Lecture 13: Graphical Models & Bayesian Inference Networks

Prof. Krishna R. Pattipati
Dept. of Electrical and Computer Engineering
University of Connecticut
Contact: krishna@engr.uconn.edu **(860) 486-2890**

Fall 2018
December 5, 2018

Lecture Outline

- Graphical Models
- Bayesian Inference in Graphical Models
- Forward-Backward Methods of Inference
- Advanced Methods
- Summary

Reading List

- Bishop, Chapters 8 and 11
- Murphy, Chapters 19-24
- Theodiridis, Chapter 15

Bayes' Theorem

- Basic Axioms of probability
  - Probability of event *A*, $P(A) \in [0,1]$
  - $P(A) = 1 \Leftrightarrow$ A is certain
  - $P(AUB) = P(A \text{ or } B) = P(A) + P(B) - P(A\,B)$

  - Bayes' theorem
    - $P(AB) = P(A|B)P(B) = P(B|A)P(A)$
    - $P(A|B) = \dfrac{P(B|A)P(A)}{P(B)}$
      - **Interested in *A***
      - **Begin with a *priori* probability *P(A)* for our belief about *A***
      - **Observe *B***
      - **Bayes' theorem provides the revised belief about *A*, that is, the posterior probability *P(A|B)***

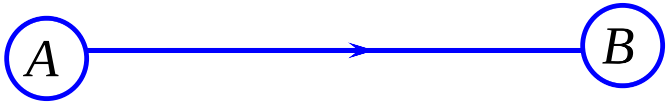
Causality and Inference

- **Likelihood of *A*: The quantity *P(B|A)*, as a function of varying *A* for a fixed *B***
- ***posterior* $\propto$ *prior* $\times$ *likelihood***

  $$\Rightarrow P(A|B) \propto P(A)\,.\,P(B|A)$$

We are inferring A given data B

- Graphical representation of the cause-effect process



A causes *B* (*A is the cause and B is the effect)*

- Why Graphical Structures?
  - Provide a representation for the joint distribution of a set of variables in terms of conditional and prior probabilities
    - Orientation of the arrows represent influence (causation)
    - Corresponding conditional probabilities are obtained from data or elicited from an expert

Bayesian Inference as Operations on a Graph

- Probabilistic (Bayesian) Inference
  - When data is observed, inferencing is required
    - **Involves calculating marginal probabilities of causes conditioned on the observed data using Bayes' theorem**
    - **Diagrammatically equivalent to reversing one or more of the arrows**

" From the observed effect *B* to the inferred cause *A*"

$$P(AB) = P(B) \cdot P(A|B)$$

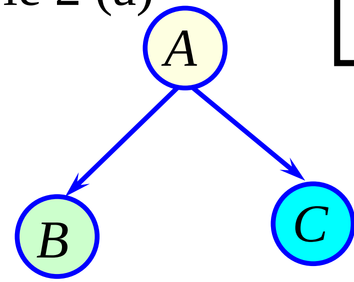
- Example 1

$$P(A \mid B=b) = \frac{P(B=b \mid A)P(A)}{P(B=b)}$$

BN provide a means to infer the distributions of unobserved variables based on observed ones

Tail-to-Tail Dependency

- Example 2 (a)



“tail-to-tail” dependency

- Naïve Bayes
- iid observations
- prediction (sufficient statistics)

$$P(ABC) = P(A)P(B \mid A)P(C \mid A)$$

When A is not observed, B & C are dependent.
When A is observed, B &C are conditionally independent!

- “Factorization of Joint Distribution”, suppose
  - **Know $P(A)$, $P(B|A)$ and $P(C|A)$**
  - **Observed $B = b$**
  - **Calculate $P(C|B = b)$**

$B \perp C \mid A$

"B is independent of C given A"

why?

$$P(B,C) = \sum_A P(B \mid A)P(C \mid A)P(A) \neq P(B)P(C)$$

$$P(B,C \mid A) = \frac{P(A,B,C)}{P(A)} = P(B \mid A)P(C \mid A)$$

- One way of computing it:

1. Calculate $P(ABC)$

2. Compute $P(B) = \sum_A \sum_C P(A,B,C) \Rightarrow$ can get $P(B = b)$

3. Compute $P(B,C) = \sum_A P(A,B,C) \Rightarrow$ can get $P(C, B = b)$

4. Calculate $P(C \mid B = b) = \dfrac{P(C, B = b)}{P(B = b)}$

$$P(A,B,C) = \frac{\overbrace{P(A,B)}^{clique}\overbrace{P(A,C)}^{clique}}{\underbrace{P(A)}_{separator}}$$

Exploiting Dependency Structure

- Problem:

  Need to compute $|A||B||C|$ entries to compute $P(A,B,C)$

  – If $|A| = |B| = |C| = 10 \rightarrow$ need 1000 entries

- Alternate way: Exploit the graph structure

  1. **Calculate** $P(A \mid B=b) = \dfrac{P(B=b \mid A)P(A)}{P(B=b)}$ **using Bayes' rule,**

     where $P(B=b) = \sum_A P(B=b \mid A)P(A)$ $\rightarrow$ **arc reversal or inferencing**

  2. **Find**

$$P(C \mid B=b) = \sum_A P(C, A \mid B=b)$$
$$= \sum_A P(C \mid A, B=b)P(A \mid B=b)$$
$$= \sum_A P(C \mid A)P(A \mid B=b)$$

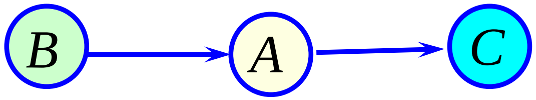
  ▪ Advantage: Need to store only 100 entries when $|A| = |B| = |C| = 10$

Head-to-Tail Dependency

- Example 2 (b)

"head-to-tail" dependency

- Markov Chains
- HMMs



$$P(ABC) = P(B)P(A|B)P(C|A)$$

$$Also,\ P(ABC) = \frac{P(AB)P(CA)}{P(A)}$$

$\{AB\}$, $\{CA\}$ "cliques"
A "separator"
P(AB), P(CA) are "clique potentials"
P(A) "separator potential"

When A is not observed, B & C are dependent.
When A is observed, B &C are conditionally independent!

$$B \perp C \mid A$$

"B is independent of C given A"

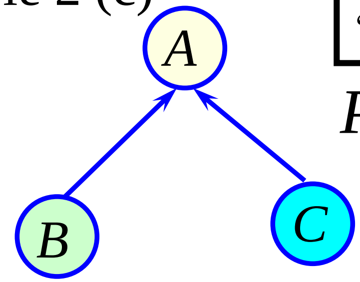
why?

$$P(B,C) = P(B)\sum_{A} P(A|B)P(C|A) \neq P(B)P(C)$$

$$P(B,C|A) = \frac{P(A,B,C)}{P(A)} = \frac{P(A|B)P(B)P(C|A)}{P(A)} = P(B|A)P(C|A)$$

Head-to-Head Dependency

- Example 2 (c)



"head-to-head" dependency

$$P(ABC) = P(B)P(C)P(A|B,C)$$

When A is not observed, B & C are **independent**!!
When A is observed, B & C are conditionally **dependent**!!

$B \perp C \mid \varnothing$

"B is independent of C given no evidence"

why?

$$P(B,C) = \sum_A P(B)P(C)P(A|B,C) = P(B)P(C)$$

They are not independent given A

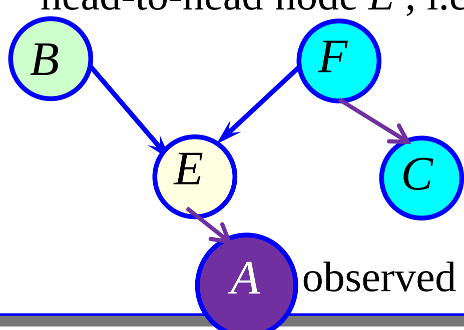
$$P(B,C|A) = \frac{P(A,B,C)}{P(A)} = \frac{P(B)P(C)P(A|B,C)}{P(A)}$$

When A is not observed, A blocks the path from B to C. However, when A is observed, it unblocks the path from B to C $\Rightarrow$ *they become dependent*

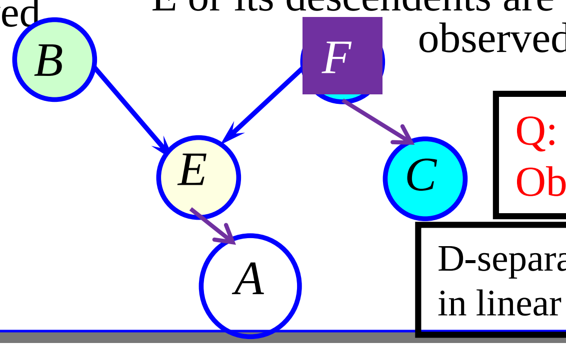
D-Separation

- D (dependency)-separation: ideas extend to general directed graphs and subsets of nodes.
  - To check conditional independence of $B \perp C \mid A$ for subsets of nodes *A*, *B* and *C*. Consider all possible paths from any node in *B* to any node in *C* . Any such path is blocked if it includes a node such that either
    - The arrows on the path are either *tail-to-tail* or *head-to-tail* at the node, and the node is in the set *A (observed)*, or
    - The arrows meet *head-to-head* at the node, and *neither the node, nor any of its descendents are in the set A (i.e., observed)*.
  - If all paths are blocked, then *B* is d-separated from *C* by *A*.

B & *C* are not d-separated because *F* is not observed (tail-to-tail) and descendent of head-to-head node *E* , i.e., *A* is observed



B & *C* are d-separated because F is observed (tail-to-tail) and head-to-head node E or its descendents are not observed.



Q: What if A is also Observed?

D-separation can be computed in linear time

Historical Perspective - 1

- Formalization of ideas
  - Graphical structures . . . A historical perspective from a communication perspective
  - Sewall Wright (1921) . . . Developed “*path analysis*” as a means to study statistical relationships in biological data
  - 1960’s . . . Statisticians use graphs to describe restrictions in log-linear statistical models
  - Gallagher (1963) . . . Error correcting codes as probabilistic graphs
  - Viterbi algorithm (Forney, 1973)

Historical Perspective - 2

- AI literature
  - Taxonomic hierarchies (Woods, 1975)
  - Medical diagnosis (Spiegelharter, 1990)
  - Exact algorithms for computing the joint probability distribution (Lauritzen and Spiegelharter, 1988; Pearl, 1986)
  - Learning parameters in graph-based log-linear models (Hinton and Sejnowski, 1986)
  - Bayesian networks (belief networks, causal networks or inference diagrams)
    - o Approximate algorithm based on Monte Carlo methods
    - o Helmholtz machines
    - o Variational techniques

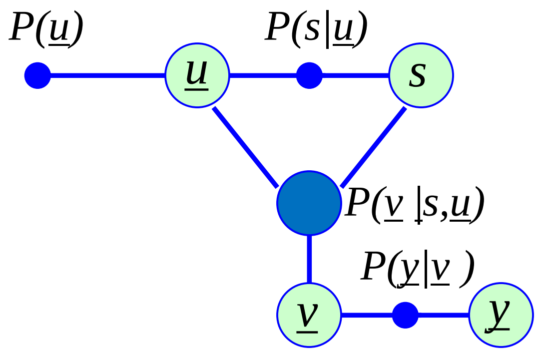
See books by Frey and M.I. Jordan
Also, Bishop's book and the book by Koller and Freidman

Similar to GMM we discussed earlier

Factor Graphs & Markov Random Fields

- Factor graphs
  - Suppose $P(\underline{u}, s, \underline{v}, \underline{y}) = P(\underline{u}) P(s \mid \underline{u}) P(\underline{v} \mid s, \underline{u}) P(\underline{y} \mid \underline{v})$

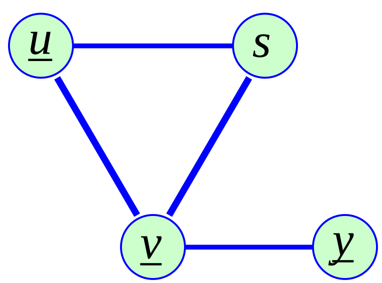
Show Bi-partite graph



$\{\underline{u}, s\} \perp \underline{y} \mid \underline{v}$

All paths that pass from $\underline{u}$ and s to $\underline{y}$ pass through $\underline{v}$

- Markov random fields (MRF)…. Undirected graphs



Markov chains in higher dimensions

Hammersley-Clifford Theorem

- Properties of MRF
  - Undirected graph with nodes corresponding to variables
  - $P(z_i \mid \mathbf{z} \setminus z_i) = P(z_i \mid n_i)$

    z_i variable

    n_i neighbors of variable z_i

    $\mathbf{z}$ = set of all variables (e.g., $\mathbf{z} = \{\underline{u}, s, \underline{v}, \underline{y}\}$)

- Local Markov Property
- Clique-based Factorization
- Global Markov Property (D-separation)

"Given its neighbors, each variable is independent of all other variables"

  - Joint Distribution is given by Hammersley-Clifford Theorem (1971)

    $P(\mathbf{z})$ = product of clique potentials

    - **Clique is a fully connected sub-graph that cannot remain fully connected if more variables are included**
    - **Cliques in the graphs**

$$C_1 = \{\underline{u}, s, \underline{v}\} \quad C_2 = \{\underline{v}, \underline{y}\}$$

$$P(\mathbf{z}) = \alpha \prod_{j=1}^{NC} \psi_j(C_j) = \alpha \exp\{-\sum_{j=1}^{NC} E_j(C_j)\}$$

NC = Number of cliques $\qquad \alpha$ = Normalization Constant

Typical model of $\psi_j(C_j)$

$$\psi_j(C_j) = \exp(\{-E_j(C_j)\})$$

$$E_j(C_j) = \text{energy function}$$

Factor Graph Example

- For the channel coding example
  - $P(\underline{u}, s, \underline{v}, \underline{y}) = \alpha\, \psi_1(\underline{u}, s, \underline{v})\, \psi_2(\underline{v}, \underline{y})$

when $\alpha = \dfrac{1}{P(\underline{v})}$

cliques

separator

$\psi_1 = P(\underline{u}, s, \underline{v})$ and

$\psi_2 = P(\underline{v}, \underline{y})$

we obtain the joint distribution

Joint distribution = product of clique potentials/Product of separator potentials

Ising Model : Image-denoising

- Observed *noisy* image described by an array of binary pixel values

$$y_i \in \{-1,+1\}, i = 1,2,.., p$$

Variation: $p(y_i | x_i)$ *Gaussian*

- Original (hidden) image has binary pixel values

$$x_i \in \{-1,+1\}, i = 1,2,.., p$$

- Joint distribution $p(\underline{x}, \underline{y})$ is the *Boltzmann* distribution

$$p(\underline{x}, \underline{y}) = \alpha \exp\{-E(\underline{x}, \underline{y})\}$$

$$E(\underline{x}, \underline{y}) = \underbrace{\sum_{i=1}^{p} h_i x_i}_{\substack{\text{to bias towards} \\ -1 \text{ or } +1}} \underbrace{-\sum_{i=1}^{p} \sum_{j \in n_i} \beta_{ij} x_i x_j}_{\substack{\text{want energy to be small} \\ \text{when } x_i \text{ and } x_j \text{ have same sign}}} \underbrace{-\sum_{i=1}^{p} \eta_i x_i y_i}_{\substack{\text{want energy to be small} \\ \text{when } x_i \text{ and } y_i \text{ have same sign}}}$$

- MAP estimate via mean field (variational approximation)

$$q(\underline{x}) = \prod_{i=1}^{p} q_i(x_i, \mu_i); \mu_i = \text{mean value of pixel } i$$

$$\log q_i(x_i) = E_{-q_i}\{\log(p(\underline{x}, \underline{y})\} + constant$$

Mean Field Method

- Keep all pixels $j \neq i$ at their mean values

$$\log q_i(x_i) = E_{q_{-i}}[\log p(\underline{x}, \underline{y})] = x_i \left(\sum_{j \in n_i} \beta_{ij} \mu_j - h_i + \eta_i y_i \right) + constant$$

$$\therefore q_i(x_i) \propto \exp(x_i \left[\left(\sum_{j \in n_i} \beta_{ij} \mu_j - h_i \right) + \eta_i y_i \right])$$

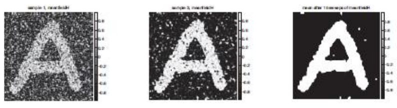
$$\Rightarrow q_i(x_i = 1) \propto \exp(a_i); q_i(x_i = -1) \propto \exp(-a_i); a_i = \left(\sum_{j \in n_i} \beta_{ij} \mu_j - h_i \right) + \eta_i y_i$$

- Update μ_i and iterate until convergence

$$\mu_i = q_i(x_i = 1) - q_i(x_i = -1) = \frac{e^{a_i} - e^{-a_i}}{e^{a_i} + e^{-a_i}} = \tanh(a_i)$$

- It is usually good to *low pass filter* the updates ($\lambda \leq 0.5$)

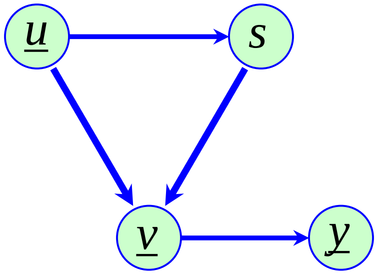
$$\mu_i^{t+1} = \lambda \mu_i^t + (1 - \lambda) \tanh(a_i^t)$$



Iteration 1 Iteration 3 Iteration 15

Directed Acyclic Graphs

- Bayesian Networks
  - Represented in terms of directed acyclic graphs



$$P(\underline{u}, s, \underline{v}, \underline{y}) = P(\underline{u})P(s \mid \underline{u})P(\underline{v} \mid s, \underline{u})P(\underline{y} \mid \underline{v})$$

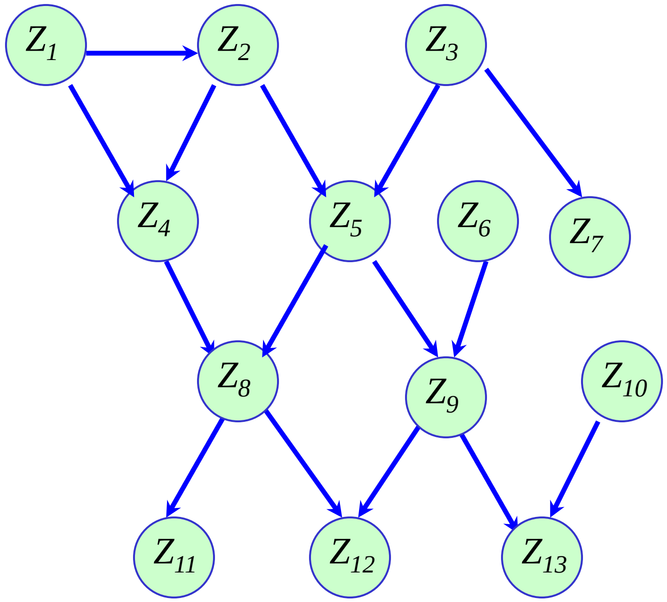
  - $\mathbf{z} = [z_1 \; z_2 \cdots\cdots z_N]$

    $P(z_k \mid a_k) \qquad a_k = \text{parents of } z_k = pa(z_k)$

    $$P(\mathbf{z}) = \prod_{k=1}^{N} P(z_k \mid pa(z_k)) = \prod_{k=1}^{N} P(z_k \mid a_k)$$

Chain Rule for Bayesian Networks

- Example 1



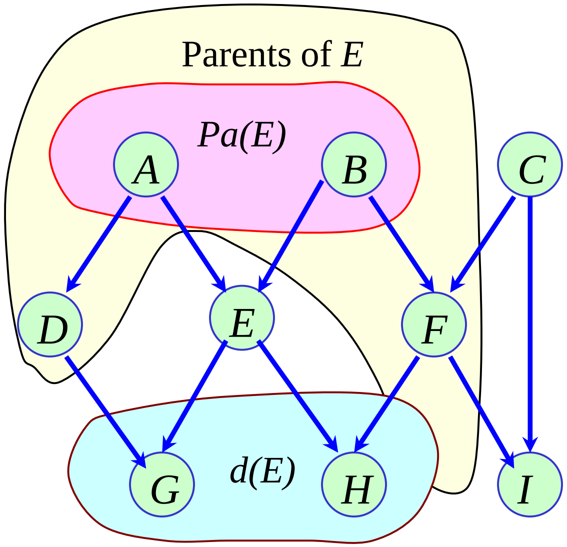
  – Topological order: $\left(z_1\ z_2\ z_3 \cdots z_{13}\right)$

$$P(\mathbf{z}) = P(z_1).P(z_2 \mid z_1).P(z_3).P(z_4 \mid z_1, z_2).P(z_5 \mid z_2, z_3).$$
$$P(z_6).P(z_7 \mid z_3).P(z_8 \mid z_4, z_5).P(z_9 \mid z_5, z_6).P(z_{10}).$$
$$P(z_{11} \mid z_8).P(z_{12} \mid z_8, z_9).P(z_{13} \mid z_9, z_{10})$$

Markov Blanket

- Example 2:

Markov Blanket of a node *E* is denoted by ∂E. ∂E = its parents, its children, and its children's other parents.



$\{A,B\}$ = Parents of E

$\{D,F\}$=its children's other parents

$$P\left(E \mid \partial E, C, I\right) = P(E \mid \partial E)$$
$$\partial E = \{A, B, D, F, G, H\}$$

$d(E)$ = Descendents of E

= Children of E={G,H}

$$P\left(A,B,C,D,E,F,G,H,I\right) = P\left(A\right)P\left(B\right)\left(C\right)P\left(D \mid A\right)P\left(E \mid A,B\right)P\left(F \mid B,C\right)$$
$$P\left(G \mid D,E\right)P\left(H \mid E,F\right)P\left(I \mid C,F\right)$$

For an undirected graph (Markov Random Field), Markov Blanket of a node is the set of its neighboring nodes

Topological Ordering

- Can arrange nodes in topological order
  - For each node *x* all of its parents *pa(x)* precede it in the ordering

- Topological orders are not unique
  - Order 1: *{A,B,C,D,E,F,G,H,I}*
  - Order 2: *{B,A,E,D,G,C,F,I,H}*

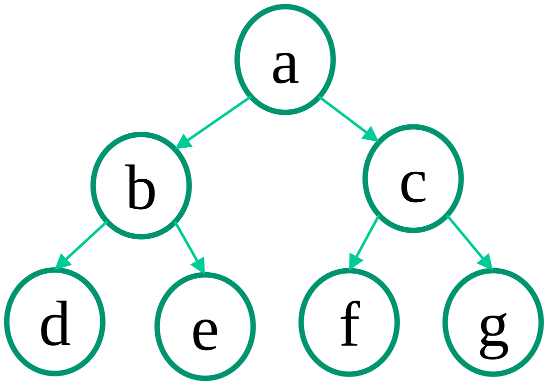
Constructing Topological Ordering

- Algorithms for finding topological ordering
  - Algorithm 1:
    - Start with the graph and an empty list
    - Successively delete from the graph any node which does not have any parents, and add it to the end of the list
    - Stop when no node has parent nodes
  - Algorithm 2:
    - Start with the graph and an empty list
    - Successively delete from the graph nodes which have no children and add them to the beginning of the list
    - Stop when no node has child node

Inference via Variable Elimination

- Consider the out-tree structure

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



- Suppose you observe *e* and *f*
- Want to compute *P*(*d*|*e*,*f*)

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

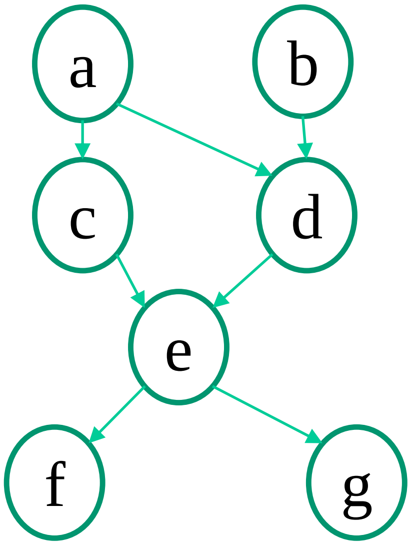
→ Too much computation by brute force

$$= \sum_b P(d|b) \sum_a P(b|a,e) \sum_c P(a|c) \sum_g P(c,g|f)$$

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

- Ordered summation: *c*, *a*, *b*
- What if the graph is a general directed acyclic graph (DAG)?

Variable Elimination for DAGs



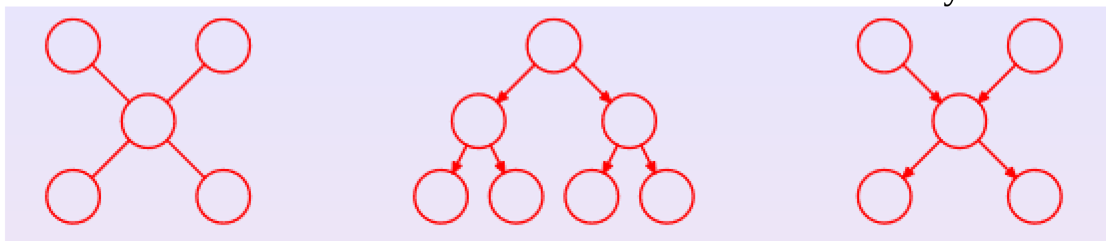
- Consider the Directed Acyclic Graph (DAG). Want to find the marginal probability $P(g)$

- Steps involve sums and products
  - Compute the product $P(a,b,d) = P(a)P(b)P(d\,|\,a,b)$
  - Sum over b to get $P(a,d) = \sum_b P(a,b,d)$
  - Multiply $P(a,d)$ by $P(c\,|\,a)$ to get $P(a,c,d) = P(c\,|\,a)P(a,d)$
  - Sum $P(a,c,d)$ over a to get $P(c,d) = \sum_a P(a,c,d)$
  - Multiply $P(c,d)$ by $P(e\,|\,c,d)$ to obtain $P(c,d,e) = P(e\,|\,c,d)P(c,d)$
  - Sum over c and d to get $P(e) = \sum_c \sum_d P(c,d,e)$
  - Multiply $P(e)$ by $P(g\,|\,e)$ to get $P(e,g)$
  - Sum $P(e,g)$ over e to get $P(g) = \sum_e P(e,g)$
- Complexity is exponential in the size of factors and optimal ordering of computations is NP-hard
- Is there a formal (and nicer) way to do inference in Bayesian networks? For trees, there is a nice *sum-product algorithm* as in HMMs. For general DAGs, *Junction tree algorithm.*

Sum-Product Algorithm using Factor Graphs

- Trees: single undirected path between each pair of nodes

Undirected Directed Polytree

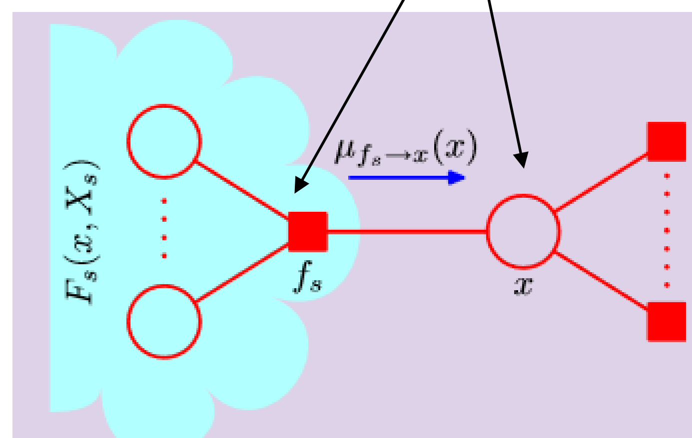


Joint probability distribution as a product of factors

$$p(\underline{x}) = \prod_s f_s(\underline{x}_s); s = \text{factor}; \underline{x}_s = \text{subset of variables in factor } s$$

Also, $p(\underline{x}) = \prod_{s \in ne(x)} F_s(x, \underline{X}_s); \underline{X}_s =$ set of all variables in the subtree connected to x via s

Factor graphs are bi-partite graphs



$$Example: p(\underline{x}) = f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4)$$

$$For\ x_1 : p(\underline{x}) = F_a(x_1, \underbrace{x_2, x_3, x_4}_{\underline{X}_a})$$

$$= f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4)$$

$$For\ x_2 : p(\underline{x}) = \underbrace{f_a(x_1, x_2)}_{F_a(x_2, X_a)} \underbrace{f_b(x_2, x_3)}_{F_b(x_2, X_b)} \underbrace{f_c(x_2, x_4)}_{F_c(x_2, X_c)};$$

$$X_a = x_1, X_b = x_3, X_c = x_4$$

$$so, |ne(x_i)| = number\ of\ terms\ involving\ x_i\ in\ p(\underline{x})$$

Message Passing between Variables and Factors: Example

- **Factors → Variables Messages (SUM)**

  messages sent by a factor node to a variable node involves multiplying all the incoming messages (except variable node *x*) with the factor and summing over all the variables except *x*

$$\mu_{f_a \to x_1}(x_1) = \sum_{x_2} f_a(x_1, x_2)\mu_{x_2 \to f_a}(x_2); \mu_{f_a \to x_2}(x_2) = \sum_{x_1} f_a(x_1, x_2)\mu_{x_1 \to f_a}(x_1)$$

$$\mu_{f_b \to x_2}(x_2) = \sum_{x_3} f_b(x_2, x_3)\mu_{x_3 \to f_b}(x_3); \mu_{f_b \to x_3}(x_3) = \sum_{x_2} f_b(x_2, x_3)\mu_{x_2 \to f_b}(x_2)$$

$$\mu_{f_c \to x_2}(x_2) = \sum_{x_4} f_c(x_2, x_4)\mu_{x_4 \to f_c}(x_4); \mu_{f_c \to x_4}(x_4) = \sum_{x_2} f_c(x_2, x_4)\mu_{x_2 \to f_c}(x_2)$$

- **Variables → Factors Messages (PRODUCT)**

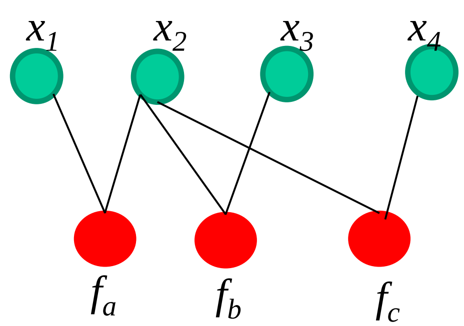
  Message sent by a variable node to a factor node is the product of all the incoming messages along all of the other links (factors)

$$\mu_{x_1 \to f_a}(x_1) = 1; \mu_{x_3 \to f_b}(x_3) = 1; \mu_{x_4 \to f_c}(x_3) = 1$$

$$\mu_{x_2 \to f_a}(x_2) = \mu_{f_b \to x_2}(x_2)\mu_{f_c \to x_2}(x_2);$$

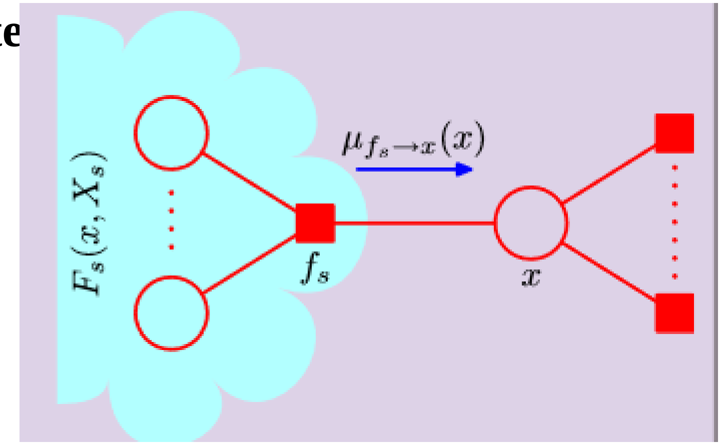
$$\mu_{x_2 \to f_b}(x_2) = \mu_{f_a \to x_2}(x_2)\mu_{f_c \to x_2}(x_2);$$

$$\mu_{x_2 \to f_c}(x_2) = \mu_{f_a \to x_2}(x_2)\mu_{f_b \to x_2}(x_2)$$



Message Passing between Factors and Variables

- **Use the node for which you want to compute marginal probability as the root**



- **Factors → Variable Messages**

Marginal probability of a variable x

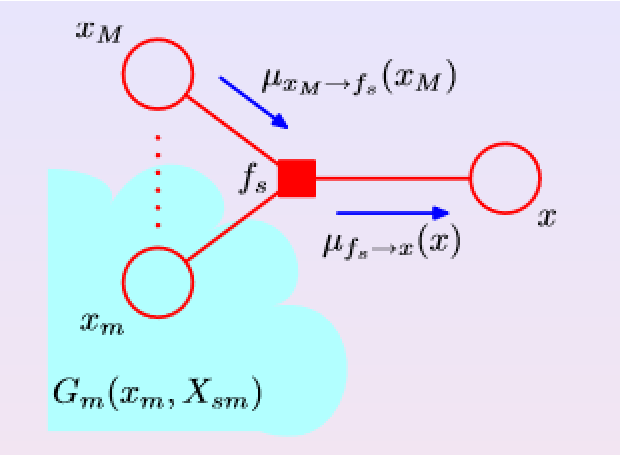
$$p(x) = \prod_{s \in ne(x)} \sum_{\underline{X}_s} F_s(x, \underline{X}_s) = \prod_{s \in ne(x)} \mu_{f_s \to x}(x)$$

Recursively,

$$F_s(x, \underline{X}_s) = f_s(\underbrace{x, x_1, .., x_M}_{\underline{x}_s}) G_1(x_1, \underline{X}_{s1}) ... G_M(x_M, \underline{X}_{sM})$$

$$\mu_{f_s \to x}(x) = \sum_{x_1} ... \sum_{x_M} f_s(x, x_1, .., x_M) \prod_{m \in ne(f_s) \backslash x} \underbrace{\sum_{\underline{X}_{sm}} G_m(x_m, \underline{X}_{sm})}_{\mu_{x_m \to f_s}(x_m)}$$

$$= \sum_{x_1} ... \sum_{x_M} f_s(x, x_1, .., x_M) \prod_{m \in ne(f_s) \backslash x} \mu_{x_m \to f_s}(x_m)$$



If factor s is a leaf node with only variable x,

set $\mu_{f_s \to x}(x) = f_s(x)$

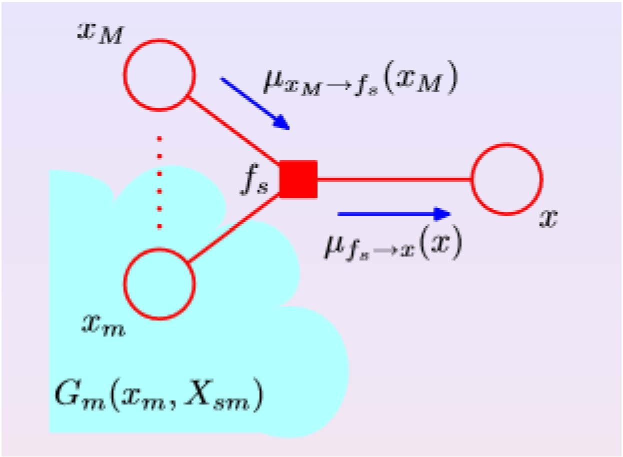
Messages passed along a link are always a function of the variable it is connected to

Message Passing between Variables and Factors

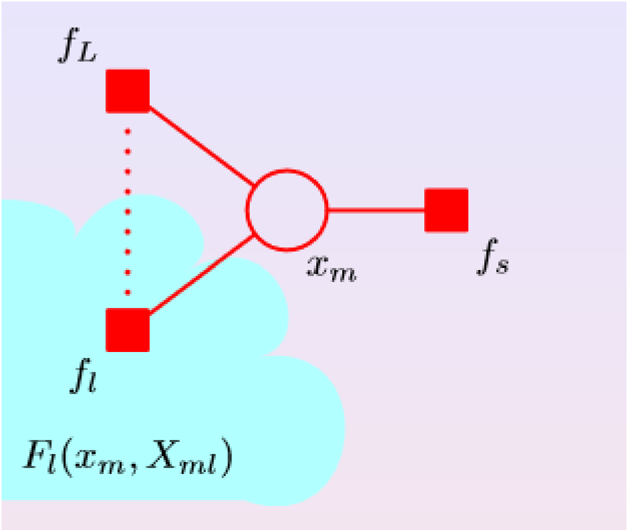
- **Variables→ Factors Messages**

$$\mu_{x_m \to f_s}(x_m) = \sum_{\underline{X}_{sM}} G(x_M, \underline{X}_{sM})$$
$$= \sum_{\underline{X}_{sM}} \prod_{l \in ne(x_m) \setminus f_s} F_l(x_m, \underline{X}_{ml})$$
$$= \prod_{l \in ne(x_m) \setminus f_s} \sum_{\underline{X}_{ml}} F_l(x_m, \underline{X}_{ml})$$
$$= \prod_{l \in ne(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m)$$

If x_m is a leaf node, $\mu_{x_m \to f_s}(x_m) = 1$



- Message sent by a variable node to a factor node is the product of all the incoming messages *along all of the other links* (factors)
- On the other hand, messages sent by a factor node to a variable node involves multiplying all the incoming messages (except variable node *x*) with the factor and summing over all the variables except *x*



Computing All Marginal Probabilities

- Select an arbitrary node as the root and propagate messages from the leaves to the root as in the sum-product algorithm for a single root node
- Send messages from the root all the way back to the leaves
- Now calculate the marginal probability at each variable and factor node via

$$p(x) = \prod_{s \in ne(x)} \mu_{f_s \to x}(x); \; p(\underline{x}_s) = f_s(\underline{x}_s) \prod_{x_i \in f_s} \mu_{x_i \to f_s}$$

- Can eliminate messages from variable nodes to factors via

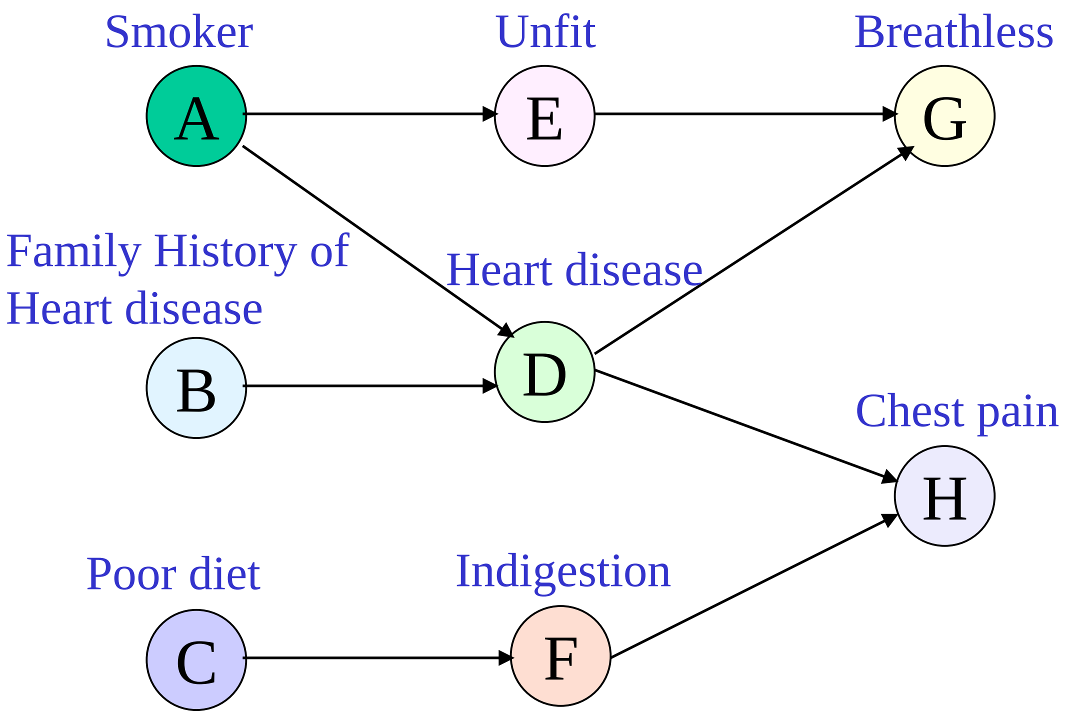
$$\mu_{f_s \to x}(x) = \sum_{x_1} \ldots \sum_{x_M} f_s(x, x_1, .., x_M) \prod_{m \in ne(f_s) \backslash x} \mu_{x_m \to f_s}(x_m)$$

$$= \sum_{x_1} \ldots \sum_{x_M} f_s(x, x_1, .., x_M) \prod_{m \in ne(f_s) \backslash x} \left(\prod_{l \in ne(x_m) \backslash f_s} \mu_{f_l \to x_m}(x_m) \right)$$

- MAP problem is called the Max-sum algorithm (Viterbi for Trees)
- For DAGs and MRFs, multiple paths may exist. If you use sum-product the usual way, it is called *Loopy Belief Propagation* and it works OK!

General DAGs

- Constructing the Inference Engine:
  - Consider an artificial medical diagnosis problem



$$P(ABCDEFGH) = P(A)P(B)P(C)P(E \mid A)P(D \mid A,B)P(F \mid C)$$
$$P(G \mid E,D)P(H \mid D,F)$$

Inference Problem in DAGs

- Typically, we are interested in computing the marginal distributions conditioned on some observation of one or more variables
- Example: what is the probability of Heart disease Given that the patient is a smoker, is Breathless and has chest pain?

$$P(D=T \mid A=G=H=T) \qquad T \Rightarrow TRUE$$

How to compute inference probabilities efficiently?

Key Steps in Inference for DAGs

- Key steps in exact Bayesian inference

1. Add undirected edges to all co-parents which are not currently joined (a process called *marrying parents*)

2. Drop all directions in the graph obtained from stage 1. The result is the so-called *moral graph*.

3. Triangulate the *moral graph*, that is, add sufficient additional undirected links between nodes such that *there are no cycles (i.e., closed paths) of length 4 or more distinct nodes without a short-cut.*

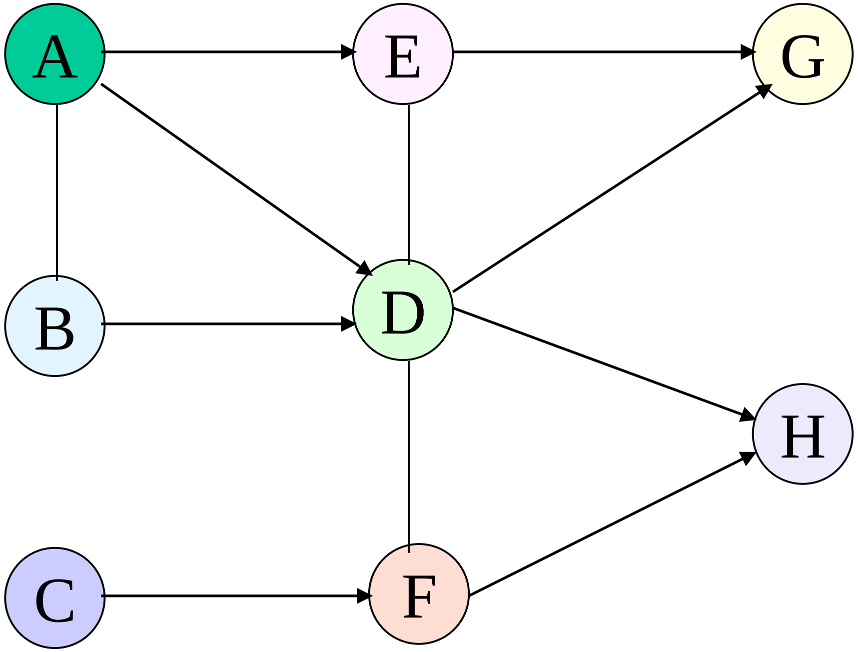
4. Identify the *cliques* of this triangulated graph

5. Join the cliques together to form the *junction tree*

6. Perform inference on the junction tree (*message passing*)

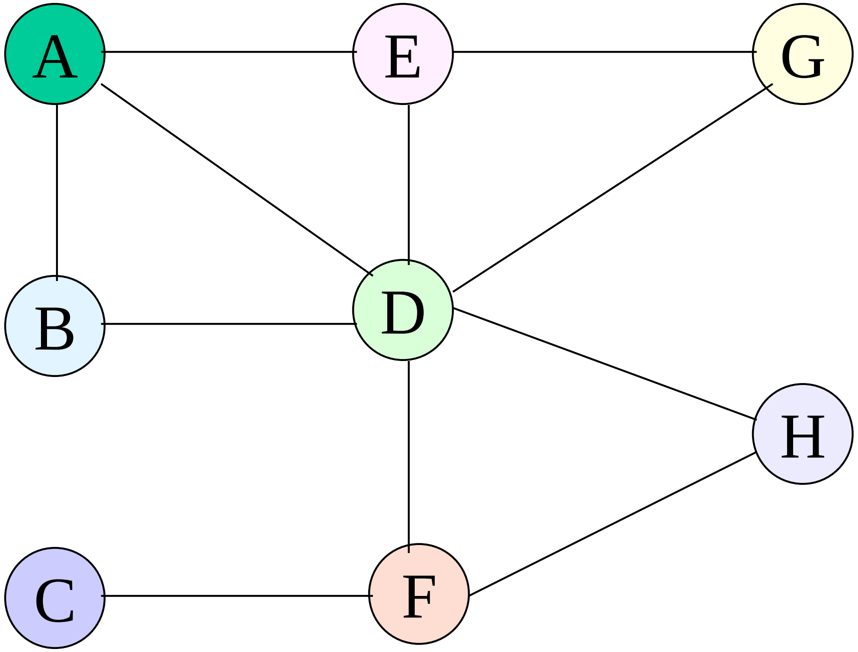
Marrying Parents

- Step 1: Marrying parents



Obtaining the Moral Graph

- Step 2: Moral graph

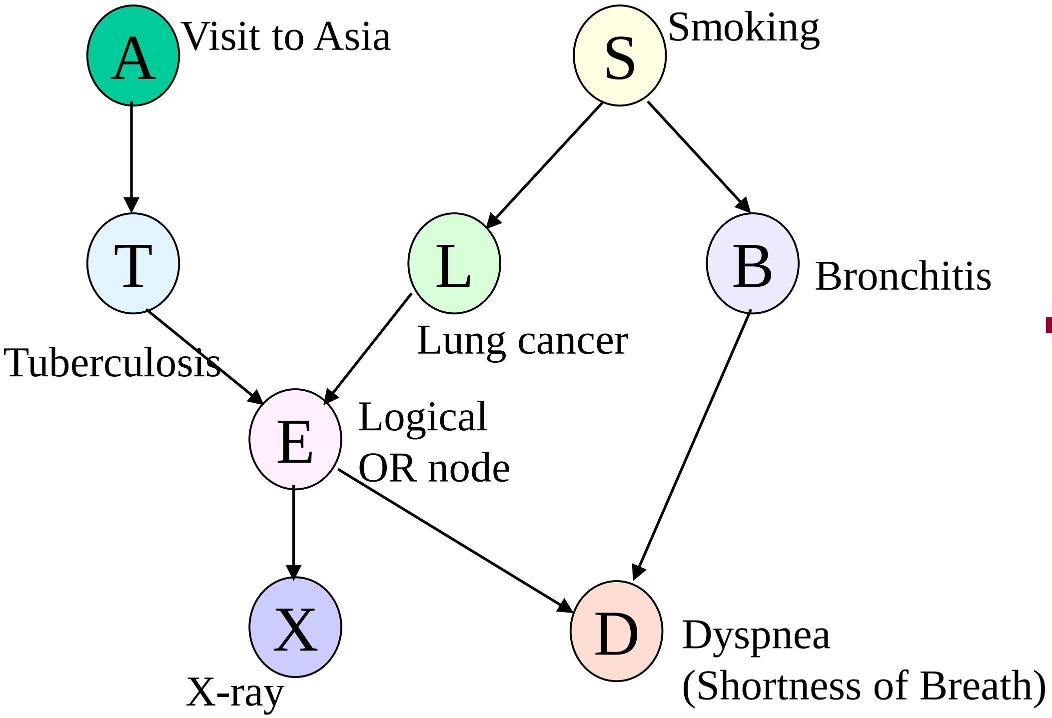


Triangulating the Moral Graph

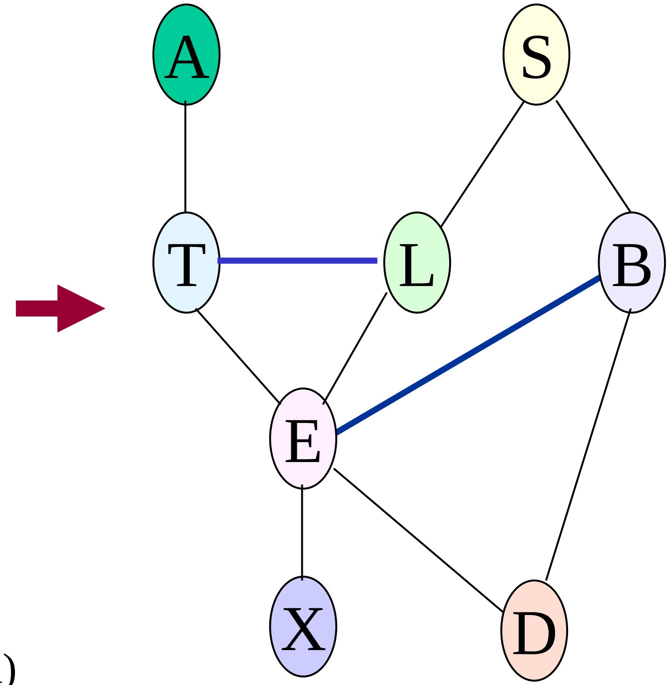
- Step 3: Triangulate the moral graph

  It is already triangulated.

  Example where triangulation is needed:



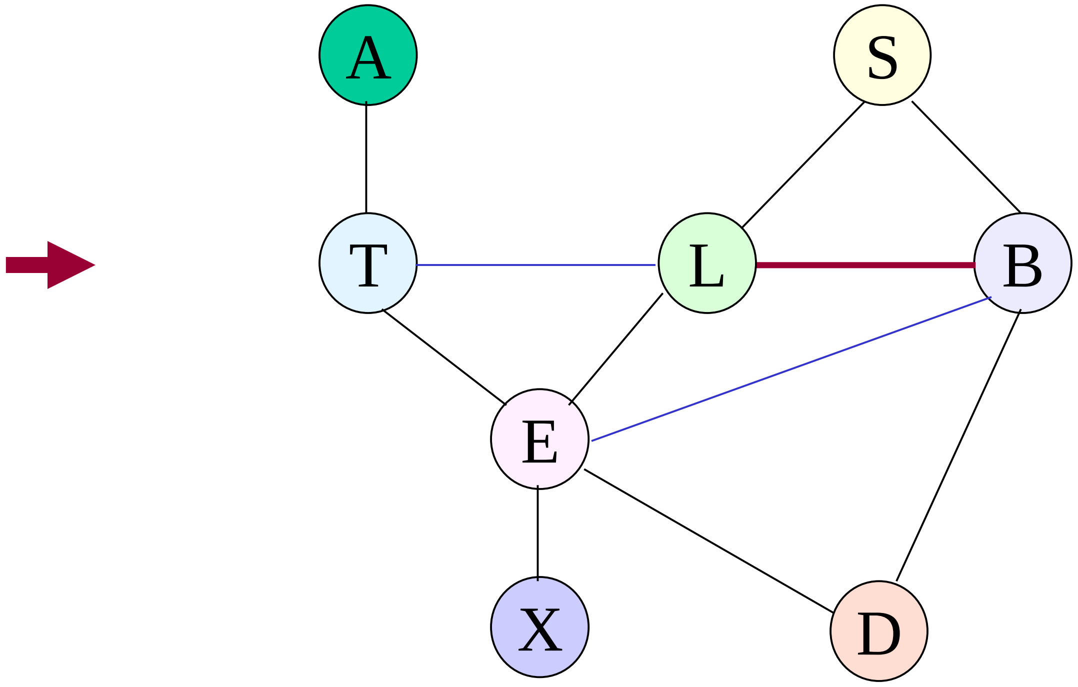
Original graph for Asia Problem



Moral graph

Triangulated Graph for the Asia Problem

$$P(ASTLBEXD) = P(A)P(S)P(T \mid A)P(L \mid S)P(B \mid S)$$
$$P(E \mid LT)P(X \mid E)P(D \mid B, E)$$



Triangulated graph

Cliques of the Triangulated Graph

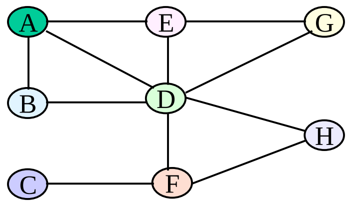
- Step 4: Cliques of the triangulated graph

  Clique: a fully connected (complete) maximal subgraph

Medical Diagnosis Problem:

C_1: ABD C_2: ADE C_3: DEG

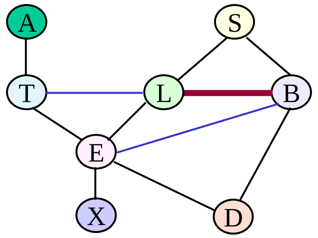
C_4: DFH C_5: CF



Asia Problem:

C_1: AT C_2: TLE C_3: BLE

C_4: SBL C_5: DBE C_6: XE

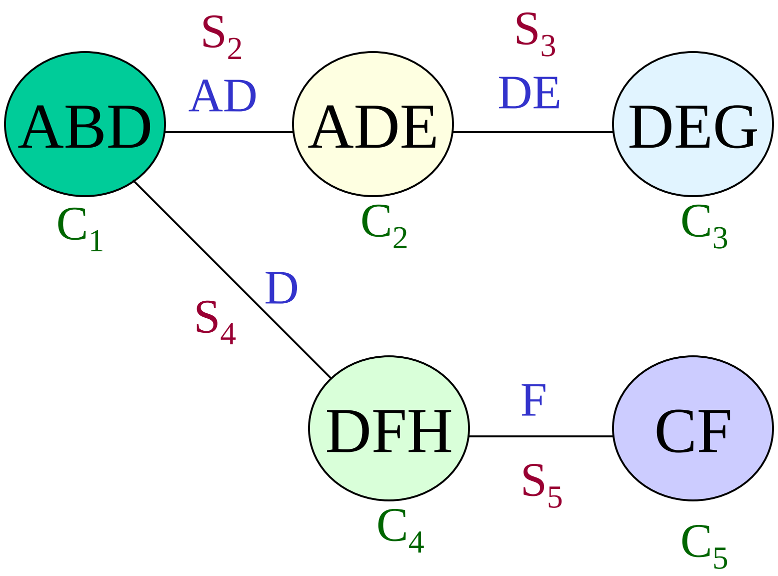


Constructing the Junction Tree

- <u>Step 5</u>: Make the junction tree

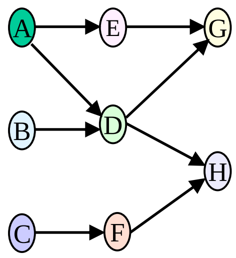
Key property: **running intersection property** ⇒ If a variable *x* is contained in two cliques, then it is contained in every clique on the path connecting the two cliques.

The edge joining two cliques is called a **<u>separator</u>**.



Joint Distribution in terms of Cliques and Separators

KEY: $P(ABCDEFGH) = \dfrac{\prod_{i=1}^{5} P(C_i)}{\prod_{i=2}^{5} P(S_i)}$ "Marginal representation"



$$= \frac{P(ABD)P(ADE)P(DEG)P(DFH)P(CF)}{P(AD)P(DE)P(D)P(F)}$$

Recall that

$$P(ABCDEFGH) = P(A)P(B)P(C)P(D \mid AB)P(E \mid A)P(F \mid C)$$
$$P(G \mid DE)P(H \mid DF)$$

Note that

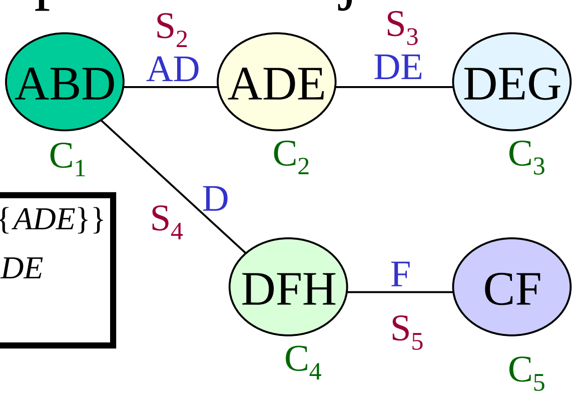
$$P(C_1) = P(ABD) = P(D \mid AB)P(A)P(B)$$
$$P(C_2) = P(ADE) = P(E \mid AD)P(AD) = P(E \mid A) \cdot P(S_2)$$
$$P(C_3) = P(G \mid DE) \cdot P(DE) = P(G \mid DE).P(S_3)$$

Joint Distribution in terms of Clique Potentials

$$P(C_4) = P(H \mid FD) \cdot P(S_4) \cdot P(S_5)$$

$$P(C_5) = P(C \mid F) \cdot P(F) = P(F \mid C) \cdot P(C)$$

So, marginal representation does indeed provide the joint distribution. In fact,



Separator $S_i = C_i \cap \{C_1 \cup C_2 \cup .... \cup C_{i-1}\}$

Let $R_i = C_i \setminus S_i \Rightarrow C_i - S_i$

$$S_3 = \{DEG\} \cap \{\{ABD\} \cup \{ADE\}\}$$
$$= \{DEG\} \cap \{ABDE\} = DE$$
$$R_3 = G$$

$$P(ABCDEFGH) = P(C_1)\prod_{i=2}^{5} P(C_i \mid S_i)$$

$$= P(C_1)\prod_{i=2}^{5} P(R_i \mid S_i) = \prod_{i=1}^{5} P(R_i \mid S_i); S_1 = \phi$$

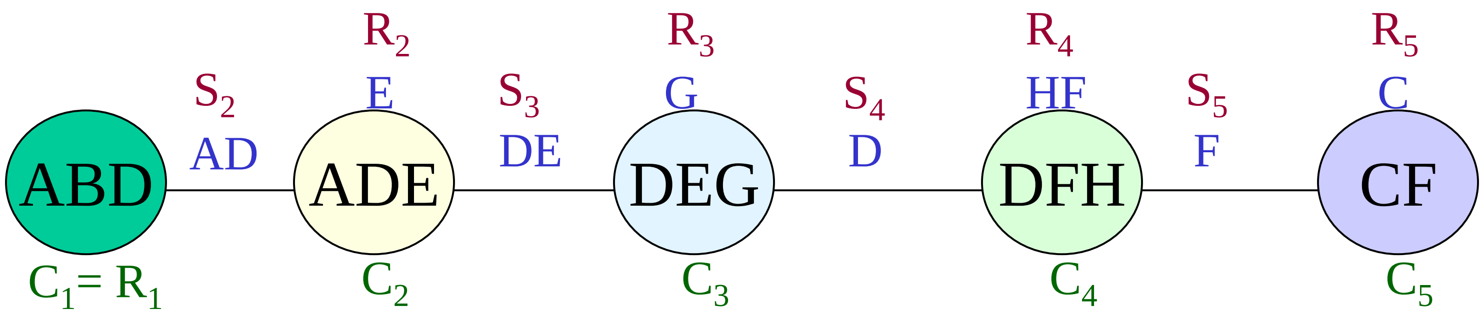
$$= P(ABD)P(E \mid AD)P(G \mid DE)P(HF \mid D)P(C \mid F)$$

$$= \prod_{i=1}^{5} \psi(C_i)$$

This is called a *potential representation* of joint distribution

Non-uniqueness of Junction Trees

Junction tree is not unique:



$$P(ABCDEFGH) = P(ABD)P(E|AD)P(G|DE)P(HF|D)P(C|F)$$

$$= \prod_{i=1}^{5} \psi(C_i)$$

$$= P(A)P(B)P(D|AB)\frac{P(C)P(F|C)}{P(F)}P(E|A)P(G|DE)P(H|DF)P(F)$$

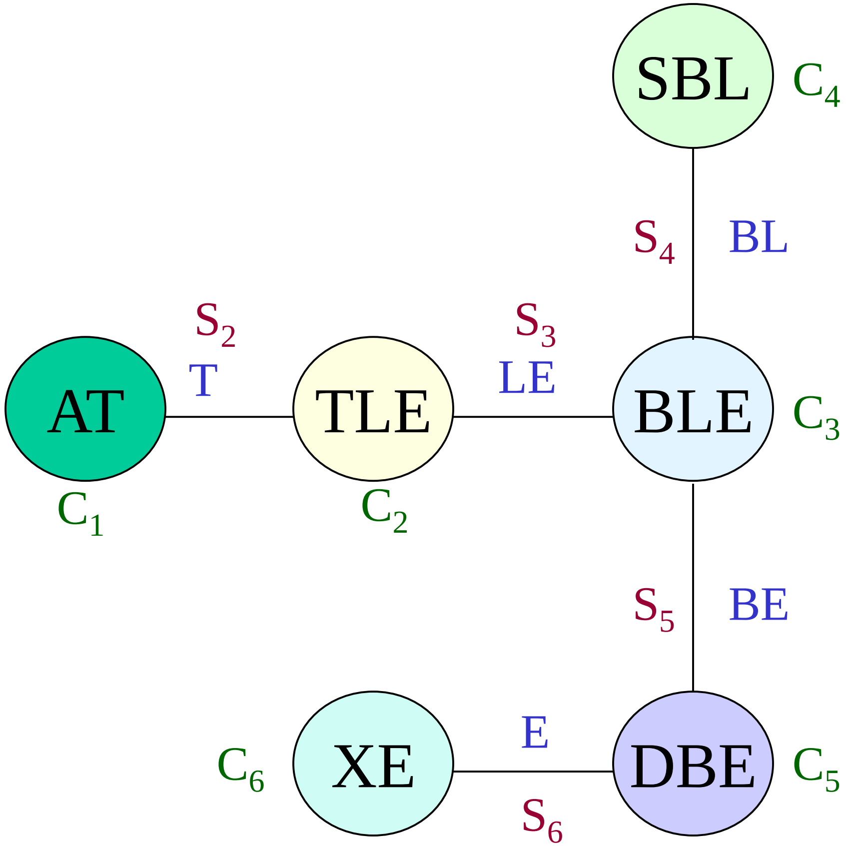
Note that ψ can be any function of cliques with a suitable normalization at the end

$$Example: \overline{\psi}(C_i) = \psi(C_i)\sum_{R_i}\psi(C_i) = P(R_i|S_i)P(S_i) = P(R_i,S_i)$$

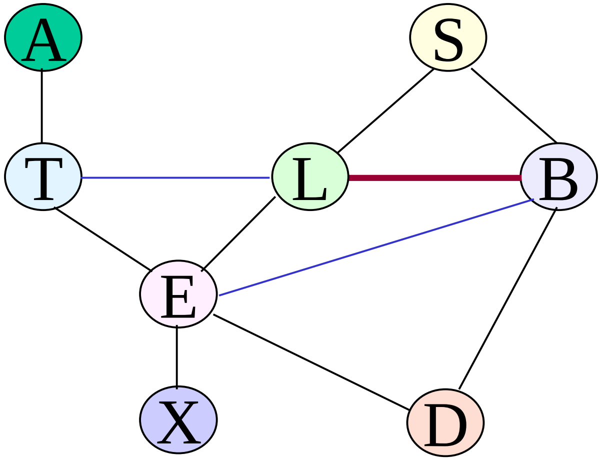
$$P(R_i|S_i) = \frac{P(R_i,S_i)}{P(S_i)} = \frac{\overline{\psi}(C_i)}{\sum_{R_i}\overline{\psi}(C_i)} \Rightarrow \text{can work with } \overline{\psi}(C_i) \text{ directly}$$

Junction Tree for the Asia Problem

One junction tree for the Asia Problem:



Junction graph



Triangulated graph

Cliques, Separators & Potentials

$$P(ASTLBEXD) = P(A)P(S)P(T \mid A)P(L \mid S)P(B \mid S)$$
$$P(E \mid LT)P(X \mid E)P(D \mid BE)$$

Chain Rule

Cliques & Separators

$$= \frac{P(AT)P(TLE)P(BLE)P(SBL)P(DBE)P(XE)}{P(T)P(LE)P(BL)P(BE)P(E)}$$

$$= P(AT)P(LE \mid T)P(B \mid LE)P(S \mid BL)$$
$$P(D \mid BE)P(X \mid E)$$

Clique Potentials/ Factors

$$= \prod_{i=1}^{6} \psi(C_i)$$

Although it looks strange, it does work!!!

- Step 6: Inference on the junction tree: sum-product algorithm on cliques

for each node in the junction tree, store

clique, $R_i(= C_i \setminus S_i)$, S_i *and* $\psi(clique)$

Other Methods for Inference

- Bayesian Inference
  - The junction tree approach becomes intractable for dense graphs
  - Alternate Approaches
    - Probabilistic logic sampling on
      - » DAGs
      - » Junction tree
    - **Gibbs sampling**
    - Botzmann Machines
      - » Gibbs sampling
      - » Mean Field Approximation
    - Lagrangian Relaxation (Variational approximation)
    - Expectation Propagation

- Learning BN Parameters and Structure from Data

What is a Gibbs Sampler?

- It is a Markov Chain Monte Carlo method (recall particle filter)
  - Updates one variable at a time
  - Samples from a conditional distribution of a variable when other variables are fixed
  - Ideally suited for Bayesian networks
- Suppose you want to sample from a distribution of p variables $p(x_1,x_2,..,x_p)$
  - Initialize $\{x_i^0\}_{i=1}^p$
  - For $t = 1, 2, ..., T$

$$\text{Sample } x_1^{(t+1)} \sim p(x_1 \mid x_2^t, x_3^t, ..., x_p^t)$$

$$\text{Sample } x_2^{(t+1)} \sim p(x_2 \mid x_1^{(t+1)}, x_3^t, ..., x_p^t)$$

....

$$\text{Sample } x_i^{(t+1)} \sim p(x_i \mid x_1^{(t+1)}, ..., x_{i-1}^{(t+1)}, x_{i+1}^{(t)}, ..., x_p^t)$$

....

$$\text{Sample } x_p^{(t+1)} \sim p(x_p \mid x_1^{(t+1)}, x_2^{(t+1)}, ..., x_{p-1}^{(t+1)})$$

- Need a burn-in period
- Subsample to minimize correlations

Summary

- Graphical Models
- Bayesian Inference in Graphical Models
- Forward-Backwards Methods of Inference
- Simulation-based Methods