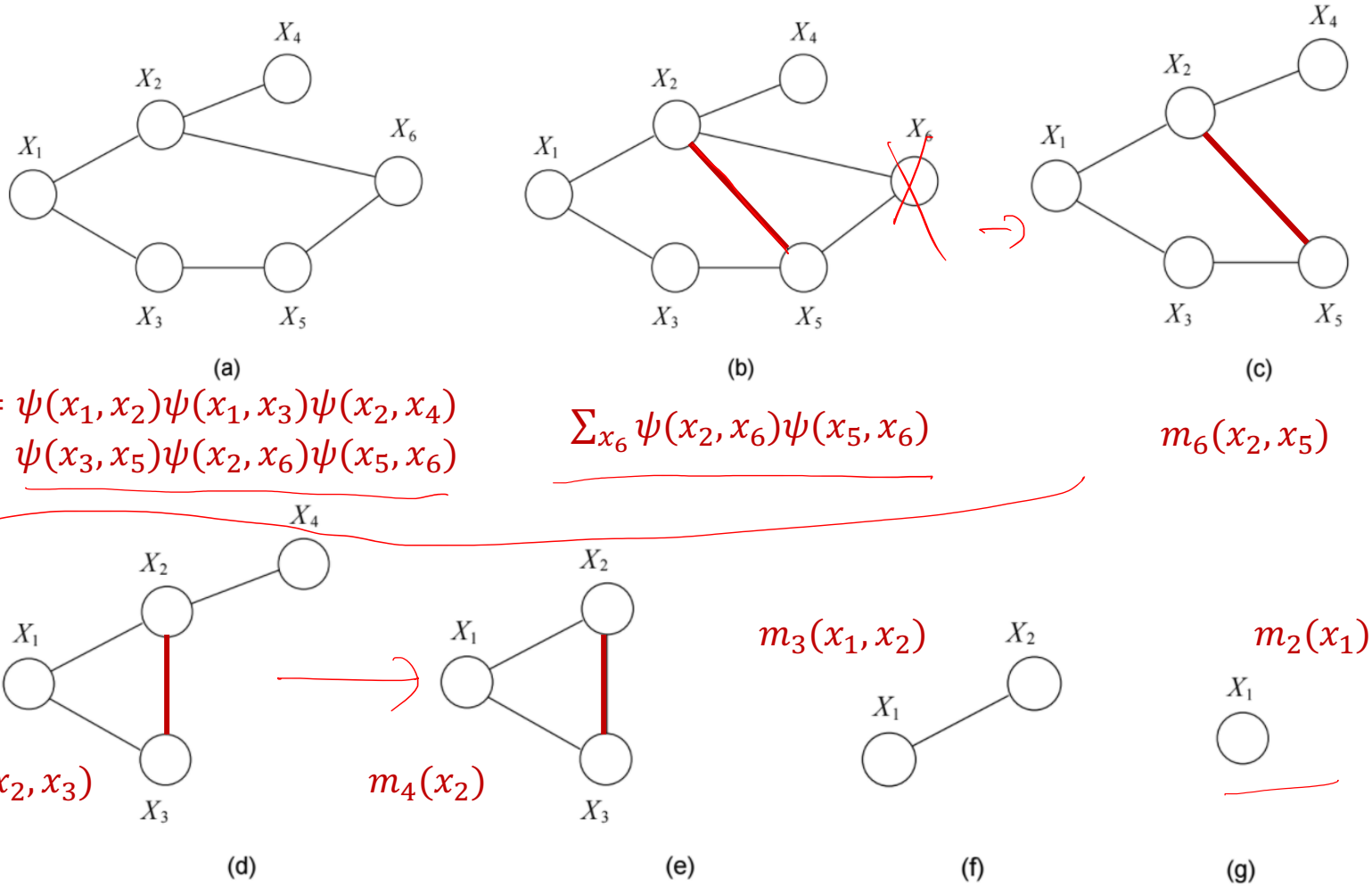
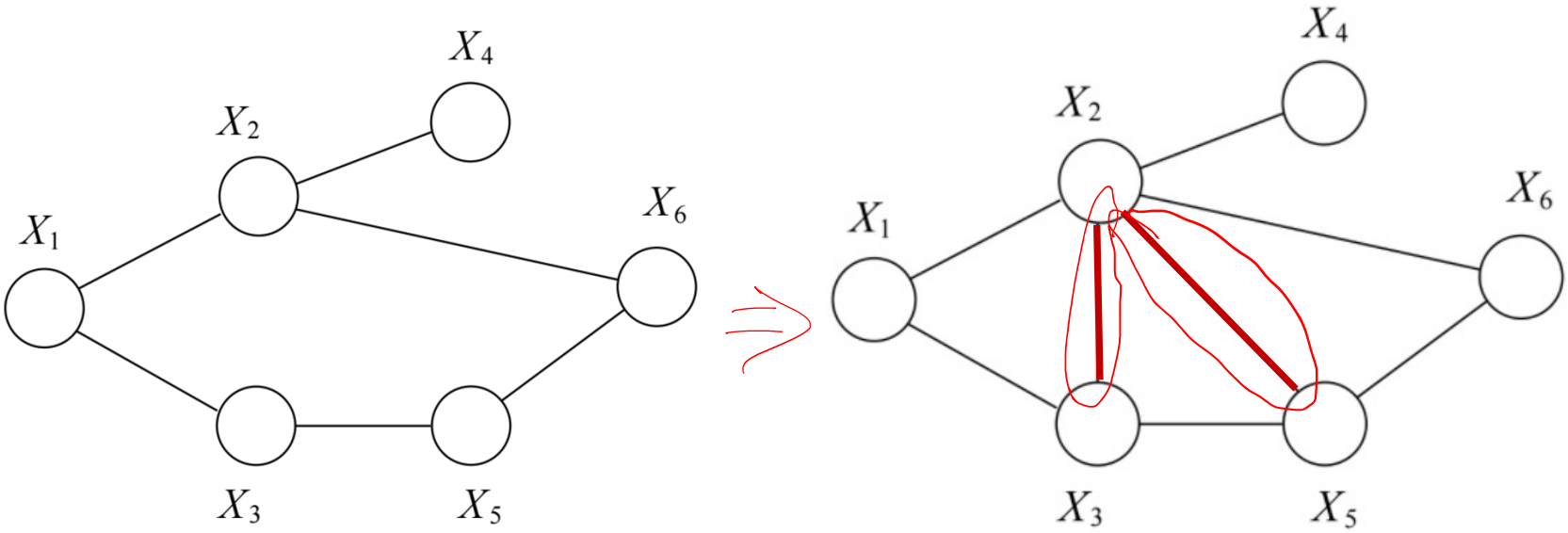


Image source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Reconstituted Graph

Example: Elimination ordering (6; 5; 4; 3; 2; 1)



Original undirected graph

Reconstituted graph: additional edges (red) added during the elimination process

Image source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Reconstituted Graph

- A simple greedy algorithm for eliminating nodes in an undirected graph.
- The additional edges added during the elimination process forms the reconstituted graph.

```
UndirectedGraphEliminate(G, I)
    for each node X_i in I
        connect all of the remaining neighbors of X_i
        remove X_i from the graph
    end
```

Source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Computational Complexity

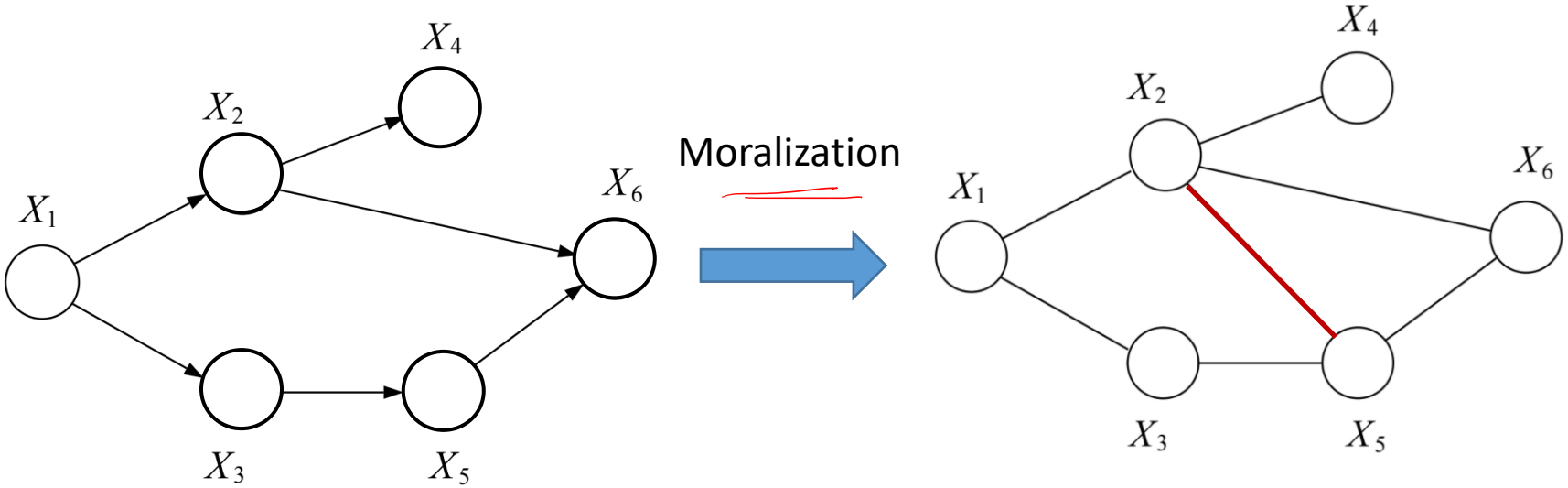
- Computation complexity of DGMs can be analyzed in the same way as UGMs by moralization.

DIRECTEDGRAPHELIMINATE(G, I)
    $G^m = \text{MORALIZE}(G)$
    UNDIRECTEDGRAPHELIMINATE(G^m, I)

MORALIZE(G)
    for each node X_i in I
        connect all of the parents of X_i
    end drop the orientation of all edges
    return G

Source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Computational Complexity



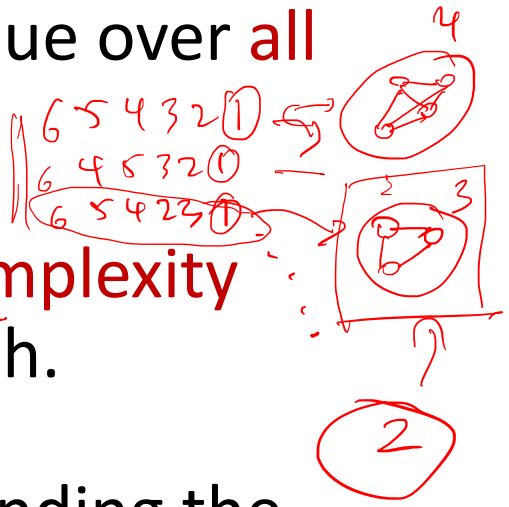
- A DGM is converted into UGM, where the computational complexity can be analyzed.

Computational Complexity

- Elimination process adds new edges between (remaining) neighbors of the node.
- This creates new "elimination cliques" in the graph.
- Overall complexity depends on the size of the largest elimination clique.
  - $O(nk^M)$ where M is the size of the largest elimination clique
- Which depends on the choice of elimination ordering.

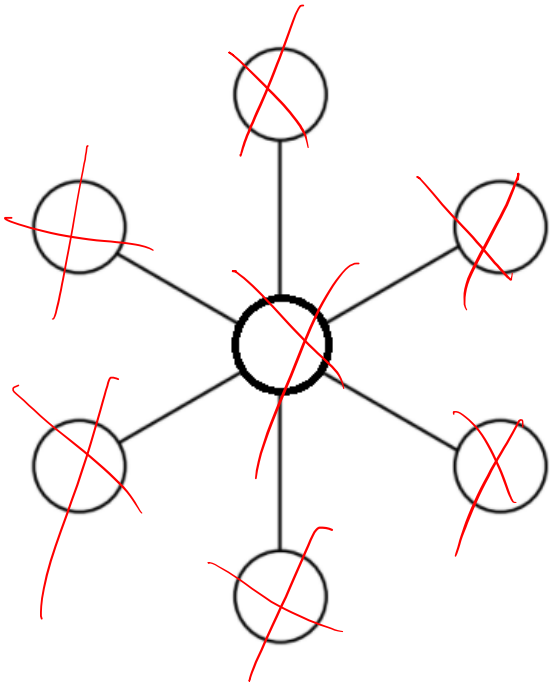
Computational Complexity

- **Treewidth**: one less than the smallest achievable cardinality of the largest elimination clique over all possible elimination orderings.
- Elimination ordering with the lowest complexity has to achieve the treewidth of the graph.
- Unfortunately, the general problem of finding the best elimination ordering that achieves the treewidth is NP-hard.



Computational Complexity

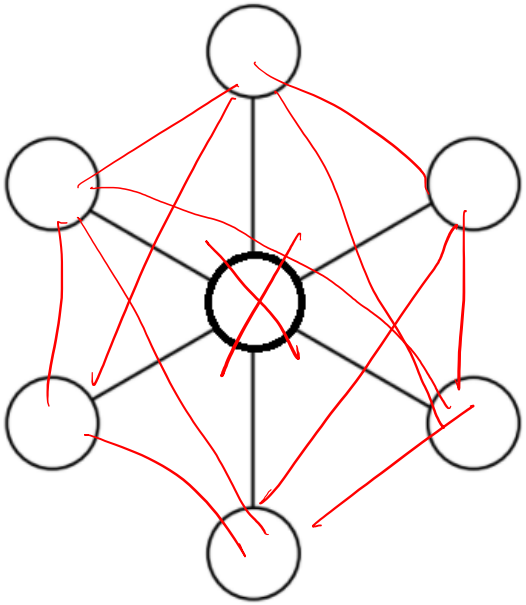
Example:



- A graph whose treewidth is equal to one.
- However, the wrong choice of eliminating the center node would immediately leads to a elimination clique with all the neighbors!

Computational Complexity

Example:



- A graph whose treewidth is equal to one.
- However, the wrong choice of eliminating the center node would immediately leads to a elimination clique with all the neighbors!

How to find the elimination ordering?

- Finding the optimal ordering is **NP-Hard**
- Resort to heuristics:
  - **Min-neighbors:** Fewest number of Dependent Variables
  - **Min-weight:** Minimize product of cardinalities of variables
  - **Min-fill:** Minimize the size of the factor (elimination clique) that will be added.
- **In practice:** these can be very effective.

Computational complexity of probabilistic inference?

- Our goal is to calculate $p(X_F|X_E)$ for arbitrary subsets E and F.
  - This is the general probabilistic inference problem for graphical models (directed or undirected).

- In general: Inference is **NP-hard**
  - Proof via reduction from 3-SAT
  - Difficult in the **worst case**.
    - Even optimal order can lead to exponential computation
    - "Compact" Graphical Models don't necessarily lead to easy inference.

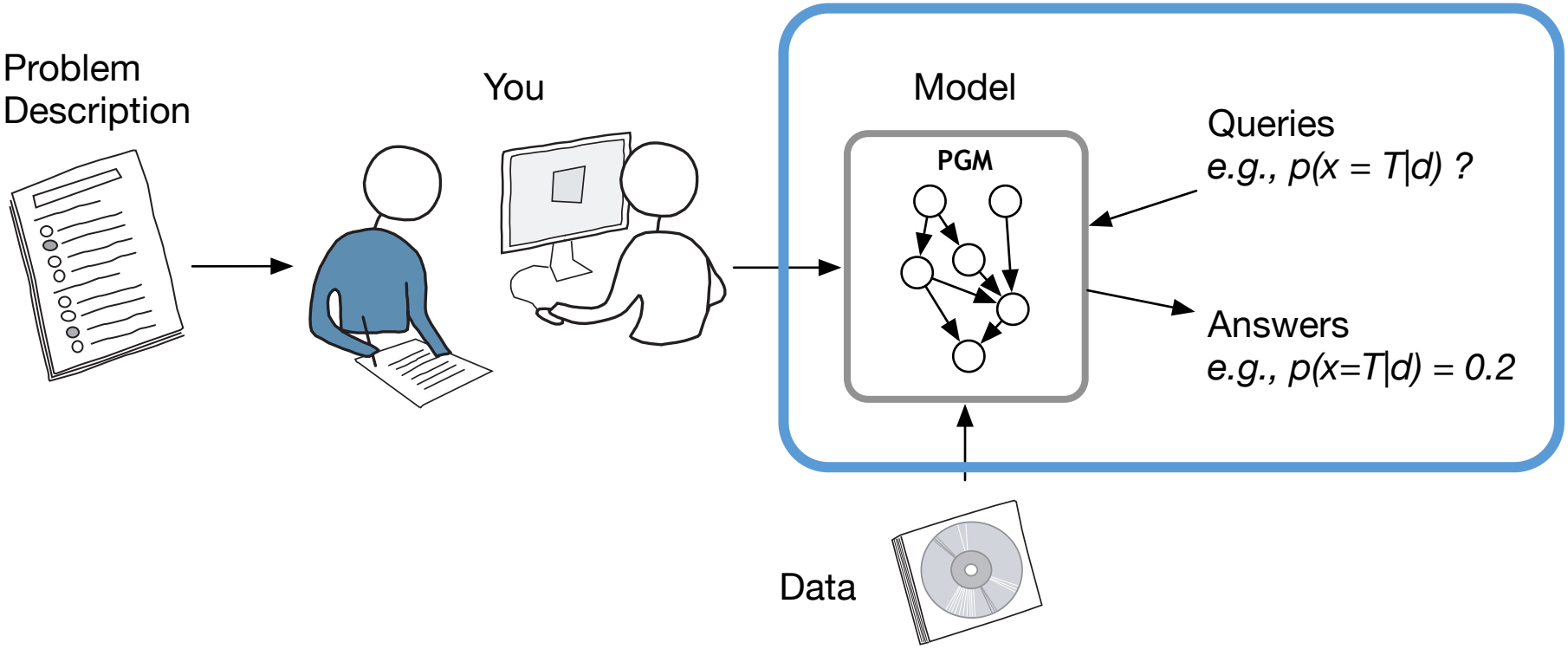
NUS National University of Singapore | School of Computing

Sum Product Algorithm or Belief Propagation

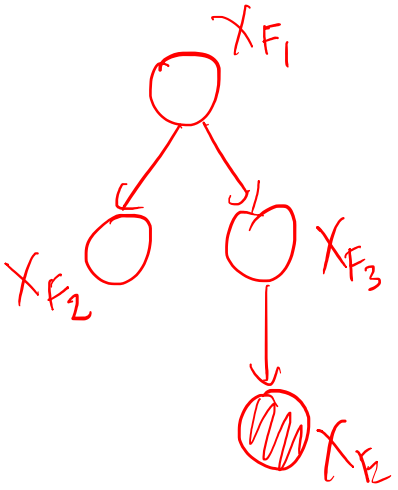
Trees and Message Passing

CS5340 in a nutshell

CS5340 is about how to "**represent**" and "**reason**" with uncertainty in a computer.



Single Node Queries



$p(X_{F_1} | X_E)$

Variable Elimination.

$p(X_{F_2} | X_E)$

$p(X_{F_3} | X_E)$

Limitation of Variable Elimination

Limitation:

- We have to re-run the variable elimination algorithm with every new query node.

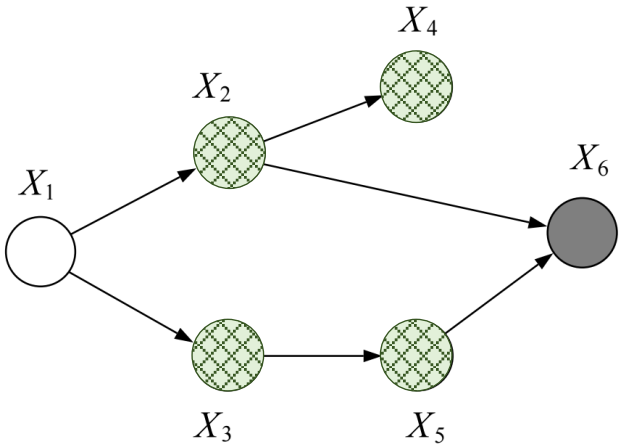
Solution:

- The sum-product or belief propagation algorithm allows us to compute all single-node marginals (conditioned on the evidence) for certain "tree-like" graphs in a single run.

Recall: Messages in Variable Elimination

Conditional probability:

$$p(x_1|\bar{x}_6) = \frac{p(x_1, \bar{x}_6)}{p(\bar{x}_6)}$$



Marginal probability:

$$p(x_1, \bar{x}_6) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)p(x_5 \mid x_3)p(\bar{x}_6 \mid x_2, x_5)$$

$$= p(x_1)\sum_{x_2} p(x_2 \mid x_1)\sum_{x_3} p(x_3 \mid x_1)\sum_{x_4} p(x_4 \mid x_2)\sum_{x_5} p(x_5 \mid x_3)p(\bar{x}_6 \mid x_2, x_5)$$

$m_5(x_2, x_3)$ — eliminate X_5

X_2	X_3	$M_5(X_2, X_3)$
0	0	
0	1	
1	0	
1	1	

- Summands can be pushed in due to the distributive law.
- $m_i(x_{S_i})$ denote the expression from performing Σ_{x_i}, where X_{S_i} are the variables, other than X_i, that appear in the summand.

Key idea: Reuse Messages

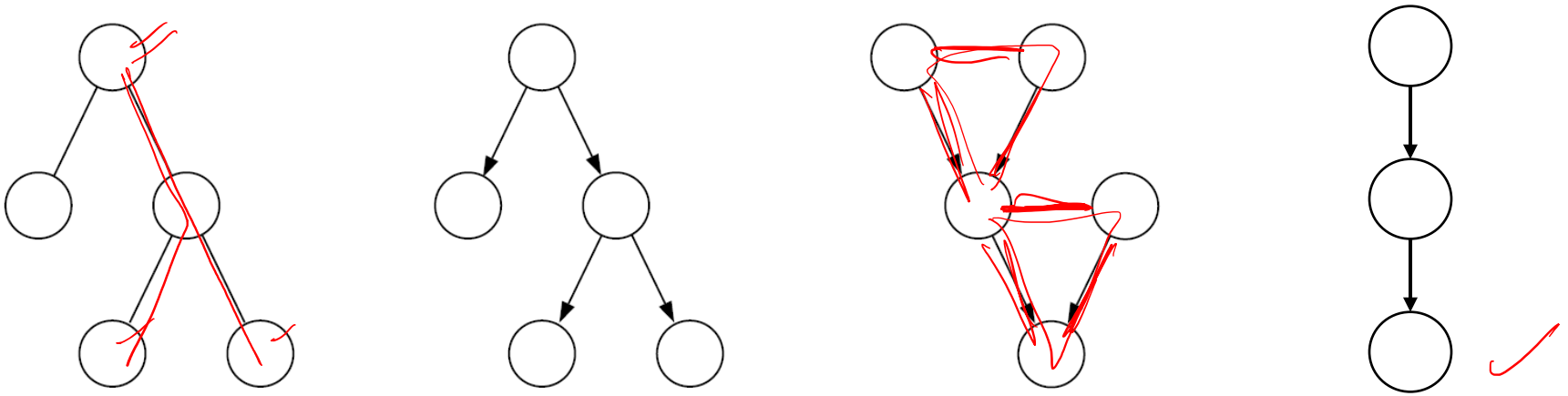
- We want to obtain all of the marginals in the tree.

Key idea:

Messages can be "reused"!

We'll learn how to reuse messages to perform efficient inference.

"Tree-Like" Graphs



a) Undirected tree: without any loop (only one path between any two pair of nodes)

b) Directed tree: only 1 single parent for every node, moralizations lead to an undirected tree.

c) Polytree: nodes with more than 1 parent. Not a directed tree, moralizations lead to loops.

d) Chain: this is also a directed tree (more on chains when we look at Hidden Markov Models).

Source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Parameterization

Undirected Trees:

- The cliques are single and pairs of nodes, thus the joint probability is:

$$p(x) = \frac{1}{Z}\left(\prod_{i \in \mathcal{V}} \psi(x_i) \prod_{(i,j) \in \mathcal{E}} \psi(x_i, x_j)\right)$$

where $\mathcal{V}$ and $\mathcal{E}$ are the nodes and edges of a tree $\mathcal{T}(\mathcal{V}, \mathcal{E})$.

Parameterization

Directed Trees:

- Joint probability is given by:

$$p(x) = p(x_r) \prod_{(i,j)\in\mathcal{E}} p(x_j \mid x_i)$$

where

- $p(x_r)$: marginal probability at the root, and
- $\{p(x_j|x_i)\}$: conditional probabilities at all other nodes.
- (i,j) is a directed edge such that i is the parent of j.

Parameterization

Directed Tree → Undirected Tree:

We define

$$\begin{aligned} \psi(x_r) &= p(x_r) \\ \psi(x_i, x_j) &= p(x_j \mid x_i), \end{aligned}$$

for i the parent of j, and define all other singleton potentials $\psi(x_i) = 1 \;\; \forall \;\; i \neq r$.

The partition function $Z = 1$.

We will not make any distinction between directed and undirected trees since they are similar *in this context*.

Conditioning

- To capture conditioning i.e. $p(x\,|\bar{x}_E)$ for some subset E, let us define the local potentials as:

$$\psi_i^E(x_i) \triangleq \begin{cases} \psi_i(x_i)\delta(x_i, \bar{x}_i) & i \in E \\ \psi_i(x_i) & i \notin E \end{cases}$$

where $\delta(x_i, \bar{x}_i)$ is the "evidence potential" defined earlier.

Conditioning

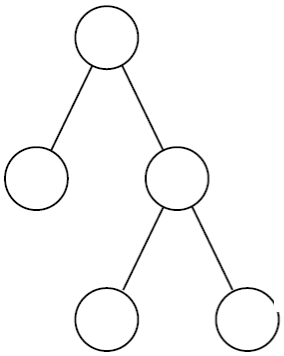
- Making use of the joint probability of undirected trees, we get the conditional probability:

$$p(x \mid \bar{x}_E) = \frac{1}{Z^E} \left(\prod_{i \in V} \psi^E(x_i) \prod_{(i,j) \in \mathcal{E}} \psi(x_i, x_j) \right)$$

where the original Z vanishes and

$$Z^E = \sum_x \left(\prod_{i \in V} \psi^E(x_i) \prod_{(i,j) \in \mathcal{E}} \psi(x_i, x_j) \right)$$

From Elimination to Message Passing



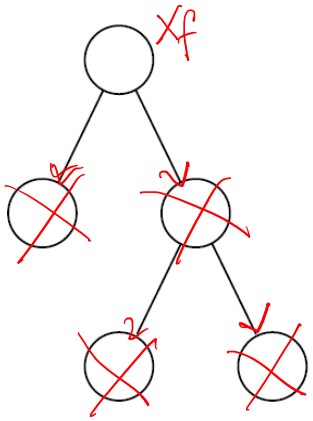
UPDATE($\mathcal{G}$)
  for each i in I
    find all potentials from the active list that reference x_i and remove them from the active list
    let $\phi_i(x_{T_i})$ denote the product of these potentials
    let $m_i(x_{S_i}) = \sum_{x_i} \phi_i(x_{T_i})$
    place $m_i(x_{S_i})$ on the active list
  end

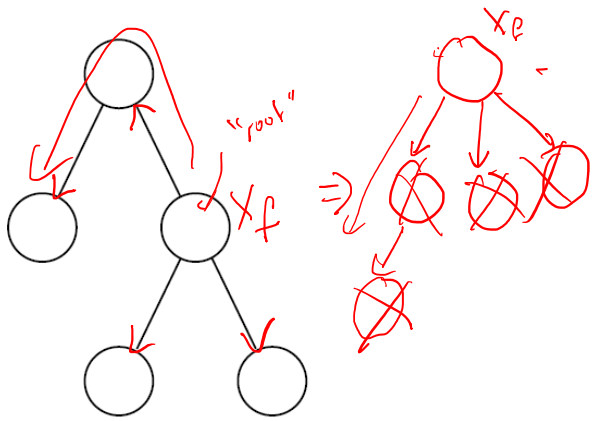
- **Question**: What is a good elimination ordering when graph is a tree?

- **Answer**: Elimination orderings that arise from a depth-first traversal of the tree.

Depth-First Tree Traversal

- Take advantage of the recursive structure of a tree to specify an elimination ordering.
- Treat query node X_f as the root.
- View the tree as a directed tree by directing all edges of the tree to point away from X_f.
- Elimination proceeds inward from the leaves, with treewidth equals to one! (**Why?**)



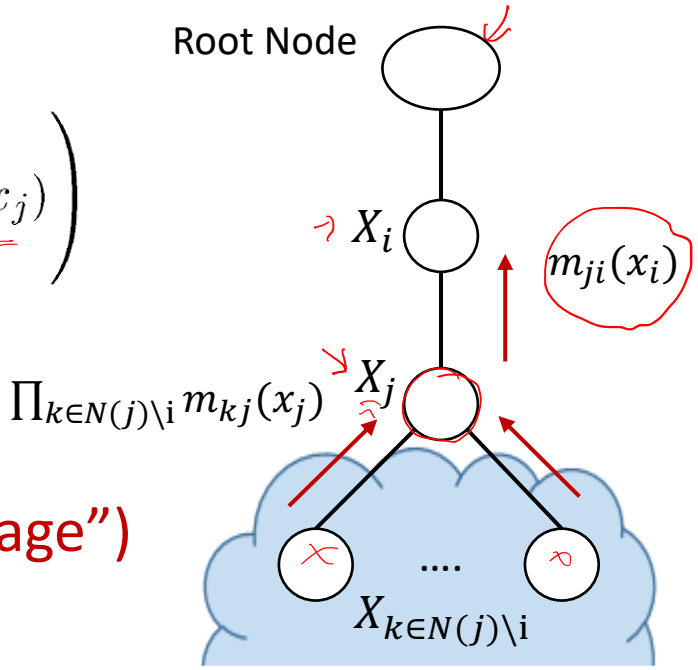


Intermediate Factor: "Message"

- Consider nodes X_i and X_j that are neighbors in the tree, where X_i is closer to the root node.

- To eliminate X_j, we take the product over all potentials that reference X_j and sum over X_j:

$$m_{ji}(x_i) = \sum_{x_j} \left(\psi^E(x_j)\psi(x_i, x_j) \prod_{k \in \mathcal{N}(j) \setminus i} m_{kj}(x_j) \right)$$

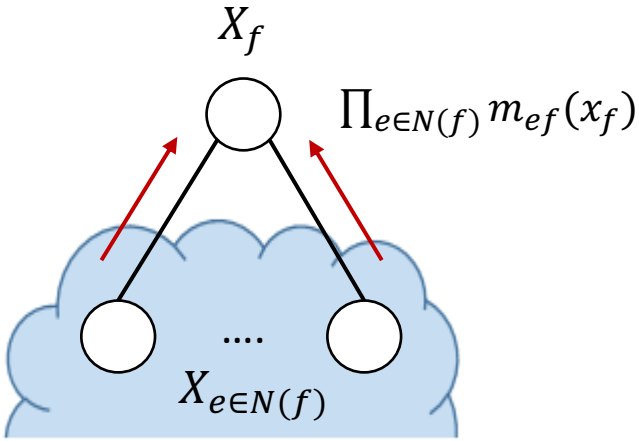


- This is the intermediate factor ("message") that X_j sends to X_i.

Message at Final Node (Root)

- All other nodes have been eliminated when we arrive at X_f.
- Thus messages $m_{ef}(x_f)$ have been computed for each of the neighbours $e \in N(f)$.
- We write the marginal of X_F as:

$$p(x_f \mid \bar{x}_E) \propto \psi^E(x_f) \prod_{e \in \mathcal{N}(f)} m_{ef}(x_f)$$



Messages

$$m_{ji}(x_i) = \sum_{x_j} \left(\psi^E(x_j)\psi(x_i, x_j) \prod_{k \in \mathcal{N}(j) \setminus i} m_{kj}(x_j) \right)$$

$$p(x_f \mid \bar{x}_E) \propto \psi^E(x_f) \prod_{e \in \mathcal{N}(f)} m_{ef}(x_f)$$

- It turns out that these messages are sufficient for obtaining not only a single marginal, but also obtaining all of the marginals in the tree!

Reuse Messages

- Obtain all of the marginals in the tree.

Key idea:

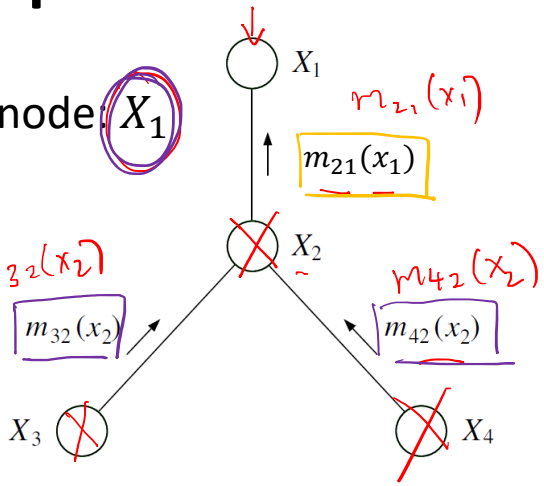
Messages can be "reused"!

We can achieve the effect of computing over all possible elimination orderings (huge number) by computing all possible messages (small number).

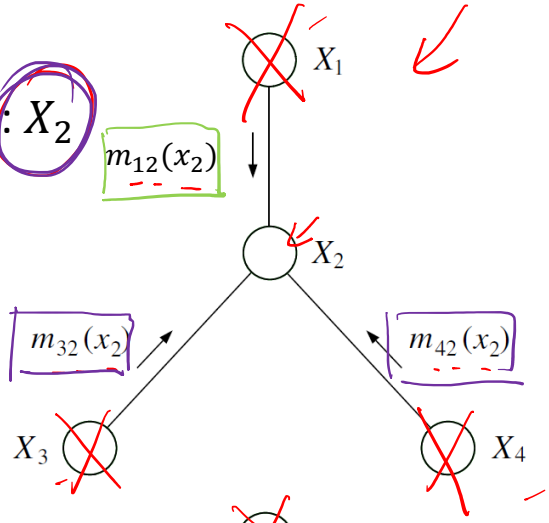
Sum-Product Algorithm

Example:

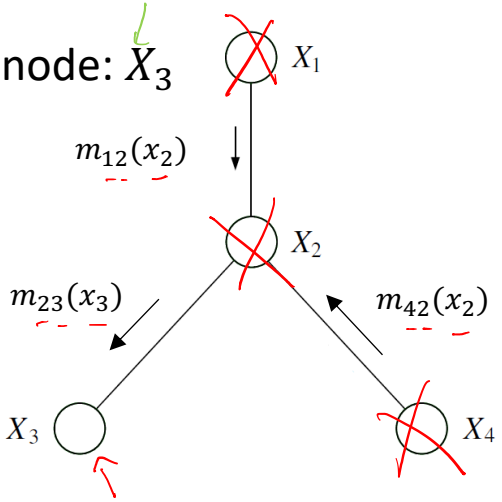
Query node: X_1



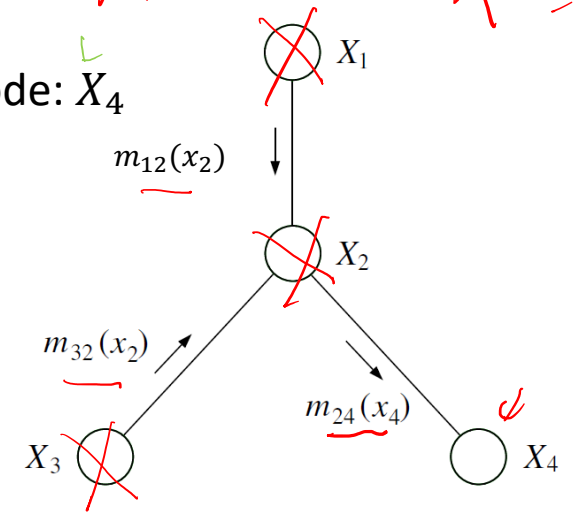
Query node: X_2



Query node: X_3



Query node: X_4

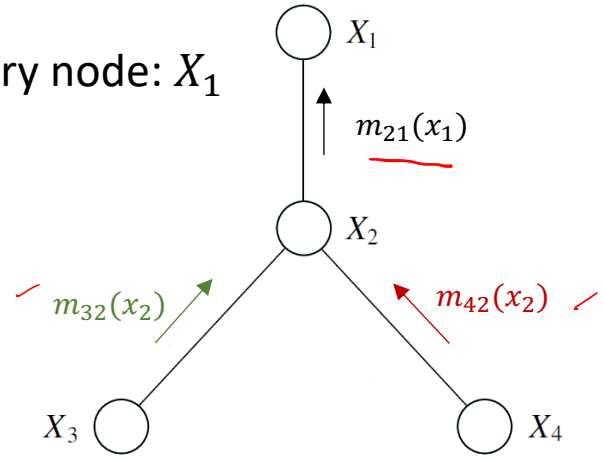


Source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

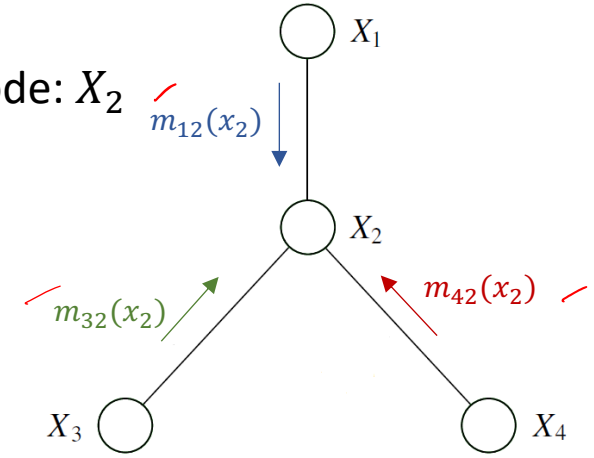
Sum-Product Algorithm

Example: Messages are "reused"!

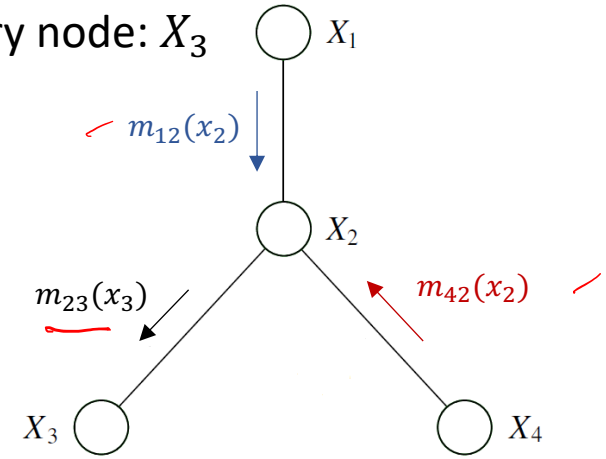
Query node: X_1



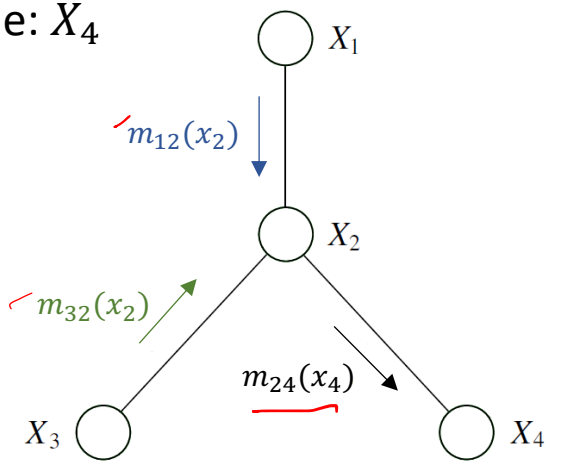
Query node: X_2



Query node: X_3



Query node: X_4

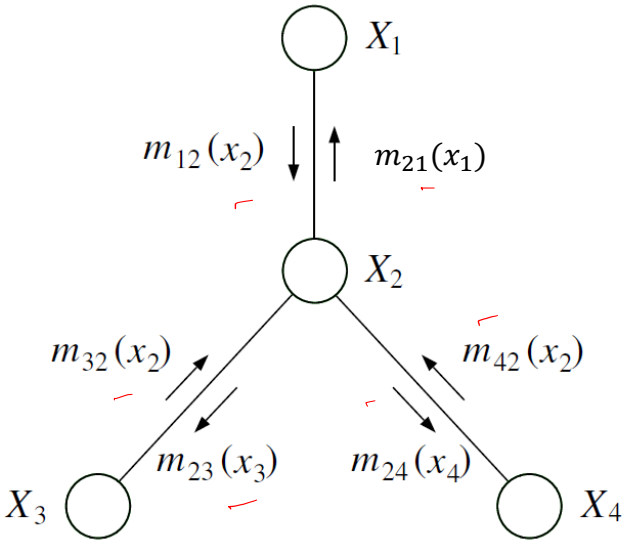


Source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Sum-Product Algorithm

Example:

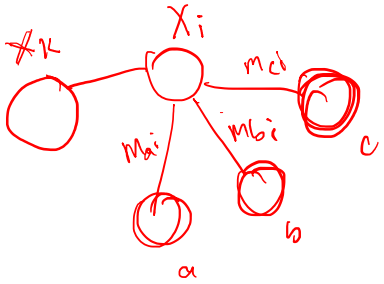
- All of the messages needed to compute all singleton marginals.
- The sum-product algorithm is an algorithm to compute all messages in a tree, and hence all singleton marginals efficiently!



Source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Message-Passing Protocol

A node can send a message to a neighboring node when (and only when) it has received messages from all of its other neighbors.



Sum-Product Algorithm

- Two phases:

1. Messages flow inward from leaves toward the root.

2. Initiated once all incoming messages have been received by the root node – messages flow outward from root toward the leaves.

Sum-Product Algorithm

SUM-PRODUCT($\mathcal{T}$, E) // main steps of the "Sum-Product Algorithm"
- EVIDENCE(E)
- f = CHOOSEROOT($\mathcal{V}$)
- **for** $e \in \mathcal{N}(f)$
  - COLLECT(f, e)
- **for** $e \in \mathcal{N}(f)$
  - DISTRIBUTE(f, e)
- **for** $i \in \mathcal{V}$
  - COMPUTEMARGINAL(i)

EVIDENCE(E) // add **evidence potentials** (convert conditioning into marginalization)
- **for** $i \in E$
  - $\psi^E(x_i) = \psi(x_i)\delta(x_i, \bar{x}_i)$
- **for** $i \notin E$
  - $\psi^E(x_i) = \psi(x_i)$

COLLECT(i, j) // messages flow **inward** from leaves toward the root
- **for** $k \in \mathcal{N}(j) \backslash i$
  - COLLECT(j, k)
- SENDMESSAGE(j, i)

DISTRIBUTE(i, j) // messages flow **outward** from root toward the leaves
- SENDMESSAGE(i, j)
- **for** $k \in \mathcal{N}(j) \backslash i$
  - DISTRIBUTE(j, k)

SENDMESSAGE(j, i) // intermediate factors (messages)

$$m_{ji}(x_i) = \sum_{x_j} (\psi^E(x_j)\psi(x_i, x_j) \prod_{k \in \mathcal{N}(j) \backslash i} m_{kj}(x_j))$$

COMPUTEMARGINAL(i) // message to final node

$$p(x_i) \propto \psi^E(x_i) \prod_{j \in \mathcal{N}(i)} m_{ji}(x_i)$$

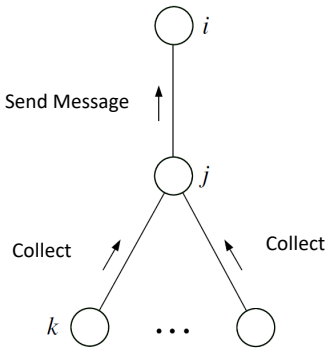
Sum-Product Algorithm

SUM-PRODUCT($\mathcal{T}$, E) // main steps of the "Sum-Product Algorithm"
  EVIDENCE(E)
  f = CHOOSEROOT($\mathcal{V}$)
  for $e \in \mathcal{N}(f)$
    COLLECT(f, e)
  for $e \in \mathcal{N}(f)$
    DISTRIBUTE(f, e)
  for $i \in \mathcal{V}$
    COMPUTEMARGINAL(i)

EVIDENCE(E) // add **evidence potentials** (convert conditioning into marginalization)
  for $i \in E$
    $\psi^E(x_i) = \psi(x_i)\delta(x_i, \bar{x}_i)$
  for $i \notin E$
    $\psi^E(x_i) = \psi(x_i)$

COLLECT(i, j) // messages flow **inward** from leaves toward the root
  for $k \in \mathcal{N}(j) \backslash i$
    COLLECT(j, k)
  SENDMESSAGE(j, i)

DISTRIBUTE(i, j) // messages flow **outward** from root toward the leaves
  SENDMESSAGE(i, j)
  for $k \in \mathcal{N}(j) \backslash i$
    DISTRIBUTE(j, k)



SENDMESSAGE(j, i) // intermediate factors (messages)

$$m_{ji}(x_i) = \sum_{x_j} (\psi^E(x_j)\psi(x_i, x_j) \prod_{k \in \mathcal{N}(j) \backslash i} m_{kj}(x_j))$$

COMPUTEMARGINAL(i) // message to final node

$$p(x_i) \propto \psi^E(x_i) \prod_{j \in \mathcal{N}(i)} m_{ji}(x_i)$$

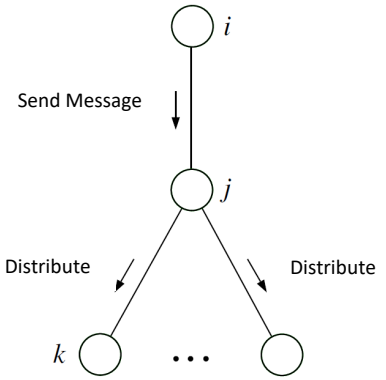
Sum-Product Algorithm

SUM-PRODUCT($\mathcal{T}$, E) // main steps of the "Sum-Product Algorithm"
- EVIDENCE(E)
- f = CHOOSEROOT($\mathcal{V}$)
- **for** $e \in \mathcal{N}(f)$
  - COLLECT(f, e)
- **for** $e \in \mathcal{N}(f)$
  - DISTRIBUTE(f, e)
- **for** $i \in \mathcal{V}$
  - COMPUTEMARGINAL(i)

EVIDENCE(E) // add **evidence potentials** (convert conditioning into marginalization)
- **for** $i \in E$
  - $\psi^E(x_i) = \psi(x_i)\delta(x_i, \bar{x}_i)$
- **for** $i \notin E$
  - $\psi^E(x_i) = \psi(x_i)$

COLLECT(i, j) // messages flow **inward** from leaves toward the root
- **for** $k \in \mathcal{N}(j) \backslash i$
  - COLLECT(j, k)
- SENDMESSAGE(j, i)

DISTRIBUTE(i, j) // messages flow **outward** from root toward the leaves
- SENDMESSAGE(i, j)
- **for** $k \in \mathcal{N}(j) \backslash i$
  - DISTRIBUTE(j, k)



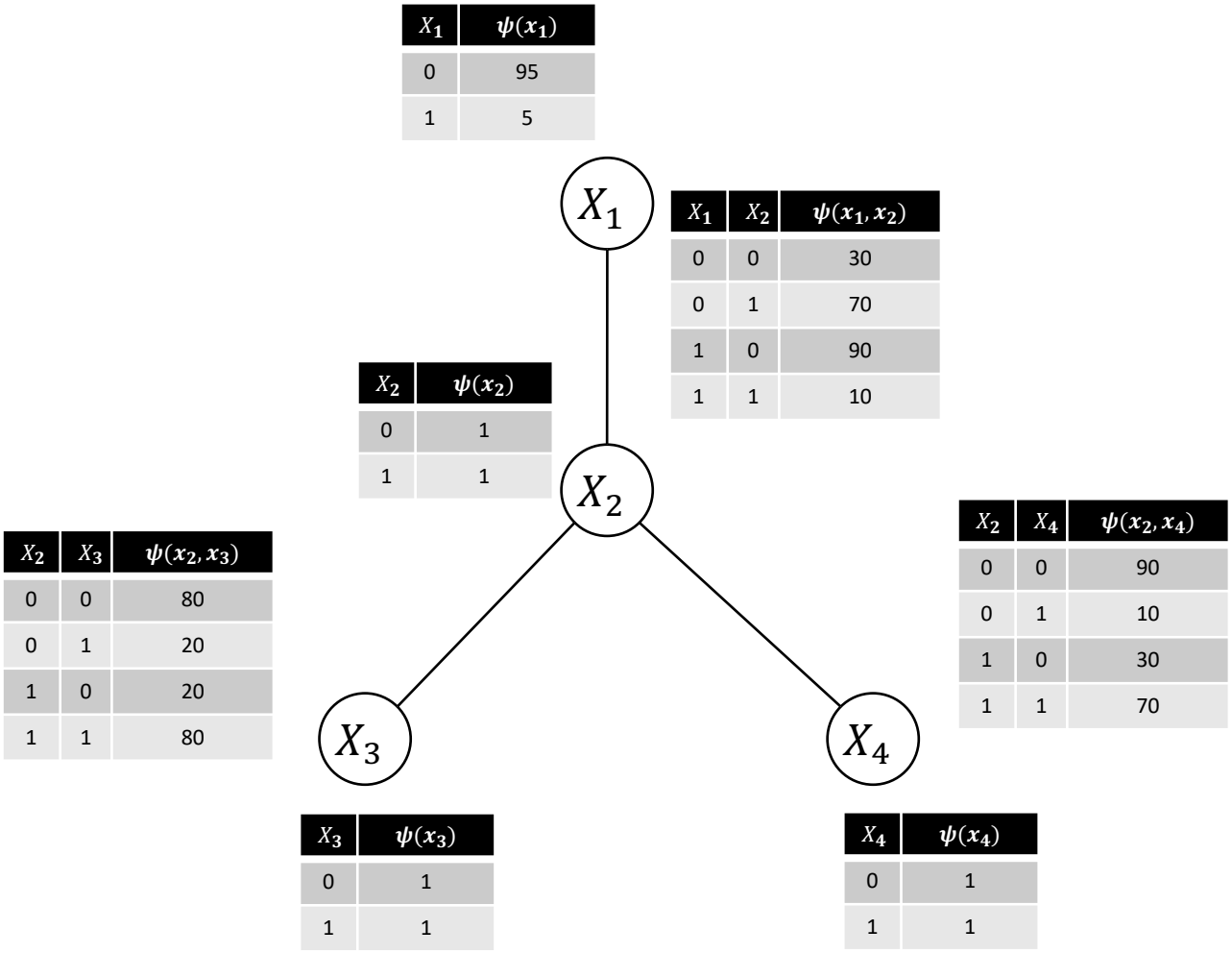
SENDMESSAGE(j, i) // intermediate factors (messages)

$$m_{ji}(x_i) = \sum_{x_j} (\psi^E(x_j)\psi(x_i, x_j) \prod_{k \in \mathcal{N}(j) \backslash i} m_{kj}(x_j))$$

COMPUTEMARGINAL(i) // message to final node

$$p(x_i) \propto \psi^E(x_i) \prod_{j \in \mathcal{N}(i)} m_{ji}(x_i)$$

Exercise: Human Belief Propagation



X_1	$\psi(x_1)$
0	95
1	5

X_1	X_2	$\psi(x_1, x_2)$
0	0	30
0	1	70
1	0	90
1	1	10

X_2	$\psi(x_2)$
0	1
1	1

X_2	X_3	$\psi(x_2, x_3)$
0	0	80
0	1	20
1	0	20
1	1	80

X_2	X_4	$\psi(x_2, x_4)$
0	0	90
0	1	10
1	0	30
1	1	70

X_3	$\psi(x_3)$
0	1
1	1

X_4	$\psi(x_4)$
0	1
1	1

SUM-PRODUCT($\mathcal{T}$, E)
  EVIDENCE(E)
  f = CHOOSEROOT($\mathcal{V}$)
  for $e \in \mathcal{N}(f)$
    COLLECT(f, e)
  for $e \in \mathcal{N}(f)$
    DISTRIBUTE(f, e)
  for $i \in \mathcal{V}$
    COMPUTEMARGINAL(i)

EVIDENCE(E)
  for $i \in E$
    $\psi^E(x_i) = \psi(x_i)\delta(x_i, \bar{x}_i)$
  for $i \notin E$
    $\psi^E(x_i) = \psi(x_i)$

COLLECT(i, j)
  for $k \in \mathcal{N}(j) \backslash i$
    COLLECT(j, k)
  SENDMESSAGE(j, i)

DISTRIBUTE(i, j)
  SENDMESSAGE(i, j)
  for $k \in \mathcal{N}(j) \backslash i$
    DISTRIBUTE(j, k)

SENDMESSAGE(j, i)

$$m_{ji}(x_i) = \sum_{x_j} (\psi^E(x_j)\psi(x_i, x_j) \prod_{k \in \mathcal{N}(j) \backslash i} m_{kj}(x_j))$$

COMPUTEMARGINAL(i)

$$p(x_i) \propto \psi^E(x_i) \prod_{j \in \mathcal{N}(i)} m_{ji}(x_i)$$

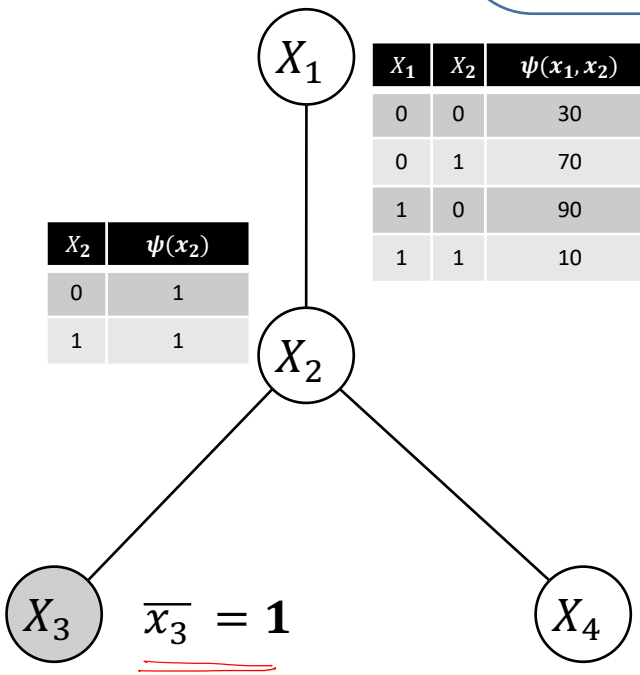
X_1	$\psi(x_1)$
0	95
1	5

X_1	X_2	$\psi(x_1, x_2)$
0	0	30
0	1	70
1	0	90
1	1	10

X_2	$\psi(x_2)$
0	1
1	1

X_2	X_3	$\psi(x_2, x_3)$
0	0	80
0	1	20
1	0	20
1	1	80

X_2	X_4	$\psi(x_2, x_4)$
0	0	90
0	1	10
1	0	30
1	1	70



X_3	$\psi(x_3)$
0	1
1	1

X_4	$\psi(x_4)$
0	1
1	1

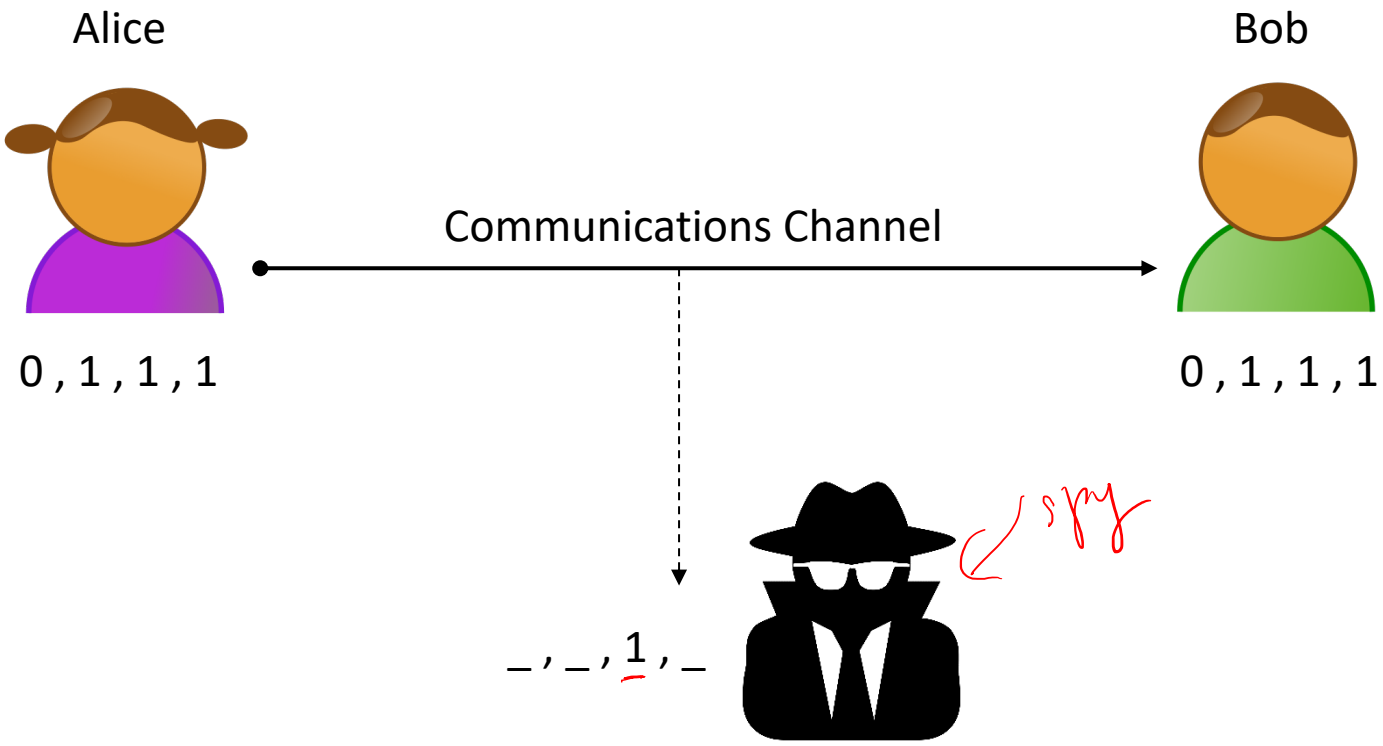
NUS National University of Singapore | School of Computing

MAP Configurations

Max-Product Algorithm

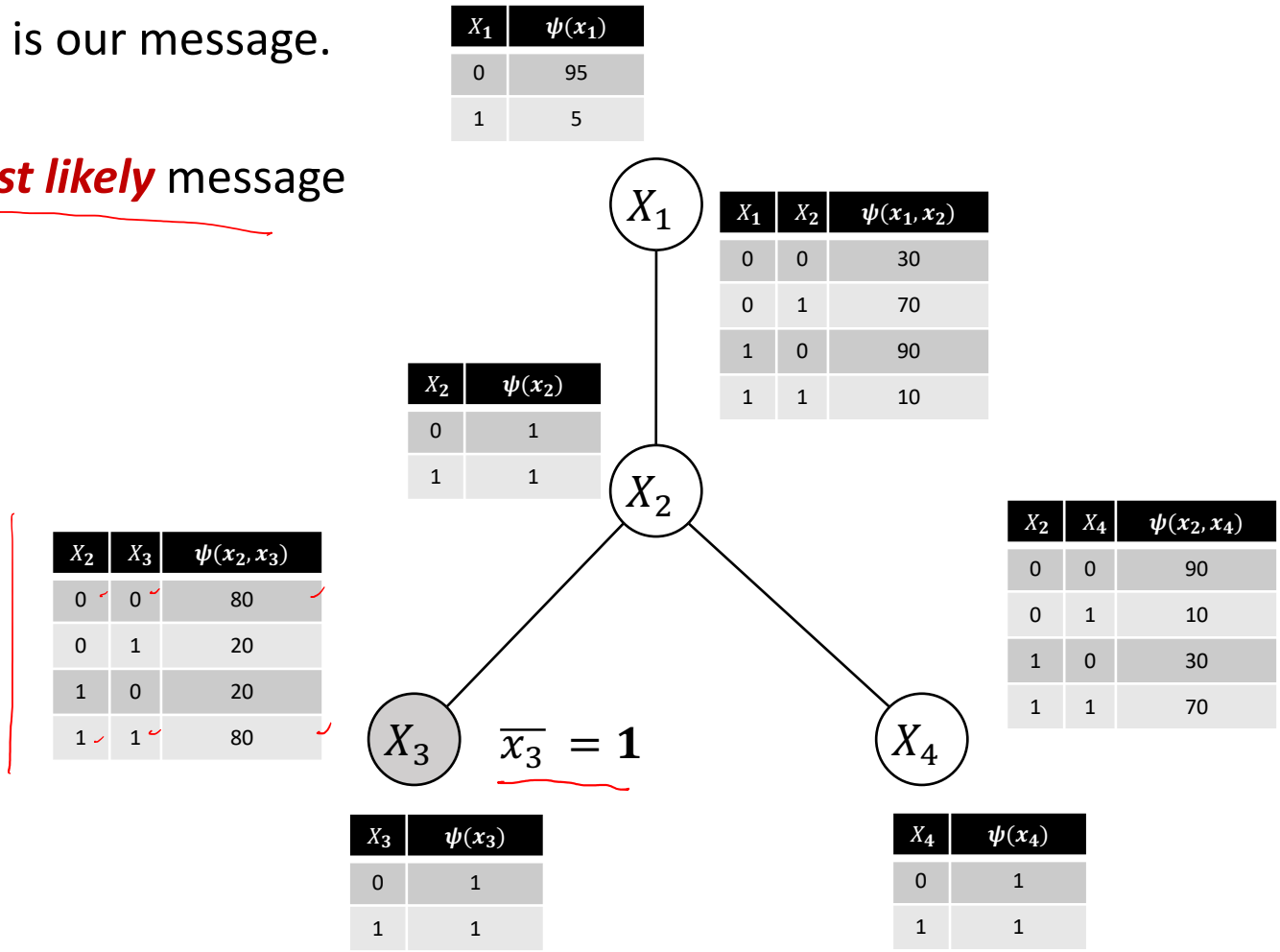
Problem: Corrupted Message

Problem: Corrupted Message



Problem: Corrupted Message

- $\{X_1, X_2, X_3, X_4\}$ is our message. We see _ _ 1 _.
- What is the ***most likely*** message that was sent?



X_1	$\psi(x_1)$
0	95
1	5

X_1	X_2	$\psi(x_1, x_2)$
0	0	30
0	1	70
1	0	90
1	1	10

X_2	$\psi(x_2)$
0	1
1	1

X_2	X_3	$\psi(x_2, x_3)$
0	0	80
0	1	20
1	0	20
1	1	80

X_2	X_4	$\psi(x_2, x_4)$
0	0	90
0	1	10
1	0	30
1	1	70

X_3	$\psi(x_3)$
0	1
1	1

X_4	$\psi(x_4)$
0	1
1	1

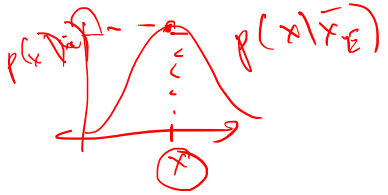
Maximum a Posterior Probabilities

- Two aspects to MAP:
    1. Finding the maximal probability.
    2. Finding a configuration that achieves the maximal probability.

- Marginalization problem: summing over all configurations of sets of random variables.

- Maximum a Posterior (MAP) problem: maximizing over all sets of random variables.

Maximal Probability



- Given a probability distribution $p(x \mid \bar{x}_E)$, the maximum a posterior probability is given by:

$$\max_x p(x \mid \bar{x}_E) = \max_x \frac{p(x, \bar{x}_E)}{p(\bar{x}_E)}$$

Can be removed since we are finding max over X.

$$= \max_x p(x, \bar{x}_E)$$

$$= \max_x p(x, x_E)\delta(x_E, \bar{x}_E)$$

$$= \max_x p(x)^E$$

where

- $\bar{X}_E$ is the set of observed variables, and
- $p(x)^E$ is the unnormalized representation of the conditional probability $p(x|\bar{x}_E)$.

Potential Algorithm

- Can we solve the MAP problem by computing the:

  1. marginal probability for each variable, and
  2. assignment of each variable that maximizes its individual marginal?

Image Source: “An introduction to probabilistic graphical models”, Michael I. Jordan, 2002.

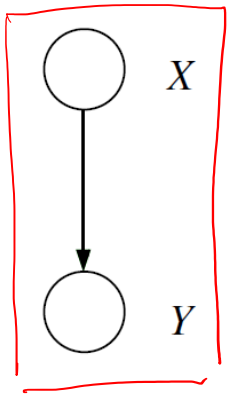
Potential Algorithm

- Can we solve the MAP problem by computing the:

  1. marginal probability for each variable, and
  2. assignment of each variable that maximizes its individual marginal?

Illustration:

NO!!!



1	.6
2	.2
3	.2

$p(x)$

x \ y	1	2	3
1	0	.6	.4
2	1	0	0
3	1	0	0

$p(y \mid x)$

1	2	3
.4	.36	.24

$p(y)$

Marginal probabilities:

$$\max_x p(x) = p(x = 1) = 0.6$$

$$\max_y p(y) = p(y = 1) = 0.4$$

But

$$\max_{x,y} p(x, y) = p(x = 1, y = 2) = 0.36$$

Image Source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Potential Algorithm

- Can we solve the MAP problem by computing the:

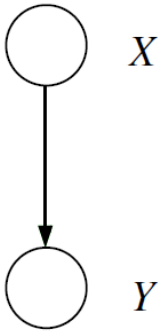
1. marginal probability for each variable, and
2. assignme…zes its individual marginal?

…nal probabilities:

$p(x) = p(x = 1) = 0.6$

$p(y) = p(y = 1) = 0.4$

Illustration:



1	.6
2	.2
3	.2

$p(x)$

x			
2	1	0	0
3	1	0	0

$p(y \mid x)$

.4	.36	.24

$p(y)$

But

$$\max_{x,y} p(x, y) = p(x = 1, y = 2) = 0.36$$

Image Source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

From Marginal to MAP Algorithms

- Distributive law of multiplication over addition:

$$a.b_1 + a.b_2 + \ \ldots + a.b_n = a.(b_1 + b_2 + \cdots + b_n)$$

- Plays a key role in elimination and sum-product algorithms:

$$p(x_1, x_2, \ldots, x_5) = \sum_{x_6} p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2)p(x_5|x_3)p(x_6|x_2, x_5)$$

$$= \sum_{x_6} a.\, p(x_6|x_2, x_5) \qquad (a)$$

$$= a.\, p(x_6 = 0|x_2, x_5) + \cdots + a.\, p(x_6 = k|x_2, x_5)$$

$$= a.\left(p(x_6 = 0|x_2, x_5) + \cdots + p(x_6 = k|x_2, x_5)\right) \quad \text{(Distributive law)}$$

$$= p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2)p(x_5|x_3)\sum_{x_6} p(x_6|x_2, x_5)$$

From Marginal to MAP Algorithms

- Distributive law applies to the "max" operator too*!

$$\max(a.b_1, a.b_2, \ldots, a.b_n) = a.\max(b_1, b_2, \ldots, b_n)$$

- Turn the elimination algorithm into the "MAP-elimination" algorithm by replacing the "sum" with "max" operator:

$$\max_{x_6} p(x_1, x_2, \ldots, x_6) = \max_{x_6} p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2)p(x_5|x_3)p(x_6|x_2, x_5)$$

"max" operator can be pushed in!

$$= \underbrace{p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2)p(x_5|x_3)}_{\text{independent of } x_6}\max_{x_6} p(x_6|x_2, x_5)$$

- Becomes the "max-product" algorithm.

*for $a \geq 0$ which always applies here

MAP-Elimination Algorithm

MAP-ELIMINATE($\mathcal{G}, E$) // main steps of the "MAP-Elimination Algorithm"

1. INITIALIZE($\mathcal{G}$)
2. EVIDENCE(E)
3. UPDATE($\mathcal{G}$)
4. MAXIMUM

1. INITIALIZE($\mathcal{G}$) // choose elimination ordering, and add local condition probabilities in **active list**

   choose an ordering I // **same** as the "variable elimination algorithm"

   for each node X_i in $\mathcal{V}$

   place $p(x_i \mid x_{\pi_i})$ on the active list

2. EVIDENCE(E) // add evidence potentials in **active list**

   // **same** as the "variable elimination algorithm"

   for each i in E

   place $\delta(x_i, \bar{x}_i)$ on the active list

3. UPDATE($\mathcal{G}$) // **maximization**, and update active list

   for each i in I

   find all potentials from the active list that reference x_i and remove them from the active list

   let $\phi_i^{\max}(x_{T_i})$ denote the product of these potentials

   let $m_i^{\max}(x_{S_i}) = \max_{x_i} \phi_i^{\max}(x_{T_i})$

   place $m_i^{\max}(x_{S_i})$ on the active list

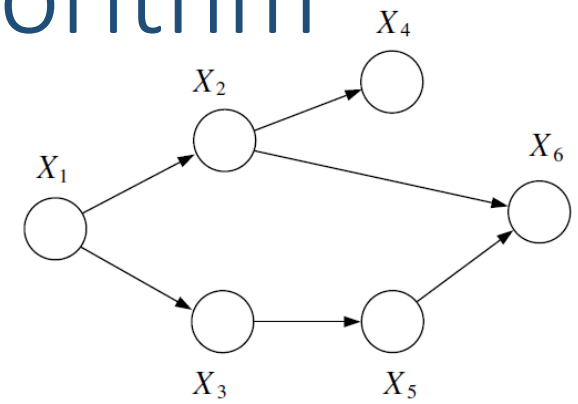
4. MAXIMUM

   $\max_x p^E(x) =$ the scalar value on the active list

Source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

MAP-Elimination Algorithm

Example:



Elimination order: $I = \{6, 5, 4, 3, 2, 1\}$

$$p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2)p(x_5|x_3)p(x_6|x_2, x_5)$$

$$\max_x p(x_1, x_2, x_3, x_4, x_5|\bar{x}_6) = \max_{x_1}\max_{x_2}\max_{x_3}\max_{x_4}\max_{x_5}\max_{x_6}\frac{p(x_1, x_2, x_3, x_4, x_5, \bar{x}_6)}{p(\bar{x}_6)}$$

$$= \max_{x_1}\max_{x_2}\max_{x_3}\max_{x_4}\max_{x_5}\max_{x_6} p(x_1, x_2, x_3, x_4, x_5, \bar{x}_6)$$

$$= \max_{x_1}\max_{x_2}\max_{x_3}\max_{x_4}\max_{x_5}\max_{x_6} p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2)p(x_5|x_3)p(x_6|x_2, x_5)\delta(x_6, \bar{x}_6)$$

$$= \max_{x_1} p(x_1) \max_{x_2} p(x_2|x_1) \max_{x_3} p(x_3|x_1) \max_{x_4} p(x_4|x_2) \max_{x_5} p(x_5|x_3) \max_{x_6} p(x_6|x_2, x_5)\delta(x_6, \bar{x}_6)$$

$m_6^{max}(x_2, x_5)$

$m_5^{max}(x_2, x_3)$

$m_4^{max}(x_2, x_3)$

$m_3^{max}(x_1, x_2)$

$m_2^{max}(x_1)$

$m_1^{max}(x_1)$

Image Source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Maximal Probability Table

Example: Evidence Node

$x_i \in \{0,1\}$ and we observed that $\bar{X}_6 = 1$

X_2	X_5	X_6	$p(x_6 \mid x_2, x_5)$
0	0	0	v_0
0	0	1	v_1
0	1	0	v_2
0	1	1	v_3
1	0	0	v_4
1	0	1	v_5
1	1	0	v_6
1	1	1	v_7

$\bar{X}_6 = 1$

$$m_6^{max}(x_2, x_5) = \max_{x_6} p(x_6 \mid x_2, x_5)\delta(x_6, \bar{x}_6)$$

X_2	X_5	$m_6^{max}(x_2, x_5)$
0	0	v_1
0	1	v_3
1	0	v_5
1	1	v_7

We are taking a 2d slice of the 3d probabilities or potentials!

Maximal Probability Table

Example:

$$\max_{x_5} p(x_5|x_3) \underbrace{\max_{x_6} p(x_6|x_2,x_5)\delta(x_6,\bar{x}_6)}_{m_6^{max}(x_2,x_5)}$$

$$m_5^{max}(x_2,x_3)$$

X_2	X_5	$m_6^{max}(x_2,x_5)$
0	0	v_1
0	1	v_3
1	0	v_5
1	1	v_7

X_3	X_5	$p(x_5\|x_3)$
0	0	b_1
0	1	b_2
1	0	b_3
1	1	b_4

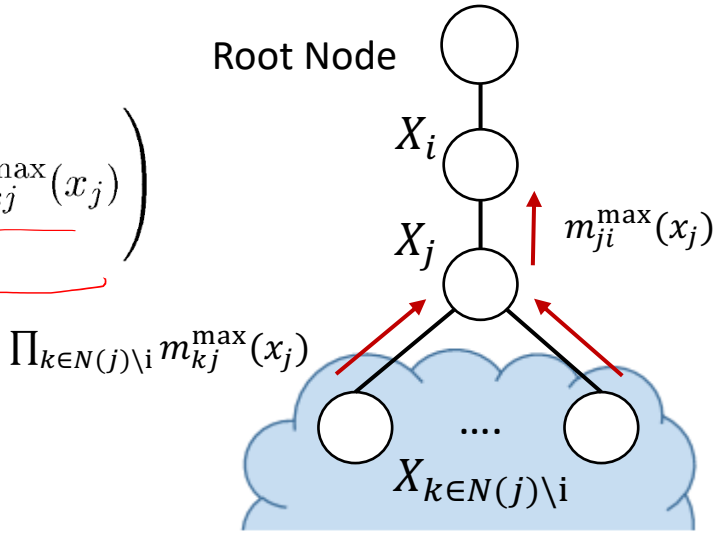
X_2	X_3	$m_5^{max}(x_2,x_3)$
0	0	$\max_{x_5} p(x_5\|x_3=0)m_6^{max}(x_2=0,x_5)$ $= \max\big(p(x_5=0\|x_3=0)m_6^{max}(x_2=0,x_5=0),$ $\quad p(x_5=1\|x_3=0)m_6^{max}(x_2=0,x_5=1)\big)$ $= \max(b_1v_1, b_2v_3)$
0	1	$\max_{x_5} p(x_5\|x_3=1)m_6^{max}(x_2=0,x_5)$ $= \max\big(p(x_5=0\|x_3=1)m_6^{max}(x_2=0,x_5=0),$ $\quad p(x_5=1\|x_3=1)m_6^{max}(x_2=0,x_5=1)\big)$ $= \max(b_3v_1, b_4v_3)$
1	0	$\max_{x_5} p(x_5\|x_3=0)m_6^{max}(x_2=1,x_5)$ $= \max\big(p(x_5=0\|x_3=0)m_6^{max}(x_2=1,x_5=0),$ $\quad p(x_5=1\|x_3=0)m_6^{max}(x_2=1\ x_5=1)\big)$ $= \max(b_1v_5, b_2v_7)$
1	1	$\max_{x_5} p(x_5\|x_3=1)m_6^{max}(x_2=1,x_5)$ $= \max\big(p(x_5=0\|x_3=1)m_6^{max}(x_2=1,x_5=0),$ $\quad p(x_5=1\|x_3=1)m_6^{max}(x_2=1,x_5=1)\big)$ $= \max(b_3v_5, b_4v_7)$

Max-Product Algorithm for Trees

- Find the MAP probability for a tree.
- We choose any node X_f as the root of the tree, and messages are propagated (inward pass) from the leaves to the root.
- Message from X_j to X_i (closer to root):

$$m_{ji}^{\max}(x_i) = \max_{x_j}\left(\psi^E(x_j)\psi(x_i,x_j)\prod_{k\in\mathcal{N}(j)\setminus i} m_{kj}^{\max}(x_j)\right)$$

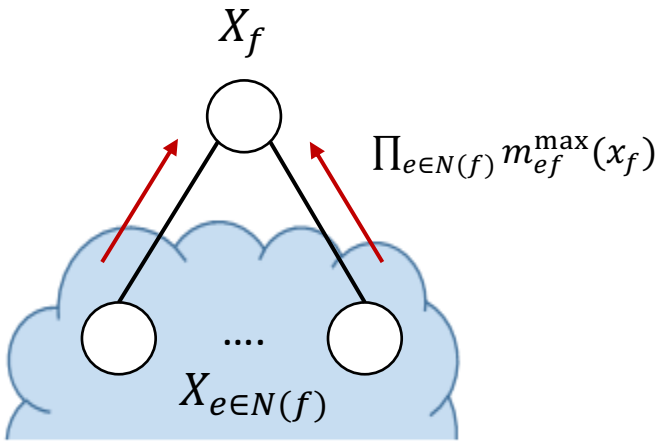
previously, this was $\sum_{x_j}$ in belief propagation



Max-Product Algorithm for Trees

- Collect all messages at the root and compute the MAP probability as:

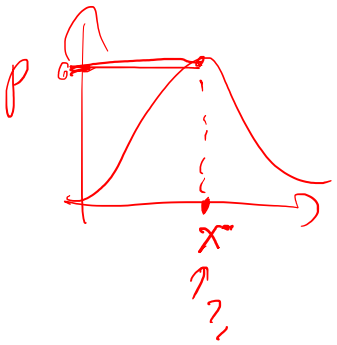
$$\max_{x} p^{E}(x) = \max_{x_i} \left(\psi^{E}(x_f) \prod_{e \in N(f)} m_{ef}^{\max}(x_f) \right)$$



Maximum a Posteriori Configurations

- Find a configuration x^* such that:

$$x^* \in \underset{x}{\operatorname{argmax}}\, p^E(x)$$



- Making use of the messages to the root X_f from the sum-product algorithm, we obtain a value:

$$x_f^* \in \arg\max_{x_f} \left(\psi^E(x_f) \prod_{e \in \mathcal{N}(f)} m_{ef}^{\max}(x_f) \right)$$

that necessarily belongs to a maximum configuration.

Maximum a Posteriori Configurations

- Can we perform an outward pass of the messages from the root to leaves so that we can find the MAP configurations for all x?

Maximum a Posteriori Configurations

- Can we perform an outward pass of the messages from the root to leaves so that we can find the MAP configurations for all x?

NO!

- No guarantee that the values x^* found this way belong to the same maximizing configuration.

Maximum a Posteriori Configurations

Example:

A lattice, or trellis, diagram shows two sets of configurations (black paths) in a chain model that give rise to the same MAP probability.

$x_n \in \{1,2,3\}$

Trellis diagram shows each possible state of the random variable.

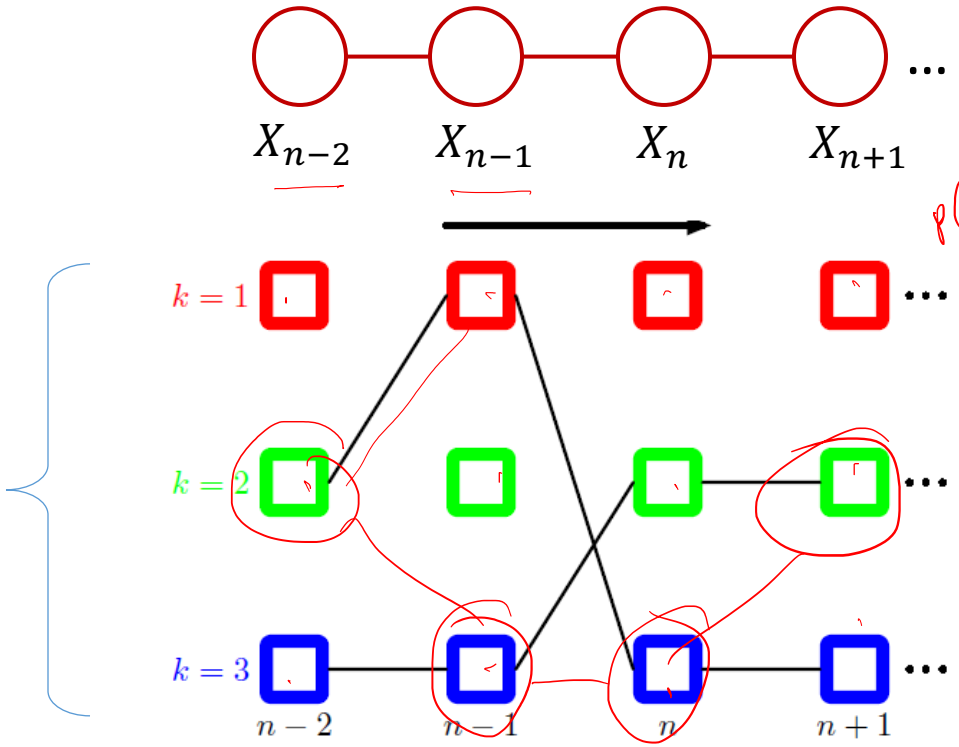


Image source: "Pattern recognition and machine learning", Christopher Bishop

Max-Product Algorithm for Trees

- **Solution**: we also have to record the maximizing values in a table $\delta_{ji}(x_i)$ when a message $m_{ji}^{\max}(x_i)$ is sent from X_j to X_i (closer to root):

$$\delta_{ji}(x_i) \in \arg\max_{x_j} \left(\psi^E(x_j)\psi(x_i, x_j) \prod_{k \in \mathcal{N}(j)\setminus i} m_{kj}^{\max}(x_j) \right)$$

- More precisely, for each X_i, the function $\delta_{ji}(x_i)$ picks out a value of X_j (can be several) that achieves the maximum.

Max-Product Algorithm for Trees

- Having defined $\delta_{ji}(x_i)$ during the inward pass, we use $\delta_{ji}(x_i)$ to define a consistent maximizing configuration during an outward pass:

1. Choose a maximizing value x_f^* at the root X_f.
2. Set $x_e^* = \delta_{ef}(x_f^*)$ for each $e \in N(f)$.
3. Procedure continues outward to the leaves.

Max-Product Algorithm for Trees

MAX-PRODUCT($\mathcal{T}$, E) // main steps of the "MAP-Product Algorithm" for a tree $\mathcal{T}(\mathcal{V}, \mathcal{E})$

EVIDENCE(E)

f = CHOOSEROOT($\mathcal{V}$)

1. **for** $e \in \mathcal{N}(f)$
   COLLECT(f, e)

$MAP = \max_{x_f}(\psi^E(x_f) \prod_{e \in \mathcal{N}(f)} m_{ef}^{\max}(x_f))$ // compute MAP probability at root

$x_f^* = \arg\max_{x_f}(\psi^E(x_f) \prod_{e \in \mathcal{N}(f)} m_{ef}^{\max}(x_f))$ // get MAP configuration at root

2. **for** $e \in \mathcal{N}(f)$
   DISTRIBUTE(f, e)

1. COLLECT(i, j) // inward message passing
   for $k \in \mathcal{N}(j) \backslash i$
   COLLECT(j, k)
   SENDMESSAGE(j, i)

2. DISTRIBUTE(i, j) // outward message passing
   SETVALUE(i, j)
   for $k \in \mathcal{N}(j) \backslash i$
   DISTRIBUTE(j, k)

SENDMESSAGE(j, i)

$m_{ji}^{\max}(x_i) = \max_{x_j}(\psi^E(x_j)\psi(x_i, x_j) \prod_{k \in \mathcal{N}(j) \backslash i} m_{kj}^{\max}(x_j))$ // compute MAP probability message

$\delta_{ji}(x_i) \in \arg\max_{x_j}(\psi^E(x_j)\psi(x_i, x_j) \prod_{k \in \mathcal{N}(j) \backslash i} m_{kj}^{\max}(x_j))$ // get MAP configurations

SETVALUE(i, j) // get MAP configuration in outward pass

$x_j^* = \delta_{ji}(x_i^*)$

Max-Product Algorithm for Trees

Example: $x_i \in \{0,1\}$

Inward message passing

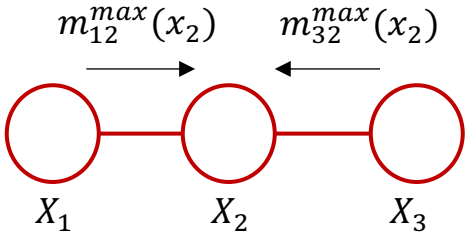
X_2	$m_{12}^{max}(x_2)$	$\delta_{12}(x_2)$
0	$\max_{x_1} \psi(x_1)\psi(x_1, x_2 = 0)$ $= \max(\psi(x_1 = 0)\psi(x_1 = 0, x_2 = 0), \psi(x_1 = 1)\psi(x_1 = 1, x_2 = 0))$ $= \max(a_1, a_2) = a_1$	$x_1^{max} = 0$
1	$\max_{x_1} \psi(x_1)\psi(x_1, x_2 = 1)$ $= \max(\psi(x_1 = 0)\psi(x_1 = 0, x_2 = 1), \psi(x_1 = 1)\psi(x_1 = 1, x_2 = 1))$ $= \max(a_3, a_4) = a_4$	$x_1^{max} = 1$

*In this example, we assume $a_1 > a_2$ and $a_4 > a_3$.

X_2	$m_{32}^{max}(x_2)$	$\delta_{32}(x_2)$
0	$\max_{x_3} \psi(x_3)\psi(x_3, x_2 = 0)$ $= \max(\psi(x_3 = 0)\psi(x_3 = 0, x_2 = 0), \psi(x_3 = 1)\psi(x_3 = 1, x_2 = 0))$ $= \max(b_1, b_3) = b_3$	$x_3^{max} = 1$
1	$\max_{x_3} \psi(x_3)\psi(x_3, x_2 = 1)$ $= \max(\psi(x_3 = 0)\psi(x_3 = 0, x_2 = 1), \psi(x_3 = 1)\psi(x_3 = 1, x_2 = 1))$ $= \max(b_2, b_4) = b_2$	$x_3^{max} = 0$

*In this example, we assume $b_3 > b_1$ and $b_2 > b_4$.

Find: $\underset{x_1,x_2,x_3}{\operatorname{argmax}}\, p(x_1, x_2, x_3)$



Root Node

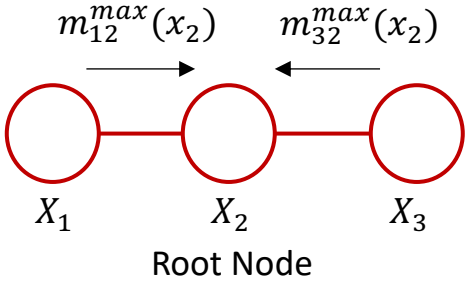
Max-Product Algorithm for Trees

Find: $\underset{x_1,x_2,x_3}{\operatorname{argmax}}\, p(x_1, x_2, x_3)$

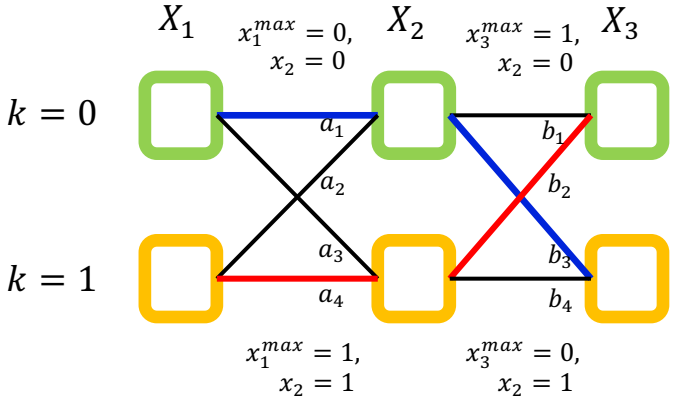
Example: $x_i \in \{0,1\}$
Root node

$m_2^{max}(x_2)$	$\delta_2(\cdot)$
$\max_{x_2} \psi(x_2) m_{12}^{max}(x_2) m_{32}^{max}(x_2)$ $= \max(\psi(x_2 = 0)a_1 b_3, \psi(x_2 = 1)a_4 b_2)$ $= \max(d_1, d_2) = d_1$ and d_2	$x_2^{max} = 0$ or 1

*In this example, we assume $d_1 = d_2$.



Trellis Diagram:



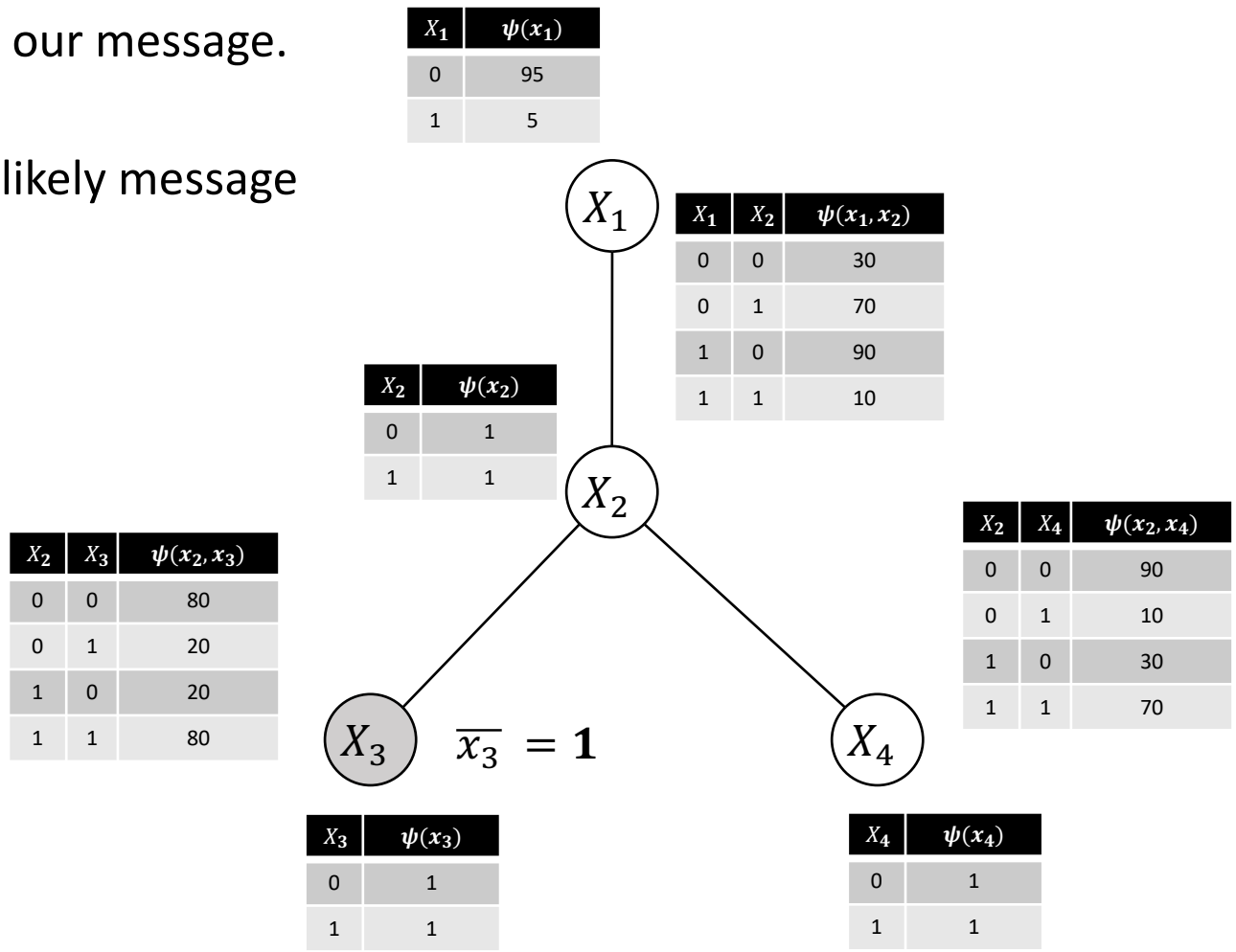
Downward pass

$\delta_{12}(x_2): x_1^{max} = 0 \leftarrow \delta_2(x_2): x_2^{max} = 0 \rightarrow \delta_{32}(x_2): x_3^{max} = 1$

$\delta_{12}(x_2): x_1^{max} = 1 \leftarrow \delta_2(x_2): x_2^{max} = 1 \rightarrow \delta_{32}(x_2): x_3^{max} = 0$

Exercise: Corrupted Message

- $\{X_1, X_2, X_3, X_4\}$ is our message. We see _ _ 1 _.
- What is the most likely message that was sent?



X_1	$\psi(x_1)$
0	95
1	5

X_1	X_2	$\psi(x_1, x_2)$
0	0	30
0	1	70
1	0	90
1	1	10

X_2	$\psi(x_2)$
0	1
1	1

X_2	X_3	$\psi(x_2, x_3)$
0	0	80
0	1	20
1	0	20
1	1	80

X_2	X_4	$\psi(x_2, x_4)$
0	0	90
0	1	10
1	0	30
1	1	70

X_3	$\psi(x_3)$
0	1
1	1

X_4	$\psi(x_4)$
0	1
1	1

MAX-PRODUCT($\mathcal{T}$, E)
  EVIDENCE(E)
  $f = \text{CHOOSEROOT}(\mathcal{V})$
  for $e \in \mathcal{N}(f)$
    COLLECT(f, e)
  $MAP = \max_{x_f}(\psi^E(x_f) \prod_{e \in \mathcal{N}(f)} m_{ef}^{\max}(x_f))$
  $x_f^* = \arg\max_{x_f}(\psi^E(x_f) \prod_{e \in \mathcal{N}(f)} m_{ef}^{\max}(x_f))$
  for $e \in \mathcal{N}(f)$
    DISTRIBUTE(f, e)

COLLECT(i, j)
  for $k \in \mathcal{N}(j) \backslash i$
    COLLECT(j, k)
  SENDMESSAGE(j, i)

DISTRIBUTE(i, j)
  SETVALUE(i, j)
  for $k \in \mathcal{N}(j) \backslash i$
    DISTRIBUTE(j, k)

SENDMESSAGE(j, i)

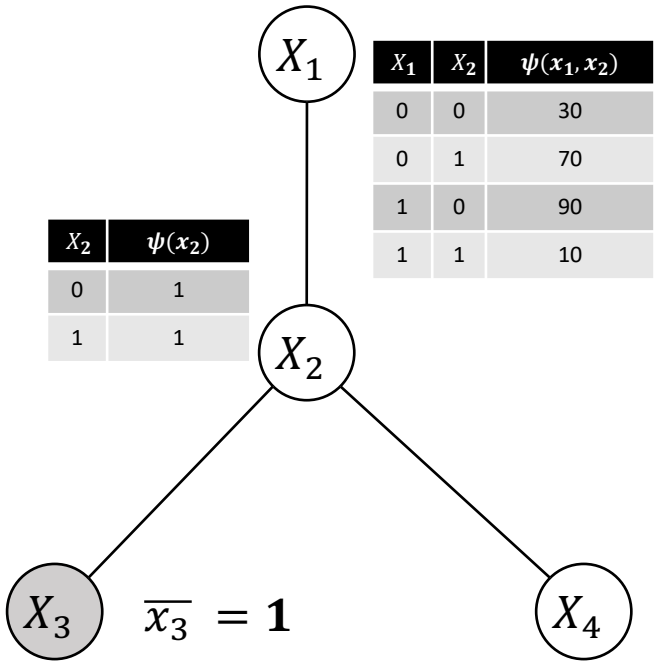
$$m_{ji}^{\max}(x_i) = \max_{x_j}(\psi^E(x_j)\psi(x_i, x_j) \prod_{k \in \mathcal{N}(j) \backslash i} m_{kj}^{\max}(x_j))$$

$$\delta_{ji}(x_i) \in \arg\max_{x_j}(\psi^E(x_j)\psi(x_i, x_j) \prod_{k \in \mathcal{N}(j) \backslash i} m_{kj}^{\max}(x_j))$$

SETVALUE(i, j)

$$x_j^* = \delta_{ji}(x_i^*)$$

X_1	$\psi(x_1)$
0	95
1	5



X_1	X_2	$\psi(x_1, x_2)$
0	0	30
0	1	70
1	0	90
1	1	10

X_2	$\psi(x_2)$
0	1
1	1

X_2	X_3	$\psi(x_2, x_3)$
0	0	80
0	1	20
1	0	20
1	1	80

X_2	X_4	$\psi(x_2, x_4)$
0	0	90
0	1	10
1	0	30
1	1	70

X_3	$\psi(x_3)$
0	1
1	1

X_4	$\psi(x_4)$
0	1
1	1

Remark: Underflow Problem

- Products of probabilities (numbers between 0 and 1) tend to underflow.
- Can be overcome by transforming to the monotone log scale:

$$\max_x p^E(x) = \max_x \log p^E(x)$$

- Fortunately, the distributive law still holds:

$$\max(a + b_1, a + b_2, \ldots, a + b_n) = a + \max(b_1, b_2, \ldots, b_n)$$

- Turns the "max-product" algorithm into the "max-sum" algorithm.

Recap

Learning Outcomes

- Students should be able to:

1. Use the Variable Elimination algorithm to compute the conditional probability of a single random variable X_f, i.e. $p(x_f | x_E)$.

2. Explain the computational complexity of variable elimination using the constituted graph.

3. Use the sum-product algorithm to compute all single-node marginals for "tree-like" graphical models.

4. Use the max-product algorithm to find the maximal probability and its configurations.

Variable Elimination Algorithm: Directed Graphs

ELIMINATE($\mathcal{G}, E, F$) // main steps of the "Variable Elimination Algorithm"
  INITIALIZE($\mathcal{G}, F$)
  EVIDENCE(E)
  UPDATE($\mathcal{G}$)
  NORMALIZE(F)

1: INITIALIZE($\mathcal{G}, F$) // choose elimination ordering, and add local condition probabilities in **active list**
  choose an ordering I such that F appears last
  for each node X_i in $\mathcal{V}$
    place $p(x_i \mid x_{\pi_i})$ on the active list
  end

2: EVIDENCE(E) // add evidence potentials in **active list**
  for each i in E
    place $\delta(x_i, \bar{x}_i)$ on the active list
  end

3: UPDATE($\mathcal{G}$) // **marginalization**, and update active list
  for each i in I
    find all potentials from the active list that reference x_i and remove them from the active list
    let $\phi_i(x_{T_i})$ denote the product of these potentials
    let $m_i(x_{S_i}) = \sum_{x_i} \phi_i(x_{T_i})$
    place $m_i(x_{S_i})$ on the active list
  end

4: NORMALIZE(F) // compute the desired **conditional probability**
  $p(x_F \mid \bar{x}_E) \leftarrow \phi_F(x_F) / \sum_{x_F} \phi_F(x_F)$

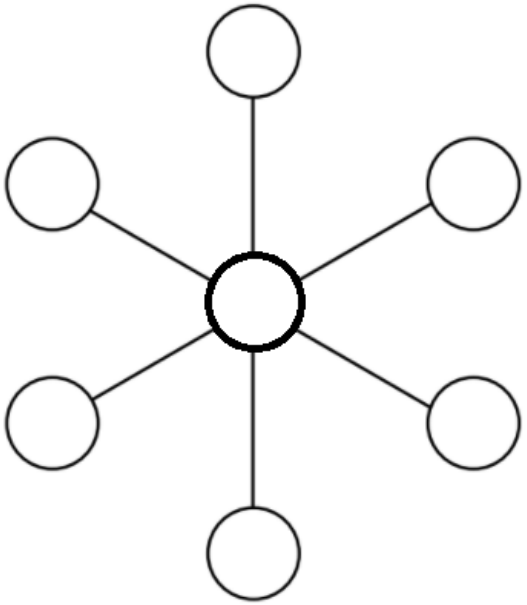
Source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Variable Elimination Algorithm: Undirected Graphs

- Entire variable eliminate algorithm for directed graph goes through without essential change to the undirected case.
- Only change needed in the initialize procedure.
- Instead of using local conditional probabilities we initialize the active list to contain the potentials of $\{\psi_{x_c}(x_C)\}$.

Computational Complexity

Example:



- A graph whose treewidth is equal to one.
- However, the wrong choice of eliminating the center node would immediately leads to a elimination clique with all the neighbors!

Sum-Product Algorithm

- Two phases:

1. Messages flow inward from leaves toward the root.

2. Initiated once all incoming messages have been received by the root node – messages flow outward from root toward the leaves.