Machine Learning

10-701/15-781, Fall 2011

Bayesian Networks
and
Exact Inference

Eric Xing

Lecture 13-14, October 24, 2011

Reading: Chap. 8, C.B book

Recall HMM
What is a Bayesian Network?
--- example from a signal transduction pathway

- A possible world for cellular signal transduction:

  - Receptor A \( X_1 \)
  - Receptor B \( X_2 \)
  - Kinase C \( X_3 \)
  - Kinase D \( X_4 \)
  - Kinase E \( X_5 \)
  - TF F \( X_6 \)
  - Gene G \( X_7 \)
  - Gene H \( X_8 \)

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?
  - 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 \( 2^8 \) configurations of the unobserved variables
What is a Bayesian Network?
--- example from a signal transduction pathway

- A possible world for cellular signal transduction:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receptor A</td>
<td>$x_1$</td>
</tr>
<tr>
<td>Receptor B</td>
<td>$x_2$</td>
</tr>
<tr>
<td>Kinase C</td>
<td>$x_3$</td>
</tr>
<tr>
<td>Kinase D</td>
<td>$x_4$</td>
</tr>
<tr>
<td>Kinase E</td>
<td>$x_5$</td>
</tr>
<tr>
<td>TFF</td>
<td>$x_6$</td>
</tr>
<tr>
<td>Gene G</td>
<td>$x_7$</td>
</tr>
<tr>
<td>Gene H</td>
<td>$x_8$</td>
</tr>
</tbody>
</table>

BN: Structure Simplifies Representation

- Dependencies among variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receptor A</td>
<td>$x_1$</td>
</tr>
<tr>
<td>Receptor B</td>
<td>$x_2$</td>
</tr>
<tr>
<td>Kinase C</td>
<td>$x_3$</td>
</tr>
<tr>
<td>Kinase D</td>
<td>$x_4$</td>
</tr>
<tr>
<td>Kinase E</td>
<td>$x_5$</td>
</tr>
<tr>
<td>TFF</td>
<td>$x_6$</td>
</tr>
<tr>
<td>Gene G</td>
<td>$x_7$</td>
</tr>
<tr>
<td>Gene H</td>
<td>$x_8$</td>
</tr>
</tbody>
</table>
If $X_i$'s are conditionally independent (as described by a BN), 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 BN?

- Representation cost: how many probability statements are needed?
  
  \[2+2+4+4+4+8+4+8 = 36, \text{ an 8-fold reduction from } 2^8!\]

- Algorithms for systematic and efficient inference/learning computation
  - Exploring the graph structure and probabilistic semantics
  - Incorporation of domain knowledge and causal (logical) structures

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”:

  $$P(X) = \prod_i P(X_i | X_{p(i)})$$

  where $X_{p(i)}$ 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
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)
  - …

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 C 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)
  ![Ancestral graph](image2)
  ![Moral ancestral graph](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 below (and plus some boundary conditions):

\[
\mathcal{I}(G) = \{X \perp Z | Y \colon \text{dsep}_G(X; Z|Y)\}
\]

- Defn: \(\mathcal{I}(\mathcal{G})\) = all independence properties that correspond to d-separation.

- D-separation is sound and complete.
Example:

```
Example:

9

Example:
```

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 $\mathcal{D}_1 \equiv \mathcal{D}_2$.
Quantitative Specification

\[ p(A,B,C) = \]

Conditional probability tables (CPTs)

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

<table>
<thead>
<tr>
<th></th>
<th>( a^0 )</th>
<th>( a^1 )</th>
<th>( b^0 )</th>
<th>( b^1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a )</td>
<td>0.75</td>
<td>0.25</td>
<td>0.33</td>
<td>0.67</td>
</tr>
<tr>
<td>( b )</td>
<td>0.45</td>
<td>0.9</td>
<td>0.7</td>
<td></td>
</tr>
<tr>
<td>( c )</td>
<td>0.55</td>
<td>0.1</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>( d )</td>
<td>0.3</td>
<td>0.5</td>
<td>0.7</td>
<td>0.5</td>
</tr>
</tbody>
</table>

© Eric Xing @ CMU, 2006-2011
Conditional probability density func. (CPDs)

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

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

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

\[ D \sim \mathcal{N}(\mu_a + C, \Sigma_d) \]

Conditional Independencies

What is this model

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

\( \theta \)

Model parameters

Data = \{y_1, \ldots, y_n\}

\( X_1 \quad X_2 \quad \cdots \quad X_{n-1} \quad X_n \)

“Plate” Notation

\( \theta \)

Model parameters

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

Plate = rectangle in graphical model

variables within a plate are replicated in a conditionally independent manner
Example: Gaussian Model

Generative model:

\[ p(x_1, \ldots, x_n | \mu, \sigma) = \prod p(x_i | \mu, \sigma) = p(\text{data} | \text{parameters}) = p(D | \theta) \]

where \( \theta = \{\mu, \sigma\} \)

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

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
More examples

Density estimation
- Parametric and nonparametric methods

Regression
- Linear, conditional mixture, nonparametric

Classification
- Generative and discriminative approach

Example, con'd
- Evolution

Tree Model
Example, con'd

- Speech recognition

![Diagram of Hidden Markov Model](image)

Example, con'd

- Genetic Pedigree

![Genetic Pedigree Diagram](image)
BN and Graphical Models

- A Bayesian network is a special case of **Graphical Models**.
- A Graphical Model 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.
- 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.
Summary

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

- The factorization theorem of the joint probability
  - Local specification $\rightarrow$ globally consistent distribution
  - Local representation for exponentially complex state-space

- 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.

Probabilistic Inference

- Computing statistical queries regarding the network, e.g.:
  - Is node $X$ independent on node $Y$ given nodes $Z,W$?
  - What is the probability of $X=true$ if $(Y=false$ and $Z=true)$?
  - What is the joint distribution of $(X,Y)$ if $Z=false$?
  - What is the likelihood of some full assignment?
  - What is the most likely assignment of values to all or a subset the nodes of the network?

- General purpose algorithms exist to fully automate such computation
  - Computational cost depends on the topology of the network
  - Exact inference:
    - The junction tree algorithm
  - Approximate inference:
    - Loopy belief propagation, variational inference, Monte Carlo sampling
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
  \[
P(X_H \mid x_v) = \frac{P(X_H, x_v)}{P(x_v)} = \frac{\sum_{X_H} P(X_H, x_v)}{P(x_v)}
  \]
  - 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
  
  ![Diagram](image)
  
  - the query node is a descendant of the evidence

- **Diagnosis**: what is the probability of disease/fault given symptoms
  
  ![Diagram](image)
  
  - 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 from all parts of the network

Inferential Query 3: Most Probable Assignment

- 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 $Y$. 

© Eric Xing @ CMU, 2006-2011
Thm:
Computing $P(X_i=x_i|x_v)$ 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 signal transduction pathway:

![Signal transduction pathway diagram]

What is the likelihood that protein E is active?

- Query: $P(e)$

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

By chain decomposition, we get

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

Elimination on 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 on 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 "eliminates" one variable from our summation argument at a "local cost".

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

- Eliminate nodes one by one all the way to the end, we get

\[ P(e) = \sum_d P(e \mid d) p(d) \]

- Complexity:
  - Each step costs \( O(|\text{Val}(X_i)| \cdot |\text{Val}(X_{i+1})|) \) operations: \( O(nk^2) \)
  - Compare to naïve evaluation that sums over joint values of \( n-1 \) variables \( O(k^n) \)

---

Inference on General BN via Variable Elimination

**General idea:**
- Write query in the form

\[ P(X_1, \epsilon) = \sum_{x_n} \cdots \sum_{x_3} \sum_{x_2} \prod_i P(x_i \mid pa_i) \]

  - this suggests an "elimination order" of latent variables to be marginalized
- Iteratively
  - Move all irrelevant terms outside of innermost sum
  - Perform innermost sum, getting a new term
  - Insert the new term into the product
- wrap-up

\[ P(X_1 \mid \epsilon) = \frac{P(X_1, \epsilon)}{P(\epsilon)} \]
From elimination to message passing

- Recall ELIMINATION algorithm:
  - Choose an ordering $Z$ 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 $m_i(x_i)$ denote the factor resulting from eliminating variables from below up to $i$, which is a function of $x_i$:

$$m_i(x_i) = \sum_{x_j} \psi_j(x_j)\psi(x_i; x_j) \prod_{k \in N(j) \setminus i} m_k(x_j)$$

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

$$m_{ji}(x_i) = \sum_{x_j} \psi(x_j)\psi(x_i; x_j) \prod_{k \in N(j) \setminus i} m_k(x_j)$$

$$p(x_f) \propto \psi_j(x_f) \prod_{e \in N(f)} m_e(x_f)$$

$m_i(x_i)$ represents a "belief" of $x_i$ from $x_f$.  

© Eric Xing @ CMU, 2006-2011
Elimination on trees is equivalent to message passing along tree branches!

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

The message passing protocol:

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

For a node of degree \( d \), whenever messages have arrived on any subset of \( d-1 \) node, compute the message for the remaining edge and send!

- A pair of messages have been computed for each edge, one for each direction
- All incoming messages are eventually computed for each node
Correctness of BP on tree

- Collollary: the synchronous implementation is "non-blocking"

- Thm: The Message Passage Guarantees obtaining all marginals in the tree

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

- What about non-tree?

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
A Sketch of the Junction Tree Algorithm

- The algorithm
  - Construction of junction trees --- a special clique tree
  - Propagation of probabilities --- a message-passing protocol
- Results in marginal probabilities of all cliques --- solves all queries in a single run
- A generic exact inference algorithm for any GM
- Complexity: exponential in the size of the maximal clique --- a good elimination order often leads to small maximal clique, and hence a good (i.e., thin) JT
- Many well-known algorithms are special cases of JT
  - Forward-backward, Kalman filter, Peeling, Sum-Product ...

The Shafer Shenoy Algorithm

- Shafer-Shenoy algorithm
  
  ![Diagram](image)
  
  - Message from clique $i$ to clique $j$:
    \[
    \mu_{i\to j} = \sum_{C_j,S_j} \psi_{C_j} \prod_{k \in j} \mu_{k\to j}(S_k)
    \]
  - Clique marginal
    \[
    p(C_i) \propto \psi_{C_i} \prod_{k} \mu_{k\to i}(S_k)
    \]
The Junction tree algorithm for HMM

- A junction tree for the HMM

\[
\begin{array}{c}
\psi(\mathbf{y}_1) \\
\psi(\mathbf{y}_2) \\
\psi(\mathbf{y}_3) \\
\vdots \\
\psi(\mathbf{y}_T)
\end{array}
\Rightarrow
\begin{array}{c}
\psi(\mathbf{x}_1) \\
\psi(\mathbf{x}_2) \\
\psi(\mathbf{x}_3) \\
\vdots \\
\psi(\mathbf{x}_T)
\end{array}
\]

- Rightward pass

\[
\begin{align*}
\mu_{t+1,1}(y_{t+1}) &= \sum_{y_t} \psi(y_{t+1}) \mu_{t,1}(y_t) \mu_t(y_{t+1}) \\
&= \sum_{y_t} p(y_{t+1} | y_t) \mu_{t,1}(y_t) \mu_t(y_{t+1}) \\
&= p(y_{t+1} | y_t) \sum_{y_t} \mu_{t,1}(y_t)
\end{align*}
\]
- This is exactly the forward algorithm!

- Leftward pass ...

\[
\begin{align*}
\mu_{t+1,1}(y_{t+1}) &= \sum_{y_t} \psi(y_{t+1}) \mu_{t+1,2}(y_t) \mu_t(y_{t+1}) \\
&= \sum_{y_t} p(y_{t+1} | y_t) \mu_{t+1,2}(y_t) \mu_t(y_{t+1})
\end{align*}
\]
- This is exactly the backward 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.
A food web

What is the probability that hawks are leaving given that the grass condition is poor?
Example: 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:
  \[
P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)P(h \mid e,f)\]
  \[
  \Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)m_g(e, f)\]

- Step 2: Eliminate \( G \)
  - Compute
    \[
m_g(e) = \sum_g p(g \mid e) = 1\]
  \[
  \Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)m_g(e, f)\]
  \[
  = P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)m_g(e, f)\]
Example: Variable Elimination

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

- Initial factors:
  \[
P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c, d)P(f \mid a)P(g \mid e)P(h \mid e, f)
  \Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c, d)P(f \mid a)P(g \mid e)m_b(e, f)
  \Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c, d)P(f \mid a)m_b(e, f)
  \]

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

- 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:
  \[
  P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)P(h \mid e,f)
  \]
  \[
  \Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)m_e(e,f)
  \]
  \[
  \Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)eP(e \mid c,d)m_e(e,f)
  \]
  \[
  \Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)eP(e \mid c,d)m_e(e,f)
  \]
  \[
  \Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)eP(e \mid c,d)m_e(e,f)
  \]
  \[
  \Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)eP(e \mid c,d)m_e(e,f)
  \]

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

- Step 6: Eliminate \( C \)
  - Compute
    \[
    m_c(a,b) = \sum_c p(c \mid b)m_d(a,c)
    \]
  \[
  \Rightarrow P(a)P(b)P(c \mid d)m_c(a,c)
  \]
Example: Variable Elimination

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

- Initial factors:
  
  \[
P(a)p(b)p(c \mid d)p(d \mid a)p(e \mid c,d)p(f \mid a)p(g \mid e)p(h \mid e,f)
  \]
  
  \[
  \Rightarrow P(a)p(b)p(c \mid d)p(d \mid a)p(e \mid c,d)p(f \mid a)p(g \mid e)m_r(e,f)
  \]
  
  \[
  \Rightarrow P(a)p(b)p(c \mid d)p(d \mid a)p(e \mid c,d)p(f \mid a)m_r(e,f)
  \]
  
  \[
  \Rightarrow P(a)p(b)p(c \mid d)p(d \mid a)m_r(a,c,e)
  \]
  
  \[
  \Rightarrow P(a)p(b)p(c \mid d)m_r(a, b)
  \]

- Step 7: Eliminate \( B \)
  - Compute
    
    \[
    m_b(a) = \sum_b p(b)m_r(a, b)
    \]
    
    \[
    \Rightarrow P(a)m_r(a)
    \]

---

Example: Variable Elimination

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

- Initial factors:
  
  \[
P(a)p(b)p(c \mid d)p(d \mid a)p(e \mid c,d)p(f \mid a)p(g \mid e)p(h \mid e,f)
  \]
  
  \[
  \Rightarrow P(a)p(b)p(c \mid d)p(d \mid a)p(e \mid c,d)p(f \mid a)p(g \mid e)m_r(e,f)
  \]
  
  \[
  \Rightarrow P(a)p(b)p(c \mid d)p(d \mid a)p(e \mid c,d)p(f \mid a)m_r(e,f)
  \]
  
  \[
  \Rightarrow P(a)p(b)p(c \mid d)p(d \mid a)m_r(a,c,e)
  \]
  
  \[
  \Rightarrow P(a)p(b)m_r(a, c)
  \]
  
  \[
  \Rightarrow P(a)m_r(a)
  \]

- Step 8: Wrap-up
  
  \[
p(a, \bar{h}) = p(a)m_r(a), \quad p(\bar{h}) = \sum_a p(a)m_r(a)
  \]
  
  \[
  \Rightarrow P(a \mid \bar{h}) = \frac{p(a)m_r(a)}{\sum_a p(a)m_r(a)}
  \]

© Eric Xing @ CMU, 2006-2011
Complexity of variable elimination

Suppose in one elimination step we compute

\[ m_y(y_1, \ldots, y_k) = \sum_x m_x(x, y_1, \ldots, y_k) \]
\[ m'(x, y_1, \ldots, y_k) = \prod_{i=1}^k m_i(x, y_i) \]

This requires

1. \( k \cdot |\text{Val}(X)| \prod_i |\text{Val}(Y_i)| \) multiplications
   - For each value of \( x, y_1, \ldots, y_k \), we do \( k \) multiplications
2. \( |\text{Val}(X)| \prod_i |\text{Val}(Y_i)| \) 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.
Understanding Variable Elimination

- A graph elimination algorithm

- Intermediate terms correspond to the cliques resulted from elimination
  - "good" elimination orderings lead to small cliques and hence reduce complexity
    (what will happen if we eliminate "e" first in the above graph?)
  - finding the optimum ordering is NP-hard, but for many graph optimum or near-optimum can often be heuristically found

- Applies to undirected GMs

Elimination Clique

- Recall that Induced dependency during marginalization is captured in elimination cliques
  - Summation <-> elimination
  - Intermediate term <-> elimination clique

\[
P(a)P(b)P(c|b)P(d|a)P(e|c)P(f|a)P(g|e)P(h|e, f)
\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c)P(f|a)P(g|e)\delta_b(e, f)
\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c)P(f|a)\delta_b(e)\delta_e(e, f)
\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c)P(f|a)\delta_b(e)\delta_a(e, f)
\Rightarrow P(a)P(b)P(c|b)P(d|a)\delta_a(e, d)
\Rightarrow P(a)P(b)P(c|b)\delta_a(e, c)
\Rightarrow P(a)\delta_b(e)
\Rightarrow \delta(a)
\]
- Can this lead to a generic inference algorithm?
A Clique Tree

From Elimination to Message Passing

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

$$m(a, c, d) = \sum_e p(e | c, d)m_e(e)m_j(a, e)$$

- Messages can be reused
From Elimination to Message Passing

- Elimination ≡ message passing on a clique tree
  - Another query ...

- Messages $m_f$ and $m_h$ are reused, others need to be recomputed

The Shafer Shenoy Algorithm

- Shafer-Shenoy algorithm

Message from clique $i$ to clique $j$:

$$
\mu_{i \rightarrow j} = \sum_{c_j, s_j} \psi_{c_j} \prod_{k \rightarrow j} \mu_{k \rightarrow i}(S_k)
$$

Clique marginal

$$
p(C_i) \propto \psi_{c_i} \prod_{k \rightarrow i} \mu_{k \rightarrow i}(S_k)
$$
A Sketch of the Junction Tree Algorithm

- **The algorithm**
  - Construction of junction trees --- a special *clique tree*
  - Propagation of probabilities --- a message-passing protocol

- Results in marginal probabilities of all cliques --- solves all queries in a single run

- A *generic* exact inference algorithm for any GM

- **Complexity**: exponential in the size of the maximal clique --- a good elimination order often leads to small maximal clique, and hence a good (i.e., thin) JT

- Many well-known algorithms are special cases of JT
  - Forward-backward, Kalman filter, Peeling, Sum-Product ...