Machine Learning

10-701, Fall 2016

Graphical Models
and
Exact Inference

Eric Xing

Lecture 17, November 7, 2016

Reading: Chap. 8, C.B book
Recap of Basic Prob. Concepts

- **Representation:** what is the joint probability dist. on multiple variables?
  \[ P(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8) \]
  - How many state configurations in total? --- \(2^8\)
  - Are they all needed to be represented?
  - Do we get any scientific/medical insight?

- **Learning:** where do we get all this probabilities?
  - Maximal-likelihood estimation? but how many data do we need?
  - Are there other est. principles?
  - Where do we put domain knowledge in terms of plausible relationships between variables, and plausible values of the probabilities?

- **Inference:** If not all variables are observable, how to compute the conditional distribution of latent variables given evidence?
  - Computing \(p(H|A)\) would require summing over all \(2^6\) configurations of the unobserved variables
What is a Graphical Model?

--- Multivariate Distribution in High-D Space

- A possible world for cellular signal transduction:

Receptor A \( x_1 \)  
Kinase C \( x_3 \)  
TF F \( x_6 \)  
Gene G \( x_7 \)  
Receptor B \( x_2 \)  
Kinase D \( x_4 \)  
Kinase E \( x_5 \)  
Gene H \( x_8 \)
GM: Structure Simplifies Representation

- Dependencies among variables

![Diagram of molecular pathways involving receptors, kinases, transcription factors, and genes with variables X1 to X8.](image)
If \( X_i \)'s are **conditionally independent** (as described by a PGM), the joint can be factored to a product of simpler terms, e.g.,

\[
P(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8)
\]

\[
= P(X_1) P(X_2) P(X_3| X_1) P(X_4| X_2) P(X_5| X_2) P(X_6| X_3, X_4) P(X_7| X_6) P(X_8| X_5, X_6)
\]

Stay tune for what are these independencies!

**Why we may favor a PGM?**

- Incorporation of domain knowledge and causal (logical) structures
  
  \[1+1+2+2+2+4+2+4=18, \text{ a 16-fold reduction from } 2^8 \text{ in representation cost!}\]
More Data Integration

- Text + Image + Network → Holistic Social Media

- Genome + Proteome + Transcritome + Phenome + … → PanOmic Biology
If $X_i$'s are **conditionally independent** (as described by a **PGM**), the joint can be factored to a product of simpler terms, e.g.,

$$P(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8) = P(X_2) P(X_4|X_2) P(X_5|X_2) P(X_7|X_5) P(X_8|X_6, X_7, X_8)$$

- Why we may favor a PGM?
  - Incorporation of domain knowledge and causal (logical) structures
    - $2+2+4+4+4+8+4+8=36$, an 8-fold reduction from $2^8$ in representation cost!
  - Modular combination of heterogeneous parts – data fusion
Rational Statistical Inference

The Bayes Theorem:

\[ p(h \mid d) = \frac{p(d \mid h)p(h)}{\sum_{h' \in H} p(d \mid h')p(h')} \]

- Posterior probability
- Likelihood
- Prior probability
- Sum over space of hypotheses

- This allows us to capture uncertainty about the model in a principled way
- But how can we specify and represent a complicated model?
  - Typically the number of genes need to be modeled are in the order of thousands!
GM: MLE and Bayesian Learning

- Probabilistic statements of $\Theta$ is conditioned on the values of the observed variables $A_{\text{obs}}$ and prior $p(\Theta | \chi)$

$A = (A,B,C,D,E,...) = (T,F,T,T,F,...)$

$\cdots$

$(A,B,C,D,E,...) = (F,T,T,F,...)$

$\Theta_{\text{Bayes}} = \int \Theta p(\Theta | A, \chi) d\Theta$
If $X_i$'s are **conditionally independent** (as described by a PGM), the joint can be factored to a product of simpler terms, e.g.,

$$P(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8) = P(X_1) P(X_2) P(X_3| X_1) P(X_4| X_2) P(X_5| X_2)$$
$$P(X_6| X_3, X_4) P(X_7| X_6) P(X_8| X_5, X_6)$$

**Why we may favor a PGM?**

- Incorporation of domain knowledge and causal (logical) structures
  
  $2+2+4+4+4+8+4+8=36$, an 8-fold reduction from $2^8$ in representation cost!

- Modular combination of heterogeneous parts – data fusion

- Bayesian Philosophy
  - Knowledge meets data
So What Is a PGM After All?

In a nutshell:

PGM = Multivariate Statistics + Structure

GM = Multivariate Obj. Func. + Structure
So What Is a PGM After All?

- The informal blurb:
  - It is a smart way to write/specify-compose/design exponentially-large probability distributions without paying an exponential cost, and at the same time endow the distributions with structured semantics

- A more formal description:
  - It refers to a family of distributions on a set of random variables that are compatible with all the probabilistic independence propositions encoded by a graph that connects these variables
Two types of GMs

- Directed edges give causality relationships (Bayesian Network or Directed Graphical Model):

\[ P(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8) = P(X_1) P(X_2) P(X_3|X_1) P(X_4|X_2) P(X_5|X_2) P(X_6|X_3, X_4) P(X_7|X_6) P(X_8|X_5, X_6) \]

- Undirected edges simply give correlations between variables (Markov Random Field or Undirected Graphical model):

\[ P(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8) = \frac{1}{Z} \exp\{E(X_1)+E(X_2)+E(X_3, X_1)+E(X_4, X_2)+E(X_5, X_2) + E(X_6, X_3, X_4)+E(X_7, X_6)+E(X_8, X_5, X_6)\} \]
Towards structural specification of probability distribution

- Separation properties in the graph imply independence properties about the associated variables
- For the graph to be useful, any conditional independence properties we can derive from the graph should hold for the probability distribution that the graph represents

- The Equivalence Theorem

For a graph $G$,

Let $\mathcal{D}_1$ denote the family of all distributions that satisfy $I(G)$,

Let $\mathcal{D}_2$ denote the family of all distributions that factor according to $G$,

Then $\mathcal{D}_1 \equiv \mathcal{D}_2$. 
Bayesian Networks

Structure: **DAG**

- Meaning: a node is **conditionally independent** of every other node in the network outside its **Markov blanket**

- Local conditional distributions (**CPD**) and the **DAG** completely determine the **joint** dist.

- Give **causality** relationships, and facilitate a **generative** process
Markov Random Fields

Structure: *undirected graph*

- Meaning: a node is *conditionally independent* of every other node in the network given its *Directed neighbors*.

- Local contingency functions (*potentials*) and the *cliques* in the graph completely determine the *joint* dist.

- Give *correlations* between variables, but no explicit way to generate samples.
GMs are your old friends

Density estimation
- Parametric and nonparametric methods

Regression
- Linear, conditional mixture, nonparametric

Classification
- Generative and discriminative approach

Clustering
An (incomplete) genealogy of graphical models

(Picture by Zoubin Ghahramani and Sam Roweis)
Fancier GMs: machine translation

The HM-BiTAM model
(B. Zhao and E.P Xing, ACL 2006)
Fancier GMs: solid state physics

Ising/Potts model
Why graphical models

- A language for communication
- A language for computation
- A language for development

Origins:
- Wright 1920’s
- Independently developed by Spiegelhalter and Lauritzen in statistics and Pearl in computer science in the late 1980’s
Why graphical models

- **Probability theory** provides the glue whereby the parts are combined, ensuring that the system as a whole is consistent, and providing ways to interface models to data.

- The **graph theoretic** side of graphical models provides both an intuitively appealing interface by which humans can model highly-interacting sets of variables as well as a data structure that lends itself naturally to the design of efficient general-purpose algorithms.

- Many of the classical multivariate probabilistic systems studied in fields such as statistics, systems engineering, information theory, pattern recognition and statistical mechanics are special cases of the general graphical model formalism.

- The graphical model framework provides a way to view all of these systems as instances of a common underlying formalism.
Bayesian Network: Factorization Theorem

Theorem:
Given a DAG, the most general form of the probability distribution that is consistent with the (probabilistic independence properties encoded in the) graph factors according to “node given its parents”:

\[ P(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8) = P(X_1) P(X_2) P(X_3| X_1) P(X_4| X_2) P(X_5| X_2) P(X_6| X_3, X_4) P(X_7| X_6) P(X_8| X_5, X_6) \]

where \( X_{\pi_i} \) is the set of parents of \( x_i \). \( d \) is the number of nodes (variables) in the graph.
Example: a pedigree of people

- Genetic Pedigree
Specification of a BN

- There are two components to any GM:
  - the *qualitative* specification
  - the *quantitative* specification

| C | D | P(F | C, D) |
|---|---|-----------|
| c | d | 0.9       | 0.1       |
| c | ̅d | 0.2     | 0.8       |
| ̅c | d | 0.9       | 0.1       |
| ̅c | ̅d | 0.01     | 0.99      |
Qualitative Specification

Where does the qualitative specification come from?

- Prior knowledge of causal relationships
- Prior knowledge of modular relationships
- Assessment from experts
- Learning from data
- We simply link a certain architecture (e.g. a layered graph)
- …
Bayesian Network: Factorization Theorem

Theorem:

Given a DAG, the most general form of the probability distribution that is consistent with the (probabilistic independence properties encoded in the) graph factors according to “node given its parents”:

\[
P(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8) = P(X_1) P(X_2) P(X_3 | X_1) P(X_4 | X_2) P(X_5 | X_2) P(X_6 | X_3, X_4) P(X_7 | X_6) P(X_8 | X_5, X_6)
\]

where \( X_{\pi_i} \) is the set of parents of \( x_i \). \( d \) is the number of nodes (variables) in the graph.
Local Structures & Independencies

- **Common parent**
  - Fixing B decouples A and C
    "given the level of gene B, the levels of A and C are independent"

- **Cascade**
  - Knowing B decouples A and C
    "given the level of gene B, the level gene A provides no extra prediction value for the level of gene C"

- **V-structure**
  - Knowing C couples A and B
    because A can "explain away" B w.r.t. C
    "If A correlates to C, then chance for B to also correlate to B will decrease"

- The language is compact, the concepts are rich!
A simple justification
Graph separation criterion

- D-separation criterion for Bayesian networks (D for Directed edges):

**Definition**: variables x and y are *D-separated* (conditionally independent) given z if they are separated in the *moralized* ancestral graph

- Example:

  
  ![original graph](image1)

  $\Rightarrow$

  ![ancestral](image2)

  $\Rightarrow$

  ![moral ancestral](image3)
Local Markov properties of DAGs

Structure: DAG

- Meaning: a node is conditionally independent of every other node in the network outside its Markov blanket

- Local conditional distributions (CPD) and the DAG completely determine the joint dist.

- Give causality relationships, and facilitate a generative process
Global Markov properties of DAGs

- $X$ is d-separated (directed-separated) from $Z$ given $Y$ if we can't send a ball from any node in $X$ to any node in $Z$ using the "Bayes-ball" algorithm illustrated bellow (and plus some boundary conditions):

- Defn: $I(G) =$ all independence properties that correspond to d-separation:

$$I(G) = \{X \perp Z \mid Y : \text{dsep}_G(X; Z \mid Y)\}$$

- D-separation is sound and complete
Example:

- Complete the $I(G)$ of this graph:

Essentially: A BN is a database of Pr. Independence statements among variables.
Towards quantitative specification of probability distribution

- Separation properties in the graph imply independence properties about the associated variables.
- For the graph to be useful, any conditional independence properties we can derive from the graph should hold for the probability distribution that the graph represents.

**The Equivalence Theorem**

For a graph $G$,

Let $\mathcal{D}_1$ denote the family of all distributions that satisfy $I(G)$,

Let $\mathcal{D}_2$ denote the family of all distributions that factor according to $G$.

Then $\mathcal{D}_1 \equiv \mathcal{D}_2$. 
Conditional probability tables (CPTs)

\[
P(a,b,c,d) = P(a)P(b)P(c|a,b)P(d|c)
\]

<table>
<thead>
<tr>
<th>a^0</th>
<th>0.75</th>
</tr>
</thead>
<tbody>
<tr>
<td>a^1</td>
<td>0.25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>b^0</th>
<th>0.33</th>
</tr>
</thead>
<tbody>
<tr>
<td>b^1</td>
<td>0.67</td>
</tr>
</tbody>
</table>

\[
\begin{array}{cccc}
  & a^0b^0 & a^0b^1 & a^1b^0 & a^1b^1 \\
c^0 & 0.45 & 1 & 0.9 & 0.7 \\
c^1 & 0.55 & 0 & 0.1 & 0.3 \\
\end{array}
\]

\[
\begin{array}{cc}
c^0 & c^1 \\
d^0 & 0.3 & 0.5 \\
d^1 & 0.7 & 0.5 \\
\end{array}
\]
Conditional probability density func. (CPDs)

\[ P(a, b, c, d) = P(a)P(b)P(c|a, b)P(d|c) \]

\[ A \sim N(\mu_a, \Sigma_a) \quad B \sim N(\mu_b, \Sigma_b) \]

\[ C \sim N(A + B, \Sigma_c) \]

\[ D \sim N(\mu_a + C, \Sigma_a) \]
Conditional Independencies

What is this model

1. When Y is observed?
2. When Y is unobserved?
Conditionally Independent Observations

Data = \{y_1, \ldots y_n\}

Model parameters

\theta

X_1 \quad X_2 \quad \cdots \quad X_{n-1} \quad X_n
“Plate” Notation

Plate = rectangle in graphical model

Variables within a plate are replicated in a conditionally independent manner

Data = \{x_1, \ldots, x_n\}

Model parameters

\( \theta \)

\( X_i \)
Example: Gaussian Model

Generative model:
\[
p(x_1,\ldots,x_n \mid \mu, \sigma) = \prod_{i=1}^{n} p(x_i \mid \mu, \sigma)
= p(\text{data} \mid \text{parameters})
= p(D \mid \theta)
\]
where \( \theta = \{\mu, \sigma\} \)

- Likelihood \[= p(\text{data} \mid \text{parameters})
= p( D \mid \theta )
= L (\theta)\]

- Likelihood tells us how likely the observed data are conditioned on a particular setting of the parameters
  - Often easier to work with \( \log L (\theta) \)
Bayesian models

$\theta$

$x_i$

$i=1:n$
Summary

- Represent dependency structure with a directed acyclic graph
  - Node <-> random variable
  - Edges encode dependencies
    - Absence of edge -> conditional independence
  - Plate representation
  - A GM is a database of probability. Independence statement on variables

- The factorization theorem of the joint probability
  - Local specification → globally consistent distribution
  - Local representation for exponentially complex state-space
  - It is a smart way to write/specify/compose/design exponentially-large probability distributions without paying an exponential cost, and at the same time endow the distributions with structured semantics

- Support efficient inference and learning
Inference and Learning

- We now have compact representations of probability distributions: **BN**
- A BN $M$ describes a unique probability distribution $P$
- Typical tasks:
  - Task 1: How do we answer *queries* about $P$?
    - We use *inference* as a name for the process of computing answers to such queries
  - Task 2: How do we estimate a *plausible model* $M$ from data $D$?
    - We use *learning* as a name for the process of obtaining point estimate of $M$.
    - But for *Bayesian*, they seek $p(M | D)$, which is actually an *inference* problem.
    - When not all variables are observable, even computing point estimate of $M$ need to do *inference* to impute the *missing data*. 
Inferential Query 1: Likelihood

- Most of the queries one may ask involve evidence

  - Evidence $x_v$ is an assignment of values to a set $X_v$ of nodes in the GM over variable set $X=\{X_1, X_2, \ldots, X_n\}$
  - Without loss of generality $X_v=\{X_{k+1}, \ldots, X_n\}$,
  - Write $X_H=X\setminus X_v$ as the set of hidden variables, $X_H$ can be $\emptyset$ or $X$

- Simplest query: compute probability of evidence

  $$P(x_v) = \sum_{x_H} P(X_H, X_v) = \sum_{x_1} \ldots \sum_{x_k} P(x_1, \ldots, x_k, x_v)$$

  - this is often referred to as computing the likelihood of $x_v$
Inferential Query 2: Conditional Probability

- Often we are interested in the **conditional probability distribution** of a variable given the evidence

\[
P(X_H \mid X_V = x_v) = \frac{P(X_H, x_v)}{P(x_v)} = \sum_{x_H} \frac{P(X_H, x_v)}{P(x_v)}
\]

- this is the **a posteriori belief** in \(X_H\), given evidence \(x_v\)

- We usually query a subset \(Y\) of all hidden variables \(X_H = \{Y, Z\}\) and "don't care" about the remaining, \(Z\):

\[
P(Y \mid x_v) = \sum_z P(Y, Z = z \mid x_v)
\]

- the process of summing out the "don't care" variables \(z\) is called **marginalization**, and the resulting \(P(Y \mid x_v)\) is called a **marginal** prob.
Applications of a *posteriori* Belief

- **Prediction**: what is the probability of an outcome given the starting condition
  - the query node is a descendent of the evidence

- **Diagnosis**: what is the probability of disease/fault given symptoms
  - the query node an ancestor of the evidence

- **Learning** under partial observation
  - fill in the unobserved values under an "EM" setting (more later)

- The directionality of information flow between variables is not restricted by the directionality of the edges in a GM
  - probabilistic inference can combine evidence form all parts of the network
In this query we want to find the most probable joint assignment (MPA) for some variables of interest.

Such reasoning is usually performed under some given evidence $x_v$, and ignoring (the values of) other variables $Z$:

$$Y^* | x_v = \arg \max_y P(Y | x_v) = \arg \max_y \sum_z P(Y, Z = z | x_v)$$

this is the maximum a posteriori configuration of $Y$. 
Complexity of Inference

Thm:

Computing $P(X_H=x_H | x_v)$ in an arbitrary GM is NP-hard

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

- **Exact inference algorithms**
  - The elimination algorithm
  - Belief propagation
  - The junction tree algorithms (but will not cover in detail here)

- **Approximate inference techniques**
  - Variational algorithms
  - Stochastic simulation / sampling methods
  - Markov chain Monte Carlo methods
Marginalization and Elimination

- A food web:

What is the probability that hawks are leaving given that the grass condition is poor?

**Query: \( P(h) \)**

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

- By chain decomposition, we get

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

a naïve summation needs to enumerate over an exponential number of terms
Variable Elimination

- Query: $P(A \mid h)$
  - Need to eliminate: $B,C,D,E,F,G,H$

- Initial factors:
  $$P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)P(h \mid e,f)$$

- Choose an elimination order: $H,G,F,E,D,C,B$

- Step 1:
  - **Conditioning** (fix the evidence node (i.e., $h$) on its observed value (i.e., $\tilde{h}$)):
    $$m_h(e,f') = p(h = \tilde{h} \mid e,f')$$
  - This step is isomorphic to a marginalization step:
    $$m_h(e,f') = \sum_h p(h \mid e,f')\delta(h = \tilde{h})$$
Example: Variable Elimination

- Query: $P(B \mid h)$
  - Need to eliminate: $B, C, D, E, F, G$

- Initial factors:

$$P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c, d)P(f \mid a)P(g \mid e)P(h \mid e, f)$$

$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c, d)P(f \mid a)P(g \mid e)m_h(e, f)$$

- Step 2: Eliminate $G$
  - compute

$$m_g(e) = \sum_g p(g \mid e) = 1$$

$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c, d)P(f \mid a)m_g(e)m_h(e, f)$$

$$= P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c, d)P(f \mid a)m_h(e, f)$$
Example: Variable Elimination

- Query: $P(B \mid h)$
  - Need to eliminate: $B, C, D, E, F$

- Initial factors:

  $$P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)P(h \mid e, f)$$

  $$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)m_h(e, f)$$

  $$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)m_h(e, f)$$

- Step 3: Eliminate $F$
  - compute

  $$m_f(e, a) = \sum_{f} p(f \mid a)m_h(e, f)$$

  $$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)m_f(a, e)$$
Example: Variable Elimination

- Query: $P(B \mid h)$
  - Need to eliminate: $B, C, D, E$

- Initial factors:
  $$P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c, d)P(f \mid a)P(g \mid e)P(h \mid e, f)$$
  $$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c, d)P(f \mid a)P(g \mid e)m_h(e, f)$$
  $$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c, d)P(f \mid a)m_h(e, f)$$
  $$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c, d)m_f(a, e)$$

- Step 4: Eliminate $E$
  - compute
    $$m_e(a, c, d) = \sum_e p(e \mid c, d)m_f(a, e)$$
  $$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)m_e(a, c, d)$$
Example: Variable Elimination

- **Query:** $P(B \mid h)$
  - Need to eliminate: $B, C, D$

- **Initial factors:**
  
  $P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c, d)P(f \mid a)P(g \mid e)P(h \mid e, f)$
  
  $\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c, d)P(f \mid a)P(g \mid e)m_h(e, f)$
  
  $\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c, d)P(f \mid a)m_h(e, f')$
  
  $\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c, d)m_f(a, e)$
  
  $\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)m_e(a, c, d)$

- **Step 5: Eliminate $D$**
  - compute
    
    $m_d(a, c) = \sum_d p(d \mid a)m_e(a, c, d)$
    
    $\Rightarrow P(a)P(b)P(c \mid d)m_d(a, c)$
Example: Variable Elimination

- **Query:** \( P(B \mid h) \)
  - Need to eliminate: \( B, C \)

- **Initial factors:**

\[
P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)P(h \mid e,f)
\]

\[
\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)m_h(e,f)
\]

\[
\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c,d)P(f \mid a)m_h(e,f)
\]

\[
\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c,d)m_f(a,e)
\]

\[
\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)m_c(a,c,d)
\]

\[
\Rightarrow P(a)P(b)P(c \mid d)m_d(a,c)
\]

- **Step 6: Eliminate** \( C \)
  - compute

\[
m_c(a,b) = \sum_c p(c \mid b)m_d(a,c)
\]

\[
\Rightarrow P(a)P(b)P(c \mid d)m_d(a,c)
\]
Example: Variable Elimination

- **Query:** $P(B \mid h)$
  - Need to eliminate: $B$

- **Initial factors:**

  $$
P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c, d)P(f \mid a)P(g \mid e)P(h \mid e, f)
  \Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c, d)P(f \mid a)P(g \mid e)m_h(e, f)
  \Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c, d)P(f \mid a)m_h(e, f)
  \Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c, d)m_f(a, e)
  \Rightarrow P(a)P(b)P(c \mid d)m_e(a, c, d)
  \Rightarrow P(a)P(b)m_d(a, c)
  \Rightarrow P(a)P(b)m_c(a, b)
  \Rightarrow P(a)m_b(a)
  \Rightarrow P(a)m_b(a)
$$

- **Step 7: Eliminate $B$**
  - compute

  $$m_b(a) = \sum_b p(b)m_c(a, b)$$

  $$\Rightarrow P(a)m_b(a)$$
Example: Variable Elimination

- Query: $P(B \mid h)$
  - Need to eliminate: $B$

- Initial factors:

\[
P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c, d)P(f \mid a)P(g \mid e)P(h \mid e, f)
\]

$\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c, d)P(f \mid a)P(g \mid e)m_h(e, f)$

$\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c, d)P(f \mid a)m_h(e, f)$

$\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c, d)m_f(a, e)$

$\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)m_e(a, c, d)$

$\Rightarrow P(a)P(b)P(c \mid d)m_d(a, c)$

$\Rightarrow P(a)P(b)m_c(a, b)$

$\Rightarrow P(a)m_b(a)$

- Step 8: Wrap-up

\[
p(a, \tilde{h}) = p(a)m_b(a), \quad p(\tilde{h}) = \sum_a p(a)m_b(a)
\]

$\Rightarrow P(a \mid \tilde{h}) = \frac{p(a)m_b(a)}{\sum_a p(a)m_b(a)}$
Complexity of variable elimination

- Suppose in one elimination step we compute

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

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

This requires

- \( k \cdot |\text{Val}(X)| \cdot \prod_{i} |\text{Val}(Y_{c_i})| \) multiplications
  - For each value of \( x, y_1, \ldots, y_k \), we do \( k \) multiplications

- \( |\text{Val}(X)| \cdot \prod_{i} |\text{Val}(Y_{c_i})| \) additions
  - 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
Induced dependency during marginalization is captured in elimination cliques

- Summation $\leftrightarrow$ elimination
- Intermediate term $\leftrightarrow$ elimination clique

\[
P(a)P(b)P(c|b)P(d|a)P(e|c, d)P(f|a)P(g|e)P(h|e, f)
\]
\[
\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c, d)P(f|a)P(g|e)\phi_h(e, f)
\]
\[
\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c, d)P(f|a)\phi_g(e)\phi_h(e, f)
\]
\[
\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c, d)\phi_f(a, e)
\]
\[
\Rightarrow P(a)P(b)P(c|b)P(d|a)\phi_e(a, c, d)
\]
\[
\Rightarrow P(a)P(b)P(c|b)\phi_d(a, c)
\]
\[
\Rightarrow P(a)P(b)\phi_c(a, b)
\]
\[
\Rightarrow P(a)\phi_b(a)
\]
\[
\Rightarrow \phi(a)
\]

- Can this lead to a generic inference algorithm?
From Elimination to Message Passing

- Elimination $\equiv$ message passing on a clique tree

$\sum_{e} p(e | c, d) m_g(e) m_f(a, e)$

- Messages can be reused
From Elimination to Message Passing

- Elimination \(\equiv\) message passing on a **clique tree**
  - Another query ...

- Messages \(m_f\) and \(m_h\) are reused, others need to be recomputed
From elimination to message passing

- Recall **ELIMINATION** algorithm:
  - Choose an ordering $\mathcal{Z}$ in which query node $f$ is the final node
  - Place all potentials on an active list
  - Eliminate node $i$ by removing all potentials containing $i$, take sum/product over $x_i$
  - Place the resultant factor back on the list

- For a **TREE** graph:
  - Choose query node $f$ as the root of the tree
  - View tree as a directed tree with edges pointing towards from $f$
  - Elimination ordering based on depth-first traversal
  - Elimination of each node can be considered as message-passing (or Belief Propagation) directly along tree branches, rather than on some transformed graphs
  - thus, we can use the tree itself as a data-structure to do general inference!!
Message passing for trees

Let $m_{ij}(x_i)$ denote the factor resulting from eliminating variables from below up to $i$, which is a function of $x_i$:

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

This is reminiscent of a message sent from $j$ to $i$.

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

$p(x_f) \propto \psi(x_f) \prod_{e \in N(f)} m_{ef}(x_f)$

$m_{ij}(x_i)$ represents a "belief" of $x_i$ from $x_j$!
Elimination on trees is equivalent to message passing along tree branches!

\[
m_{ji}(x_i) = \sum_{x_j} \psi(x_j) \psi(x_i, x_j) \prod_{k \in N(j) \setminus i} m_{kj}(x_j)
\]
The message passing protocol:

- A two-pass algorithm:
Belief Propagation (SP-algorithm): Sequential implementation

**SUM-PRODUCT**(T, E)

**EVIDENCE**(E)

\[ f = \text{CHOOSEROOT}(V) \]

for \( e \in \mathcal{N}(f) \)

\[ \text{COLLECT}(f, e) \]

for \( e \in \mathcal{N}(f) \)

\[ \text{DISTRIBUTE}(f, e) \]

for \( i \in V \)

\[ \text{COMPUTE MARGINAL}(i) \]

**EVIDENCE**(E)

for \( i \in E \)

\[ \psi^E(x_i) = \psi(x_i) \delta(x_i, \overline{x_i}) \]

for \( i \notin E \)

\[ \psi^E(x_i) = \psi(x_i) \]

**COLLECT**(i, j)

for \( k \in \mathcal{N}(j) \setminus i \)

\[ \text{COLLECT}(j, k) \]

\[ \text{SEND MESSAGE}(j, i) \]

**DISTRIBUTE**(i, j)

\[ \text{SEND MESSAGE}(i, j) \]

for \( k \in \mathcal{N}(j) \setminus i \)

\[ \text{DISTRIBUTE}(j, k) \]

\[ \text{SEND MESSAGE}(j, i) \]

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

**COMPUTE MARGINAL**(i)

\[ p(x_i) \propto \psi^E(x_i) \prod_{j \in \mathcal{N}(i)} m_{ji}(x_i) \]
Inference on general GM

- Now, what if the GM is not a tree-like graph?

- Can we still directly run message message-passing protocol along its edges?

- For non-trees, we do not have the guarantee that message-passing will be consistent!

- Then what?
  - Construct a graph data-structure from P that has a tree structure, and run message-passing on it!

→ Junction tree algorithm
Summary

- The simple Eliminate algorithm captures the key algorithmic Operation underlying probabilistic inference:
  --- That of taking a sum over product of potential functions

- The computational complexity of the Eliminate algorithm can be reduced to purely graph-theoretic considerations.

- This graph interpretation will also provide hints about how to design improved inference algorithms.

- What can we say about the overall computational complexity of the algorithm? In particular, how can we control the "size" of the summands that appear in the sequence of summation operation.