Recap of BN Representation

- Joint probability dist. on multiple variables:

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

- If \( X_i \)'s are independent: \( P(X_i | \cdot) = P(X_i) \)

\[ P(X_1, X_2, X_3, X_4, X_5, X_6) = P(X_1)P(X_2)P(X_3)P(X_4)P(X_5)P(X_6) = \prod_i P(X_i) \]

- If \( X_i \)'s are conditionally independent (as described by a GM), the joint can be factored to simpler products, e.g.,

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

Learning BNs

The goal:

Given set of independent samples (assignments of random variables), find the best (the most likely?) Bayesian Network (both DAG and CPDs)

(B,E,A,C,R)=(T,F,F,T,F)
(B,E,A,C,R)=(T,F,T,T,F)
........
(B,E,A,C,R)=(F,T,T,T,F)

| E | B | $P(A|E,B)$ |
|---|---|------------|
| $e$ | $b$ | 0.9 | 0.1 |
| $e$ | $\bar{b}$ | 0.2 | 0.8 |
| $\bar{e}$ | $b$ | 0.9 | 0.1 |
| $\bar{e}$ | $\bar{b}$ | 0.01 | 0.99 |
MLE for general BN parameters

- If we assume the parameters for each CPD are globally independent, and all nodes are fully observed, then the log-likelihood function decomposes into a sum of local terms, one per node:

\[
\ell(\theta; D) = \log p(D \mid \theta) = \log \prod_{i=1}^{n} p(x_{i, \theta_i}) = \sum_{i=1}^{n} \log p(x_{i, \theta_i})
\]

Example: decomposable likelihood of a directed model

- Consider the distribution defined by the directed acyclic GM:

\[
p(x \mid \theta) = p(x_1 \mid \theta_1) \cdot p(x_2 \mid x_1, \theta_2) \cdot p(x_3 \mid x_1, \theta_3) \cdot p(x_4 \mid x_2, x_3, \theta_4)
\]

- This is exactly like learning four separate small BNs, each of which consists of a node and its parents.
E.g.: MLE for BNs with tabular CPDs

- Assume each CPD is represented as a table (multinomial) where
  \[ \theta_{jk} = \log p(X_j = j \mid X_{z_i} = k) \]
  - Note that in case of multiple parents, \( X_z \) will have a composite state, and the CPD will be a high-dimensional table
  - The sufficient statistics are counts of family configurations
    \[ n_{jk} = \sum x^j x^{X_{z_i}} \]
  - The log-likelihood is
    \[ \ell(\theta, \mathcal{D}) = \log \prod_{i,j,k} \theta_{ijk}^{n_{ijk}} = \sum_{i,j,k} n_{jk} \log \theta_{jk} \]
  - Using a Lagrange multiplier to enforce \( \sum_j \theta_{ijk} = 1 \), we get:
    \[ \theta_{jk}^{ML} = \frac{n_{jk}}{\sum_{i,j,k} n_{ijk}^{n_{ijk}}} \]

What if some nodes are not observed?

- Consider the distribution defined by the directed acyclic GM:
  \[ p(x \mid \theta) = p(x_1 \mid \theta_1) p(x_2 \mid x_1, \theta_1) p(x_3 \mid x_1, \theta_2) p(x_4 \mid x_2, x_3, \theta_3) \]

- Need to compute \( p(x_{id} \mid X_{ij}) \rightarrow \text{inference} \)
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 $X_H = X \backslash X_v$, 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

$$P(X_H | X_V = x_v) = \frac{P(X_H, x_v)}{P(x_V)} = \sum_{x_H} \frac{P(X_H, x_v)}{P(x_V)}$$

- this is the a posteriori belief in $X_H$, given evidence $x_V$.

- We usually query a subset $Y$ of all hidden variables $H = \{Y, Z\}$ and "don’t care" about the remaining, $Z$:

$$P(Y | x_v) = \sum_z P(Y, Z = z | x_v)$$

- the process of summing out the "don’t care" variables $Z$ is called marginalization, and the resulting $P(Y | x_v)$ 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 descendant 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 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 $Z$:

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

- this is the maximum a posteriori configuration of $Y$.

Complexity of Inference

Thm:

Computing $P(x_{H=x_H} \mid x_v)$ in an arbitrary BN 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 BNs
  - For particular families of BNs, we can have provably efficient procedures
Approaches to inference

- Exact inference algorithms
  - The elimination algorithm
  - The junction tree algorithms

- Approximate inference techniques
  - Stochastic simulation / sampling methods
  - Markov chain Monte Carlo methods
  - Variational algorithms (will be covered in advanced ML courses)

Marginalization and Elimination

- A signal transduction pathway:

  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

  $= \sum_{d} \sum_{c} \sum_{b} \sum_{a} P(a)P(b | a)P(c | b)P(d | c)P(e | d)$
Elimination on Chains

- Rearranging terms...

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

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

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 \(O(|\text{Val}(X_i)| \times |\text{Val}(X_{i+1})|)\) 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
  \[ P(X_1, e) = \sum_{x_n} \cdots \sum_{x_3} \sum_{x_2} \prod_i P(x_i | pa_i) \]
- 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
  \[ P(X_1 | e) = \frac{P(X_1, e)}{P(e)} \]

A more complex network

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
    $$m_g(e) = \sum_g p(g \mid e) = 1$$
  $\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c, d)P(f \mid a)m_g(e, f)$
  $= P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c, d)P(f \mid a)m_e(e, f)$
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:**
  $P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c, d)P(f \mid a)P(g \mid e)P(h \mid e, f)$
  $\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c, d)P(f \mid a)P(g \mid e)m_1(e, f)$
  $\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)eP(e \mid c, d)m_1(e, f)$
  $\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)eP(e \mid c, d)m_1(e, f)$
  $\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)m_1(a, e)$
  $\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)m_1(a, c, d)$

- **Step 5: Eliminate $D$**
  - Compute $m_2(a, c) = \sum_d p(d \mid a)m_1(a, c, d)$
  $\Rightarrow P(a)P(b)P(c \mid d)m_2(a, c)$

Example: Variable Elimination

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

- **Initial factors:**
  $P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c, d)P(f \mid a)P(g \mid e)P(h \mid e, f)$
  $\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c, d)P(f \mid a)P(g \mid e)m_1(e, f)$
  $\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c, d)P(f \mid a)m_1(e, f)$
  $\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c, d)m_1(a, e)$
  $\Rightarrow P(a)P(b)P(c \mid d)m_1(a, c, d)$
  $\Rightarrow P(a)P(b)P(c \mid d)m_1(a, c)$

- **Step 6: Eliminate $C$**
  - Compute $m_2(a, b) = \sum_c p(c \mid b)m_1(a, c)$
  $\Rightarrow P(a)P(b)P(c \mid d)m_2(a, c)$
Example: Variable Elimination

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

- Initial factors:
  \[
  P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c, d)P(f \mid a)P(g \mid e)P(h \mid e, f) \\
  \Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c, d)P(f \mid a)P(g \mid e)m_{l}(e, f) \\
  \Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c, d)P(f \mid a)m_{l}(e, f) \\
  \Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)e_{l}(a, c, d) \\
  \Rightarrow P(a)P(b)P(c \mid d)m_{l}(a, c) \\
  \Rightarrow P(a)P(b)m_{l}(a, b) \\
  \]

- Step 7: Eliminate \( B \)
  - Compute \( m_{b}(a) = \sum_{b} p(b)m_{l}(a, b) \)
  \[
  \Rightarrow P(a)m_{b}(a) \\
  \]

Example: Variable Elimination

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

- Initial factors:
  \[
  P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c, d)P(f \mid a)P(g \mid e)P(h \mid e, f) \\
  \Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c, d)P(f \mid a)P(g \mid e)m_{l}(e, f) \\
  \Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c, d)P(f \mid a)m_{l}(e, f) \\
  \Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)e_{l}(a, c, d) \\
  \Rightarrow P(a)P(b)e_{l}(a, c) \\
  \Rightarrow P(a)P(b)m_{l}(a, b) \\
  \Rightarrow P(a)m_{l}(a) \\
  \]

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

© Eric Xing @ CMU, 2006-2010
Suppose in one elimination step we compute
\[ m_i(y_1, \ldots, y_k) = \sum_x m^i_i(x, y_1, \ldots, y_k) \]
\[ m^i_i(x, y_1, \ldots, y_k) = \prod_{i=1}^k m_i(x, Y_{e_i}) \]

This requires
- \( k \cdot |\text{Val}(X)| \cdot \prod_{i} |\text{Val}(Y_{e_i})| \) multiplications
  - For each value of \( x \), \( y_1 \), ..., \( y_k \), we do \( k \) multiplications
- \( |\text{Val}(X)| \cdot \prod_{i} |\text{Val}(Y_{e_i})| \) additions
  - For each value of \( y_1 \), ..., \( y_k \), we do \( |\text{Val}(X)| \) additions

Complexity is exponential in number of variables in the intermediate factor.

Understanding Variable Elimination

- A graph elimination algorithm

\[ \text{moralization} \quad \text{graph elimination} \]
Elimination Cliques

m_h(e, f) → m_g(e) → m_f(e, a) → m_e(a, c, d) → m_d(a, c) → m_c(a, b) → m_e(a)

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
A clique tree

\[ m(a,c,d) = \sum_e p(e|c,d)m_x(e)m_y(a,e) \]

From Elimination to Message Passing

- Our algorithm so far answers only one query (e.g., on one node), do we need to do a complete elimination for every such query?
- Elimination \(\equiv\) message passing on a clique tree

\[ m(a,c,d) = \sum_e p(e|c,d)m_x(e)m_y(a,e) \]

- Messages can be reused
From Elimination to Message Passing

- Our algorithm so far answers only one query (e.g., on one node), do we need to do a complete elimination for every such query?
- Elimination \(\equiv\) message passing on a clique tree
  - Another query ...

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

The Junction Tree Algorithm

- Shafer-Shenoy algorithm

  \[
  \mu_{i\rightarrow j} = \sum_{c_j|\text{S}_{ij}} \psi_{c_i} \prod_{k\in j} \mu_{k\rightarrow i}(S_{ik})
  \]

- Clique marginal

  \[
  p(C_i) \propto \psi_{c_i} \prod_k \mu_{k\rightarrow i}(S_{ik})
  \]

© Eric Xing @ CMU, 2006-2010
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 Junction tree algorithm for HMM

- A junction tree for the HMM

\[
\begin{align*}
H_{m+1}(Y_{m+1}) &= \sum_{Y_i} \psi(Y_i, Y_{m+1}) H_{m+1}(Y_i) H_{m+1}(Y_{m+1}) \\
&= \sum_{Y_i} p(Y_{m+1} | Y_i) H_{m+1}(Y_i) H_{m+1}(Y_{m+1}) \\
&= p(Y_{m+1} | Y_i) \sum_{Y_i} a(Y_{m+1}) H_{m+1}(Y_i)
\end{align*}
\]

- Rightward pass

This is exactly the forward algorithm!

- Leftward pass ...

\[
\begin{align*}
H_{m+2}(Y_{m+1}) &= \sum_{Y_i} \psi(Y_i, Y_{m+1}) H_{m+2}(Y_i) H_{m+1}(Y_{m+1}) \\
&= \sum_{Y_i} p(Y_{m+1} | Y_i) H_{m+2}(Y_i) H_{m+1}(Y_{m+1})
\end{align*}
\]

- This is exactly the backward algorithm!
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 → globally consistent distribution
  - Local representation for exponentially complex state-space

- Support efficient inference and learning