NUS National University of Singapore | School of Computing

CS5340 Uncertainty Modeling in AI

Lecture 4: Variable Elimination and Belief Propagation

Assoc. Prof. Lee Gim Hee

AY 2022/23

Semester 1

Course Schedule

Week	Date	Topic	Remarks
1	10 Aug	Introduction to probabilistic reasoning	**Assignment 0:** Python Numpy Tutorial (Ungraded)
2	17 Aug	Bayesian networks (Directed graphical models)	
3	24 Aug	Markov random Fields (Undirected graphical models)	
4	31 Aug	Variable elimination and belief propagation	**Assignment 1:** Belief propagation and maximal probability (15%)
5	07 Sep	Factor graph and the junction tree algorithm	
6	14 Sep	Parameter learning with complete data	**Assignment 1:** Due **Assignment 2:** Junction tree and parameter learning (15%)
-	21 Sep	Recess week	**No lecture**
7	28 Sep	Mixture models and the EM algorithm	**Assignment 2:** Due
8	05 Oct	Hidden Markov Models (HMM)	**Assignment 3:** Hidden Markov model (15%)
9	12 Oct	Monte Carlo inference (Sampling)	
*	15 Oct	Variational inference	Makeup Lecture (LT15) Time: 9.30am – 12.30pm (Saturday)
10	19 Oct	Variational Auto-Encoder and Mixture Density Networks	**Assignment 3:** Due **Assignment 4:** MCMC Sampling (15%)
11	26 Oct	No Lecture	I will be traveling
12	02 Nov	Graph-cut and alpha expansion	**Assignment 4:** Due
13	09 Nov	-	

Acknowledgements

- A lot of slides and content of this lecture are adopted 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

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.

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).

Probabilistic Inference Problem

Conditional probability:

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

Marginals from joint probability:

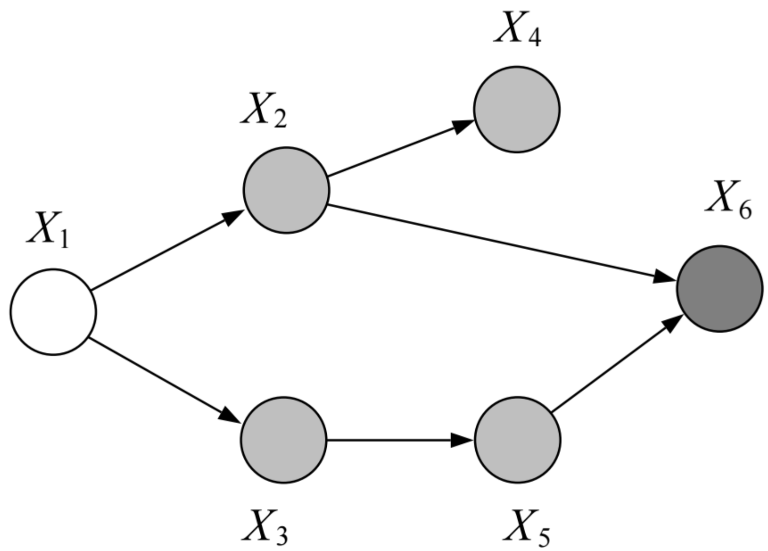
$$p(x_E, x_F) = \sum_{x_R} p(x_E, x_F, x_R),$$

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

X_R: nuisance variables

$\{X_E, X_R, X_F\}$: all random variables in the graphical model

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.
- Lightly shaded nodes $\mathrm{X}_R = X_V \setminus (X_E, X_F)$ are the nodes that must be marginalized out of the joint probability.

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

Probabilistic Inference Problem

Marginals from joint probability:

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

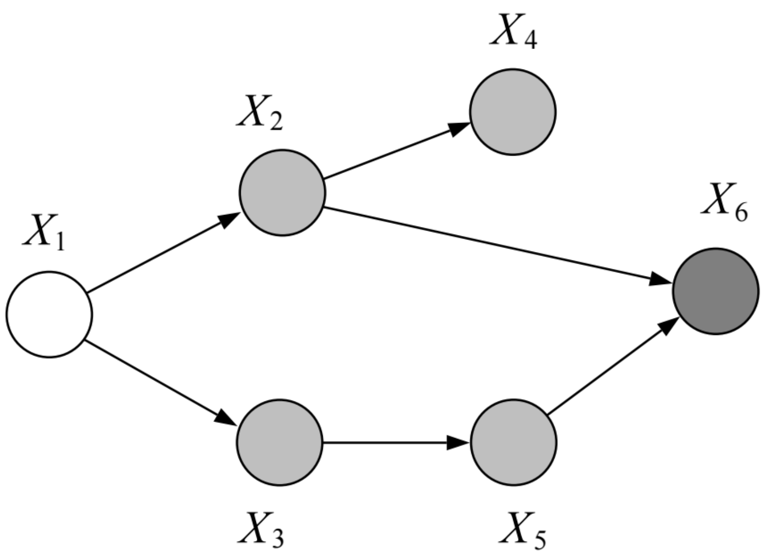
- Σ_{x_R} expands into a sequence of summations, one for each of the random variables indexed by R.
- A naïve summation over the joint distribution of n variables that takes k states will incur a computational complexity of $O(k^n)$!

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.
- Refer to X_F as the "query node", and X_E as the "evidence nodes".
- Variable elimination algorithm: an efficient algorithm based on marginalization and conditional independence of the graphical model.

Naive Summation

Naïve summation is intractable!



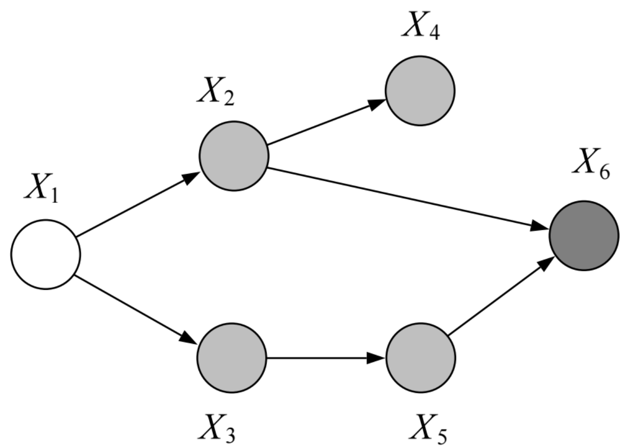
Consider:

$$p(x_1, x_2, x_3, x_4, x_5) = \sum_{x_6} \underbrace{p(x_1, x_2, x_3, x_4, x_5, x_6)}^{\text{Joint probability table size is } k^6}$$

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

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

Variable Elimination



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

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

$O(k^6)$ 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.

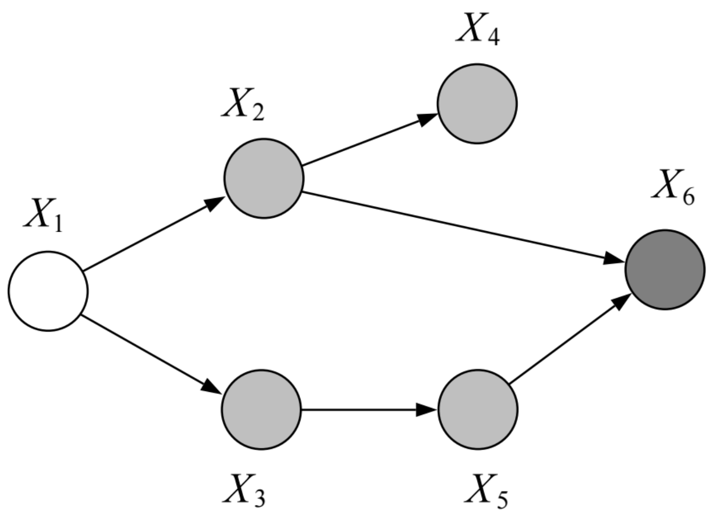
Evidence Node

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 $\bar{X}_i$:

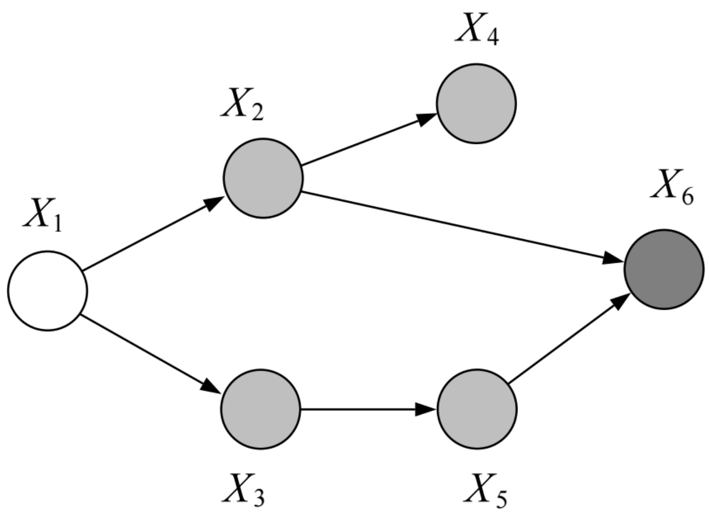
$$p(x_1|\bar{x}_6) = \frac{p(x_1, \bar{x}_6)}{p(\bar{x}_6)}$$

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

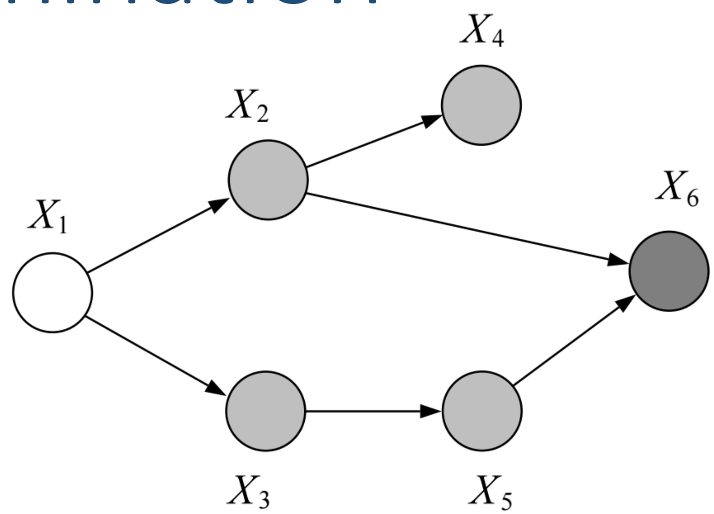
We are taking a 2d slice of the 3d probabilities or potentials!

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:

$$\begin{aligned}
p(x_1, \bar{x}_6) &= \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)p(x_5 \mid x_3)p(\bar{x}_6 \mid x_2, x_5) \\
&= p(x_1)\sum_{x_2} p(x_2 \mid x_1)\sum_{x_3} p(x_3 \mid x_1)\sum_{x_4} p(x_4 \mid x_2)\underbrace{\sum_{x_5} p(x_5 \mid x_3)p(\bar{x}_6 \mid x_2, x_5)}_{\text{eliminate } X_5} \\
&= p(x_1)\sum_{x_2} p(x_2 \mid x_1)\sum_{x_3} p(x_3 \mid x_1)\sum_{x_4} p(x_4 \mid x_2)m_5(x_2, x_3)
\end{aligned}$$

- 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:

$$\begin{aligned}
p(x_1, \bar{x}_6) &= \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)p(x_5 \mid x_3)p(\bar{x}_6 \mid x_2, x_5) \\
&= p(x_1)\sum_{x_2} p(x_2 \mid x_1)\sum_{x_3} p(x_3 \mid x_1)\sum_{x_4} p(x_4 \mid x_2)\boxed{\sum_{x_5} p(x_5 \mid x_3)p(\bar{x}_6 \mid x_2, x_5)} \\
&= p(x_1)\sum_{x_2} p(x_2 \mid x_1)\sum_{x_3} p(x_3 \mid x_1)\boxed{m_5(x_2, x_3)}\boxed{\sum_{x_4} p(x_4 \mid x_2)} \\
&= p(x_1)\sum_{x_2} p(x_2 \mid x_1)\boxed{m_4(x_2)}\boxed{\sum_{x_3} p(x_3 \mid x_1)m_5(x_2, x_3)} \\
&= p(x_1)\boxed{\sum_{x_2} p(x_2 \mid x_1)m_4(x_2)\boxed{m_3(x_1, x_2)}} \\
&= p(x_1)\boxed{m_2(x_1)}
\end{aligned}$$

eliminate X_5, independent of X_4

eliminated X_4, Independent of X_3

eliminated X_3

eliminated X_2

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:

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

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:

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

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

Proof:

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

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

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

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

Conditioning to Marginalization Trick

Example:

(Directed Graph)

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

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

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

(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 $p(x_i \,|\, x_{\pi_i})$ 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**
  $p(x_F \,|\, \bar{x}_E) \leftarrow \phi_F(x_F) / \sum_{x_F} \phi_F(x_F)$

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.

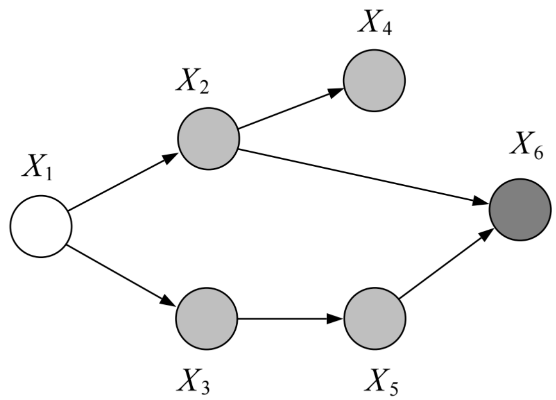


Image source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Variable Elimination Algorithm: Directed Graphs

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

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

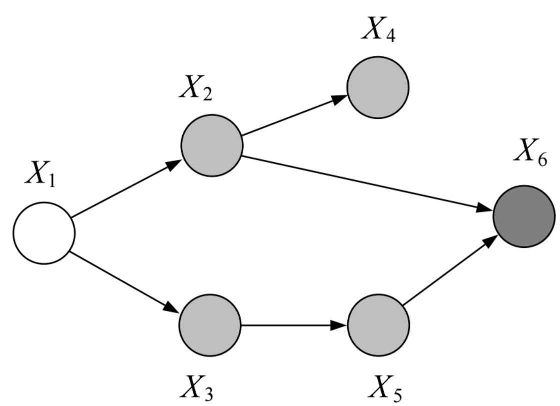
choose an ordering I such that F appears last

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

place $p(x_i \,|\, x_{\pi_i})$ 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**

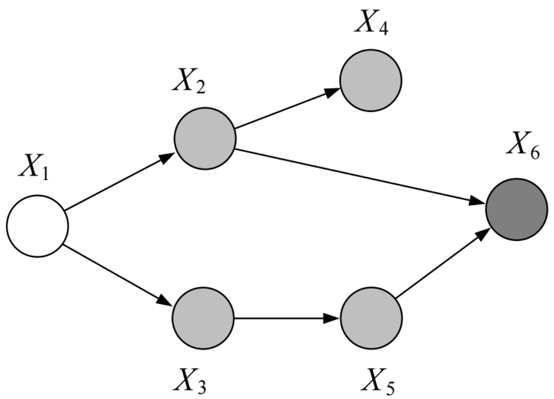
2: EVIDENCE(E)

for each i in E

place $\delta(x_i, \bar{x}_i)$ 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: $I = \{6, 5, 4, 3, 2, 1\}$

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

$$\Downarrow$$

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

$$\Downarrow$$

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

$$\Downarrow$$

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

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

***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: $\text{UPDATE}(\mathcal{G})$

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

$\boldsymbol{i = 2}$:

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

$$\Downarrow$$

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

$$\Downarrow$$

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

$$\Downarrow$$

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

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

$\boldsymbol{i = 1}$:

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

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

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)

$$\boxed{p(x_F \,|\, \bar{x}_E) \leftarrow \phi_F(x_F) / \sum_{x_F} \phi_F(x_F)}$$

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

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

$\boldsymbol{i = 6}$:

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

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

$\boldsymbol{i = 5}$:

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

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

Variable Elimination Algorithm: Directed Graphs

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

$\boldsymbol{i = 4}$:

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

$\boldsymbol{i = 3}$:

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

$\boldsymbol{i = 2}$:

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

$\boldsymbol{i = 1}$:

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

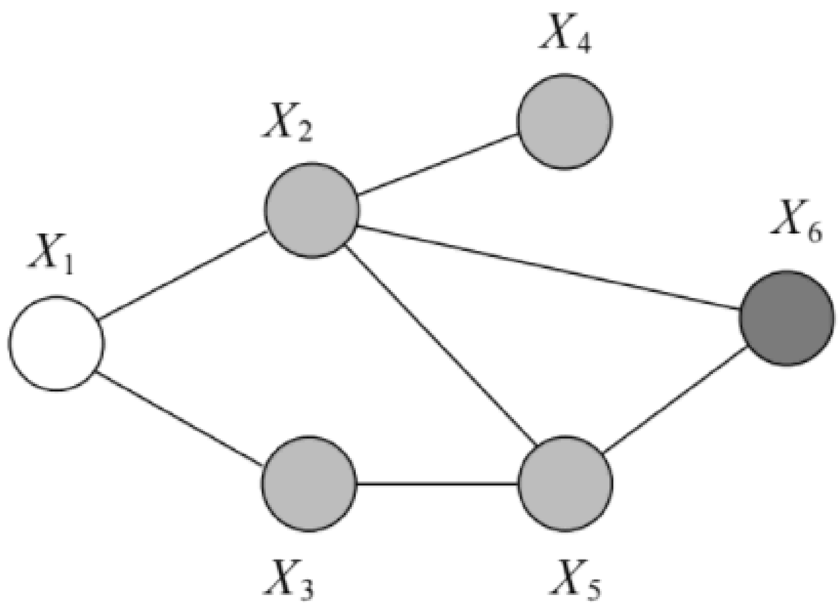


Image source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Variable Elimination Algorithm: Undirected Graphs

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

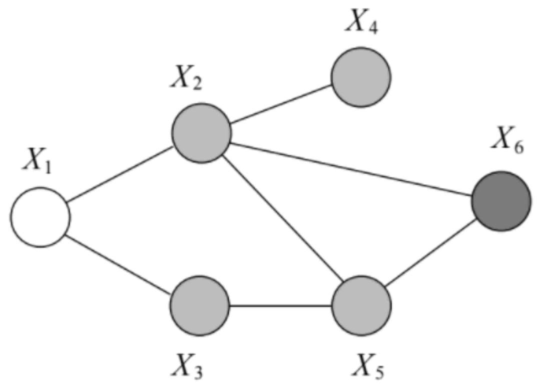


Image source: “An introduction to probabilistic graphical models”, Michael I. Jordan, 2002.

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.

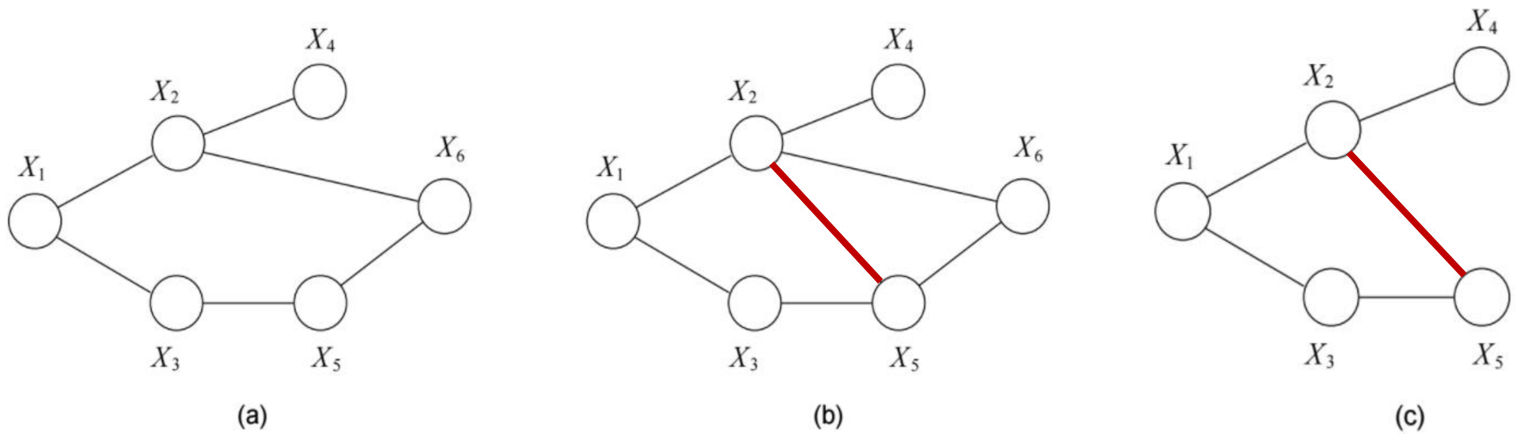
- **Important note**: For a marginal probability, the normalization factor Z does not cancel, and must be calculated explicitly.

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 $\tilde{\mathcal{G}} = (\mathcal{V}, \tilde{\mathcal{E}})$.

Reconstituted Graph

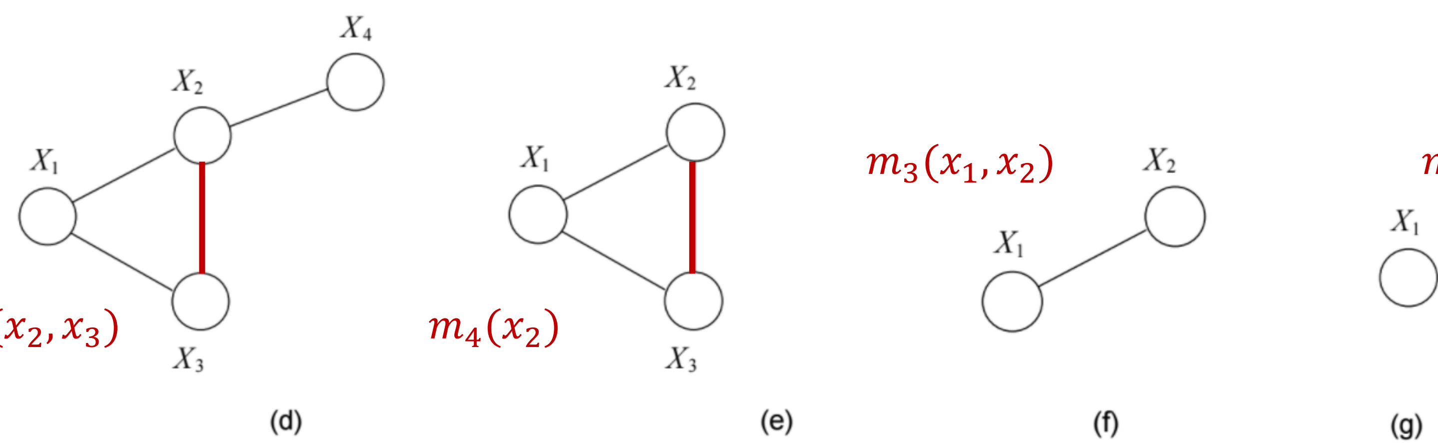
Example: Elimination ordering (6; 5; 4; 3; 2; 1)



$$p(x) = \psi(x_1, x_2)\psi(x_1, x_3)\psi(x_2, x_4)$$
$$\psi(x_3, x_5)\psi(x_2, x_6)\psi(x_5, x_6)$$

$\sum_{x_6} \psi(x_2, x_6)\psi(x_5, x_6)$

$m_6(x_2, x_5)$



$m_5(x_2, x_3)$

$m_4(x_2)$

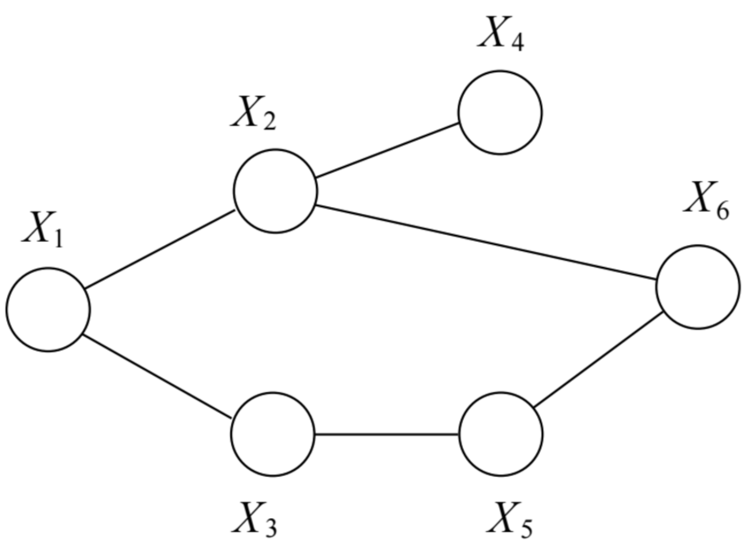
$m_3(x_1, x_2)$

$m_2(x_1)$

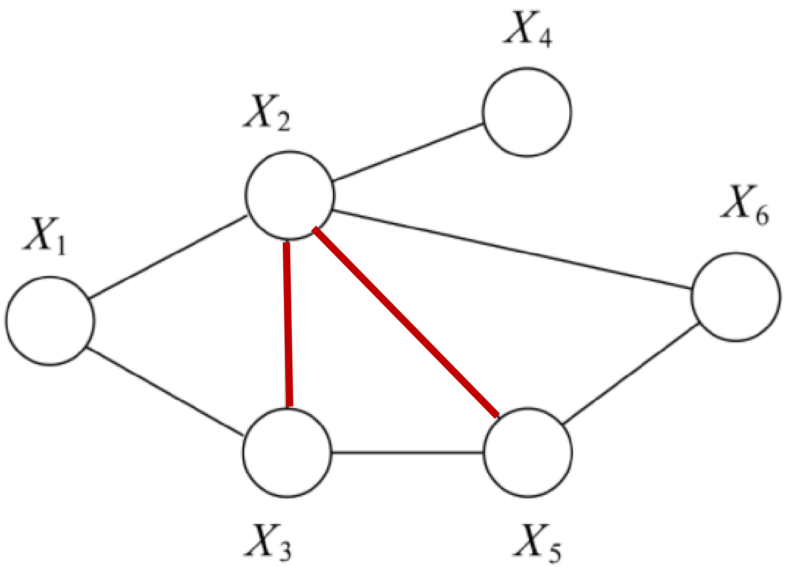
Image source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

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

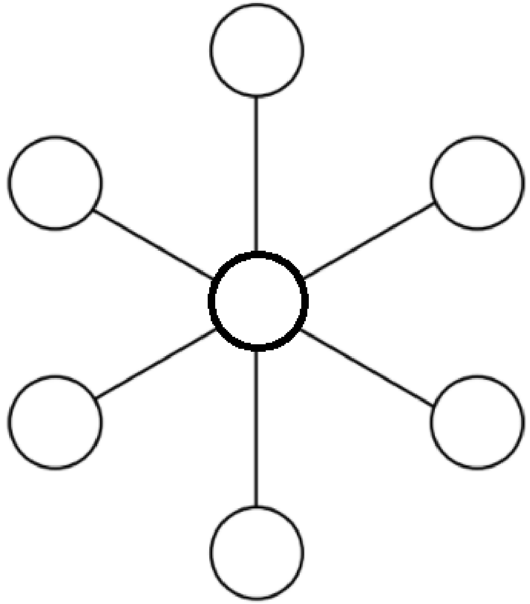
- Elimination process adds new edges between (remaining) neighbors of the node.
- This creates new “elimination cliques” in the graph.
- Overall complexity depends on the size of the largest elimination clique.
- Which depends on the choice of elimination ordering.

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 must 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 an elimination clique with all the neighbors!

Computational Complexity

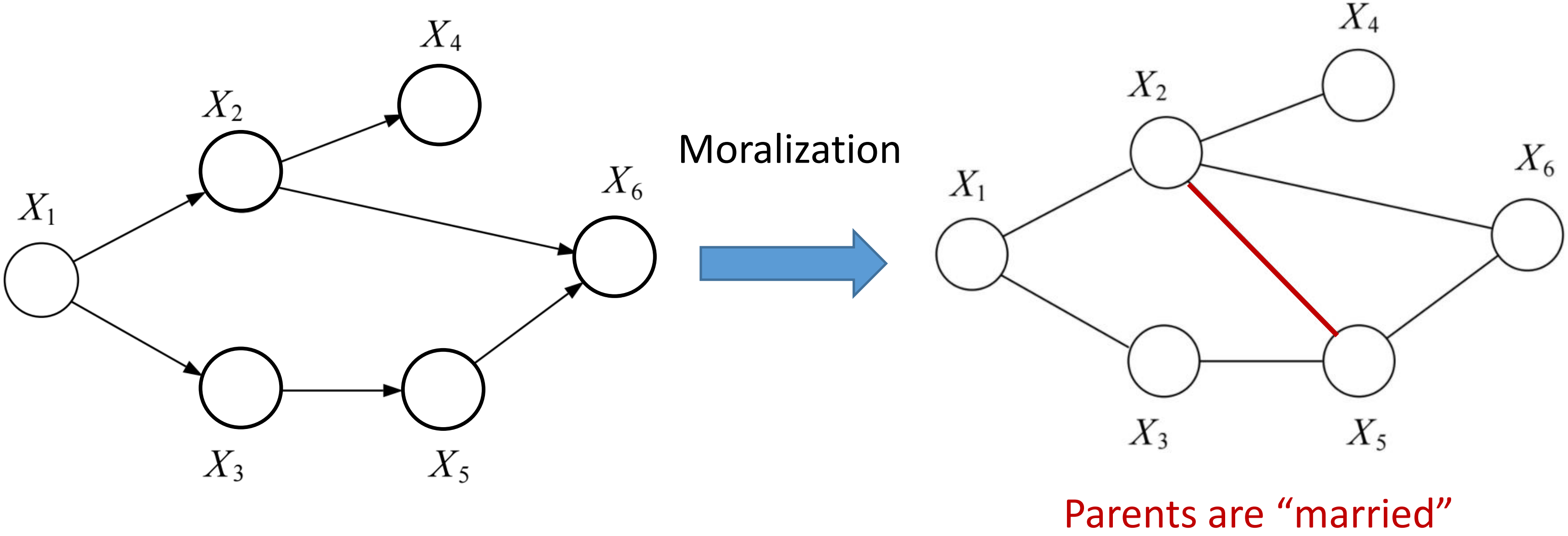
- Computation complexity of DGMs can be analyzed in the same way as UGMs by moralization.

```
DirectedGraphEliminate(G, I)
    G^m = 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.

Limitation of Variable Elimination

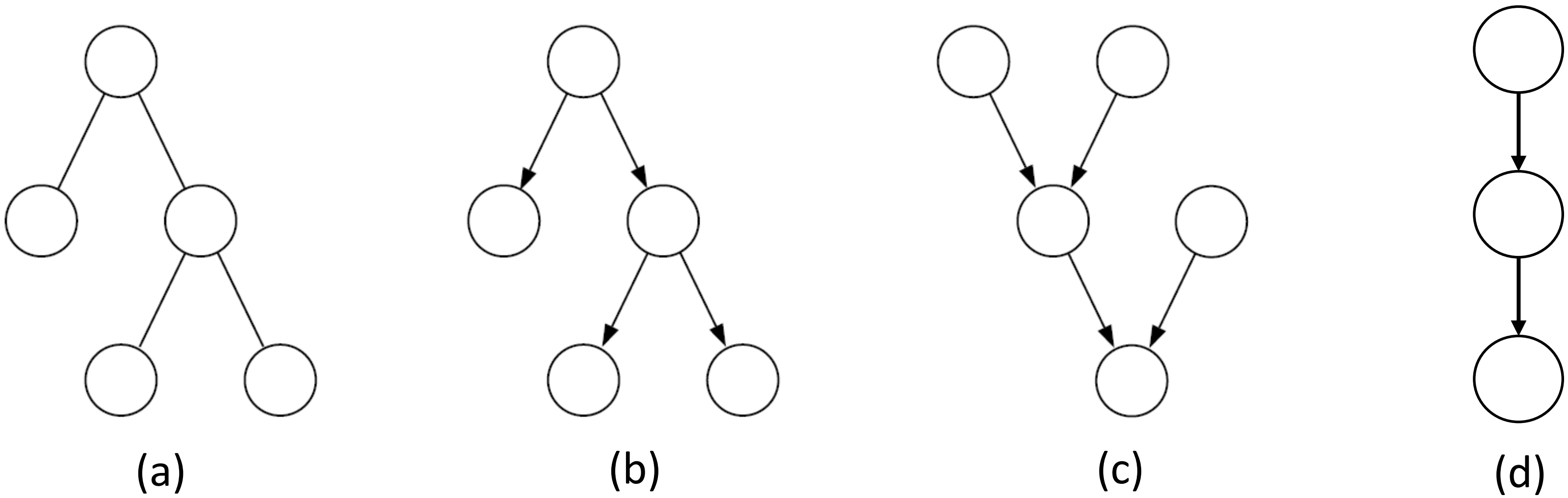
Limitation:

- 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 for certain "tree-like" graphs in a single run.

"Tree-Like" Graphs



a) Undirected tree: without any loop.

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:

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

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:

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

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

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

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 formally identical!

Conditioning

- To capture conditioning i.e. $p(x \mid \bar{x}_E)$ 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

- **Question**: What are the special features of the variable elimination algorithm when the graph is a tree?

- **Answer**: We can consider elimination orderings that arise from a depth-first traversal of the tree.

Depth-First Tree Traversal

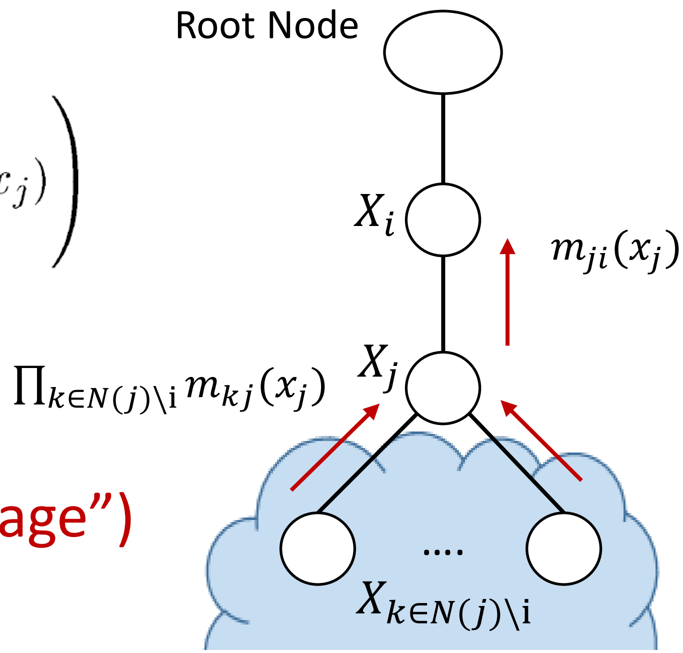
- Take advantage of the recursive structure of a tree to specify an elimination ordering.
- Treat query node X_f as the root.
- View the tree as a directed tree by directing all edges of the tree to point away from X_f.
- Elimination proceeds inward from the leaves, with treewidth equals to one!

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:

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

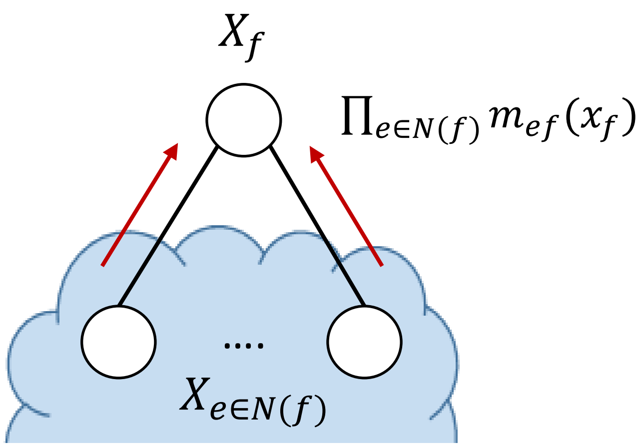


- This is the intermediate factor ("message") that X_j sends to X_i.

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

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

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

- 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:

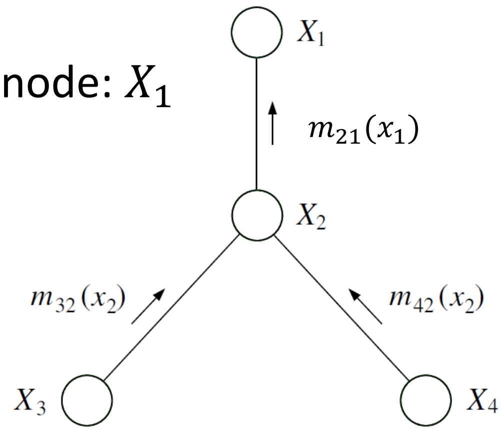
Messages can be "reused"!

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

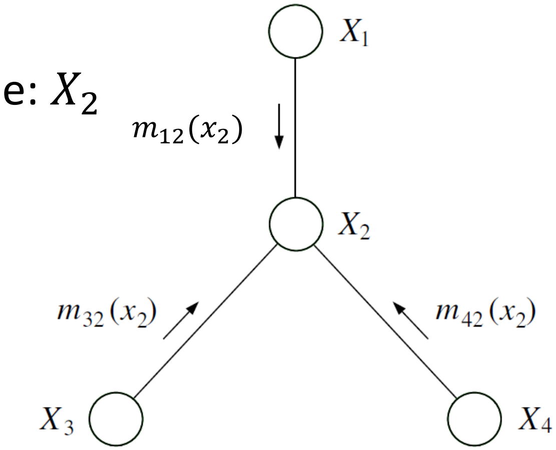
Sum-Product Algorithm

Example:

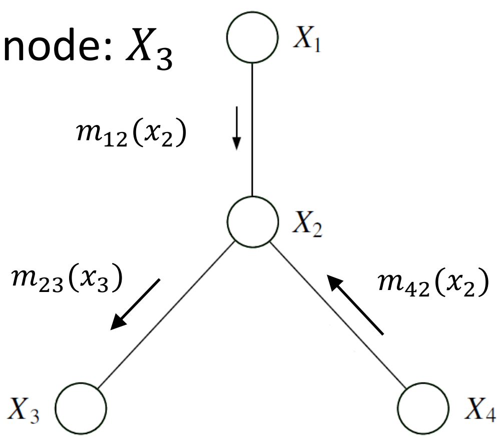
Query node: X_1



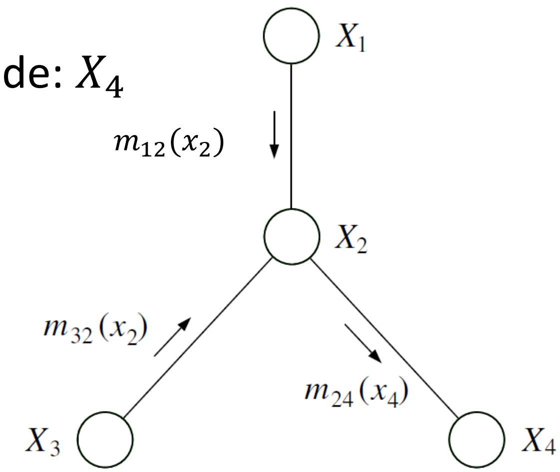
Query node: X_2



Query node: X_3



Query node: X_4

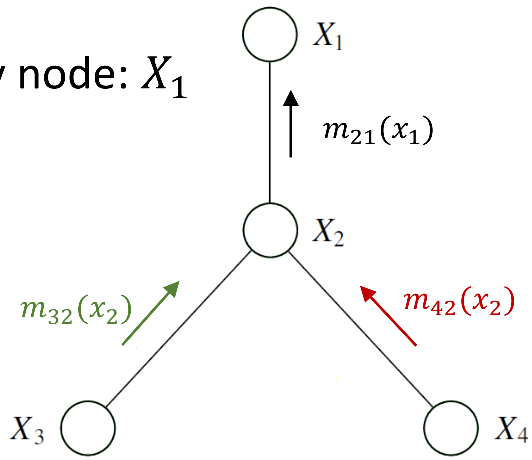


Source: “An introduction to probabilistic graphical models”, Michael I. Jordan, 2002.

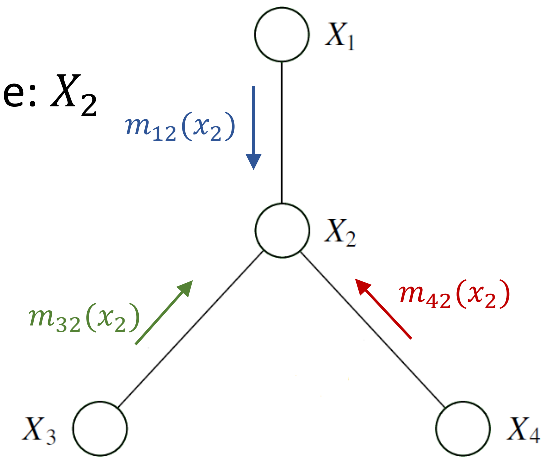
Sum-Product Algorithm

Example: Messages are "reused"!

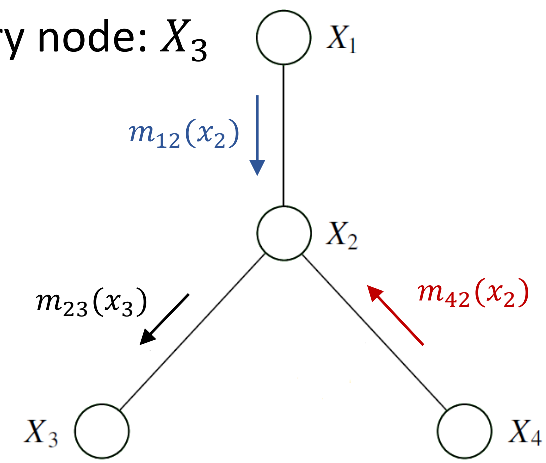
Query node: X_1



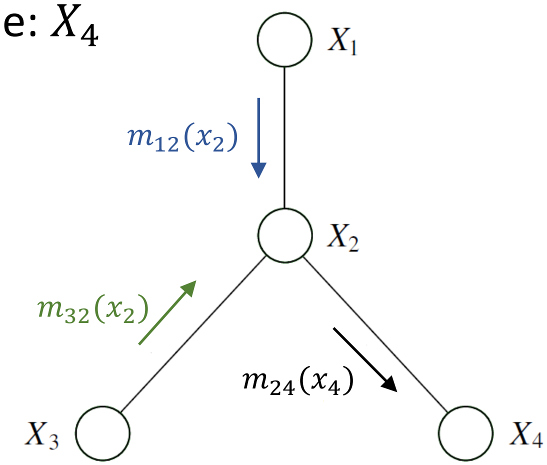
Query node: X_2



Query node: X_3



Query node: X_4

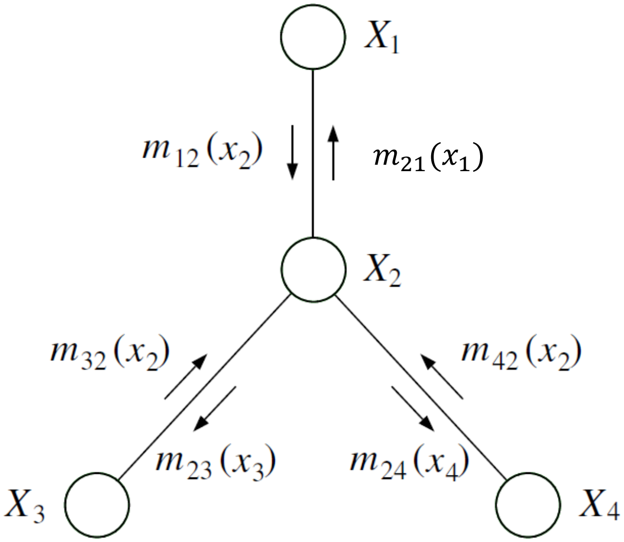


Source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

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(𝒯, E)                    // main steps of the "Sum-Product Algorithm"
    EVIDENCE(E)
    f = CHOOSEROOT(𝒱)
    for e ∈ 𝒩(f)
        COLLECT(f, e)
    for e ∈ 𝒩(f)
        DISTRIBUTE(f, e)
    for i ∈ 𝒱
        COMPUTEMARGINAL(i)
```

```
EVIDENCE(E)                          // add evidence potentials (convert conditioning into marginalization)
    for i ∈ E
        ψ^E(x_i) = ψ(x_i) δ(x_i, x̄_i)
    for i ∉ E
        ψ^E(x_i) = ψ(x_i)

COLLECT(i, j)                        // messages flow inward from leaves toward the root
    for k ∈ 𝒩(j)\i
        COLLECT(j, k)
    SENDMESSAGE(j, i)

DISTRIBUTE(i, j)                     // messages flow outward from root toward the leaves
    SENDMESSAGE(i, j)
    for k ∈ 𝒩(j)\i
        DISTRIBUTE(j, k)
```

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

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

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"

$\quad$ EVIDENCE(E)

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

$\quad$ **for** $e \in \mathcal{N}(f)$

$\quad\quad$ COLLECT(f, e)

$\quad$ **for** $e \in \mathcal{N}(f)$

$\quad\quad$ DISTRIBUTE(f, e)

$\quad$ **for** $i \in \mathcal{V}$

$\quad\quad$ COMPUTEMARGINAL(i)

EVIDENCE(E) // add **evidence potentials** (convert conditioning into marginalization)

$\quad$ **for** $i \in E$

$\quad\quad$ $\psi^E(x_i) = \psi(x_i)\delta(x_i, \bar{x}_i)$

$\quad$ **for** $i \notin E$

$\quad\quad$ $\psi^E(x_i) = \psi(x_i)$

COLLECT(i, j) // messages flow **inward** from leaves toward the root

$\quad$ **for** $k \in \mathcal{N}(j) \backslash i$

$\quad\quad$ COLLECT(j, k)

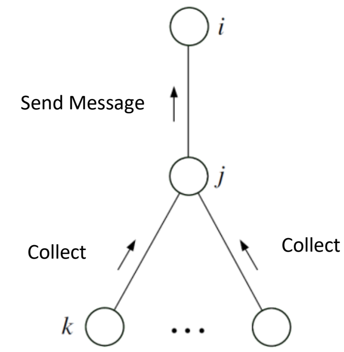
$\quad$ SENDMESSAGE(j, i)

DISTRIBUTE(i, j) // messages flow **outward** from root toward the leaves

$\quad$ SENDMESSAGE(i, j)

$\quad$ **for** $k \in \mathcal{N}(j) \backslash i$

$\quad\quad$ DISTRIBUTE(j, k)



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

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

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

$\textsc{Sum-Product}(\mathcal{T},\ E)$ // main steps of the "Sum-Product Algorithm"

$\quad \textsc{Evidence}(E)$

$\quad f = \textsc{ChooseRoot}(\mathcal{V})$

$\quad$ **for** $e \in \mathcal{N}(f)$

$\qquad \textsc{Collect}(f, e)$

$\quad$ **for** $e \in \mathcal{N}(f)$

$\qquad \textsc{Distribute}(f, e)$

$\quad$ **for** $i \in \mathcal{V}$

$\qquad \textsc{ComputeMarginal}(i)$

$\textsc{Evidence}(E)$ // add **evidence potentials** (convert conditioning into marginalization)

$\quad$ **for** $i \in E$

$\qquad \psi^E(x_i) = \psi(x_i)\delta(x_i, \bar{x}_i)$

$\quad$ **for** $i \notin E$

$\qquad \psi^E(x_i) = \psi(x_i)$

$\textsc{Collect}(i, j)$ // messages flow **inward** from leaves toward the root

$\quad$ **for** $k \in \mathcal{N}(j) \backslash i$

$\qquad \textsc{Collect}(j, k)$

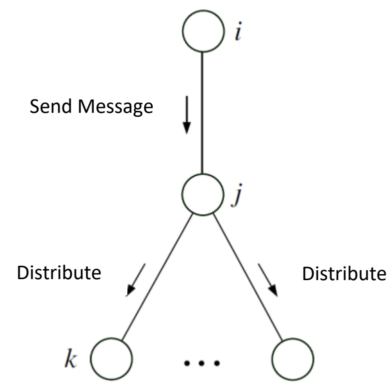
$\quad \textsc{SendMessage}(j, i)$

$\textsc{Distribute}(i, j)$ // messages flow **outward** from root toward the leaves

$\quad \textsc{SendMessage}(i, j)$

$\quad$ **for** $k \in \mathcal{N}(j) \backslash i$

$\qquad \textsc{Distribute}(j, k)$



$\textsc{SendMessage}(j, i)$ // intermediate factors (messages)

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

$\textsc{ComputeMarginal}(i)$ // message to final node

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

Summary

- You have learned how 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.