

Machine Learning

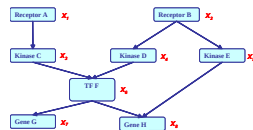
10-701/15-781, Fall 2011

Bayesian Networks and Exact Inference

Eric Xing

Lecture 13-14, October 24, 2011

Reading: Chap. 8, C.B book



© Eric Xing @ CMU, 2006-2011

1

Recall HMM

© Eric Xing @ CMU, 2006-2011

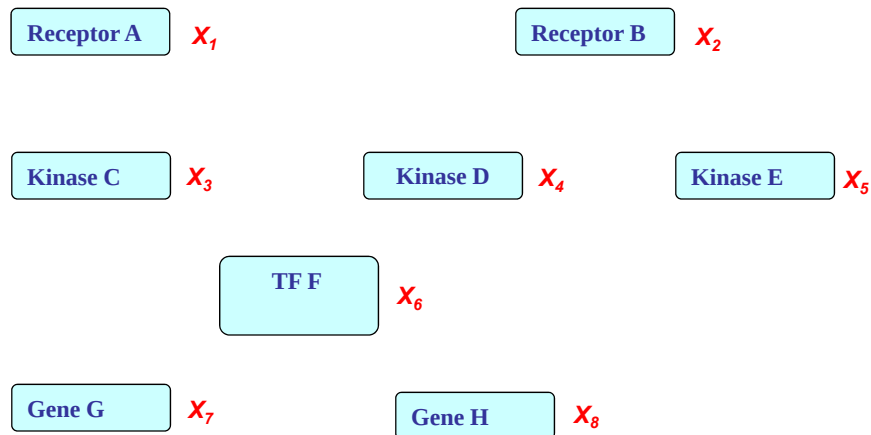
2

What is a Bayesian Network?

--- example from a signal transduction pathway



- A possible world for cellular signal transduction:



© Eric Xing @ CMU, 2006-2011

3

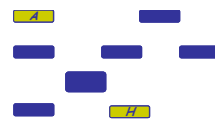
Recap of Basic Prob. Concepts



- 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 2^6 configurations of the unobserved variables

© Eric Xing @ CMU, 2006-2011

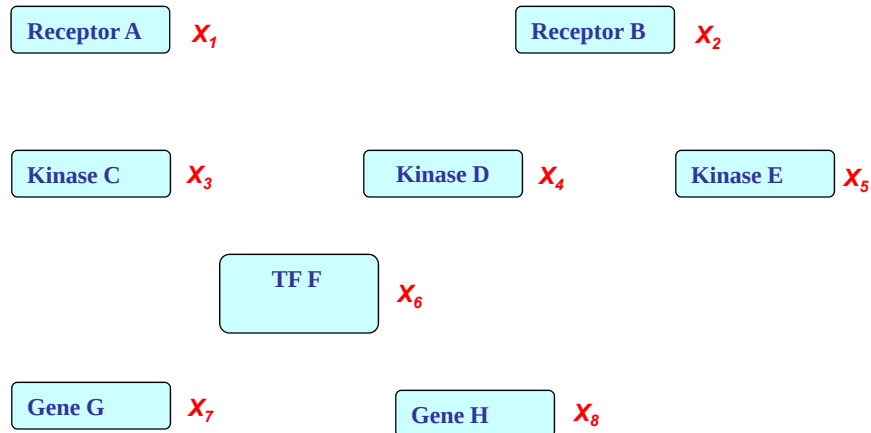
4

What is a Bayesian Network?

--- example from a signal transduction pathway



- A possible world for cellular signal transduction:



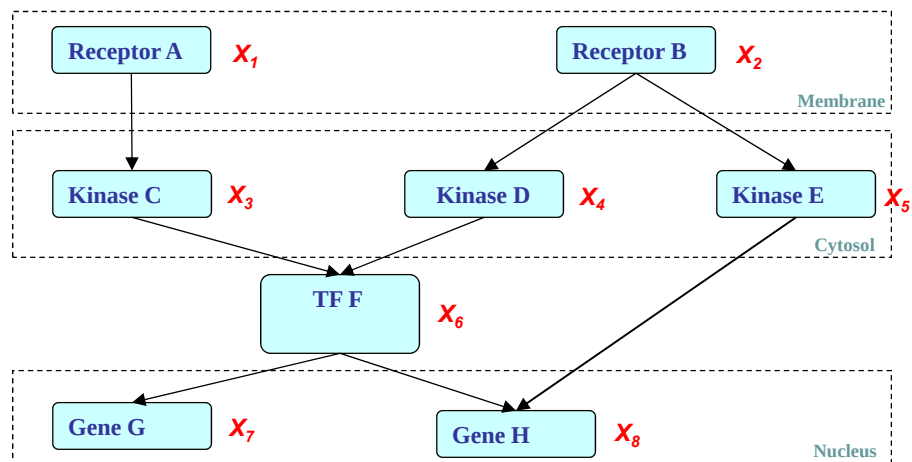
© Eric Xing @ CMU, 2006-2011

5

BN: Structure Simplifies Representation



- Dependencies among variables

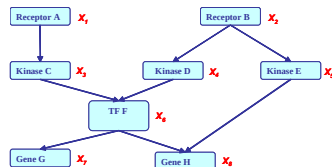


© Eric Xing @ CMU, 2006-2011

6

Bayesian Network

- 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.,



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

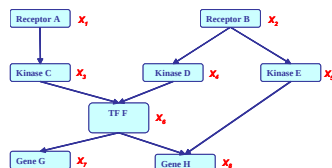
- Why we may favor a BN?
 - Representation cost: how many probability statements are needed?

$$2+2+4+4+4+8+4+8=36$$
, 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

© Eric Xing @ CMU, 2006-2011

7

Bayesian Network: Factorization Theorem



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

• 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”:

$$P(\mathbf{X}) = \prod_i P(X_i | \mathbf{X}_{\pi_i})$$

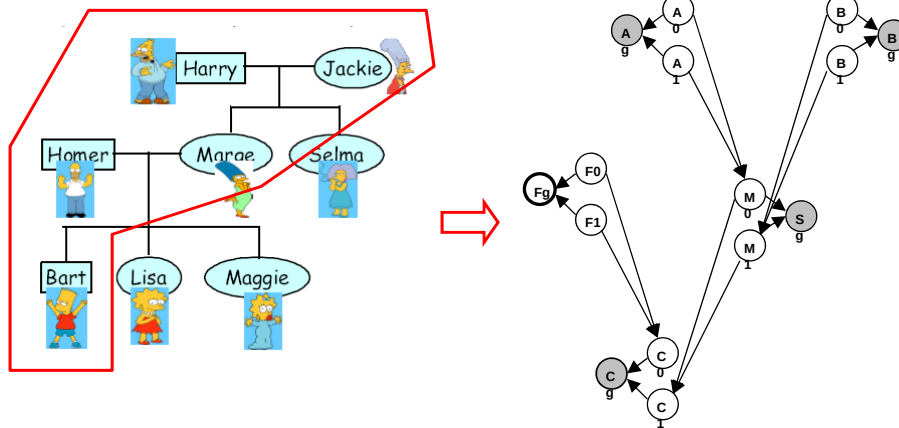
where $\mathbf{X}_{\pi_i}$ is the set of parents of x_i . d is the number of nodes (variables) in the graph.

© Eric Xing @ CMU, 2006-2011

8

Example: a pedigree of people

- Genetic Pedigree



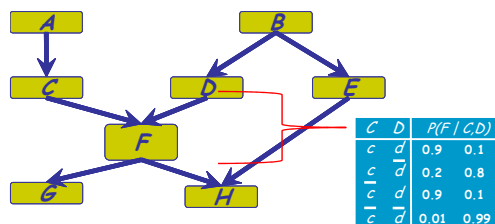
© Eric Xing @ CMU, 2006-2011

9

Specification of a BN

- There are two components to any GM:

- the *qualitative* specification
- the *quantitative* specification



© Eric Xing @ CMU, 2006-2011

10

Qualitative Specification

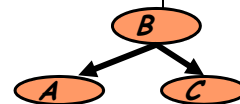


- Where does the qualitative specification come from?
 - Prior knowledge of causal relationships
 - Prior knowledge of modular relationships
 - Assessment from experts
 - Learning from data
 - We simply link a certain architecture (e.g. a layered graph)
 -

Local Structures & Independencies

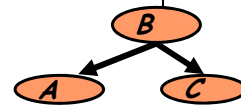


- Common parent
 - Fixing B decouples A and C
 "given the level of gene B , the levels of A and C are independent"
- Cascade
 - Knowing B decouples A and C
 "given the level of gene B , the level gene A provides no extra prediction value for the level of gene C "
- V-structure
 - Knowing C couples A 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 B will decrease"



- The language is compact, the concepts are rich!

A simple justification



© Eric Xing @ CMU, 2006-2011

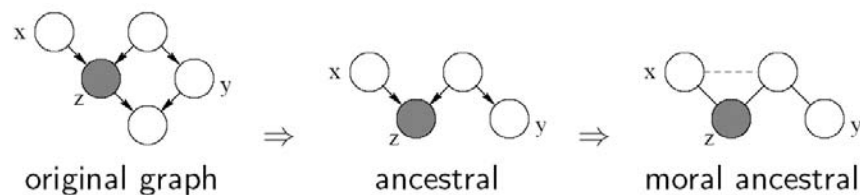
13

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:



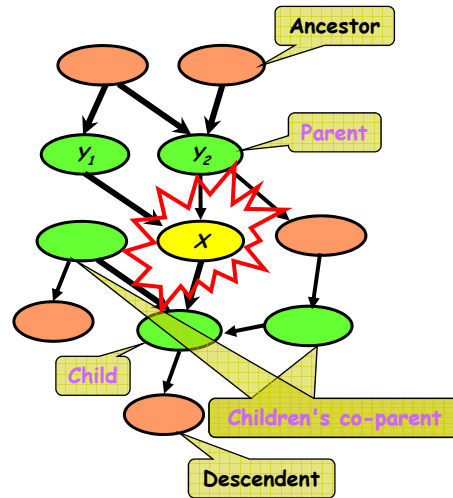
© Eric Xing @ CMU, 2006-2011

14

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**

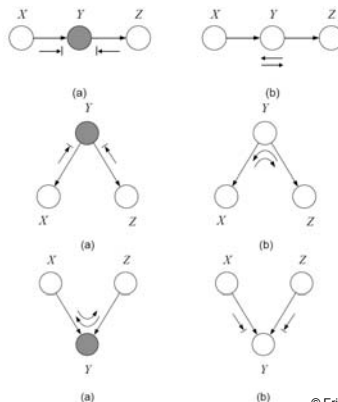


© Eric Xing @ CMU, 2006-2011

15

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):



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

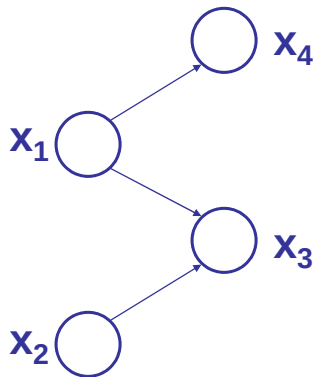
$$I(G) = \{X \perp Z | Y : \text{dsep}_G(X; Z | Y)\}$$

- D-separation is sound and complete

© Eric Xing @ CMU, 2006-2011

16

Example:



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

Essentially: A BN is a database of Pr. Independence statements among variables.

© Eric Xing @ CMU, 2006-2011

17

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 $I(G)$,

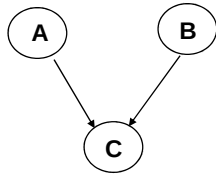
Let $\mathcal{D}_2$ denote the family of all distributions that factor according to G ,

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

© Eric Xing @ CMU, 2006-2011

18

Quantitative Specification



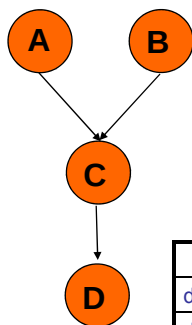
$$p(A,B,C) =$$

Conditional probability tables (CPTs)

a^0	0.75
a^1	0.25

b^0	0.33
b^1	0.67

$$P(a,b,c,d) = P(a)P(b)P(c|a,b)P(d|c)$$



	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

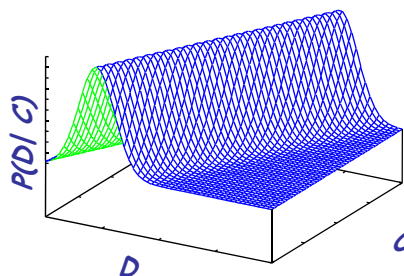
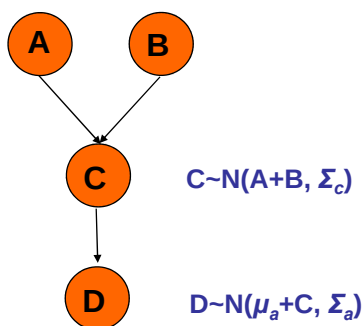
	c^0	c^1
d^0	0.3	0.5
d^1	0.7	0.5

Conditional probability density func. (CPDs)



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

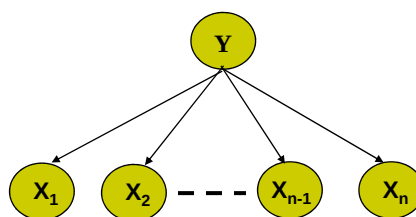
$$P(a,b,c,d) = P(a)P(b)P(c|a,b)P(d|c)$$



© Eric Xing @ CMU, 2006-2011

21

Conditional Independencies



Label

Features

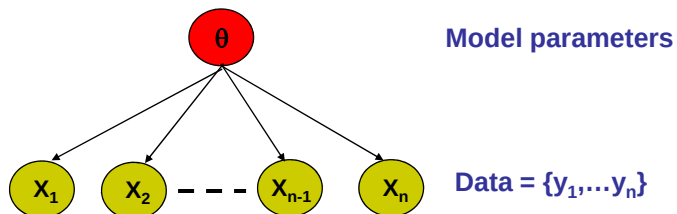
What is this model

1. When Y is observed?
2. When Y is unobserved?

© Eric Xing @ CMU, 2006-2011

22

Conditionally Independent Observations



© Eric Xing @ CMU, 2006-2011

23

“Plate” Notation

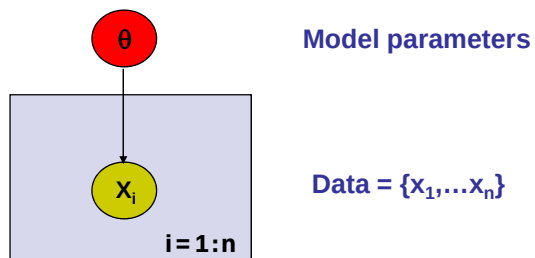


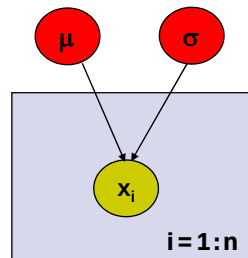
Plate = rectangle in graphical model

variables within a plate are replicated
in a conditionally independent manner

© Eric Xing @ CMU, 2006-2011

24

Example: Gaussian Model



Generative model:

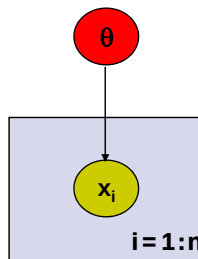
$$\begin{aligned} p(x_1, \dots, x_n \mid \mu, \sigma) &= \prod p(x_i \mid \mu, \sigma) \\ &= p(\text{data} \mid \text{parameters}) \\ &= p(D \mid \theta) \\ &\text{where } \theta = \{\mu, \sigma\} \end{aligned}$$

- 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 $\log L(\theta)$

© Eric Xing @ CMU, 2006-2011

25

Bayesian models



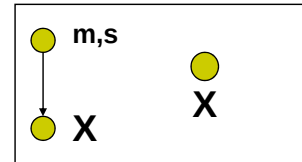
© Eric Xing @ CMU, 2006-2011

26

More examples

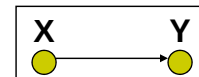
Density estimation

Parametric and nonparametric methods



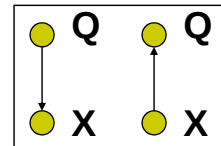
Regression

Linear, conditional mixture, nonparametric



Classification

Generative and discriminative approach

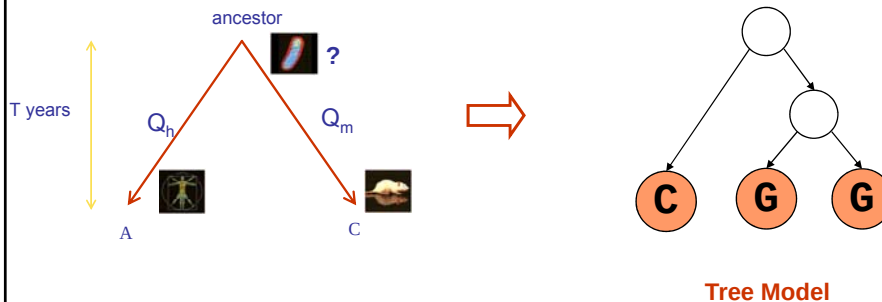


© Eric Xing @ CMU, 2006-2011

27

Example, con'd

- Evolution

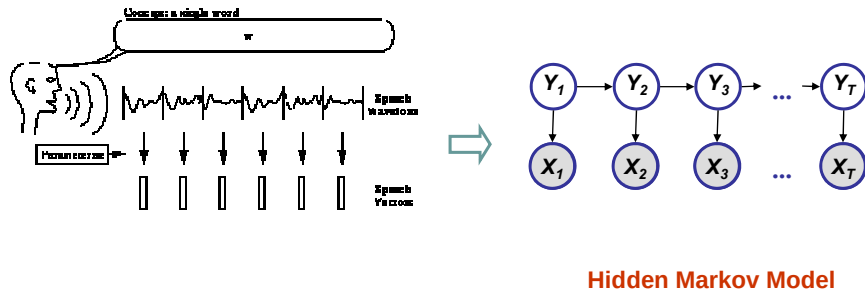


© Eric Xing @ CMU, 2006-2011

28

Example, con'd

- Speech recognition

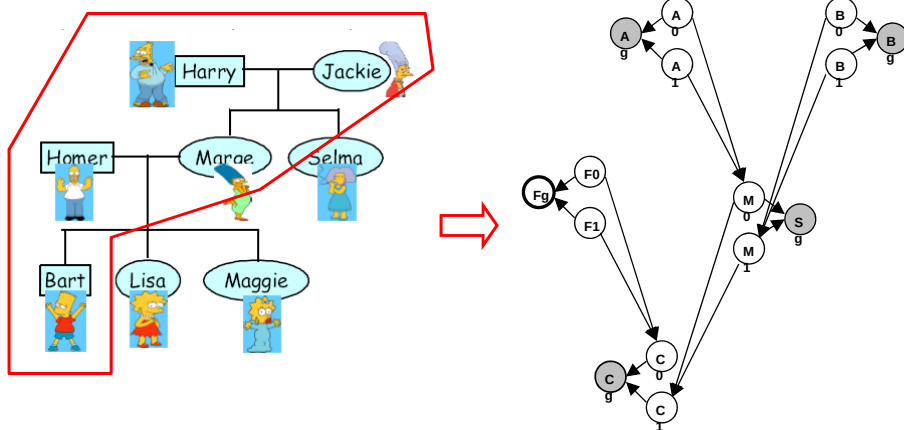


© Eric Xing @ CMU, 2006-2011

29

Example, con'd

- Genetic Pedigree

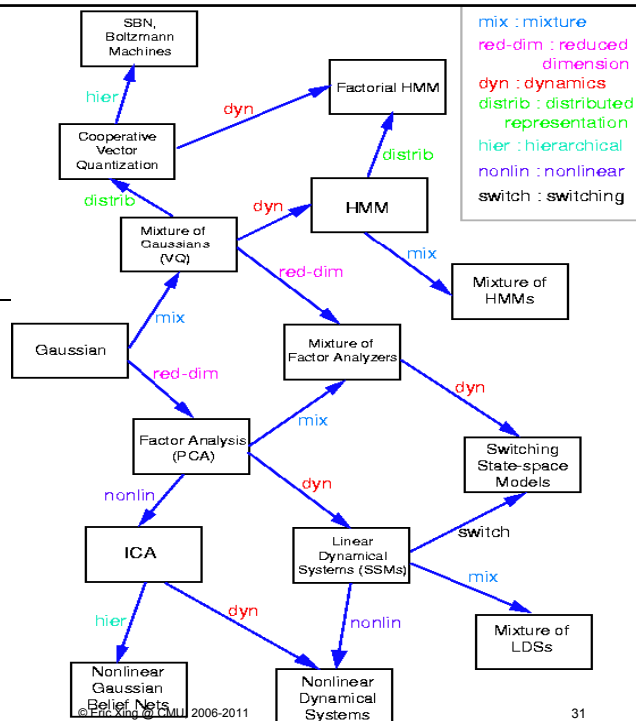


© Eric Xing @ CMU, 2006-2011

30

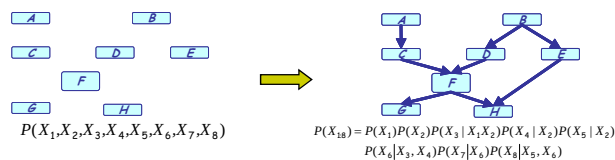
An (incomplete) genealogy of BNs

(Picture by
Zoubin
Ghahramani and
Sam Roweis)



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 $\leftrightarrow$ random variable
 - Edges encode dependencies
 - Absence of edge $\rightarrow$ 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



© Eric Xing @ CMU, 2006-2011

33

Machine Learning

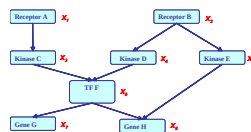
10-701/15-781, Fall 2011

Bayesian Networks and Exact Inference

Eric Xing

Lecture 13-14, October 24, 2011

Reading: Chap. 8, C.B book



© Eric Xing @ CMU, 2006-2011

34

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 ?
 - i. We use **learning** as a name for the process of obtaining point estimate of M .
 - ii. But for *Bayesian*, they seek $p(M|D)$, which is actually an **inference** problem.
 - iii. 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

35

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

© Eric Xing @ CMU, 2006-2011

36

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

$$P(\mathbf{x}_v) = \sum_{\mathbf{x}_H} P(\mathbf{X}_H, \mathbf{x}_v) = \sum_{x_1} \dots \sum_{x_k} P(x_1, \dots, x_k, \mathbf{x}_v)$$

- this is often referred to as computing the **likelihood** of x_v

© Eric Xing @ CMU, 2006-2011

37

Inferential Query 2: Conditional Probability



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

$$P(X_H | X_v = \mathbf{x}_v) = \frac{P(X_H, \mathbf{x}_v)}{P(\mathbf{x}_v)} = \frac{P(X_H, \mathbf{x}_v)}{\sum_{\mathbf{x}_H} P(X_H = \mathbf{x}_H, \mathbf{x}_v)}$$

- 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 :

$$P(Y | \mathbf{x}_v) = \sum_z P(Y, Z = z | \mathbf{x}_v)$$

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

© Eric Xing @ CMU, 2006-2011

38

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

39

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 $\mathbf{Y}$.

© Eric Xing @ CMU, 2006-2011

40

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 signal transduction pathway:

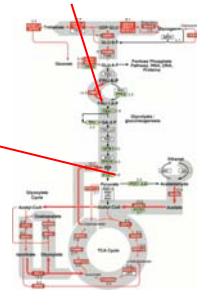


What is the likelihood that protein E is active?

- Query: $P(e)$

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

a naive 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

43

Elimination on Chains



- Rearranging terms ...

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

© Eric Xing @ CMU, 2006-2011

44

Elimination on Chains



- Now we can perform innermost summation

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

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

© Eric Xing @ CMU, 2006-2011

45

Elimination in Chains



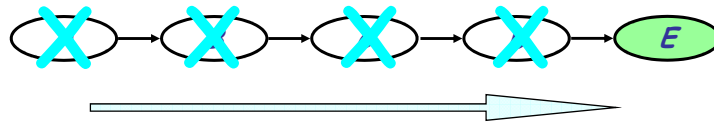
- Rearranging and then summing again, we get

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

© Eric Xing @ CMU, 2006-2011

46

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 $O(k^n)$

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

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

From elimination to message passing



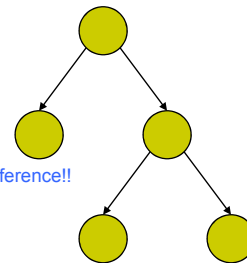
- Recall **ELIMINATION** algorithm:

- 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:

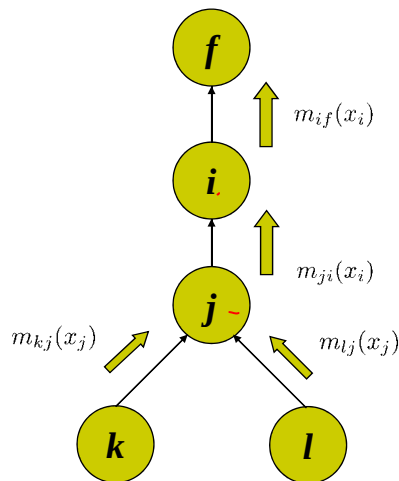
- Choose query node f as the root of the tree
 - View tree as a directed tree with edges pointing towards from f
 - Elimination ordering based on depth-first traversal
 - Elimination of each node can be considered as **message-passing (or Belief Propagation)** directly along tree branches, rather than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



© Eric Xing @ CMU, 2006-2011

49

Message passing for trees



Let $m_{ji}(x_i)$ denote the factor resulting from eliminating variables from bellow 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 .

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

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

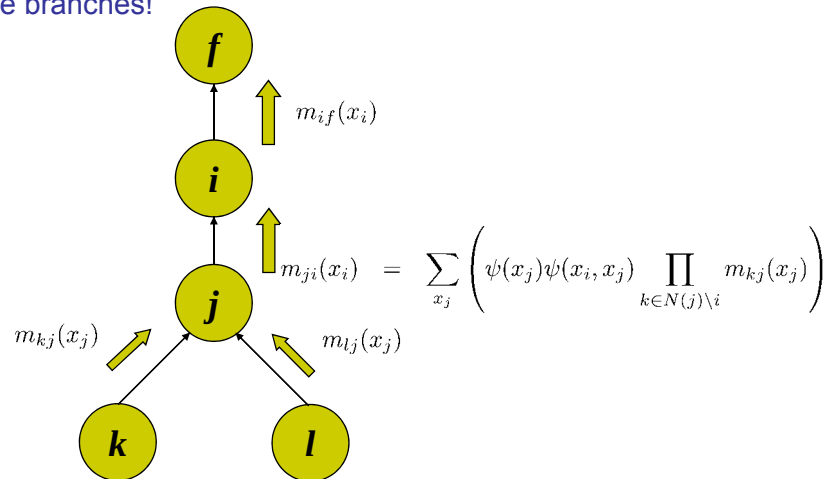
$m_{ji}(x_i)$ represents a "belief" of x_i from x_j !

© Eric Xing @ CMU, 2006-2011

50



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



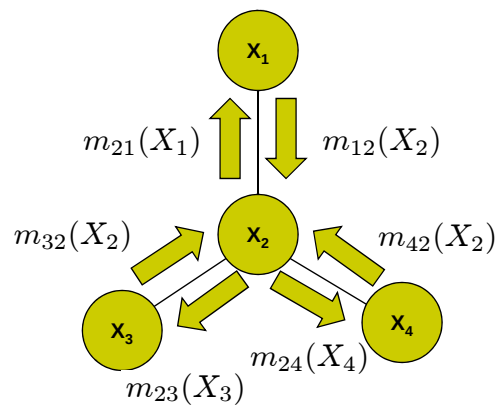
© Eric Xing @ CMU, 2006-2011

51



The message passing protocol:

- A two-pass algorithm:



© Eric Xing @ CMU, 2006-2011

52

Belief Propagation (SP-algorithm): Sequential implementation



```

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

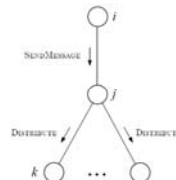
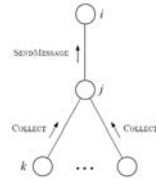
EVIDENCE( $E$ )
  for  $i \in E$ 
     $\psi^E(x_i) = \psi(x_i)\delta(x_i, \bar{x}_i)$ 
  for  $i \notin E$ 
     $\psi^E(x_i) = \psi(x_i)$ 

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

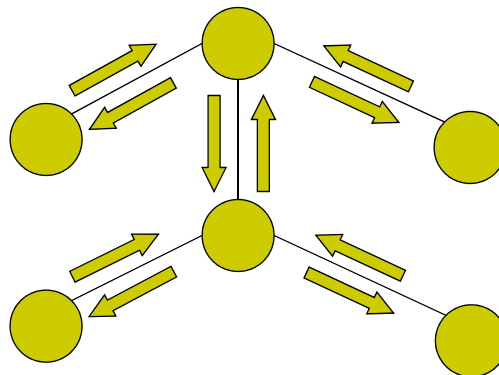
COMPUTEMARGINAL( $i$ )
 $p(x_i) \propto \psi^E(x_i) \prod_{j \in \mathcal{N}(i)} m_{ji}(x_i)$ 
    
```



© Eric Xing @ CMU, 2006-2011

53

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

54

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?

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 ...

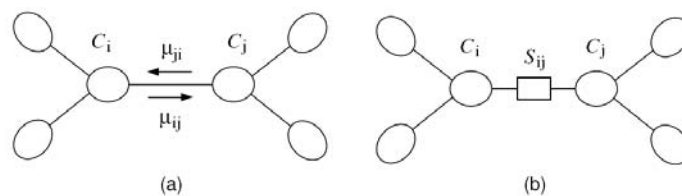
© Eric Xing @ CMU, 2006-2011

57

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

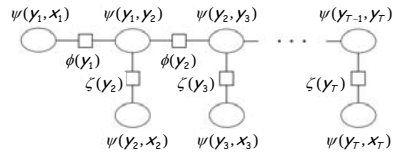
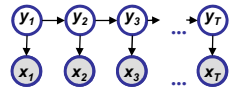
$$p(C_i) \propto \psi_{C_i} \prod_k \mu_{k \rightarrow i}(S_{ki})$$

© Eric Xing @ CMU, 2006-2011

58

The Junction tree algorithm for HMM

- A junction tree for the HMM



- Rightward pass

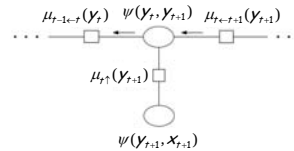
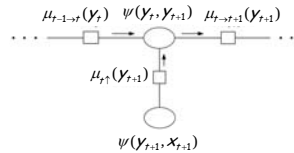
$$\begin{aligned}\mu_{t \rightarrow t+1}(y_{t+1}) &= \sum_{y_t} \psi(y_t, y_{t+1}) \mu_{t-1 \rightarrow t}(y_t) \mu_{t \uparrow}(y_{t+1}) \\ &= \sum_{y_t} p(y_{t+1} | y_t) \mu_{t-1 \rightarrow t}(y_t) p(x_{t+1} | y_{t+1}) \\ &= p(x_{t+1} | y_{t+1}) \sum_{y_t} a_{y_t, y_{t+1}} \mu_{t-1 \rightarrow t}(y_t)\end{aligned}$$

- 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

59

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

60

Extra reading:



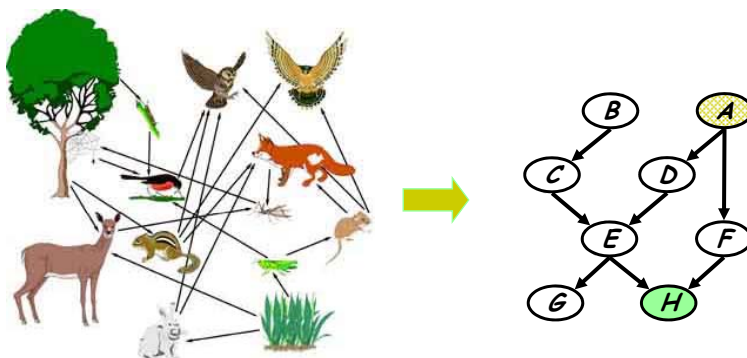
© Eric Xing @ CMU, 2006-2011

61

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

62

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

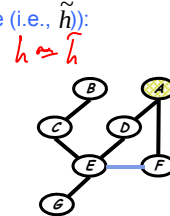
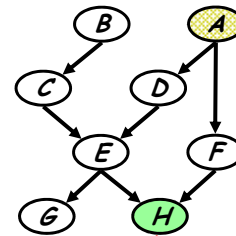
- Step 1:

- 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

63

Example: Variable Elimination

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

- Initial factors:

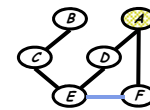
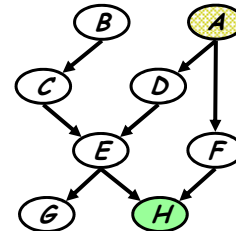
$$P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f) \\ \Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e, f)$$

- 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

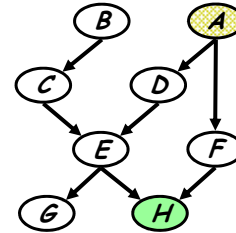
64

Example: Variable Elimination

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

- Initial factors:

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

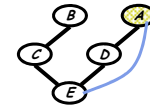


- Step 3: Eliminate F

- 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

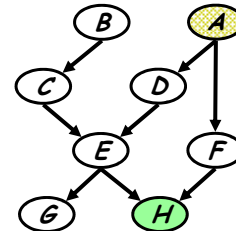
65

Example: Variable Elimination

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

- Initial factors:

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

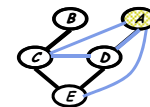


- Step 4: Eliminate E

- 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

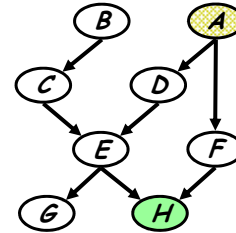
66

Example: Variable Elimination

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

- Initial factors:

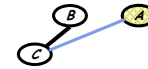
$$\begin{aligned}
 &P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f) \\
 \Rightarrow &P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f) \\
 \Rightarrow &P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)m_h(e,f) \\
 \Rightarrow &P(a)P(b)P(c|b)P(d|a)P(e|c,d)m_f(a,e) \\
 \Rightarrow &P(a)P(b)P(c|b)P(d|a)m_e(a,c,d)
 \end{aligned}$$



- 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

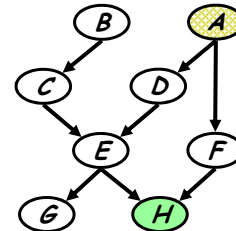
67

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

68

Example: Variable Elimination

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

- Initial factors:

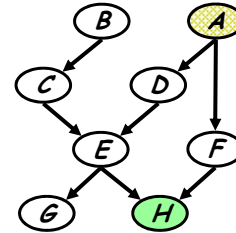
$$\begin{aligned}
 &P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f) \\
 \Rightarrow &P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f) \\
 \Rightarrow &P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)m_h(e,f) \\
 \Rightarrow &P(a)P(b)P(c|d)P(d|a)P(e|c,d)m_f(a,e) \\
 \Rightarrow &P(a)P(b)P(c|d)P(d|a)m_e(a,c,d) \\
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

- 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

69

Example: Variable Elimination

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

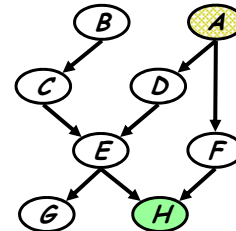
- Initial factors:

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

- 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

70

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

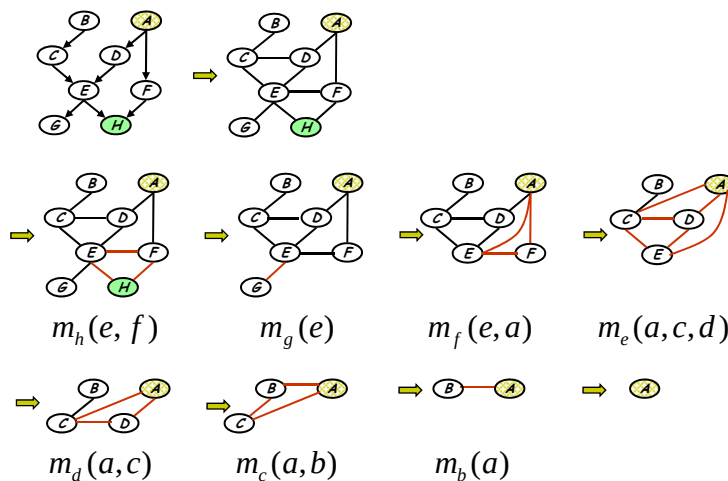
- $k \cdot |\text{Val}(X)| \cdot \prod_i |\text{Val}(Y_{c_i})|$ **multiplications**
 - For each value of $x, y_1, \dots, y_k$ we do k multiplications
- $|\text{Val}(X)| \cdot \prod_i |\text{Val}(Y_{c_i})|$ **additions**
 - For each value of $y_1, \dots, y_k$, we do $|\text{Val}(X)|$ additions

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

© Eric Xing @ CMU, 2006-2011

71

Elimination Cliques

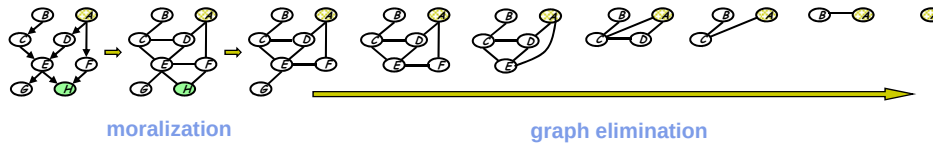


© Eric Xing @ CMU, 2006-2011

72

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

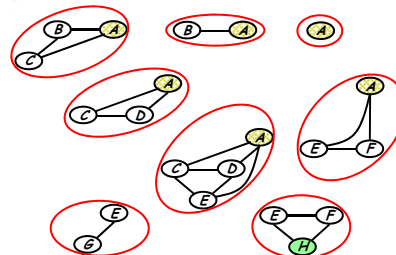
73

Elimination Clique

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

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

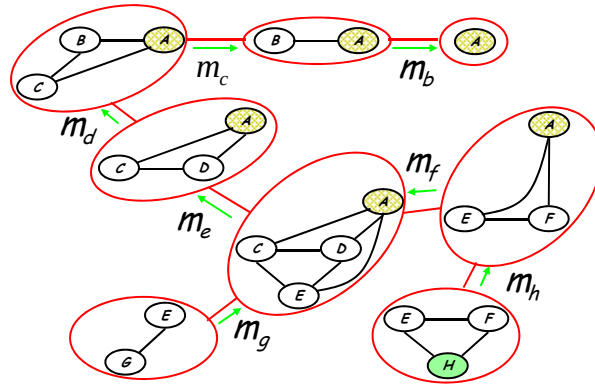
- Can this lead to a generic inference algorithm?



© Eric Xing @ CMU, 2006-2011

74

A Clique Tree



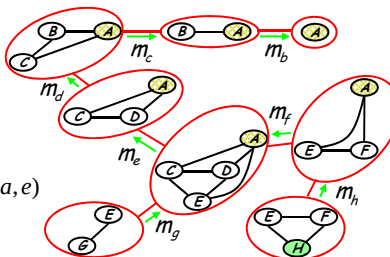
$$m_c(a, c, d) = \sum_e p(e | c, d) m_g(e) m_f(a, e)$$

© Eric Xing @ CMU, 2006-2011

75

From Elimination to Message Passing

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



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

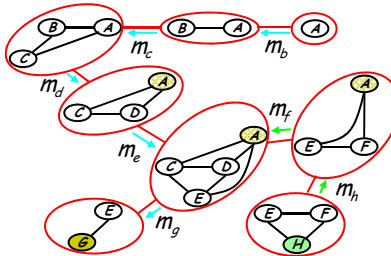
- Messages can be reused

© Eric Xing @ CMU, 2006-2011

76

From Elimination to Message Passing

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



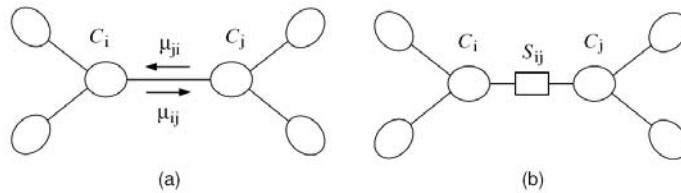
- Messages m_f and m_h are reused, others need to be recomputed

© Eric Xing @ CMU, 2006-2011

77

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

78

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 ...