• Readings:
  – handouts

• Today’s Lecture:
  – d-Separation, minimal I-Maps
  – Bayesian Networks
  – Markov Networks
  – Inference

• Upcoming Due Dates:
  – P2 due 3/14
Summary of Last Class

We defined the following concepts

- The Markov Independences of a DAG $G$
  - $I(X_i, \text{NonDesc}(X_i) | \text{Pa}_i)$

- $G$ is an I-Map of a distribution $P$
  - If $P$ satisfies the Markov independencies implied by $G$

We showed

$$G \text{ is an I-Map of } P \iff P(X_1, \ldots, X_n) = \prod_{i} P(X_i | \text{Pa}_i)$$
d-seperation

- A procedure $d$-$sep(X; Y / Z, G)$ that given a DAG $G$, and sets $X$, $Y$, and $Z$ returns either yes or no

- **Goal:**
  $$d$-sep(X; Y / Z, G) = yes$$ iff $I(X; Y|Z)$ follows from $Markov(G)$

- $X$ is **d-separated** from $Y$, given $Z$, if all paths from a node in $X$ to a node in $Y$ are blocked, given $Z$. 
Paths

• We want to know when a path is
  – **active** -- creates dependency between end nodes
  – **blocked** -- cannot create dependency end nodes

• We want to classify situations in which paths are active.
Path Blockage

- Common cause
- Intermediate cause
- Common Effect
Minimal I-Maps

A DAG $G$ is a \textbf{minimal I-Map} of $P$ if

- $G$ is an I-Map of $P$
- If $G' \subset G$, then $G'$ is not an I-Map of $P$

Removing any arc from $G$ introduces (conditional) independencies that do not hold in $P$
Constructing minimal I-Maps

The factorization theorem gives an algorithm

- Fix an ordering $X_1, \ldots, X_n$
- For each $i$,
  - select $Pa_i$ to be a minimal subset of $\{X_1, \ldots, X_{i-1}\}$, such that $I(X_i; \{X_1, \ldots, X_{i-1}\} - Pa_i / Pa_i)$

- Clearly, the resulting graph is a minimal I-Map.
Non-uniqueness of minimal I-Map

• Unfortunately, there may be several minimal I-Maps for the same distribution
  – Applying I-Map construction procedure with different orders can lead to different structures

Order: $C, R, A, E, B$

Original I-Map
Choosing Ordering & Causality

• The choice of order can have drastic impact on the complexity of minimal I-Map

• Heuristic argument:
  construct I-Map using *causal* ordering among variables

• Justification?
  – It is often reasonable to assume that graphs of causal influence should satisfy the Markov properties.
P-Maps

- A DAG $G$ is P-Map (perfect map) of a distribution $P$ if
  - $I(X; Y \mid Z)$ if and only if
    $$d-sep(X; Y \mid Z, G) = yes$$

Notes:
- A P-Map captures all the independencies in the distribution
- P-Maps are unique, up to DAG equivalence
P-Maps

• Unfortunately, some distributions do not have a P-Map

• Example: \( P(A, B, C) = \begin{cases} 
\frac{1}{12} & \text{if } A \oplus B \oplus C = 0 \\
\frac{1}{6} & \text{if } A \oplus B \oplus C = 1 
\end{cases} \)

• A minimal I-Map:

• This is not a P-Map since \( I(A;C) \) but \( d-sep(A;C) = no \)
Bayesian Networks

• A Bayesian network specifies a probability distribution via two components:
  
  – A DAG $G$
  – A collection of conditional probability distributions $P(X_i|Pa_i)$

• The joint distribution $P$ is defined by the factorization

$$P(X_1,\ldots,X_n) = \prod_i P(X_i \mid Pa_i)$$

• Additional requirement: $G$ is a minimal I-Map of $P$
Summary

- We explored DAGs as a representation of conditional independencies:
  - Markov independencies of a DAG
  - Tight correspondence between $\text{Markov}(G)$ and the factorization defined by $G$
  - d-separation, a sound & complete procedure for computing the consequences of the independencies
  - Notion of minimal I-Map
  - P-Maps
- This theory is the basis for defining Bayesian networks
Markov Networks

- We now briefly consider an alternative representation of conditional independencies

- Let \( U \) be an **undirected graph**
- Let \( N_i \) be the set of neighbors of \( X_i \)
- Define \( \text{Markov}(U) \) to be the set of independencies
  \[
  I( X_i ; \{X_1, \ldots, X_n\} - N_i - \{X_i\} / N_i )
  \]

- \( U \) is an I-Map of \( P \) if \( P \) satisfies \( \text{Markov}(U) \)
Example

This graph implies that
- $I(A; C \mid B, D )$
- $I(B; D \mid A, C )$

- Note: this example does not have a directed P-Map
Markov Network Factorization

**Thm:** if

- $P$ is **strictly positive**, that is $P(x_1, \ldots, x_n) > 0$ for all assignments

then

- $U$ is an I-Map of $P$

if and only if

- there is a factorization $P(X_1, \ldots, X_n) = \prod_{i} f(C_i)$

where $C_1, \ldots, C_k$ are the maximal cliques in $U$

**Alternative form:**

$$P(X_1, \ldots, X_n) = \frac{1}{Z} e^{\sum_i g(C_i)}$$
Relationship between Directed & Undirected Models
CPDs

- So far, we focused on how to represent independencies using DAGs
- The “other” component of a Bayesian networks is the specification of the **conditional probability distributions** (CPDs)

- We start with the simplest representation of CPDs and then discuss additional structure
Tabular CPDs

• When the variable of interest are all discrete, the common representation is as a table:

• For example $P(C|A,B)$ can be represented by

| $A$ | $B$ | $P(C = 0 | A, B)$ | $P(C = 1 | A, B)$ |
|-----|-----|------------------|------------------|
| 0   | 0   | 0.25             | 0.75             |
| 0   | 1   | 0.50             | 0.50             |
| 1   | 0   | 0.12             | 0.88             |
| 1   | 1   | 0.33             | 0.67             |
Tabular CPDs

Pros:
- Very flexible, can capture any CPD of discrete variables
- Can be easily stored and manipulated

Cons:
- Representation size grows exponentially with the number of parents!
- Unwieldy to assess probabilities for more than few parents
Structured CPD

• To avoid the exponential blowup in representation, we need to focus on specialized types of CPDs
• This comes at a cost in terms of expressive power
• We now consider several types of structured CPDs
Causal Independence

- Consider the following situation

  - In tabular CPD, we need to assess the probability of fever in eight cases
  - These involve all possible interactions between diseases

  - For three diseases, this might be feasible.
    For ten diseases, not likely....
Causal Independence

- Simplifying assumption:
  - Each disease attempts to cause fever, *independently* of the other diseases
  - The patient has fever if one of the diseases “succeeds”
- We can model this using a Bayesian network fragment
Noisy-Or CPD

• Models $P(X|Y_1,\ldots,Y_k)$, $X$, $Y_1,\ldots,Y_k$ are all binary

• Parameters:
  - $p_i$ -- probability of $X = 1$ due to $Y_i = 1$
  - $p_0$ -- probability of $X = 1$ due to other causes

• Plugging these in the model we get

\[
P(X = 0 | Y_1,\ldots,Y_k) = (1 - p_0) \prod_i (1 - p_i)^{y_i}
\]

\[
P(X = 1 | Y_1,\ldots,Y_k) = 1 - P(X = 0 | Y_1,\ldots,Y_k)
\]
Noisy-or CPD

• Benefits of noisy-or
  – “Reasonable” assumptions in many domains
    • e.g., medical domain
  – Few parameters.
  – Each parameter can be estimated independently of the others

• The same idea can be extended to other functions: noisy-max, noisy-and, etc.

• Frequently used in large medical expert systems
Context Specific Independence

• Consider the following examples:

• Alarm sound depends on
  – Whether the alarm was set before leaving the house
  – Burglary
  – Earthquake

• Arriving on time depends on
  – Travel route
  – The congestion on the two possible routes
Context-Specific Independence

• In both of these examples we have **context-specific independence (CSI)**
  – Independencies that depend on a particular value of one or more variables

• In our examples:
  - $\text{Ind}(A; B, E | S = 0)$
    Alarm sound is independent of $B$ and $E$ when the alarm is not set
  - $\text{Ind}(A; R_2 | T = 1)$
    Arrival time is independent of traffic on route 2 if we choose to travel on route 1
Representing CSI

- When we have such CSI, $P(X / Y_1, \ldots, Y_k)$ is the same for several values of $Y_1, \ldots, Y_k$
- There are many ways of representing these regularities

- A natural representation: decision trees
  - Internal nodes: tests on parents
  - Leaves: probability distributions on $X$

- Evaluate $P(X / Y_1, \ldots, Y_k)$ by traversing tree
Detecting CSI

• Given evidence on some nodes, we can identify the “relevant” parts of the trees
  – This consists of the paths in the tree that are consistent with context

• Example
  – Context $S = 0$
  – Only one path of tree is relevant

• A parent is independent given the context if it does not appear on one of the relevant paths
Decision Tree CPDs

Benefits
- Decision trees offer a flexible and intuitive language to represent CSI
- Incorporated into several commercial tools for constructing Bayesian networks

Comparison to noisy-or
- Noisy-or CPDs require full trees to represent
- General decision tree CPDs cannot be represented by noisy-or
Inference in Bayesian Networks
Inference

• We now have compact representations of probability distributions:
  – Bayesian Networks
  – Markov Networks

• Network describes a unique probability distribution $P$

• How do we answer queries about $P$?

• We use inference as a name for the process of computing answers to such queries
Queries: Likelihood

• There are many types of queries we might ask.
• Most of these involve evidence
  – An evidence $e$ is an assignment of values to a set $E$ variables in the domain
  – Without loss of generality $E = \{X_{k+1}, \ldots, X_n\}$
• Simplest query: compute probability of evidence

$$P(e) = \sum_{x_1} \ldots \sum_{x_k} P(x_1, \ldots, x_k, e)$$

• This is often referred to as computing the likelihood of the evidence
Queries: A posteriori belief

• Often we are interested in the conditional probability of a variable given the evidence
  \[ p(X \mid e) = \frac{p(X, e)}{p(e)} \]

• This is the **a posteriori belief** in \( X \), given evidence \( e \)

• A related task is computing the term \( p(X, e) \)
  – i.e., the likelihood of \( e \) and \( X = x \) for values of \( X \)
  – we can recover the a posteriori belief by

\[
p(X = x \mid e) = \frac{p(X = x, e)}{\sum_{x'} p(X = x', e)}
\]
A posteriori belief

This query is useful in many cases:

• **Prediction**: what is the probability of an outcome given the starting condition
  – Target is a descendent of the evidence

• **Diagnosis**: what is the probability of disease/fault given symptoms
  – Target is an ancestor of the evidence

• As we shall see, the direction between variables does not restrict the directions of the queries
  – Probabilistic inference can combine evidence form all parts of the network
Queries: A posteriori joint

• In this query, we are interested in the conditional probability of several variables, given the evidence

\[ P(X, Y, \ldots \mid e) \]

• Note that the size of the answer to query is exponential in the number of variables in the joint
Queries: MAP

- In this query we want to find the **maximum a posteriori** assignment for some variable of interest (say $X_1,\ldots,X_l$)
- That is, $x_1,\ldots,x_l$ maximize the probability

$$P(x_1,\ldots,x_l | e)$$

- Note that this is equivalent to maximizing

$$P(x_1,\ldots,x_l, e)$$
Queries: MAP

We can use MAP for:

- **Classification**
  - find most likely label, given the evidence

- **Explanation**
  - What is the most likely scenario, given the evidence
Queries: MAP

Cautionary note:
- The MAP depends on the set of variables
- Example:
  - MAP of $X$
  - MAP of $(X, Y)$
Complexity of Inference

**Thm:**

Computing $P(X = x)$ in a Bayesian network is NP-hard

Not surprising, since we can simulate Boolean gates.
Hardness

- Hardness does not mean we cannot solve inference
  - It implies that we cannot find a general procedure that works efficiently for all networks
  - For particular families of networks, we can have provably efficient procedures
Approaches to inference

• Exact inference
  – Inference in Simple Chains
  – Variable elimination
  – Clustering / join tree algorithms

• Approximate inference
  – Stochastic simulation / sampling methods
  – Markov chain Monte Carlo methods
  – Mean field theory
Inference in Simple Chains

How do we compute $P(X_2)$?

$$P(x_2) = \sum_{x_1} P(x_1, x_2) = \sum_{x_1} P(x_1)P(x_2 \mid x_1)$$
Inference in Simple Chains (cont.)

How do we compute $P(X_3)$?

$$P(x_3) = \sum_{x_2} P(x_2, x_3) = \sum_{x_2} P(x_2)P(x_3 \mid x_2)$$

• we already know how to compute $P(X_2)$...

$$P(x_2) = \sum_{x_1} P(x_1, x_2) = \sum_{x_1} P(x_1)P(x_2 \mid x_1)$$
How do we compute $P(X_n)$?

- Compute $P(X_1)$, $P(X_2)$, $P(X_3)$, ...
- We compute each term by using the previous one

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

Complexity:
- Each step costs $O(|\text{Val}(X_i)| \times |\text{Val}(X_{i+1})|)$ operations
- Compare to naïve evaluation, that requires summing over joint values of $n-1$ variables
Inference in Simple Chains (cont.)

- Suppose that we observe the value of $X_2 = x_2$
- How do we compute $P(X_1 | x_2)$?
  - Recall that we it suffices to compute $P(X_1, x_2)$

$$P(x_1, x_2) = P(x_2 | x_1)P(x_1)$$
Inference in Simple Chains (cont.)

- Suppose that we observe the value of $X_3 = x_3$
- How do we compute $P(X_1, x_3)$?
  \[
P(x_1, x_3) = P(x_1)P(x_3 | x_1)
  \]
- How do we compute $P(x_3 / x_1)$?
  \[
P(x_3 | x_1) = \sum_{x_2} P(x_2, x_3 | x_1) = \sum_{x_2} P(x_2 | x_1)P(x_3 | x_1, x_2)
  = \sum_{x_2} P(x_2 | x_1)P(x_3 | x_2)
  \]
Inference in Simple Chains (cont.)

Suppose that we observe the value of \( X_n = x_n \)

How do we compute \( P(X_1, x_n) \)?

\[
P(x_1, x_n) = P(x_1)P(x_n | x_1)
\]

We compute \( P(x_n / x_{n-1}), P(x_n / x_{n-2}), \ldots \) iteratively

\[
P(x_n | x_i) = \sum_{x_{i+1}} P(x_{i+1}, x_n | x_i)
\]

\[
= \sum_{x_i} P(x_{i+1} | x_i)P(x_n | x_{i+1})
\]
Inference in Simple Chains (cont.)

- Suppose that we observe the value of $X_n = x_n$
- We want to find $P(X_k/x_n)$
- How do we compute $P(x_k, x_n)$?

$$P(x_k, x_n) = P(x_k)P(x_n | x_k)$$

- We compute $P(X_k)$ by forward iterations
- We compute $P(x_n / X_k)$ by backward iterations
Elimination in Chains

- We now try to understand the simple chain example using first-order principles

\[ P(e) = \sum_{d} \sum_{c} \sum_{b} \sum_{a} P(a, b, c, d, e) \]
Elimination in Chains

By chain decomposition, we get

$$P(e) = \sum_{d} \sum_{c} \sum_{b} \sum_{a} P(a, b, c, d, e)$$

$$= \sum_{d} \sum_{c} \sum_{b} \sum_{a} P(a)P(b \mid a)P(c \mid b)P(d \mid c)P(e \mid d')$$
Elimination in Chains

• Rearranging terms ...

\[
P(e) = \sum_{d} \sum_{c} \sum_{b} \sum_{a} P(a)P(b \mid a)P(c \mid b)P(d \mid c)P(e \mid d')
\]

\[
= \sum_{d} \sum_{c} \sum_{b} P(c \mid b)P(d \mid c)P(e \mid d') \sum_{a} P(a)P(b \mid a)
\]
Elimination in Chains

- Now we can perform innermost summation

\[ P(e) = \sum_d \sum_c \sum_b P(c \mid b)P(d \mid c)P(e \mid d)\sum_a P(a)P(b \mid a) \]

\[ = \sum_d \sum_c \sum_b P(c \mid b)P(d \mid c)P(e \mid d)p(b) \]

- This summation, is exactly the first step in the forward iteration we describe before
Elimination in Chains

- Rearranging and then summing again, we get

\[
P(e) = \sum_{d} \sum_{c} \sum_{b} P(c \mid b)P(d \mid c)P(e \mid d)p(b)
\]

\[
= \sum_{d} \sum_{c} P(d \mid c)P(e \mid d)\sum_{b} P(c \mid b)p(b)
\]

\[
= \sum_{d} \sum_{c} P(d \mid c)P(e \mid d)p(c)
\]
Elimination in Chains with Evidence

- Similarly, we understand the backward pass

\[ P(a, e) = \sum_{b} \sum_{c} \sum_{d} P(a, b, c, d, e) \]

\[ = \sum_{b} \sum_{c} \sum_{d} P(a)P(b \mid a)P(c \mid b)P(d \mid c)P(e \mid d) \]
Elimination in Chains with Evidence

- Eliminating $d$, we get

$$P(a, e) = \sum_b \sum_c \sum_d P(a)P(b \mid a)P(c \mid b)P(d \mid c)P(e \mid d)$$

$$= \sum_b \sum_c \sum_d P(a)P(b \mid a)P(c \mid b) \sum_d P(d \mid c)P(e \mid d)$$

$$= \sum_b \sum_c \sum_d P(a)P(b \mid a)P(c \mid b)P(e \mid c)$$
Elimination in Chains with Evidence

- Eliminating $c$, we get

$$P(a, e) = \sum_b \sum_c P(a) P(b \mid a) P(c \mid b) P(e \mid c)$$

$$= \sum_b P(a) P(b \mid a) \sum_c P(c \mid b) P(e \mid c)$$

$$= \sum_b P(a) P(b \mid a) P(e \mid b)$$
Elimination in Chains with Evidence

- Finally, we eliminate $b$

\[
P(a, e) = \sum_{b} P(a)P(b \mid a)p(e \mid b)
\]

\[
= P(a)\sum_{b} P(b \mid a)p(e \mid b)
\]

\[
= P(a)P(e \mid a)
\]
Variable Elimination

General idea:
• Write query in the form

\[ P(X_n, e) = \sum_{x_k} \cdots \sum_{x_3} \sum_{x_2} \prod_i P(x_i \mid pa_i) \]

• Iteratively
  – Move all irrelevant terms outside of innermost sum
  – Perform innermost sum, getting a new term
  – Insert the new term into the product
A More Complex Example

• “Asia” network:
• We want to compute $P(d)$
• Need to eliminate: $v, s, x, t, l, a, b$

Initial factors

$$P(v)P(s)P(t | v)P(l | s)P(b | s)P(a | t, l)P(x | a)P(d | a, b)$$
- We want to compute \( P(d) \)
- Need to eliminate: \( v, s, x, t, l, a, b \)

**Initial factors**

\[
\begin{align*}
\mathbb{P}(v) \mathbb{P}(s) \mathbb{P}(t|v) \mathbb{P}(l|s) \mathbb{P}(b|s) \mathbb{P}(a|t,l) \mathbb{P}(x|a) \mathbb{P}(d|a,b)
\end{align*}
\]

Eliminate: \( v \)

Compute:

\[
f_v(t) = \sum_v \mathbb{P}(v) \mathbb{P}(t|v)
\]

\[
\Rightarrow f_v(t) \mathbb{P}(s) \mathbb{P}(l|s) \mathbb{P}(b|s) \mathbb{P}(a|t,l) \mathbb{P}(x|a) \mathbb{P}(d|a,b)
\]

**Note:** \( f_v(t) = P(t) \)

In general, result of elimination is not necessarily a probability term
• We want to compute $P(d)$
• Need to eliminate: $s, x, t, l, a, b$

• Initial factors

\[
P(v)P(s)P(t \mid v)P(l \mid s)P(b \mid s)P(a \mid t, l)P(x \mid a)P(d \mid a, b)
\]

\[
\Rightarrow f_v(t)P(s)P(l \mid s)P(b \mid s)P(a \mid t, l)P(x \mid a)P(d \mid a, b)
\]

Eliminate: $s$

Compute:

\[
f_s(b, l) = \sum_s P(s)P(b \mid s)P(l \mid s)
\]

\[
\Rightarrow f_v(t)f_s(b, l)P(a \mid t, l)P(x \mid a)P(d \mid a, b)
\]

Summing on $s$ results in a factor with two arguments $f_s(b, l)$

In general, result of elimination may be a function of several variables
• We want to compute $P(d)$
• Need to eliminate: $x, t, l, a, b$

Initial factors

$$P(v)P(s)P(t \mid v)P(l \mid s)P(b \mid s)P(a \mid t, l)P(x \mid a)P(d \mid a, b)$$

$$\Rightarrow f_v(t)P(s)P(l \mid s)P(b \mid s)P(a \mid t, l)P(x \mid a)P(d \mid a, b)$$

$$\Rightarrow f_v(t)f_s(b, l)P(a \mid t, l)P(x \mid a)P(d \mid a, b)$$

Eliminate: $x$

Compute:

$$f_x(a) = \sum_x P(x \mid a)$$

$$\Rightarrow f_v(t)f_s(b, l)f_x(a)P(a \mid t, l)P(d \mid a, b)$$

Note: $f_x(a) = 1$ for all values of $a$!!
• We want to compute $P(d)$
• Need to eliminate: $t,l,a,b$

• Initial factors

$$
P(v)P(s)P(t \mid v)P(l \mid s)P(b \mid s)P(a \mid t,l)P(x \mid a)P(d \mid a,b)$$

$$\Rightarrow f_v(t)P(s)P(l \mid s)P(b \mid s)P(a \mid t,l)P(x \mid a)P(d \mid a,b)$$

$$\Rightarrow f_v(t)f_s(b,l)P(a \mid t,l)P(x \mid a)P(d \mid a,b)$$

$$\Rightarrow f_v(t)f_s(b,l)f_x(a)P(a \mid t,l)P(d \mid a,b)$$

Eliminate: $t$

Compute: $f_t(a,l) = \sum_t f_v(t)P(a \mid t,l)$

$$\Rightarrow f_s(b,l)f_x(a)f_t(a,l)P(d \mid a,b)$$
• We want to compute $P(d)$
• Need to eliminate: $l,a,b$

• Initial factors

$$P(v)P(s)P(t \mid v)P(l \mid s)P(b \mid s)P(a \mid t,l)P(x \mid a)P(d \mid a,b)$$
$$\Rightarrow f_v(t)P(s)P(l \mid s)P(b \mid s)P(a \mid t,l)P(x \mid a)P(d \mid a,b)$$
$$\Rightarrow f_v(t)f_s(b,l)P(a \mid t,l)P(x \mid a)P(d \mid a,b)$$
$$\Rightarrow f_v(t)f_s(b,l)f_x(a)P(a \mid t,l)P(d \mid a,b)$$
$$\Rightarrow \boxed{f_s(b,l)f_x(a)f_t(a,l)}P(d \mid a,b)$$

Eliminate: $l$
Compute: $f_l(a,b) = \sum_l f_s(b,l)f_t(a,l)$
$$\Rightarrow \boxed{f_l(a,b) f_x(a)}P(d \mid a,b)$$
• We want to compute $P(d)$
• Need to eliminate: $b$

• Initial factors

\[
P(v)P(s)P(t \mid v)P(l \mid s)P(b \mid s)P(a \mid t, l)P(x \mid a)P(d \mid a, b)
\Rightarrow f_v(t)P(s)P(l \mid s)P(b \mid s)P(a \mid t, l)P(x \mid a)P(d \mid a, b)
\Rightarrow f_v(t)f_s(b, l)P(a \mid t, l)P(x \mid a)P(d \mid a, b)
\Rightarrow f_v(t)f_s(b, l)f_x(a)P(a \mid t, l)P(d \mid a, b)
\Rightarrow f_s(b, l)f_x(a)f_t(a, l)P(d \mid a, b)
\Rightarrow f_l(a, b)f_x(a)P(d \mid a, b) \Rightarrow f_a(b, d) \Rightarrow f_b(d')
\]

Eliminate: $a, b$
Compute:

\[
f_a(b, d) = \sum_a f_l(a, b)f_x(a)P(d \mid a, b) \quad f_b(d') = \sum_b f_a(b, d')
\]
Variable Elimination

- We now understand variable elimination as a sequence of **rewriting** operations

- Actual computation is done in elimination step

- Exactly the same computation procedure applies to Markov networks

- Computation depends on order of elimination
Complexity of variable elimination

• Suppose in one elimination step we compute

\[ f_x(y_1, \ldots, y_k) = \sum_{x} f'_x(x, y_1, \ldots, y_k) \]

\[ f'_x (x, y_1, \ldots, y_k) = \prod_{i=1}^{m} f_i (x, y_{1,i}, \ldots, y_{1,i}) \]

This requires

- Multiplications:
  \[ m \cdot |\text{Val}(X)| \cdot \prod_{i} |\text{Val}(Y_i)| \]
  - For each value for \( x, y_1, \ldots, y_k \) we do \( m \) multiplications

- Additions:
  \[ |\text{Val}(X)| \cdot \prod_{i} |\text{Val}(Y_i)| \]
  - For each value of \( y_1, \ldots, y_k \), we do \(|\text{Val}(X)|\) additions

Complexity is exponential in number of variables in the intermediate factor!
Understanding Variable Elimination

• We want to select “good” elimination orderings that reduce complexity

• We start by attempting to understand variable elimination via the graph we are working with

• This will reduce the problem of finding good ordering to graph-theoretic operation that is well-understood
Undirected graph representation

• At each stage of the procedure, we have an algebraic term that we need to evaluate
• In general this term is of the form:
  \[ P(x_1, \ldots, x_k) = \sum \cdots \sum \prod f_i(Z_i) \]
  where \( Z_i \) are sets of variables

• We now plot a graph where there is undirected edge \( X--Y \) if \( X,Y \) are arguments of some factor
  – that is, if \( X,Y \) are in some \( Z_i \)

• Note: this is the Markov network that describes the probability on the variables we did not eliminate yet
Chordal Graphs

- elimination ordering $\Rightarrow$ undirected chordal graph

Graph:
- Maximal cliques are factors in elimination
- Factors in elimination are cliques in the graph
- Complexity is exponential in size of the largest clique in graph
Induced Width

- The size of the largest clique in the induced graph is thus an indicator for the complexity of variable elimination.

- This quantity is called the **induced width** of a graph according to the specified ordering.

- Finding a good ordering for a graph is equivalent to finding the minimal induced width of the graph.
General Networks

- From graph theory:

**Thm:**
- Finding an ordering that minimizes the induced width is NP-Hard

However,
- There are reasonable heuristic for finding “relatively” good ordering
- There are provable approximations to the best induced width
- If the graph has a small induced width, there are algorithms that find it in polynomial time
Elimination on Trees

- Formally, for any tree, there is an elimination ordering with induced width $= 1$

**Thm**

- Inference on trees is linear in number of variables
PolyTrees

- A polytree is a network where there is at most one path from one variable to another

**Thm:**
- Inference in a polytree is linear in the representation size of the network
  - This assumes tabular CPT representation
Approaches to inference

• Exact inference
  – Inference in Simple Chains
  – Variable elimination
  – Clustering / join tree algorithms

• **Approximate inference**
  – Stochastic simulation / sampling methods
  – Markov chain Monte Carlo methods
  – Mean field theory
Stochastic simulation

- Suppose you are given values for some subset of the variables, G, and want to infer values for unknown variables, U
- Randomly generate a very large number of instantiations from the BN
  - Generate instantiations for all variables – start at root variables and work your way “forward”
- Only keep those instantiations that are consistent with the values for G
- Use the frequency of values for U to get estimated probabilities
- Accuracy of the results depends on the size of the sample (asymptotically approaches exact results)
Markov chain Monte Carlo methods

• So called because
  – Markov chain – each instance generated in the sample is dependent on the previous instance
  – Monte Carlo – statistical sampling method

• Perform a random walk through variable assignment space, collecting statistics as you go
  – Start with a random instantiation, consistent with evidence variables
  – At each step, for some nonevidence variable, randomly sample its value, consistent with the other current assignments

• Given enough samples, MCMC gives an accurate estimate of the true distribution of values
References

• Nir Friedman’s excellent lecture notes, [http://www.cs.huji.ac.il/~pmai/](http://www.cs.huji.ac.il/~pmai/)