

Machine Learning

10-701/15-781, Fall 2011

Inference and Learning For Bayesian Networks

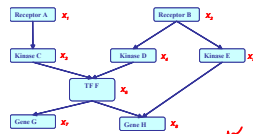
Eric Xing

Lecture 14, October 31, 2011

Reading: Chap. 8, C.B book

© Eric Xing @ CMU, 2006-2011

1



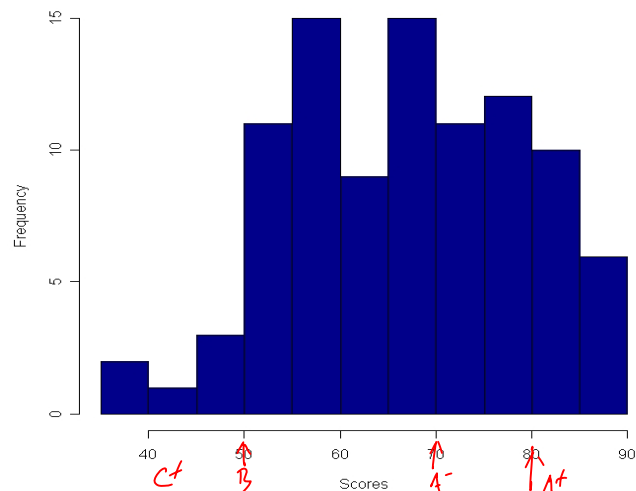
✓

✓

✓
✓
✓
✓

Midterm

Histogram of midterm scores



2

Recap of BN Representation

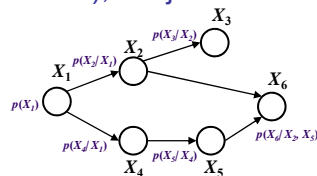
- Joint probability dist. on multiple variables:

$$\frac{P(X_1, X_2, X_3, X_4, X_5, X_6)}{= P(X_1)P(X_2 | X_1)P(X_3 | X_1, X_2)P(X_4 | X_1, X_2, X_3)P(X_5 | X_1, X_2, X_3, X_4)P(X_6 | X_1, X_2, X_3, X_4, X_5)}$$

- 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.,



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

© Eric Xing @ CMU, 2006-2011

3

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**.

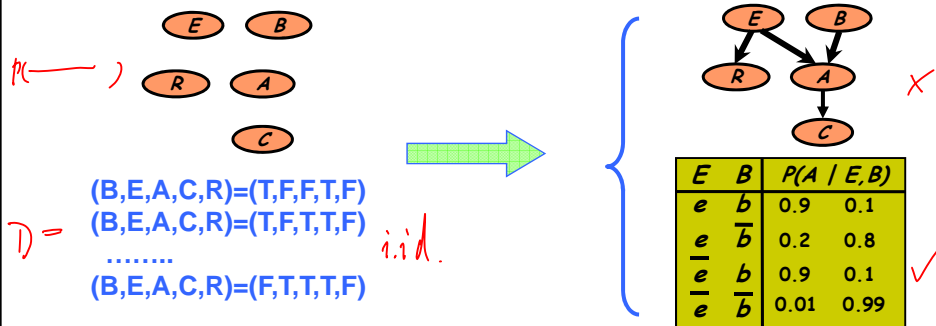
© Eric Xing @ CMU, 2006-2011

4

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)



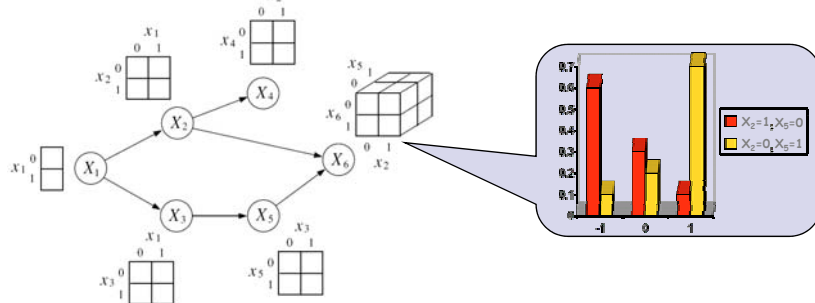
Learning Graphical Models

- Scenarios:
 - completely observed GMs
 - directed ✓
 - undirected
 - partially observed GMs
 - directed ✓
 - undirected (an open research topic)
- Estimation principles:
 - Maximal likelihood estimation (MLE) ✓
 - Bayesian estimation
 - Maximal conditional likelihood
 - Maximal "Margin"
- We use **learning** as a name for the process of **estimating the parameters**, and in some cases, the topology of the network, from data.

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:

$$\mathcal{L}(\theta; D) = \log p(D | \theta) = \log \prod_i \left(\prod_{n,i} p(x_{n,i} | \mathbf{x}_{n,\pi_i}, \theta_i) \right) = \sum_i \left(\sum_n \log p(x_{n,i} | \mathbf{x}_{n,\pi_i}, \theta_i) \right)$$



7

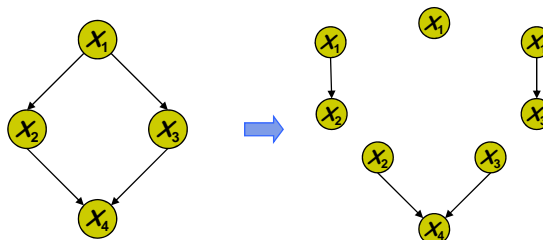
Example: decomposable likelihood of a directed model

- Consider the distribution defined by the directed acyclic GM:

$$p(x | \theta) = p(x_1 | \theta_1) p(x_2 | x_1, \theta_2) p(x_3 | x_1, \theta_3) p(x_4 | x_2, x_3, \theta_4)$$

by $\log p(x | \theta) = \log p(x_1 | \theta_1) + \log p(x_2 | x_1, \theta_2) + \log p(x_3 | x_1, \theta_3) + \log p(x_4 | x_2, x_3, \theta_4)$

- This is exactly like learning four separate small BNs, each of which consists of a node and its parents.



© Eric Xing @ CMU, 2006-2011

8

E.g.: MLE for BNs with tabular CPDs



- Assume each CPD is represented as a table (multinomial) where

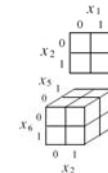
$$\theta_{ijk} \stackrel{\text{def}}{=} p(X_i = j \mid X_{\pi_i} = k)$$

- Note that in case of multiple parents, \mathbf{X}_{π_i} will have a composite state, and the CPD will be a high-dimensional table
- The sufficient statistics are counts of family configurations

$$n_{ijk} \stackrel{\text{def}}{=} \sum_n x_{n,i}^j x_{n,\pi_i}^k$$

- The log-likelihood is $\mathcal{L}(\theta; \mathcal{D}) = \log \prod_{i,j,k} \theta_{ijk}^{n_{ijk}} = \sum_{i,j,k} n_{ijk} \log \theta_{ijk}$
- Using a Lagrange multiplier to enforce $\sum_j \theta_{ijk} = 1$, we get:

$$\theta_{ijk}^{ML} = \frac{n_{ijk}}{\sum_{i,j,k} n_{ijk}}$$



$$X_i = \begin{pmatrix} x_{i,1} \\ x_{i,2} \\ \vdots \\ x_{i,m} \end{pmatrix}$$

© Eric Xing @ CMU, 2006-2011

9

Recall definition of HMM



- Transition probabilities between any two states

$$p(y_t^j \mid y_{t-1}^i = 1) = a_{i,j},$$

or $p(y_t \mid y_{t-1} = 1) \sim \text{Multinomial}(a_{i,1}, a_{i,2}, \dots, a_{i,M}), \forall i \in I.$

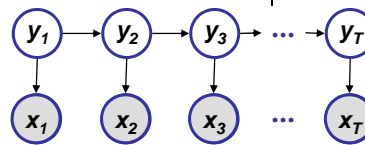
- Start probabilities

$$p(y_1) \sim \text{Multinomial}(\pi_1, \pi_2, \dots, \pi_M).$$

- Emission probabilities associated with each state

$$p(x_t \mid y_t^i = 1) \sim \text{Multinomial}(b_{i,1}, b_{i,2}, \dots, b_{i,K}), \forall i \in I.$$

or in general: $p(x_t \mid y_t^i = 1) \sim f(\cdot \mid \theta_i), \forall i \in I.$



© Eric Xing @ CMU, 2006-2011

10

Supervised ML estimation

- Given $\mathbf{x} = x_1 \dots x_N$ for which the true state path $\mathbf{y} = y_1 \dots y_N$ is known,

$$\mathcal{L}(\theta; \mathbf{x}, \mathbf{y}) = \log p(\mathbf{x}, \mathbf{y}) = \log \prod_n \left(p(y_{n,1}) \prod_{t=2}^T p(y_{n,t} | y_{n,t-1}) \prod_{t=1}^T p(x_{n,t} | x_{n,t-1}) \right)$$

- Define:

A_{ij} = # times state transition $i \rightarrow j$ occurs in \mathbf{y}

B_{ik} = # times state i in \mathbf{y} emits k in \mathbf{x}

- We can show that the maximum likelihood parameters θ are:

$$a_{ij}^{ML} = \frac{\#(i \rightarrow j)}{\#(i \rightarrow \bullet)} = \frac{\sum_n \sum_{t=2}^T y_{n,t-1}^i y_{n,t}^j}{\sum_n \sum_{t=2}^T y_{n,t-1}^i} = \frac{A_{ij} + A'_{ij}}{\sum_j A_{ij} + A'_{ij}} = 1$$

$$b_{ik}^{ML} = \frac{\#(i \rightarrow k)}{\#(i \rightarrow \bullet)} = \frac{\sum_n \sum_{t=1}^T y_{n,t}^i x_{n,t}^k}{\sum_n \sum_{t=1}^T y_{n,t}^i} = \frac{B_{ik}}{\sum_k B_{ik}}$$

- If \mathbf{y} is continuous, we can treat $\{(x_{n,t}, y_{n,t}) : t=1:T, n=1:N\}$ as $N \times T$ observations of, e.g., a Gaussian, and apply learning rules for Gaussian ...

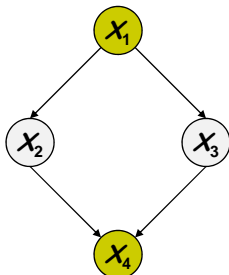
© Eric Xing @ CMU, 2006-2011

11

What if some nodes are not observed?

- Consider the distribution defined by the directed acyclic GM:

$$p(\mathbf{x} | \theta) = p(x_1 | \theta_1) p(x_2 | x_1, \theta_1) p(x_3 | x_1, \theta_3) p(x_4 | x_2, x_3, \theta_1)$$



$$p(\mathcal{D} | \theta) = \sum_{\mathbf{x}_u, \mathbf{x}_v} p(\mathbf{x}_1 \dots \mathbf{x}_4 | \theta)$$

$$\log p = \log \sum p$$

$$= \frac{p(\mathbf{x}_H, \mathbf{x}_V)}{p(\mathbf{x}_V)}$$

- Need to compute $p(\mathbf{x}_H | \mathbf{x}_V) \rightarrow$ inference

$$p(\mathbf{x}_V)$$

© Eric Xing @ CMU, 2006-2011

12

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 $X=\text{true}$ if ($Y=\text{false}$ and $Z=\text{true}$)?
 - What is the joint distribution of (X, Y) if $Z=\text{false}$?
 - 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 $X = \{X_1, X_2, \dots, X_n\}$
 - Without loss of generality $X_v = \{X_{k+1}, \dots, X_n\}$,
 - 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

$$P(x_v) = \sum_{x_H} P(x_H, x_v) = \sum_{x_1} \dots \sum_{x_k} P(x_1, \dots, x_k, x_v)$$

- 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

$$P(\mathbf{X}_H | \mathbf{X}_V = \mathbf{x}_V) = \frac{P(\mathbf{X}_H, \mathbf{x}_V)}{P(\mathbf{x}_V)} = \frac{P(\mathbf{X}_H, \mathbf{x}_V)}{\sum_{\mathbf{x}_H} P(\mathbf{X}_H = \mathbf{x}_H, \mathbf{x}_V)}$$

- this is the **a posteriori belief** in \mathbf{X}_H , given evidence \mathbf{x}_V
- We usually query a subset \mathbf{Y} of all hidden variables $\mathbf{X}_H = \{\mathbf{Y}, \mathbf{Z}\}$ and "don't care" about the remaining, \mathbf{Z} :

$$P(\mathbf{Y} | \mathbf{x}_V) = \sum_{\mathbf{z}} P(\mathbf{Y}, \mathbf{Z} = \mathbf{z} | \mathbf{x}_V)$$

- the process of summing out the "don't care" variables \mathbf{z} is called **marginalization**, and the resulting $P(\mathbf{Y} | \mathbf{x}_V)$ is called a **marginal prob.**

© Eric Xing @ CMU, 2006-2011

15

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 from all parts of the network

© Eric Xing @ CMU, 2006-2011

16

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 \mathbf{x}_v , and ignoring (the values of) other variables \mathbf{Z} :

$$\mathbf{Y}^* | \mathbf{x}_v = \arg \max_{\mathbf{y}} P(\mathbf{Y} | \mathbf{x}_v) = \arg \max_{\mathbf{y}} \sum_{\mathbf{z}} P(\mathbf{Y}, \mathbf{Z} = \mathbf{z} | \mathbf{x}_v)$$

- this is the **maximum a posteriori** configuration of \mathbf{Y} .

Complexity of Inference



Thm:

Computing $P(\mathbf{X}_H = \mathbf{x}_H | \mathbf{x}_v)$ 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

© Eric Xing @ CMU, 2006-2011

19

Marginalization and Elimination

- A signal transduction pathway:

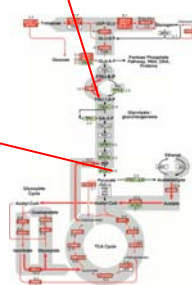


What is the likelihood that protein E is active?

- Query: $P(e)$

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

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



- By chain decomposition, we get

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

© Eric Xing @ CMU, 2006-2011

20

Elimination on Chains



- Rearranging terms ...

$$\begin{aligned}
 P(e) &= \sum_d \sum_c \sum_b \sum_a P(a)P(b|a)P(c|b)P(d|c)P(e|d) \\
 &= \sum_d \sum_c \sum_b P(c|b)P(d|c)P(e|d) \sum_a P(a)P(b|a)
 \end{aligned}$$

$f(b)$
 $f(b)$

© Eric Xing @ CMU, 2006-2011

21

Elimination on Chains



- Now we can perform innermost summation

$$\begin{aligned}
 P(e) &= \sum_d \sum_c \sum_b P(c|b)P(d|c)P(e|d) \sum_a P(a)P(b|a) \\
 &= \sum_d \sum_c \sum_b P(c|b)P(d|c)P(e|d)p(b)
 \end{aligned}$$

$f(b)$

- This summation "eliminates" one variable from our summation argument at a "local cost".

© Eric Xing @ CMU, 2006-2011

22

Elimination in Chains



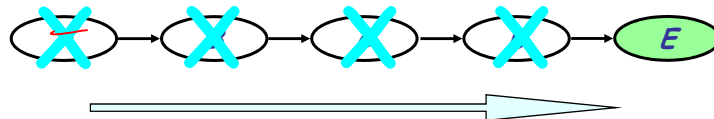
- Rearranging and then summing again, we get

$$\begin{aligned}
 P(e) &= \sum_d \sum_c \sum_b P(c|b) P(d|c) P(e|d) p(b) \\
 &= \sum_d \sum_c P(d|c) P(e|d) \sum_b P(c|b) p(b) \\
 &= \sum_d \sum_c P(d|c) P(e|d) p(c)
 \end{aligned}$$

© Eric Xing @ CMU, 2006-2011

23

Elimination in Chains



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

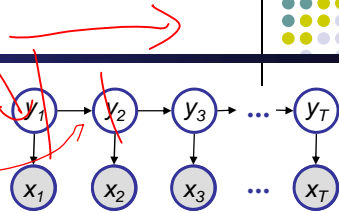
$$P(e) = \sum_d P(e|d) p(d)$$

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

© Eric Xing @ CMU, 2006-2011

24

Hidden Markov Model



$$p(\mathbf{x}, \mathbf{y}) = p(x_1, \dots, x_T, y_1, \dots, y_T)$$

$$= p(y_1) p(x_1 | y_1) p(y_2 | y_1) p(x_2 | y_2) \dots p(y_T | y_{T-1}) p(x_T | y_T)$$

Conditional probability:

$$p(y_i | x_1, \dots, x_T) = \sum_{y_1} \dots \sum_{y_{i-1}} \sum_{y_{i+1}} \dots \sum_{y_T} p(y_1, \dots, y_T, x_1, \dots, x_T)$$

$$= \sum_{y_1} \dots \sum_{y_{i-1}} \sum_{y_{i+1}} \dots \sum_{y_T} p(y_1) p(x_1 | y_1) \dots p(y_T | y_{T-1}) p(x_T | y_T)$$

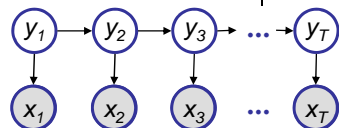
$$= \sum_{y_1} \dots \sum_{y_T} p(x_1 | y_1) p(y_2 | y_1) p(x_2 | y_2) \dots p(x_T | y_T) \sum_{y_1} p(y_1) p(x_1 | y_1) p(y_2 | y_1)$$

$$= \sum_{y_1} \dots \sum_{y_T} p(x_1 | y_1) p(y_2 | y_1) \dots p(x_T | y_T) \sum_{y_1} f_1(y_1) p(y_2 | y_1) = p(y_2, x_1) = \alpha_1$$

$$p(x_{i+1} | y_i) = f_i(y_i) = p(x_{i+1}, y_i) / p(y_i)$$

© Eric Xing @ CMU, 2006-2011

Hidden Markov Model



Conditional probability:

$$p(y_i | x_1, \dots, x_T) = \sum_{y_1} \dots \sum_{y_{i-1}} \sum_{y_{i+1}} \dots \sum_{y_T} p(y_i, \dots, y_T, x_1, \dots, x_T)$$

$$= \sum_{y_1} \dots \sum_{y_{i-1}} \sum_{y_{i+1}} \dots \sum_{y_T} p(y_1) p(x_1 | y_1) \dots p(y_T | y_{T-1}) p(x_T | y_T)$$

≈

© Eric Xing @ CMU, 2006-2011

26

Inference on General BN via Variable Elimination



General idea:

- Write query in the form

$$P(X_1, \mathbf{e}) = \sum_{x_n} \cdots \sum_{x_3} \sum_{x_2} \prod_i P(x_i | pa_i)$$

- 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

$$P(X_1 | \mathbf{e}) = \frac{P(X_1, \mathbf{e})}{P(\mathbf{e})}$$

© Eric Xing @ CMU, 2006-2011

27

The Sum-Product Operation



- In general, we can view the task at hand as that of computing the value of an expression of the form:

$$\sum_{\mathbf{z}} \prod_{\phi \in \mathcal{F}} \phi$$

where \mathcal{F} is a set of factors

k^n
↓
 k ~~#~~ scope(x)
 $\text{Scope}(x) \equiv \# \text{ of v.r. in } \phi$

- We call this task the sum-product inference task.

© Eric Xing @ CMU, 2006-2011

28

Outcome of elimination



- Let X be some set of variables,
let \mathcal{F} be a set of factors such that for each $\phi \in \mathcal{F}$, $\text{Scope}[\phi] \subseteq X$,
let $Y \subseteq X$ be a set of query variables,
and let $Z = X - Y$ be the variable to be eliminated

- The result of eliminating the variable Z is a factor

$$\tau(Y) = \sum_Z \prod_{\phi \in \mathcal{F}} \phi$$

- This factor does not necessarily correspond to any probability or conditional probability in this network. (example forthcoming)

Dealing with evidence



- Conditioning as a Sum-Product Operation

- The evidence potential: $\delta(E_i, \bar{e}_i) = \begin{cases} 1 & \text{if } E_i \equiv \bar{e}_i \\ 0 & \text{if } E_i \neq \bar{e}_i \end{cases}$

- Total evidence potential: $\delta(\mathbf{E}, \bar{\mathbf{e}}) = \prod_{i \in I_E} \delta(E_i, \bar{e}_i)$

- Introducing evidence --- restricted factors:

$$\tau(Y, \bar{\mathbf{e}}) = \sum_Z \prod_{\phi \in \mathcal{F}} \phi \times \delta(\mathbf{E}, \bar{\mathbf{e}})$$

The elimination algorithm



Procedure Elimination (

G , // the GM
 E , // evidence
 Z , // Set of variables to be eliminated
 X , // query variable(s)
)

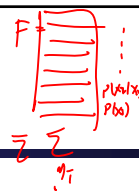
1. Initialize (G)
2. Evidence (E)
3. Sum-Product-Elimination ($\mathcal{F}, Z, <$)
4. Normalization (\mathcal{F})

$$P(X|E)$$

© Eric Xing @ CMU, 2006-2011

31

The elimination algorithm



Procedure Initialize (G, Z)

1. Let Z_1, \dots, Z_k be an ordering of Z such that $Z_i < Z_j$ iff $i < j$
2. Initialize \mathcal{F} with the full the set of factors

Procedure Evidence (E)

1. for each $i \in I_E$,
 $\mathcal{F} = \mathcal{F} \cup \delta(E_i, e_i)$

Procedure Sum-Product-Variable-Elimination ($\mathcal{F}, Z, <$)

1. for $i = 1, \dots, k$
 $\mathcal{F} \leftarrow \text{Sum-Product-Eliminate-Var}(\mathcal{F}, Z_i)$
2. $\phi^* \leftarrow \prod_{\phi \in \mathcal{F}} \phi$
3. return ϕ^*
4. Normalization (ϕ^*)

© Eric Xing @ CMU, 2006-2011

32

The elimination algorithm

$O(n^k)$



Procedure Initialize (G, Z)

1. Let Z_1, \dots, Z_k be an ordering of Z such that $Z_i < Z_j$ iff $i < j$
2. Initialize \mathcal{F} with the full the set of factors

Procedure Evidence (E)

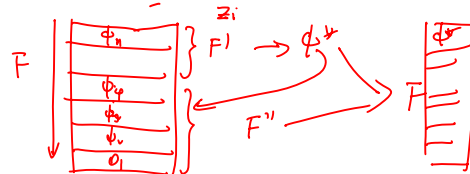
1. for each $i \in I_E$,
 $\mathcal{F} = \mathcal{F} \cup \delta(E_i, e_i)$

Procedure Sum-Product-Variable-Elimination ($\mathcal{F}, Z, <$)

1. for $i = 1, \dots, k$
 $\mathcal{F} \leftarrow \text{Sum-Product-Eliminate-Var}(\mathcal{F}, Z_i)$
2. $\phi \leftarrow \prod_{\phi \in \mathcal{F}} \phi$
3. return ϕ
4. Normalization (ϕ)

Procedure Normalization (ϕ)

1. $P(X|E) = \phi(X) / \sum_X \phi(X)$



Procedure Sum-Product-Eliminate-Var (\mathcal{F}, Z)

1. $\mathcal{F}' \leftarrow \{\phi \in \mathcal{F} : Z \in \text{Scope}(\phi)\}$
2. $\mathcal{F}'' \leftarrow \mathcal{F} - \mathcal{F}'$
3. $\psi \leftarrow \prod_{\phi \in \mathcal{F}'} \phi$
4. $\psi \leftarrow \sum_Z \psi$
5. return $\mathcal{F}'' \cup \{\psi\}$

© Eric Xing @ CMU, 2006-2011

33

From elimination to message passing

$P(x_i | E)$



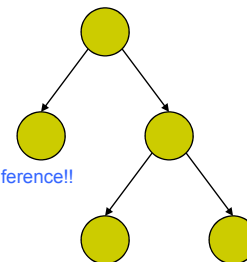
Recall ELIMINATION algorithm:

- 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:

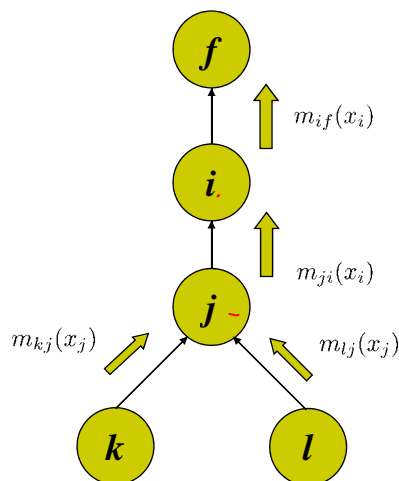
- Choose query node f as the root of the tree
 - View tree as a directed tree with edges pointing towards from f
 - Elimination ordering based on depth-first traversal
 - Elimination of each node can be considered as message-passing (or Belief Propagation) directly along tree branches, rather than on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



© Eric Xing @ CMU, 2006-2011

34

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 .

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

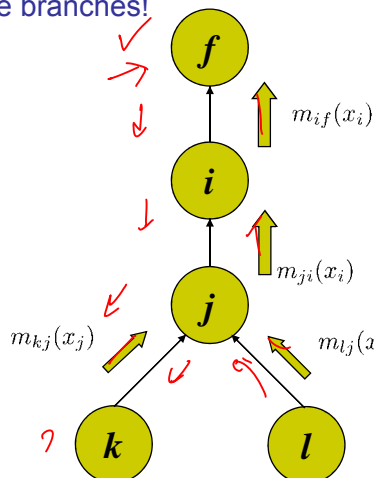
$$p(x_f) \propto \psi(x_f) \prod_{e \in N(f)} m_{ef}(x_f)$$

$m_{ij}(x_i)$ represents a "belief" of x_i from x_j !

© Eric Xing @ CMU, 2006-2011

35

- Elimination on trees is equivalent to message passing along tree branches!



$F \setminus \{x_i\} \rightarrow \text{new } \phi_i$
message

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

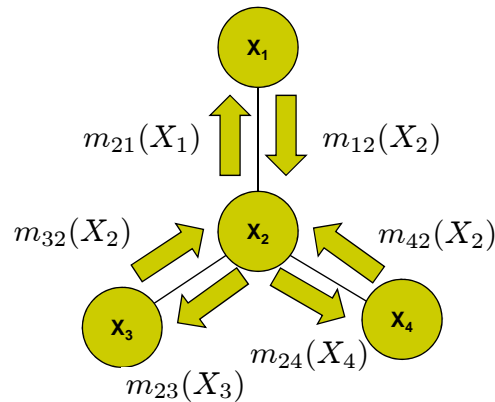
$$p(x_i) \propto m(x_i)$$

© Eric Xing @ CMU, 2006-2011

36

The message passing protocol:

- A two-pass algorithm:



© Eric Xing @ CMU, 2006-2011

37

Belief Propagation (SP-algorithm): Sequential implementation

```

SUM-PRODUCT( $T, E$ )
  EVIDENCE( $E$ )
   $f = \text{CHOOSE-ROOT}(V)$ 
  for  $e \in \mathcal{N}(f)$ 
    COLLECT( $f, e$ )
  for  $e \in \mathcal{N}(f)$ 
    DISTRIBUTE( $f, e$ )
  for  $i \in V$ 
    COMPUTE-MARGINAL( $i$ )

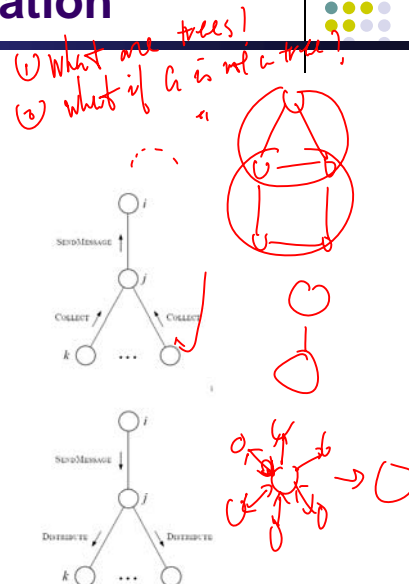
EVIDENCE( $E$ )
  for  $i \in E$ 
     $\psi^E(x_i) = \psi(x_i)\delta(x_i, \bar{x}_i)$ 
  for  $i \notin E$ 
     $\psi^E(x_i) = \psi(x_i)$ 

COLLECT( $i, j$ )
  for  $k \in \mathcal{N}(j) \setminus i$ 
    COLLECT( $j, k$ )
  SENDMESSAGE( $j, i$ )

DISTRIBUTE( $i, j$ )
  SENDMESSAGE( $i, j$ )
  for  $k \in \mathcal{N}(j) \setminus i$ 
    DISTRIBUTE( $j, k$ )

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

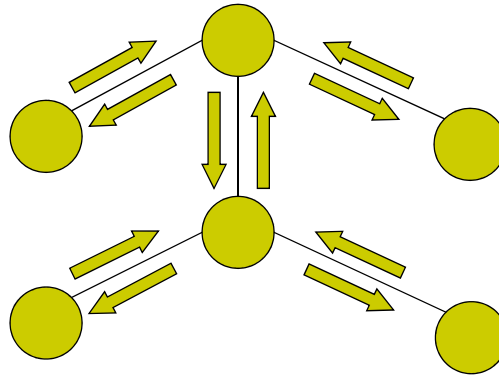
COMPUTE-MARGINAL( $i$ )
 $p(x_i) \propto \psi^E(x_i) \prod_{j \in \mathcal{N}(i)} m_{ji}(x_i)$ 
    
```



© Eric Xing @ CMU, 2006-2011

38

Belief Propagation (SP-algorithm): Parallel synchronous 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

© Eric Xing @ CMU, 2006-2011

39

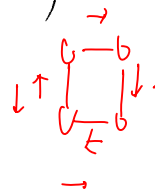
Correctness of BP on tree



- Collollary: the synchronous implementation is "non-blocking"
- Thm: The Message Passage Guarantees obtaining all marginals in the tree

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

- What about non-tree?

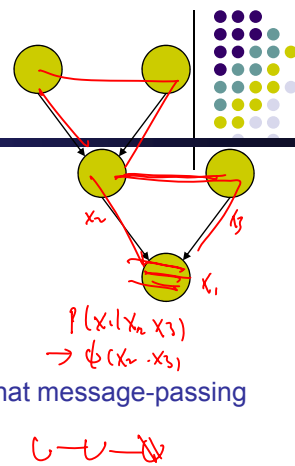


© Eric Xing @ CMU, 2006-2011

40

Inference on general GM

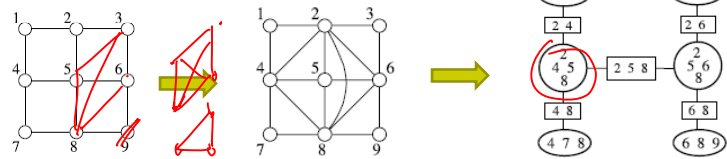
- Now, what if the GM is not a tree-like graph?
- Can we still directly run message-passing protocol along its edges?
- For non-trees, we do not have the guarantee that message-passing will be consistent!
- Then what?
 - Construct a graph data-structure from P that has a tree structure, and run message-passing on it!



→ Junction tree algorithm

Junction Tree

- General Algorithm on Graphs with Cycles



Steps: **=> Triangularization** **=> Construct JTs**

=> Message Passing on Clique Trees

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 ...

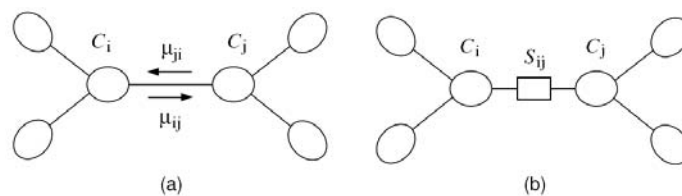
© Eric Xing @ CMU, 2006-2011

43

The Shafer Shenoy Algorithm



- Shafer-Shenoy algorithm



- Message from clique i to clique j :

$$\mu_{i \rightarrow j} = \sum_{C_i \setminus S_{ij}} \psi_{C_i} \prod_{k \neq j} \mu_{k \rightarrow i}(S_{ki})$$

- Clique marginal

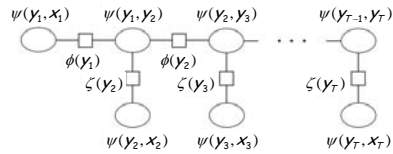
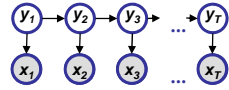
$$p(C_i) \propto \psi_{C_i} \prod_k \mu_{k \rightarrow i}(S_{ki})$$

© Eric Xing @ CMU, 2006-2011

44

The Junction tree algorithm for HMM

- A junction tree for the HMM



- Rightward pass

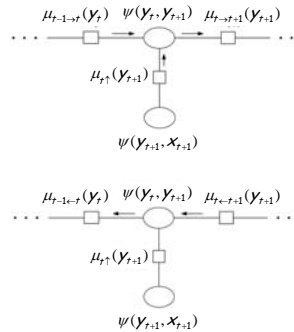
$$\begin{aligned}\mu_{t \rightarrow t+1}(y_{t+1}) &= \sum_{y_t} \psi(y_t, y_{t+1}) \mu_{t-1 \rightarrow t}(y_t) \mu_{t \uparrow}(y_{t+1}) \\ &= \sum_{y_t} p(y_{t+1} | y_t) \mu_{t-1 \rightarrow t}(y_t) p(x_{t+1} | y_{t+1}) \\ &= p(x_{t+1} | y_{t+1}) \sum_{y_t} a_{y_t, y_{t+1}} \mu_{t-1 \rightarrow t}(y_t)\end{aligned}$$

- This is exactly the **forward algorithm**!

- Leftward pass ...

$$\begin{aligned}\mu_{t-1 \leftarrow t}(y_t) &= \sum_{y_{t+1}} \psi(y_t, y_{t+1}) \mu_{t \leftarrow t+1}(y_{t+1}) \mu_{t \uparrow}(y_t) \\ &= \sum_{y_{t+1}} p(y_{t+1} | y_t) \mu_{t \leftarrow t+1}(y_{t+1}) p(x_{t+1} | y_{t+1})\end{aligned}$$

- This is exactly the **backward algorithm**!



© Eric Xing @ CMU, 2006-2011

45

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.

© Eric Xing @ CMU, 2006-2011

46

Extra reading:



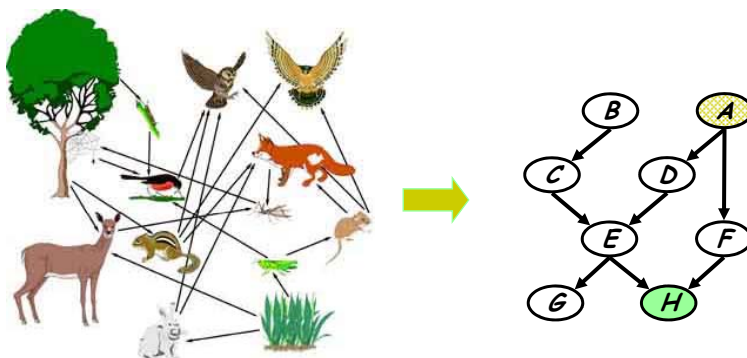
© Eric Xing @ CMU, 2006-2011

47

From Elimination to JT on a general Bayesian network



A food web



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

© Eric Xing @ CMU, 2006-2011

48

Example: Variable Elimination

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

- Initial factors:

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

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

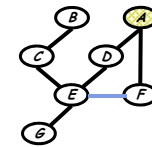
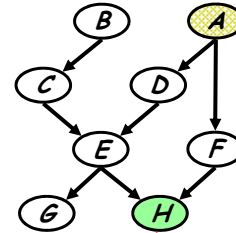
- Step 1:

- Conditioning (fix the evidence node (i.e., h) on its observed value (i.e., \tilde{h}):

$$m_h(e, f) = p(h = \tilde{h} | e, f)$$

- This step is isomorphic to a marginalization step:

$$m_h(e, f) = \sum_h p(h | e, f) \delta(h = \tilde{h})$$



© Eric Xing @ CMU, 2006-2011

49

Example: Variable Elimination

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

- Initial factors:

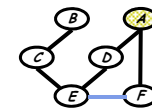
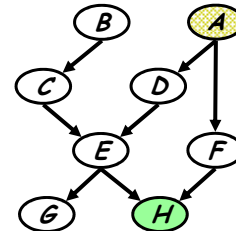
$$P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f) \\ \Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e, f)$$

- Step 2: Eliminate G

- compute

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

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



© Eric Xing @ CMU, 2006-2011

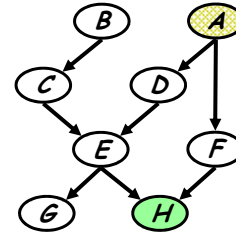
50

Example: Variable Elimination

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

- Initial factors:

$$\begin{aligned}
 &P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f) \\
 \Rightarrow &P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f) \\
 \Rightarrow &P(a)P(b)P(c|b)P(d|a)P(e|c,d)\underline{P(f|a)m_h(e,f)}
 \end{aligned}$$

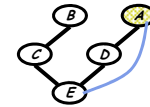


- Step 3: Eliminate F

- compute

$$m_f(e, a) = \sum_f p(f | a) m_h(e, f)$$

$$\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c,d)\underline{m_f(a,e)}$$



© Eric Xing @ CMU, 2006-2011

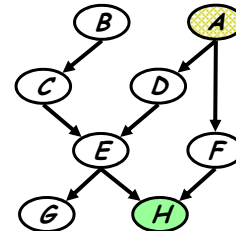
51

Example: Variable Elimination

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

- Initial factors:

$$\begin{aligned}
 &P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f) \\
 \Rightarrow &P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f) \\
 \Rightarrow &P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)m_h(e,f) \\
 \Rightarrow &P(a)P(b)P(c|b)P(d|a)\underline{P(e|c,d)m_f(a,e)}
 \end{aligned}$$

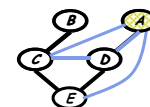


- Step 4: Eliminate E

- compute

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

$$\Rightarrow P(a)P(b)P(c|b)P(d|a)\underline{m_e(a,c,d)}$$



© Eric Xing @ CMU, 2006-2011

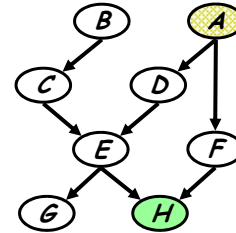
52

Example: Variable Elimination

- Query: $P(B | h)$
 - Need to eliminate: B, C, D

- Initial factors:

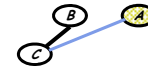
$$\begin{aligned}
 &P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f) \\
 \Rightarrow &P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f) \\
 \Rightarrow &P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)m_h(e,f) \\
 \Rightarrow &P(a)P(b)P(c|b)P(d|a)P(e|c,d)m_f(a,e) \\
 \Rightarrow &P(a)P(b)P(c|b)P(d|a)m_e(a,c,d)
 \end{aligned}$$



- Step 5: Eliminate D

- compute $m_d(a,c) = \sum_d p(d|a)m_e(a,c,d)$

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



© Eric Xing @ CMU, 2006-2011

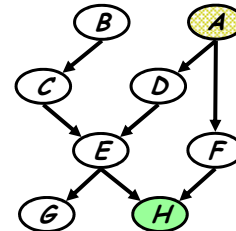
53

Example: Variable Elimination

- Query: $P(B | h)$
 - Need to eliminate: B, C

- Initial factors:

$$\begin{aligned}
 &P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f) \\
 \Rightarrow &P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f) \\
 \Rightarrow &P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)m_h(e,f) \\
 \Rightarrow &P(a)P(b)P(c|d)P(d|a)P(e|c,d)m_f(a,e) \\
 \Rightarrow &P(a)P(b)P(c|d)P(d|a)m_e(a,c,d) \\
 \Rightarrow &P(a)P(b)P(c|d)m_d(a,c)
 \end{aligned}$$



- Step 6: Eliminate C

- compute $m_c(a,b) = \sum_c p(c|b)m_d(a,c)$

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



© Eric Xing @ CMU, 2006-2011

54

Example: Variable Elimination

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

- Initial factors:

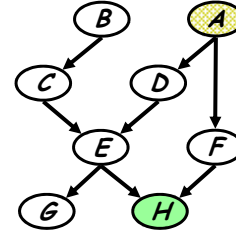
$$\begin{aligned}
 &P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f) \\
 \Rightarrow &P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f) \\
 \Rightarrow &P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)m_h(e,f) \\
 \Rightarrow &P(a)P(b)P(c|d)P(d|a)P(e|c,d)m_f(a,e) \\
 \Rightarrow &P(a)P(b)P(c|d)P(d|a)m_e(a,c,d) \\
 \Rightarrow &P(a)P(b)P(c|d)m_d(a,c) \\
 \Rightarrow &P(a)P(b)m_c(a,b)
 \end{aligned}$$

- Step 7: Eliminate B

- compute

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

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



(A)

© Eric Xing @ CMU, 2006-2011

55

Example: Variable Elimination

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

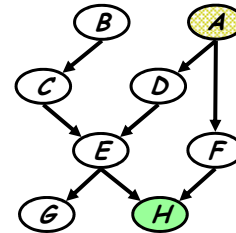
- Initial factors:

$$\begin{aligned}
 &P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f) \\
 \Rightarrow &P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f) \\
 \Rightarrow &P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)m_h(e,f) \\
 \Rightarrow &P(a)P(b)P(c|d)P(d|a)P(e|c,d)m_f(a,e) \\
 \Rightarrow &P(a)P(b)P(c|d)P(d|a)m_e(a,c,d) \\
 \Rightarrow &P(a)P(b)P(c|d)m_d(a,c) \\
 \Rightarrow &P(a)P(b)m_c(a,b) \\
 \Rightarrow &P(a)m_b(a)
 \end{aligned}$$

- Step 8: Wrap-up

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

$$\Rightarrow P(a | \tilde{h}) = \frac{p(a)m_b(a)}{\sum_a p(a)m_b(a)}$$



© Eric Xing @ CMU, 2006-2011

56

Complexity of variable elimination



- Suppose in one elimination step we compute

$$m_x(y_1, \dots, y_k) = \sum_x m'_x(x, y_1, \dots, y_k)$$

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

This requires

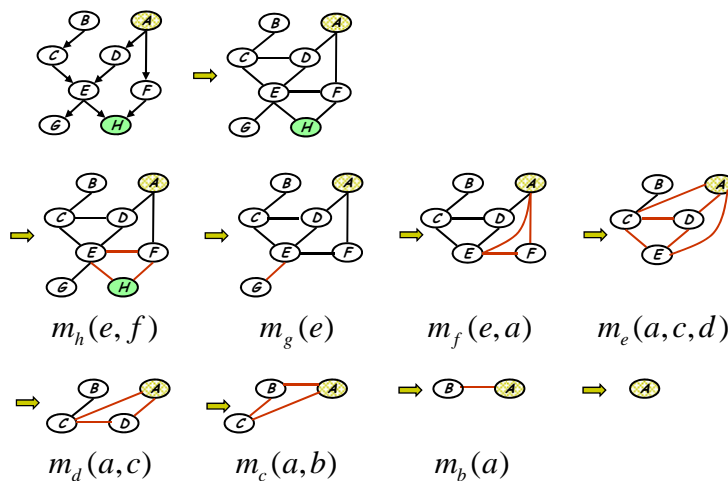
- $k \cdot |\text{Val}(X)| \cdot \prod_i |\text{Val}(\mathbf{Y}_{c_i})|$ **multiplications**
 - For each value of x, y_1, \dots, y_k we do k multiplications
- $|\text{Val}(X)| \cdot \prod_i |\text{Val}(\mathbf{Y}_{c_i})|$ **additions**
 - For each value of y_1, \dots, y_k , we do $|\text{Val}(X)|$ additions

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

© Eric Xing @ CMU, 2006-2011

57

Elimination Cliques

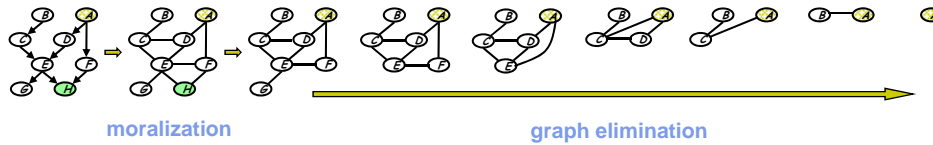


© Eric Xing @ CMU, 2006-2011

58

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

© Eric Xing @ CMU, 2006-2011

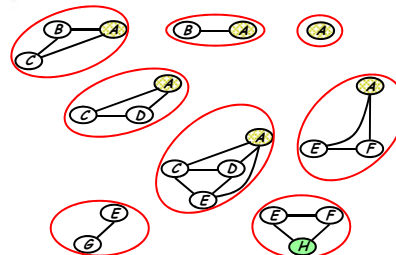
59

Elimination Clique

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

$$\begin{aligned}
 &P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f) \\
 \Rightarrow &P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)\phi_h(e,f) \\
 \Rightarrow &P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)\phi_g(e)\phi_h(e,f) \\
 \Rightarrow &P(a)P(b)P(c|b)P(d|a)P(e|c,d)\phi_f(a,e) \\
 \Rightarrow &P(a)P(b)P(c|b)P(d|a)\phi_e(a,c,d) \\
 \Rightarrow &P(a)P(b)P(c|b)\phi_d(a,c) \\
 \Rightarrow &P(a)P(b)\phi_c(a,b) \\
 \Rightarrow &P(a)\phi_b(a) \\
 \Rightarrow &\phi(a)
 \end{aligned}$$

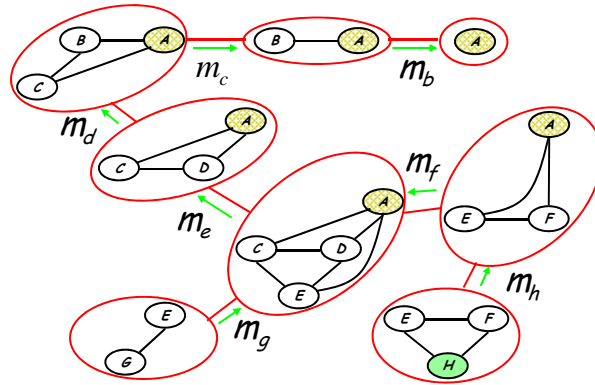
- Can this lead to a generic inference algorithm?



© Eric Xing @ CMU, 2006-2011

60

A Clique Tree



$$m_e(a, c, d) = \sum_e p(e | c, d) m_g(e) m_f(a, e)$$

© Eric Xing @ CMU, 2006-2011

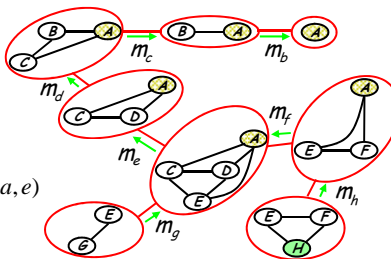
61

From Elimination to Message Passing

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



III



$$m_e(a, c, d) = \sum_e p(e | c, d) m_g(e) m_f(a, e)$$

- Messages can be reused

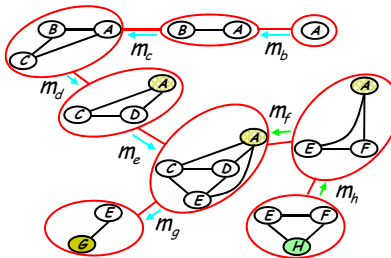
© Eric Xing @ CMU, 2006-2011

62

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

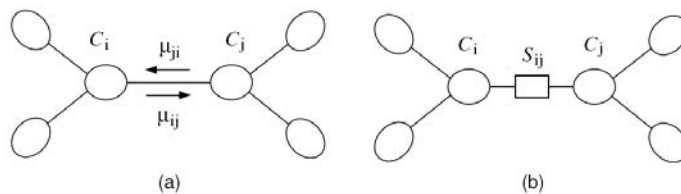
© Eric Xing @ CMU, 2006-2011

63

The Shafer Shenoy Algorithm



- Shafer-Shenoy algorithm



- Message from clique i to clique j :

$$\mu_{i \rightarrow j} = \sum_{C_i \setminus S_{ij}} \psi_{C_i} \prod_{k \neq j} \mu_{k \rightarrow i}(S_{ki})$$

- Clique marginal

$$p(C_i) \propto \psi_{C_i} \prod_k \mu_{k \rightarrow i}(S_{ki})$$

© Eric Xing @ CMU, 2006-2011

64

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 ...