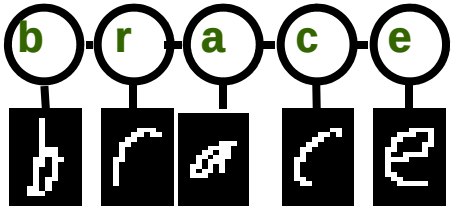
Machine Learning

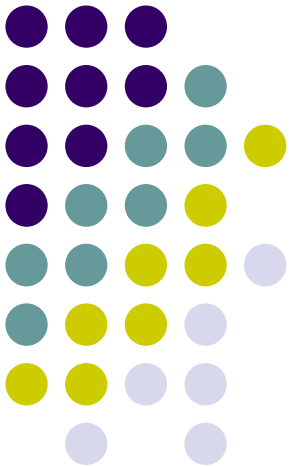
10-701/15-781, Fall 2012

Inference and Learning in GM

Eric Xing

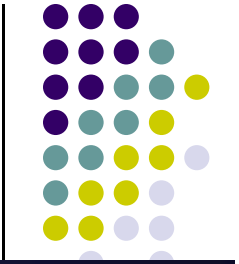
Lecture 14, October 31, 2012





Reading: Chap. 8, C.B book

Recap of BN Representation



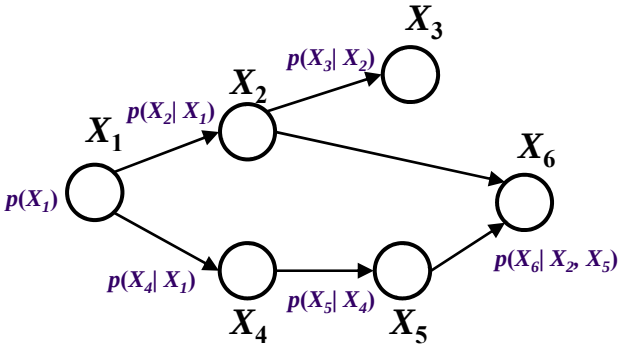
- Joint probability dist. on multiple variables:

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

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

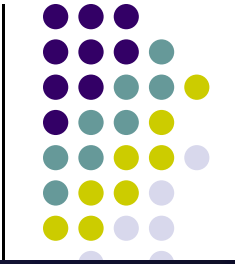
$$P(X_1, X_2, X_3, X_4, X_5, X_6)$$
$$= P(X_1)P(X_2)P(X_3)P(X_4)P(X_5)P(X_6) = \prod_i P(X_i)$$

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



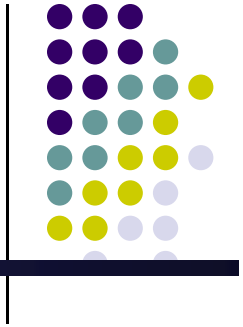
$$P(X_1, X_2, X_3, X_4, X_5, X_6)$$
$$= P(X_1)\, P(X_2|\, X_1)\, P(X_3|\, X_2)\, P(X_4|\, X_1)\, P(X_5|\, X_4)\, P(X_6|\, X_2, X_5)$$

Inference and Learning



- We now have compact representations of probability distributions: **BN**
- A BN *M* describes a unique probability distribution *P*
- Typical tasks:
  - Task 1: How do we answer **queries** about P?
    - We use **inference** as a name for the process of computing answers to such queries
  - Task 2: How do we estimate a **plausible model** *M* from data *D*?
    i. We use **learning** as a name for the process of obtaining point estimate of *M*.
    ii. But for *Bayesian*, they seek $p(M \mid D)$, 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*.

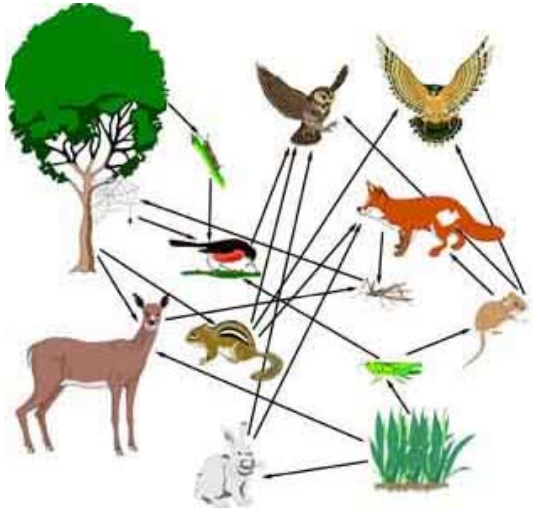
Approaches to inference

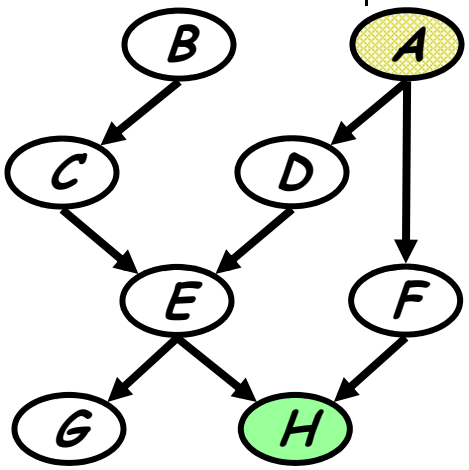


- Exact inference algorithms
  - The elimination algorithm
  - Belief propagation
  - The junction tree algorithms (but will not cover in detail here)
- Approximate inference techniques
  - Variational algorithms
  - Stochastic simulation / sampling methods
  - Markov chain Monte Carlo methods

Marginalization and Elimination

- A food web:





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

Query: $P(h)$

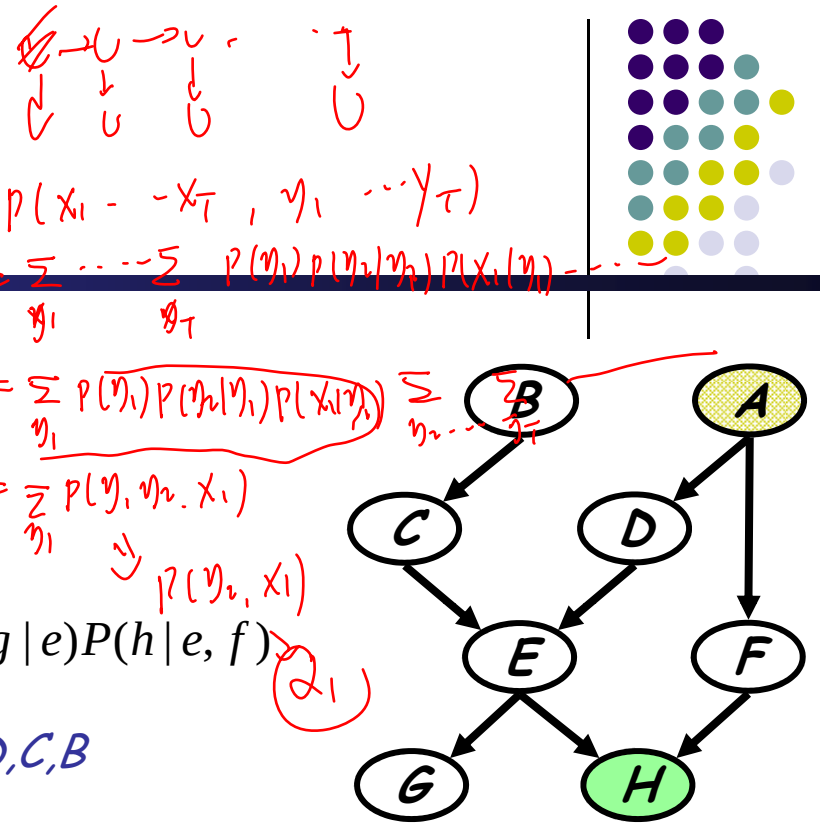
$$P(h) = \sum_g \sum_f \sum_e \sum_d \sum_c \sum_b \sum_a P(a,b,c,d,e,f,g,h)$$

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

- By chain decomposition, we get

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

Variable Elimination



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

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

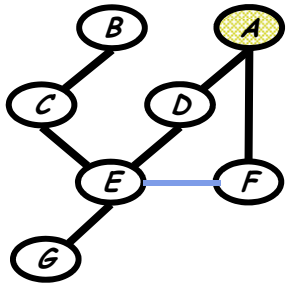
- Choose an elimination order: H,G,F,E,D,C,B

- Step 1:
  - **Conditioning** (fix the evidence node (i.e., h) on its observed value (i.e., $\tilde{h}$)):

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

  - This step is isomorphic to a marginalization step:

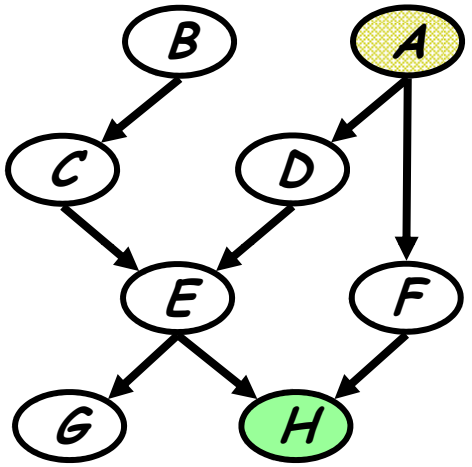
$$m_h(e,f) = \sum_h p(h \mid e,f)\delta(h = \tilde{h})$$



Example: Variable Elimination

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

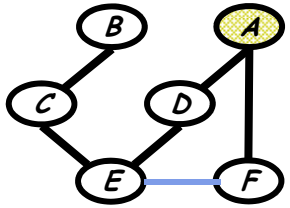
$$P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)P(h \mid e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)m_h(e,f)$$



- Step 2: Eliminate G
  - compute

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

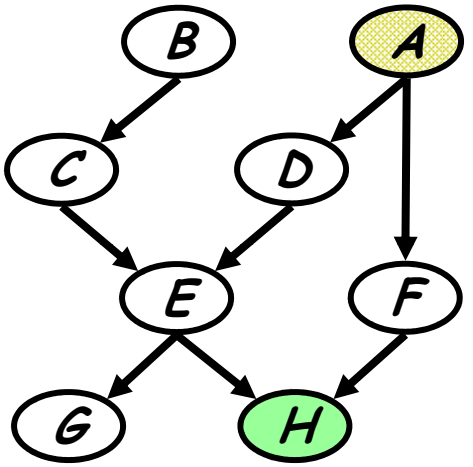
$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)m_g(e)m_h(e,f)$$
$$= P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)m_h(e,f)$$



Example: Variable Elimination

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

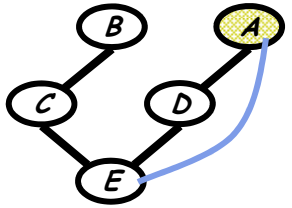
$$P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)P(h \mid e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)m_h(e,f)$$



- Step 3: Eliminate F
  - compute

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

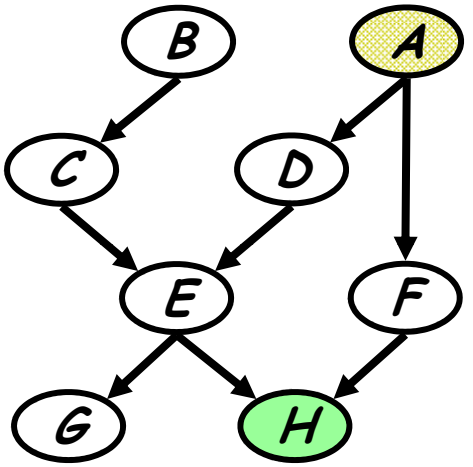
$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)m_f(a,e)$$



Example: Variable Elimination

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

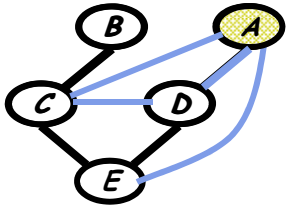
$$P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)P(h \mid e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid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



- Step 4: Eliminate E
  - compute

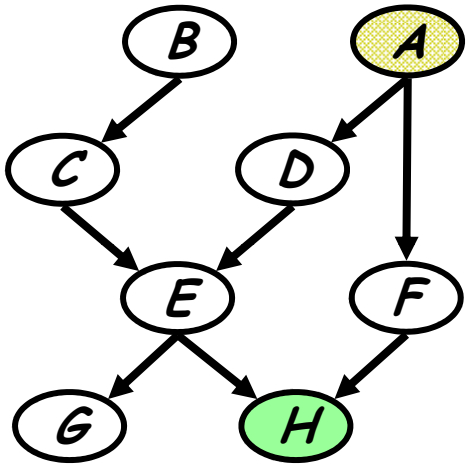
$$m_e(a,c,d) = \sum_e p(e \mid c,d)m_f(a,e)$$

$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)m_e(a,c,d)$$



Example: Variable Elimination

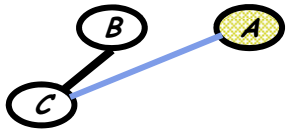
- Query: $P(B \mid h)$
  - Need to eliminate: B, C, D



- Initial factors:

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

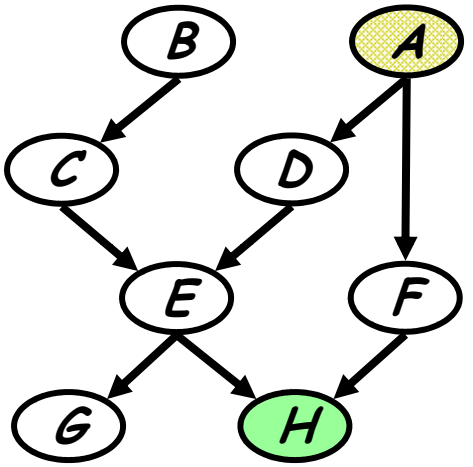
- Step 5: Eliminate D
  - compute $m_d(a,c) = \sum_d p(d \mid a)m_e(a,c,d)$



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

Example: Variable Elimination

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



- Initial factors:

$$P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)P(h \mid e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c,d)P(f \mid a)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c,d)m_f(a,e)$$
$$\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)m_e(a,c,d)$$
$$\Rightarrow P(a)P(b)P(c \mid d)m_d(a,c)$$

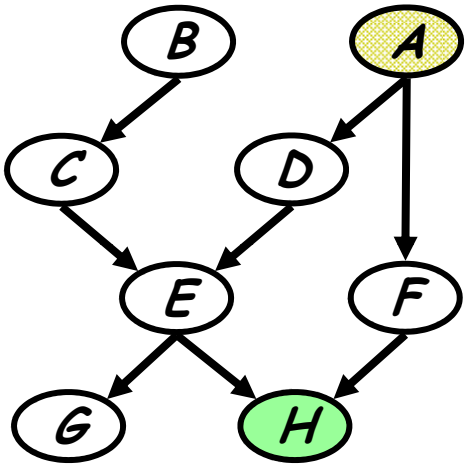
- Step 6: Eliminate C
  - compute $m_c(a,b) = \sum_c p(c \mid b)m_d(a,c)$

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

Example: Variable Elimination

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

$$P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)P(h \mid e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c,d)P(f \mid a)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c,d)m_f(a,e)$$
$$\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)m_e(a,c,d)$$
$$\Rightarrow P(a)P(b)P(c \mid d)m_d(a,c)$$
$$\Rightarrow P(a)P(b)m_c(a,b)$$



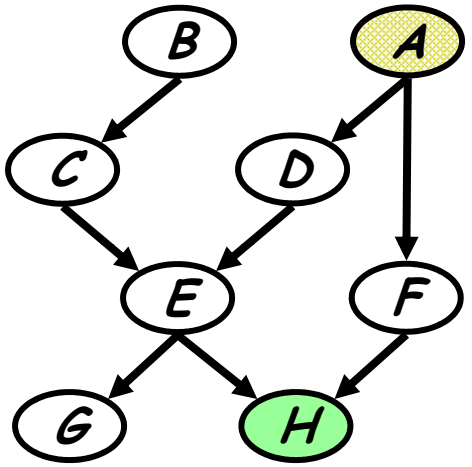
- Step 7: Eliminate B
  - compute

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

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

Example: Variable Elimination

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



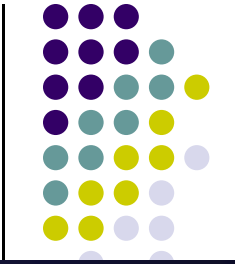
- Initial factors:

$$P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)P(h \mid e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c,d)P(f \mid a)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c,d)m_f(a,e)$$
$$\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)m_e(a,c,d)$$
$$\Rightarrow P(a)P(b)P(c \mid d)m_d(a,c)$$
$$\Rightarrow P(a)P(b)m_c(a,b)$$
$$\Rightarrow P(a)m_b(a)$$

- Step 8: Wrap-up

$$p(a,\tilde{h}) = p(a)m_b(a), \quad p(\tilde{h}) = \sum_a p(a)m_b(a)$$
$$\Rightarrow P(a \mid \tilde{h}) = \frac{p(a)m_b(a)}{\sum p(a)m_b(a)}$$

Complexity of variable elimination



- Suppose in one elimination step we compute

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

$$m'_x(x, y_1,\ldots,y_k) = \prod_{i=1}^{k} m_i(x, \mathbf{y}_{c_i})$$

This requires

- $k \bullet \left|\mathrm{Val}(X)\right| \bullet \prod_i \left|\mathrm{Val}(\mathbf{y}_{C_i})\right|$ multiplications
  - For each value of $x, y_1, \ldots, y_k$, we do k multiplications
- $\left|\mathrm{Val}(X)\right| \bullet \prod_i \left|\mathrm{Val}(\mathbf{y}_{C_i})\right|$ additions
  - For each value of $y_1, \ldots, y_k$, we do $|Val(X)|$ additions

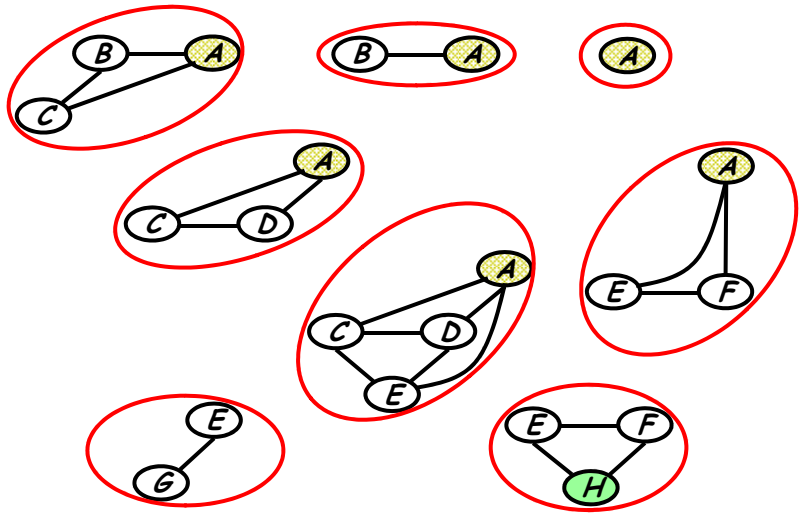
Complexity is **exponential** in number of variables in the intermediate factor

Elimination Clique

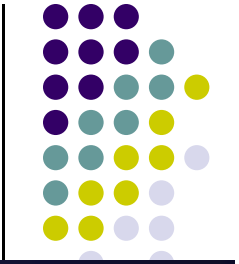
- Induced dependency during marginalization is captured in elimination cliques
  - Summation <-> elimination
  - Intermediate term <-> elimination clique

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

  - Can this lead to an generic inference algorithm?

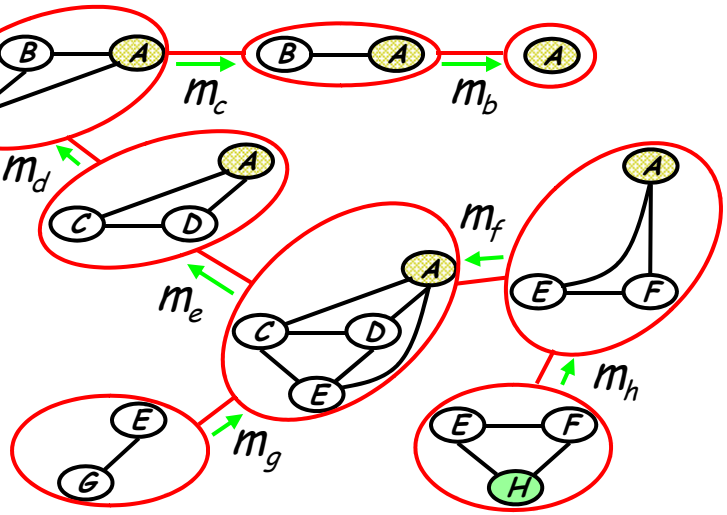


From Elimination to Message Passing



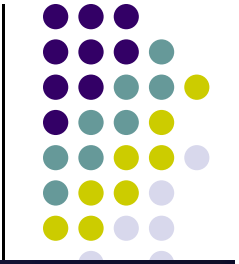
- Elimination ≡ message passing on a **clique tree**

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

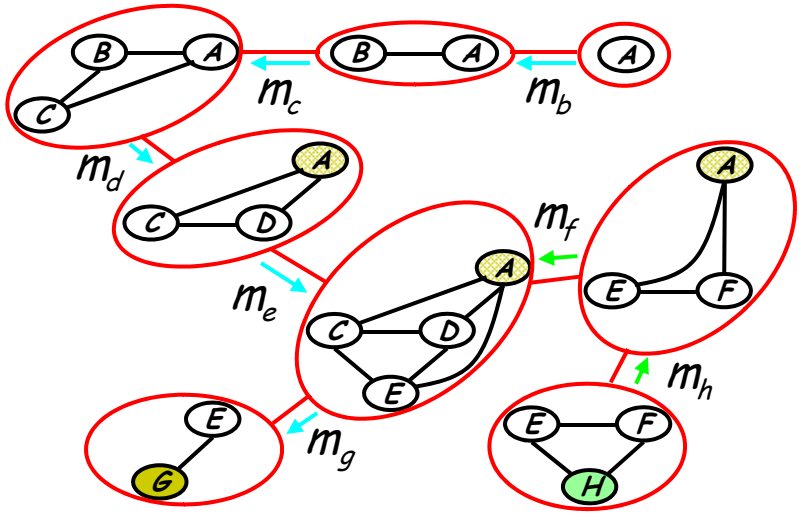


- Messages can be reused

From Elimination to Message Passing

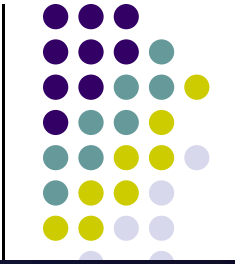


- Elimination ≡ message passing on a **clique tree**
  - **Another query ...**



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

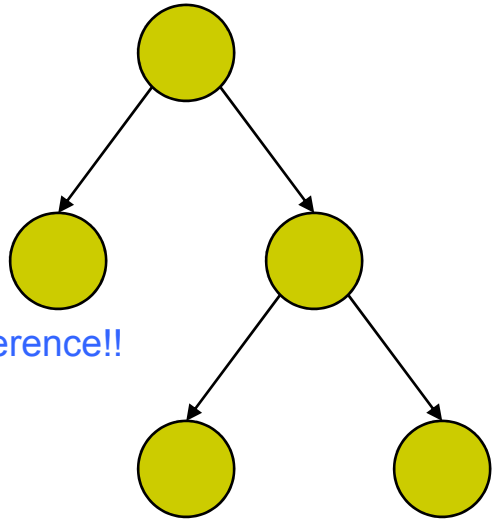
From elimination to message passing



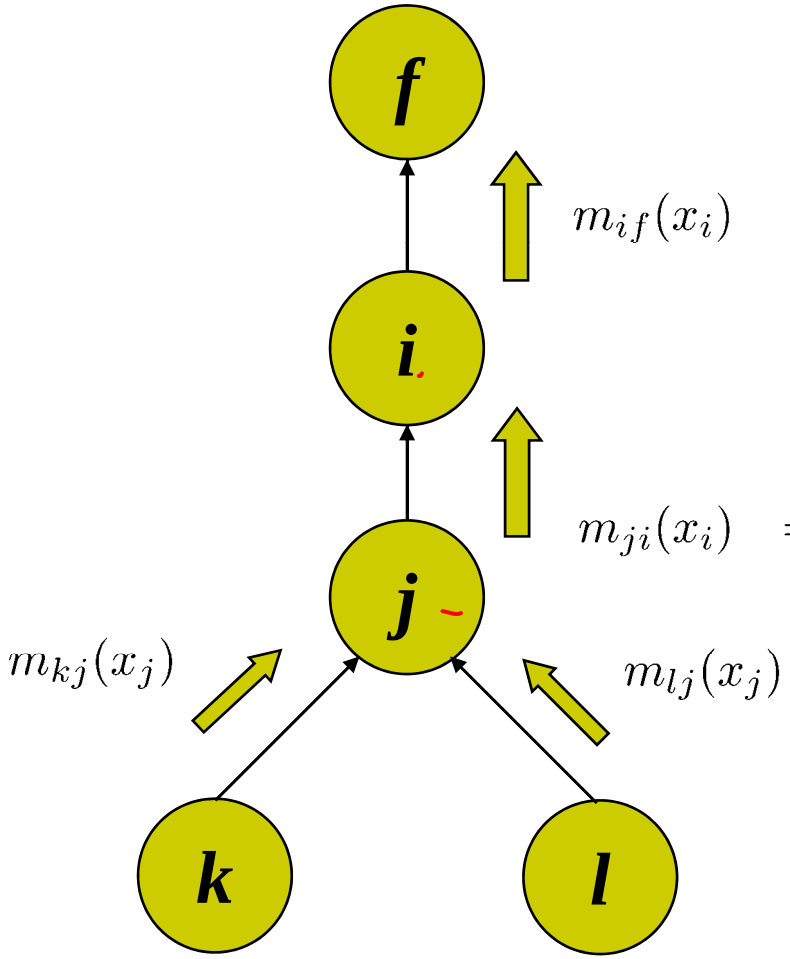
- Recall ELIMINATION algorithm:
  - Choose an ordering $\mathcal{Z}$ in which query node f is the final node
  - Place all potentials on an active list
  - Eliminate node i by removing all potentials containing i, take sum/product over x_i.
  - Place the resultant factor back on the list

- 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!!



Message passing for trees



Let $m_{ij}(x_i)$ denote the factor resulting from eliminating variables from bellow up to i, which is a function of x_i:

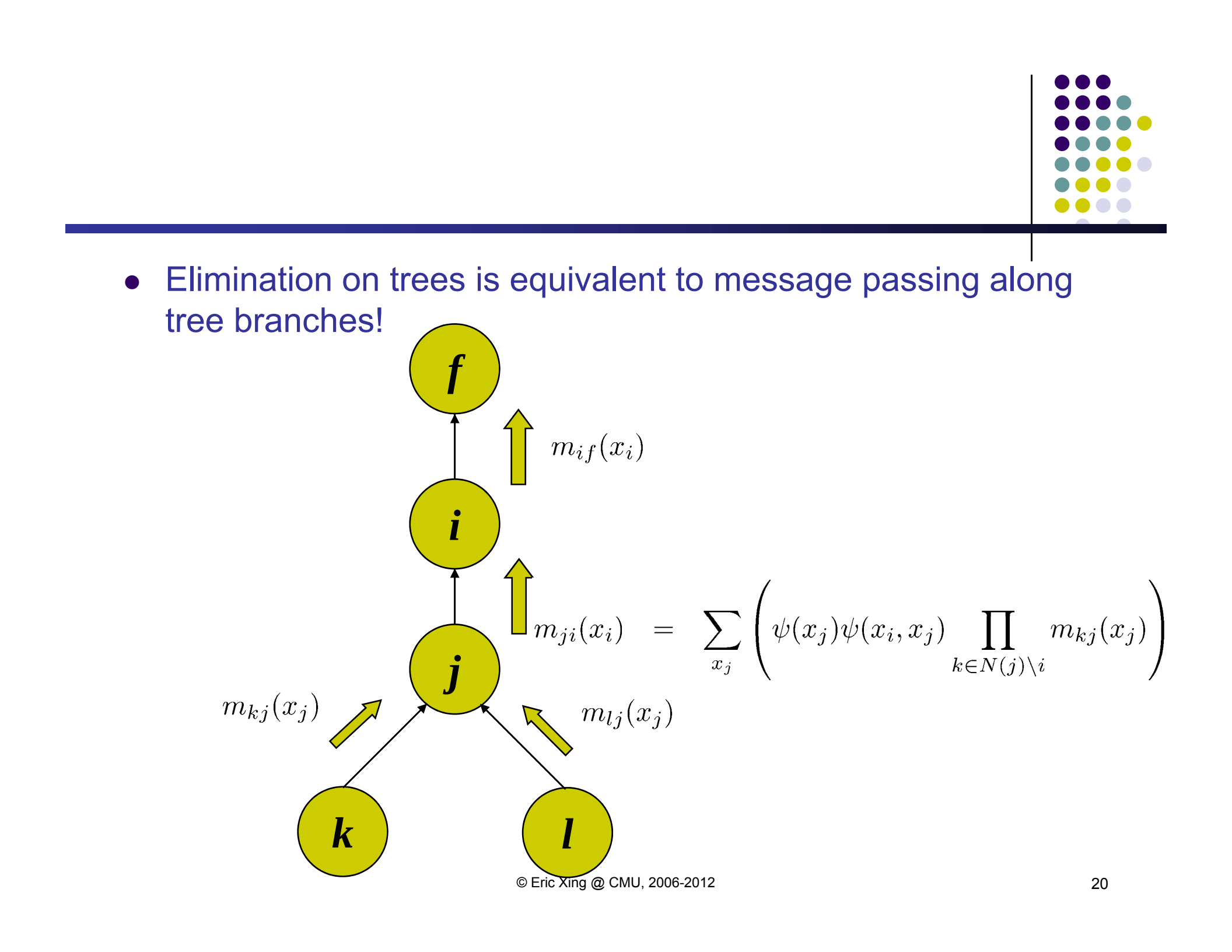
$$m_{ji}(x_i) = \sum_{x_j} \left(\psi(x_j)\psi(x_i, x_j) \prod_{k \in N(j) \setminus i} m_{kj}(x_j) \right)$$

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

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

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

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

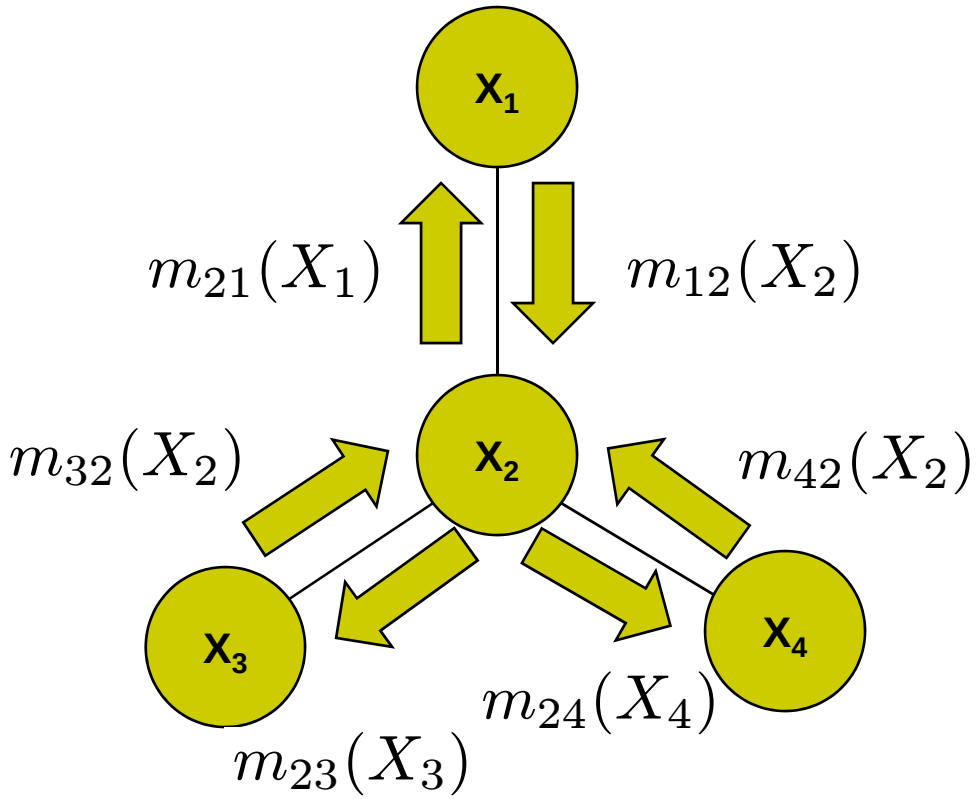


- Elimination on trees is equivalent to message passing along tree branches!

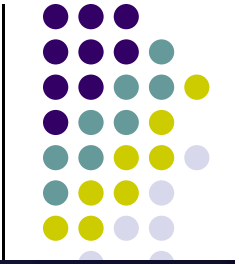
$$m_{ji}(x_i) = \sum_{x_j} \left(\psi(x_j)\psi(x_i, x_j) \prod_{k \in N(j) \setminus i} m_{kj}(x_j) \right)$$

The message passing protocol:

- A two-pass algorithm:



Belief Propagation (SP-algorithm): Sequential implementation



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

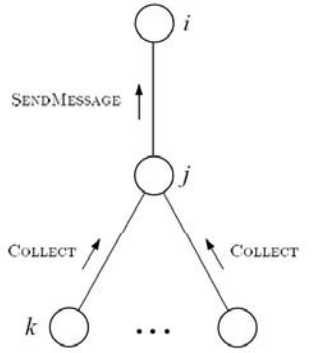
EVIDENCE(E)
  for $i \in E$
    $\psi^E(x_i) = \psi(x_i)\delta(x_i, \bar{x}_i)$
  for $i \notin E$
    $\psi^E(x_i) = \psi(x_i)$

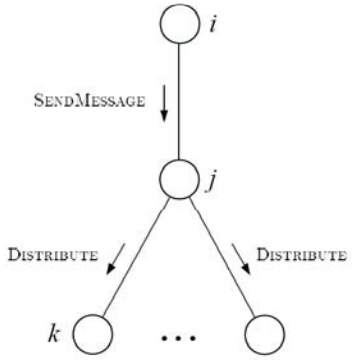
COLLECT(i, j)
  for $k \in \mathcal{N}(j) \backslash i$
    COLLECT(j, k)
  SENDMESSAGE(j, i)

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

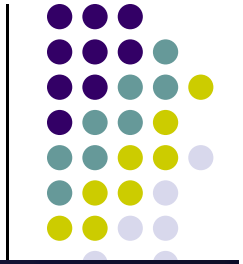
SENDMESSAGE(j, i)
  $m_{ji}(x_i) = \sum_{x_j} (\psi^E(x_j)\psi(x_i, x_j) \prod_{k \in \mathcal{N}(j) \backslash i} m_{kj}(x_j))$

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





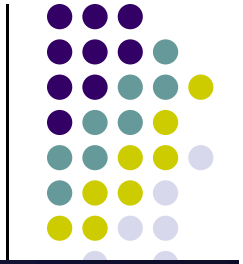
Inference on general GM



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

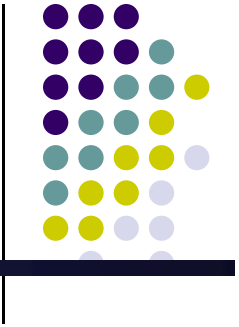
→ Junction tree algorithm

A Sketch of the Junction Tree Algorithm

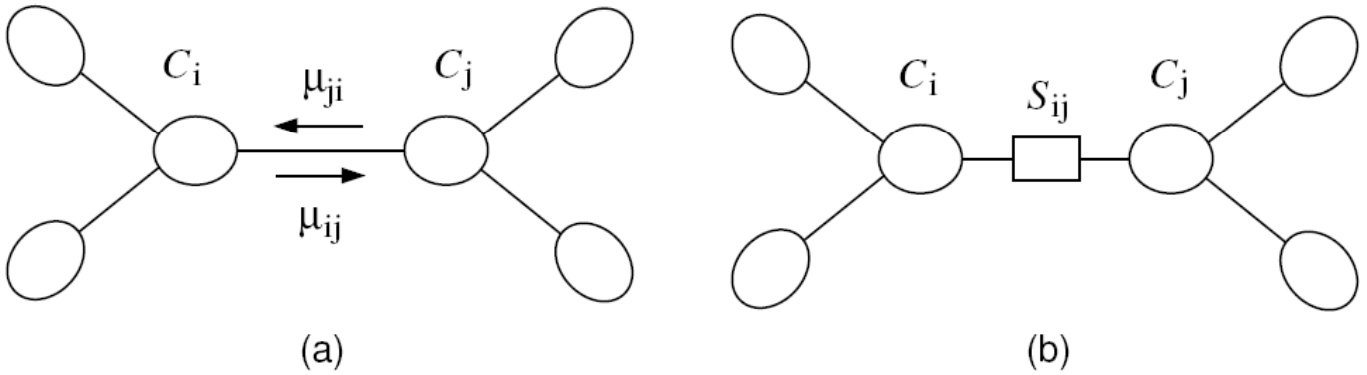


- **The algorithm**
  - Construction of junction trees --- a special **clique tree**
  - Propagation of probabilities --- a message-passing protocol
- Results in marginal probabilities of all cliques --- solves all queries in a single run
- A **generic** exact inference algorithm for any GM
- **Complexity**: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

The Shafer Shenoy Algorithm



- Shafer-Shenoy algorithm



- Message from clique i to clique j :

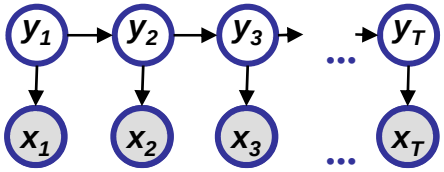
$$\mu_{i\to j} = \sum_{C_i \setminus S_{ij}} \psi_{C_i} \prod_{k\neq j} \mu_{k\to i}(S_{ki})$$

- Clique marginal

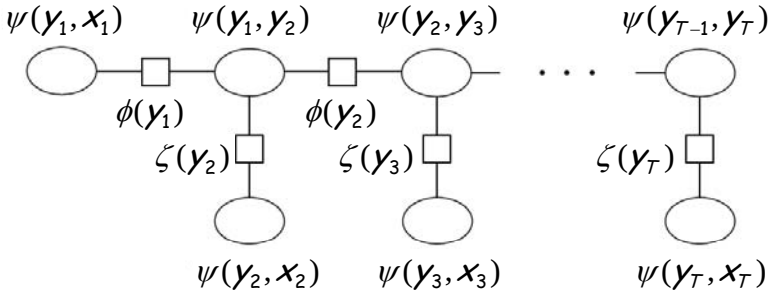
$$p(C_i) \propto \psi_{C_i} \prod_{k} \mu_{k\to i}(S_{ki})$$

The Junction tree algorithm for HMM

- A junction tree for the HMM

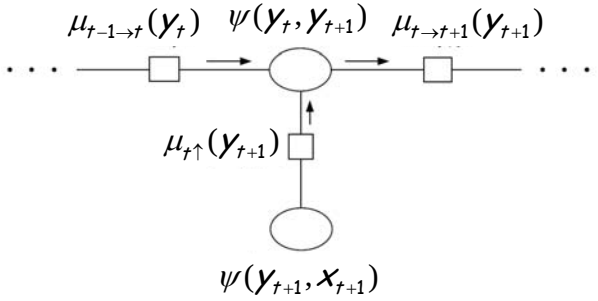


$\Rightarrow$



- Rightward pass

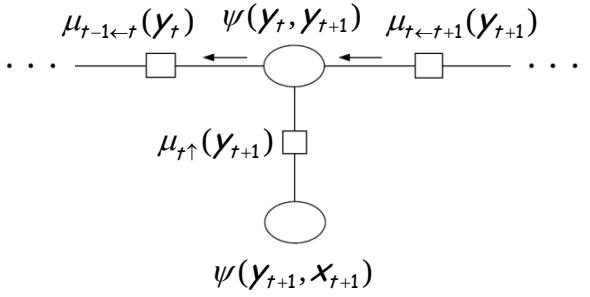
$$\mu_{t\to t+1}(y_{t+1}) = \sum_{y_t} \psi(y_t, y_{t+1})\mu_{t-1\to t}(y_t)\mu_{t\uparrow}(y_{t+1})$$
$$= \sum_{y_t} p(y_{t+1} \mid y_t)\mu_{t-1\to t}(y_t)p(x_{t+1} \mid y_{t+1})$$
$$= p(x_{t+1} \mid y_{t+1})\sum_{y_t} a_{y_t, y_{t+1}}\mu_{t-1\to t}(y_t)$$



  - This is exactly the *forward algorithm*!

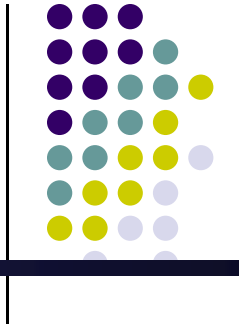
- Leftward pass …

$$\mu_{t-1\leftarrow t}(y_t) = \sum_{y_{t+1}} \psi(y_t, y_{t+1})\mu_{t\leftarrow t+1}(y_{t+1})\mu_{t\uparrow}(y_{t+1})$$
$$= \sum_{y_{t+1}} p(y_{t+1} \mid y_t)\mu_{t\leftarrow t+1}(y_{t+1})p(x_{t+1} \mid y_{t+1})$$



  - This is exactly the *backward algorithm*!

Summary: Exact Inference



- The simple Eliminate algorithm captures the key algorithmic Operation underlying probabilistic inference:

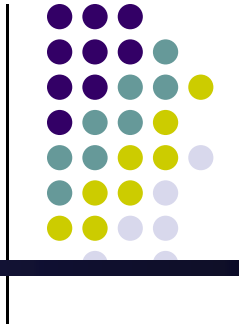
  --- That of taking a sum over product of potential functions

- 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

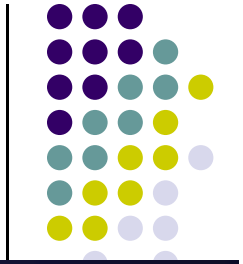
- What can we say about the overall computational complexity of the algorithm? In particular, how can we control the "size" of the summands that appear in the sequence of summation operation.

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

Monte Carlo methods



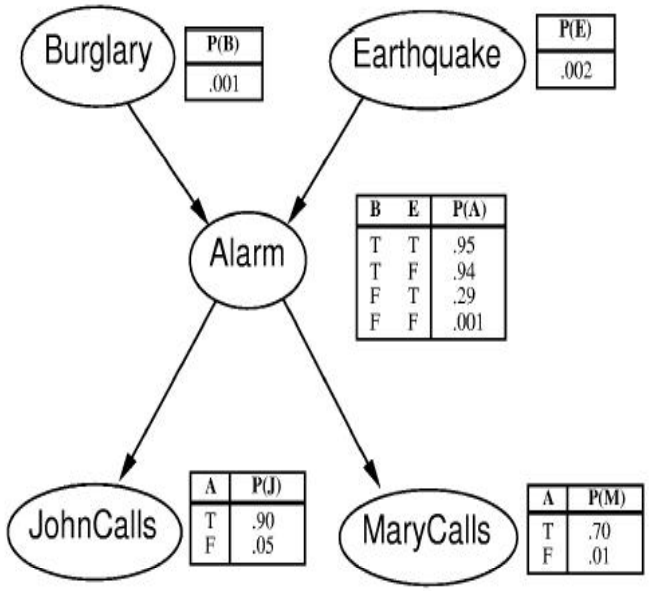
- Draw random samples from the desired distribution
- Yield a stochastic representation of a complex distribution
  - marginals and other expections can be approximated using sample-based averages

$$E[f(x)] = \frac{1}{N}\sum_{t=1}^{N} f(x^{(t)})$$

- **Asymptotically** exact and easy to apply to arbitrary models
- Challenges:
  - how to draw samples from a given dist. (not all distributions can be trivially sampled)?
  - how to make better use of the samples (not all sample are useful, or eqally useful, see an example later)?
  - how to know we've sampled enough?

Example: naive sampling

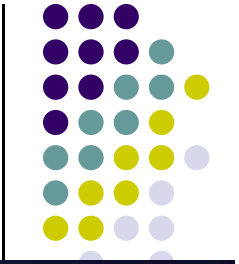
- Construct samples according to probabilities given in a BN.



E0	B0	A0	M0	J0
E0	B0	A0	M0	J0
E0	B0	A0	M0	J1
E0	B0	A0	M0	J0
E0	B0	A0	M0	J0
E0	B0	A0	M0	J0
E1	B0	A1	M1	J1
E0	B0	A0	M0	J0
E0	B0	A0	M0	J0
E0	B0	A0	M0	J0

Alarm example: (Choose the right sampling sequence)
1) Sampling:P(B)=<0.001, 0.999> suppose it is false, B0. Same for E0. P(A|B0, E0)=<0.001, 0.999> suppose it is false...
2) Frequency counting: In the samples right, P(J|A0)=P(J,A0)/P(A0)=<1/9, 8/9>.

Example: naive sampling



- Construct samples according to probabilities given in a BN.

Alarm example: (Choose the right sampling sequence)

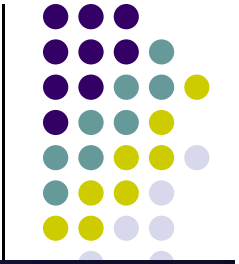
3) what if we want to compute P(J|A1) ?
we have only one sample ...
P(J|A1)=P(J,A1)/P(A1)=<0, 1>.

4) what if we want to compute P(J|B1) ?
No such sample available!
P(J|A1)=P(J,B1)/P(B1) can not be defined.

For a model with hundreds or more variables, rare events will be very hard to garner evough samples even after a long time or sampling ...

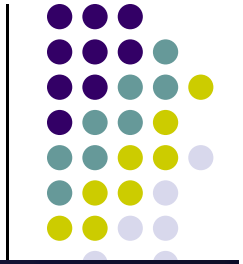
E0	B0	A0	M0	J0
E0	B0	A0	M0	J0
E0	B0	A0	M0	J1
E0	B0	A0	M0	J0
E0	B0	A0	M0	J0
E0	B0	A0	M0	J0
E1	B0	A1	M1	J1
E0	B0	A0	M0	J0
E0	B0	A0	M0	J0
E0	B0	A0	M0	J0

Markov chain Monte Carlo (MCMC)



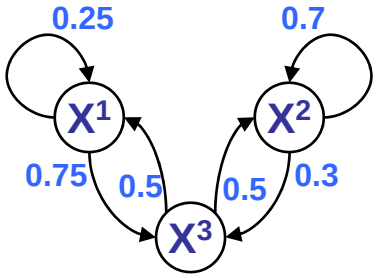
- Construct a Markov chain whose stationary distribution is the target density $= P(X|e)$.
- Run for T samples (burn-in time) until the chain converges/mixes/reaches stationary distribution.
- Then collect M (correlated) samples x_m .
- Key issues:
  - Designing proposals so that the chain mixes rapidly.
  - Diagnosing convergence.

Markov Chains

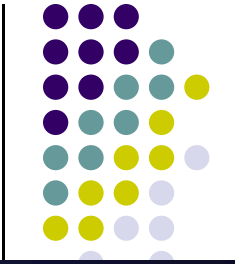


- **Definition:**
  - Given an n-dimensional state space
  - Random vector $\mathbf{X} = (x_1,\ldots,x_n)$
  - $\mathbf{x}^{(t)} = \mathbf{x}$ at time-step t
  - $\mathbf{x}^{(t)}$ transitions to $\mathbf{x}^{(t+1)}$ with prob
    $$P(\mathbf{x}^{(t+1)} \mid \mathbf{x}^{(t)},\ldots,\mathbf{x}^{(1)}) = T(\mathbf{x}^{(t+1)} \mid \mathbf{x}^{(t)}) = T(\mathbf{x}^{(t)} \rightarrow \mathbf{x}^{(t+1)})$$
- **Homogenous**: chain determined by state $\mathbf{x}^{(0)}$, fixed *transition kernel* Q (rows sum to 1)
- **Equilibrium**: $\pi(\mathbf{x})$ is a *stationary (equilibrium) distribution* if
  $$\pi(\mathbf{x}') = \Sigma_{\mathbf{x}} \pi(\mathbf{x})\, Q(\mathbf{x}\rightarrow\mathbf{x}').$$
  i.e., is a left eigenvector of the transition matrix $\pi^T = \pi^T Q$.

$$(0.2 \quad 0.5 \quad 0.3) = (0.2 \quad 0.5 \quad 0.3)\begin{pmatrix} 0.25 & 0 & 0.75 \\ 0 & 0.7 & 0.3 \\ 0.5 & 0.5 & 0 \end{pmatrix}$$



Gibbs sampling



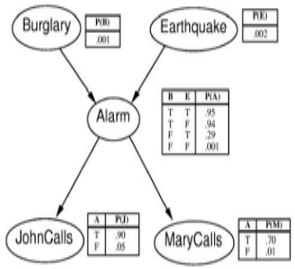
- The transition matrix updates each node one at a time using the following proposal:

$$Q\big((x_i, \mathbf{x}_{-i}) \to (x_i', \mathbf{x}_{-i})\big) = p(x_i' \mid \mathbf{x}_{-i})$$

  - It is efficient since $p(x_i' \mid \mathbf{x}_{-i})$ only depends on the values in X_i's Markov blanket

Gibbs sampling

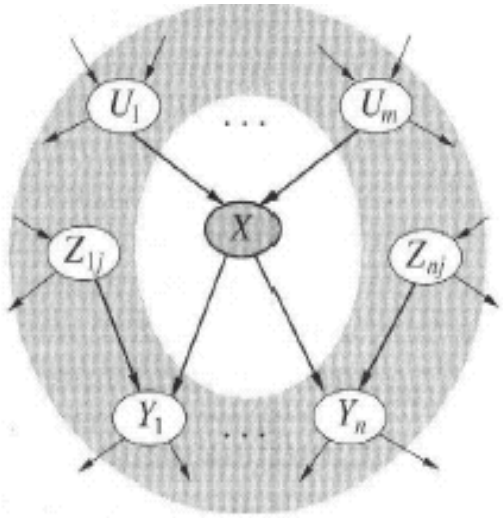
- Gibbs sampling is an MCMC algorithm that is especially appropriate for inference in graphical models.
- The procedue
  - we have variable set $\boldsymbol{X}=\{x_1, x_2, x_3, \ldots x_N\}$ for a GM
  - at each step one of the variables X_i is selected (at random or according to some fixed sequences), denote the remaining variables as $\boldsymbol{X}_{-i}$, and its current value as $\boldsymbol{x}_{-i}^{(t-1)}$
    - Using the "alarm network" as an example, say at time t we choose X_E, and we denote the current value assignments of the remaining variables, $\boldsymbol{X}_{-E}$, obtained from previous samples, as $\boldsymbol{x}_{-E}^{(t-1)} = \left\{x_B^{(t-1)}, x_A^{(t-1)}, x_J^{(t-1)}, x_M^{(t-1)}\right\}$
  - the conditonal distribution $p(X_i | \boldsymbol{x}_{-i}^{(t-1)})$ is computed
  - a value $\boldsymbol{x}_i^{(t)}$ is sampled from this distribution
  - the sample $\boldsymbol{x}_i^{(t)}$ replaces the previous sampled value of X_i in $\boldsymbol{X}$.
    - i.e., $\boldsymbol{x}^{(t)} = \boldsymbol{x}_{-E}^{(t-1)} \cup \boldsymbol{x}_E^{(t)}$



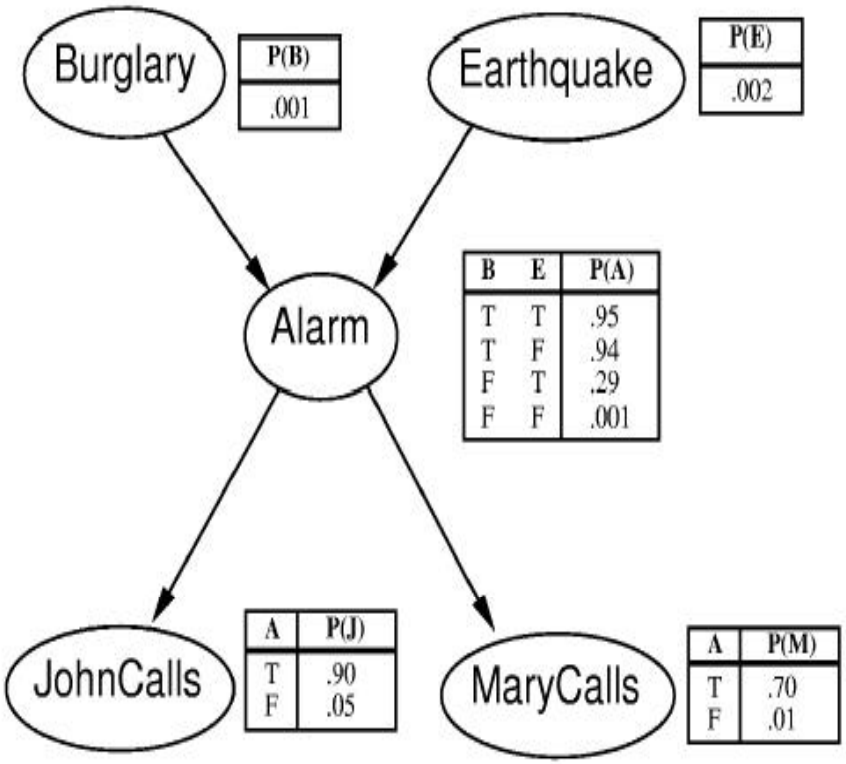
Markov Blanket

- **Markov Blanket in BN**
  - A variable is independent from others, given its parents, children and children's parents (d-separation).
- **MB in MRF**
  - A variable is independent all its non-neighbors, given all its direct neighbors.

    $\Rightarrow p(X_i \mid X_{-i}) = p(X_i \mid \mathrm{MB}(X_i))$
- **Gibbs sampling**
  - Every step, choose one variable and sample it by P(X|MB(X)) based on previous sample.



Gibbs sampling of the alarm network

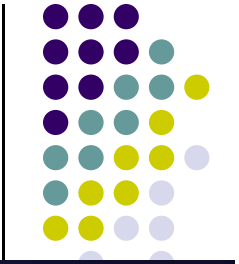


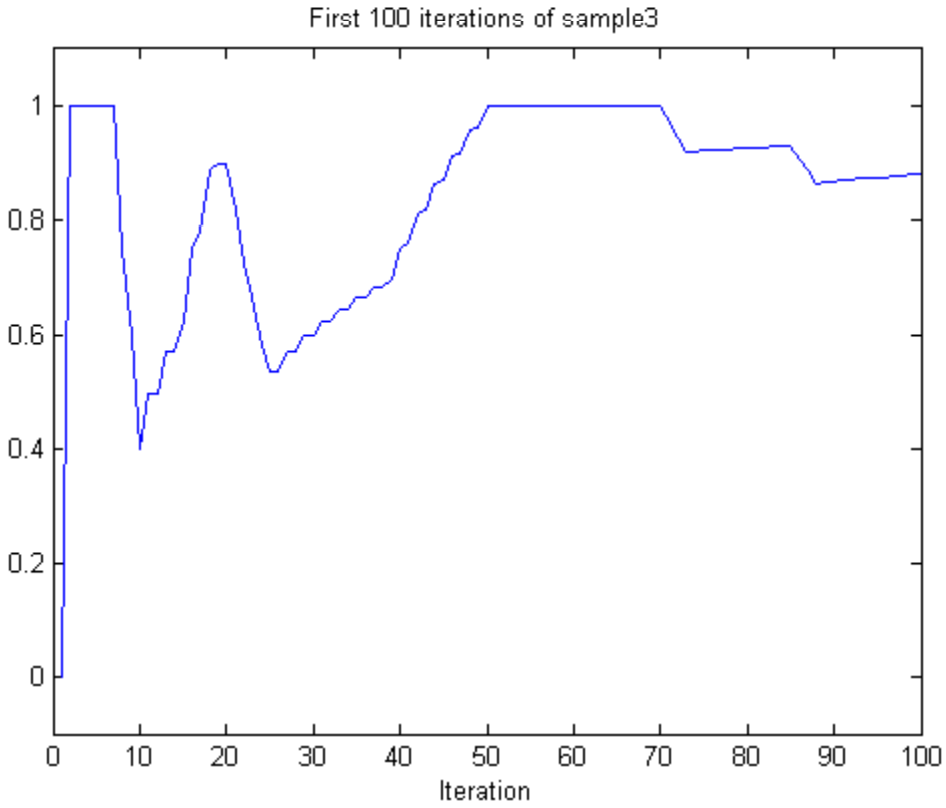
MB(A)={B, E, J, M}

MB(E)={A, B}

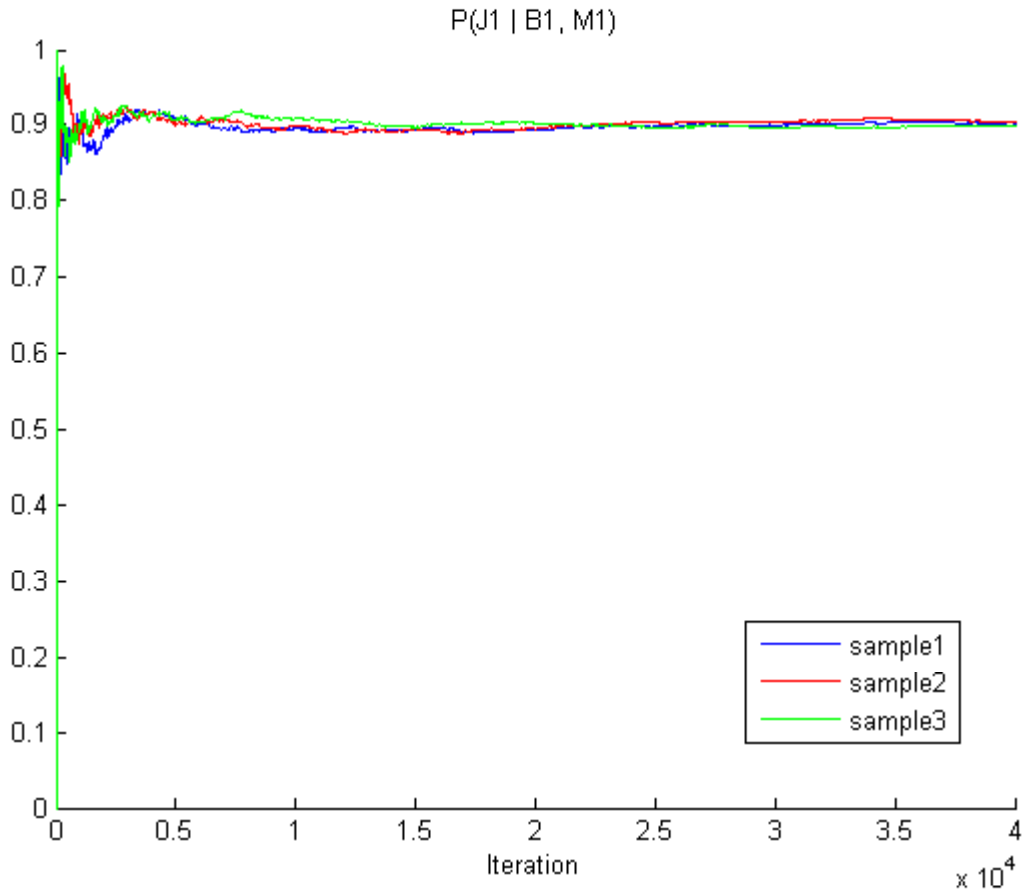
- To calculate P(J|B1,M1)
- Choose (B1,E0,A1,M1,J1) as a start
- Evidences are B1, M1, variables are A, E, J.
- Choose next variable as A
- Sample A by P(A|MB(A))=P(A|B1, E0, M1, J1) suppose to be false.
- (B1, E0, A0, M1, J1)
- Choose next random variable as E, sample E~P(E|B1,A0)
- ...

Example

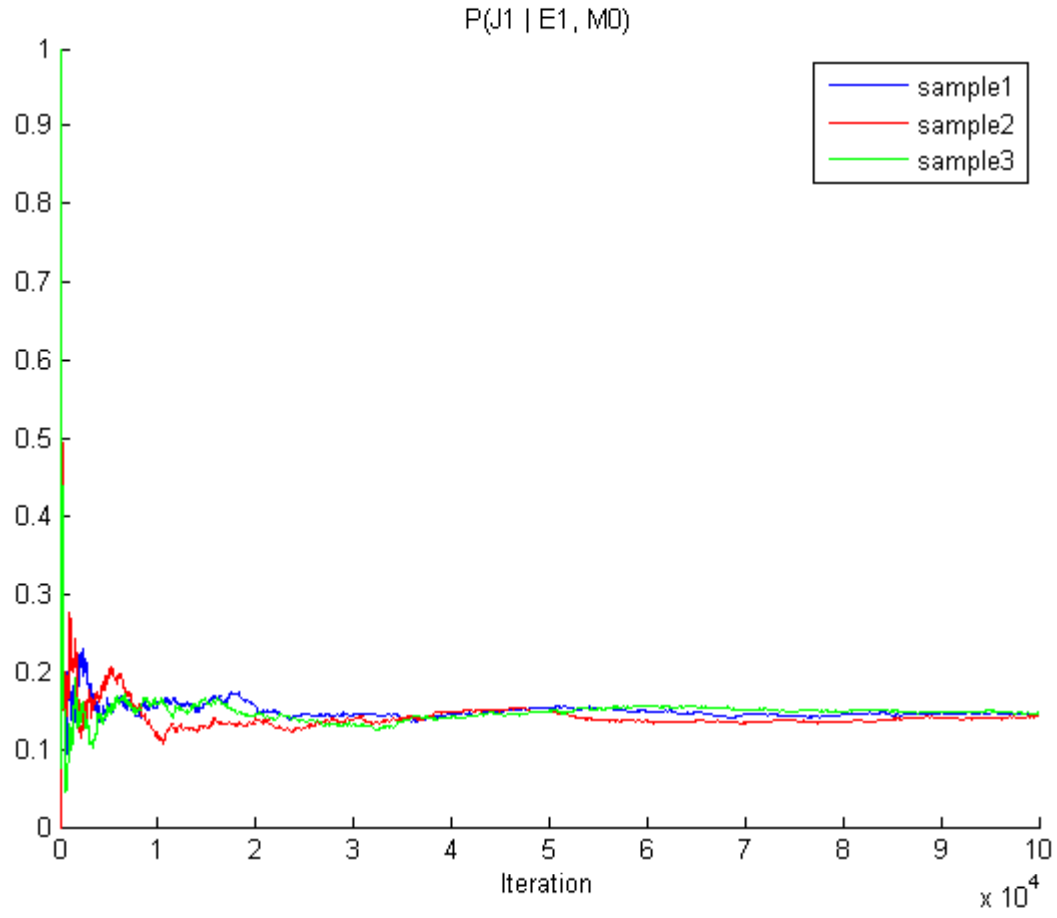




Example:



Example



Example

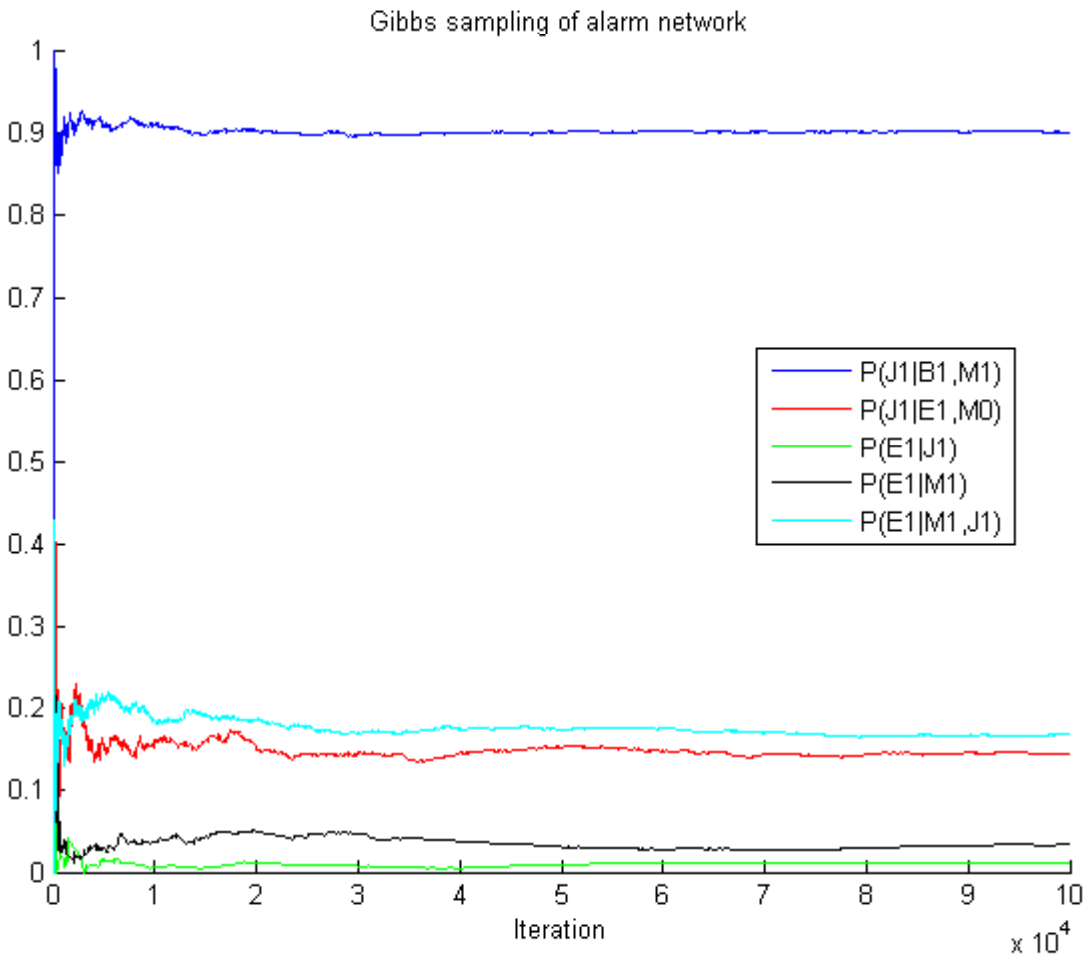
P(J1 | B1,M1) = 0.90

P(J1 | E1,M0) = 0.14

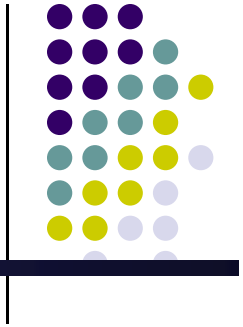
P(E1 | J1) = 0.01

P(E1 | M1) = 0.04

P(E1 | M1,J1) = 0.17

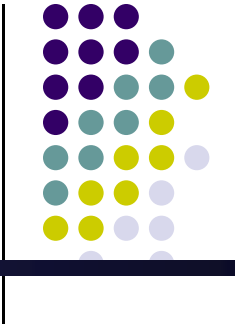


The Art of simulation



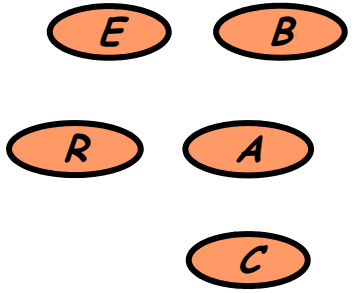
- Run several chains
- Start at over-dispersed points
- Monitor the log lik.
- Monitor the serial correlations
- Monitor acceptance ratios
- Re-parameterize (to get approx. indep.)
- Re-block (Gibbs)
- Collapse (int. over other pars.)
- Run with troubled pars. fixed at reasonable vals.

Learning Graphical Models

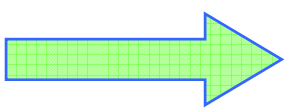


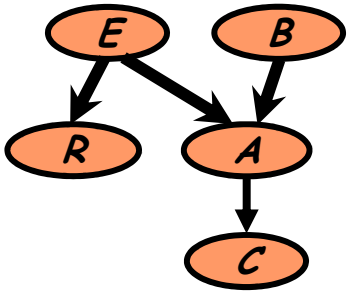
The goal:

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



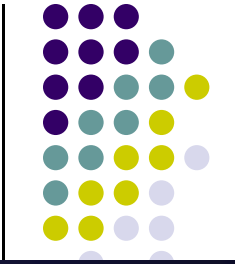
(B,E,A,C,R)=(T,F,F,T,F)
(B,E,A,C,R)=(T,F,T,T,F)
……..
(B,E,A,C,R)=(F,T,T,T,F)





E	B	P(A | E,B)	
e	$\bar{b}$	0.9	0.1
e	b	0.2	0.8
$\bar{e}$	$\bar{b}$	0.9	0.1
$\bar{e}$	b	0.01	0.99

Learning Graphical Models (cont.)

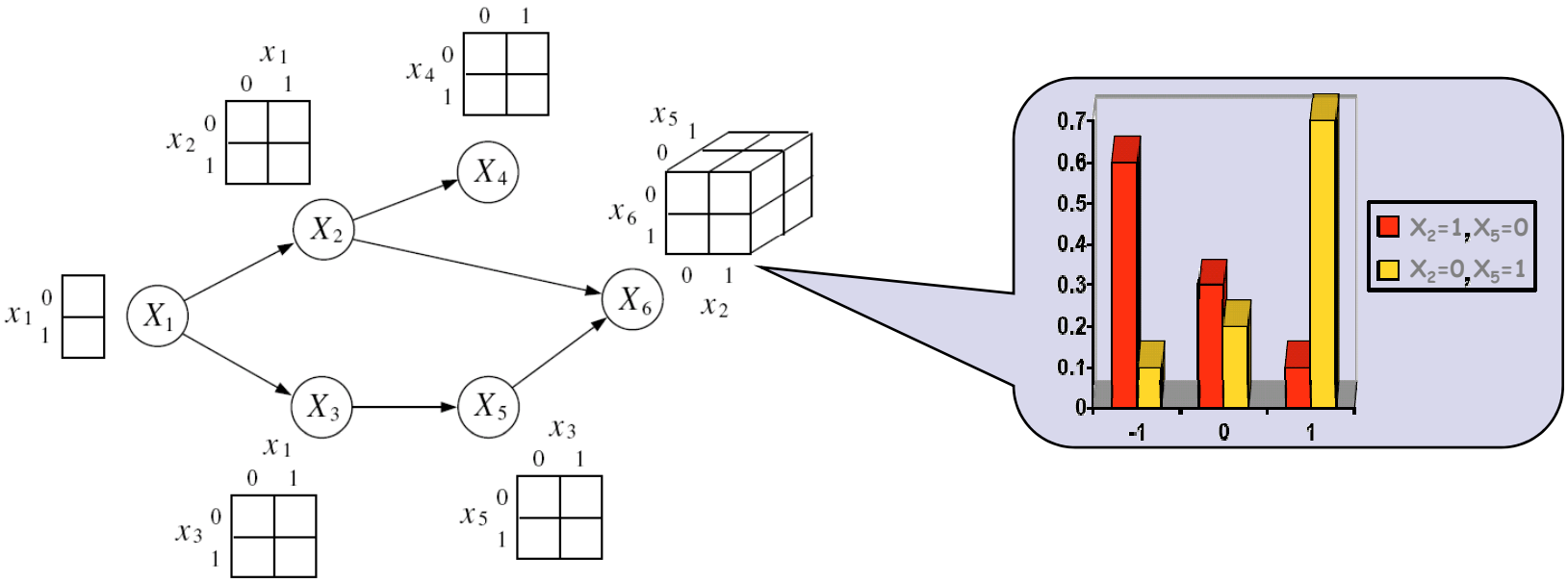


- Scenarios:
  - completely observed GMs
    - directed
    - undirected
  - partially observed GMs
    - directed
    - undirected (an open research topic)
- Estimation principles:
  - Maximal likelihood estimation (MLE)
  - Bayesian estimation
  - Maximal conditional likelihood
  - Maximal "Margin"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

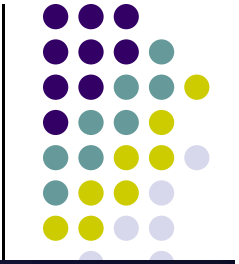
MLE for general BN parameters

- 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:

$$\ell(\theta; D) = \log p(D \mid \theta) = \log \prod_n \left(\prod_i p(x_{n,i} \mid \mathbf{x}_{n,\pi_i}, \theta_i) \right) = \sum_i \left(\sum_n \log p(x_{n,i} \mid \mathbf{x}_{n,\pi_i}, \theta_i) \right)$$



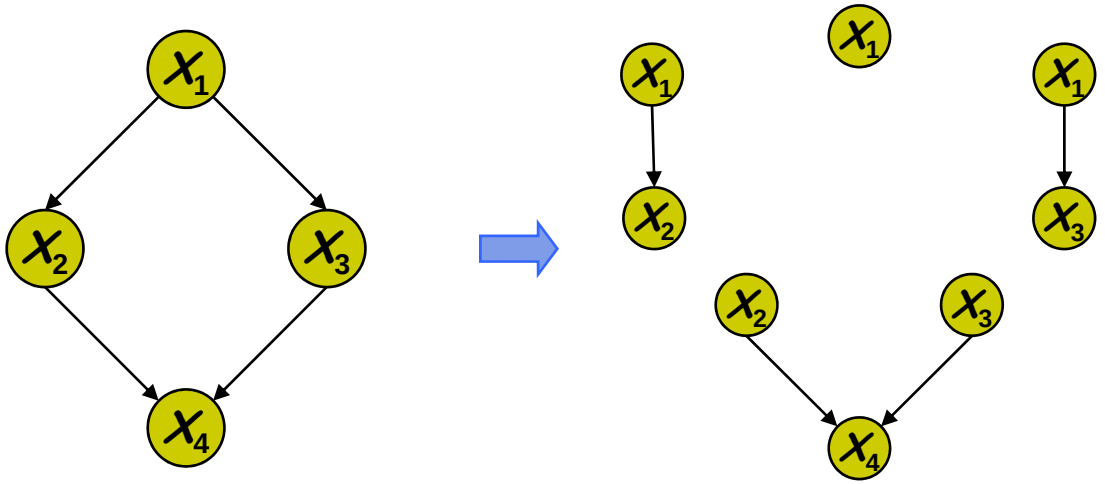
Example: decomposable likelihood of a directed model



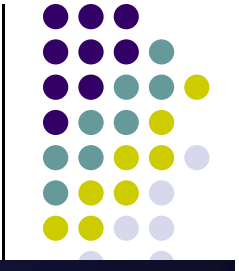
- Consider the distribution defined by the directed acyclic GM:

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

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

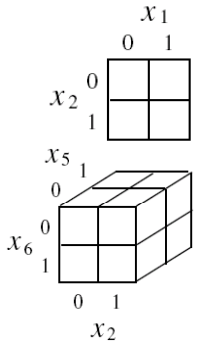


E.g.: MLE for BNs with tabular CPDs



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

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



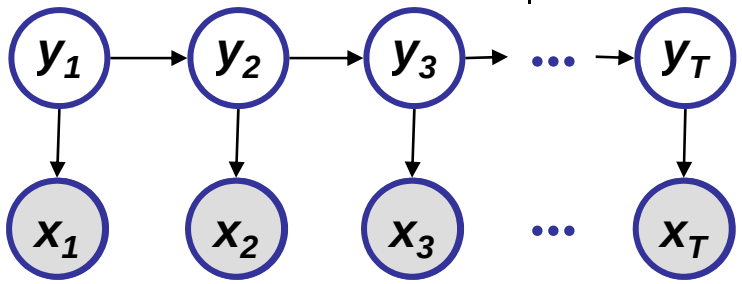
  - Note that in case of multiple parents, $\mathbf{X}_{\pi_i}$ will have a composite state, and the CPD will be a high-dimensional table
  - The sufficient statistics are counts of family configurations

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

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

Recall definition of HMM

- Transition probabilities between any two states



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

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

- Start probabilities

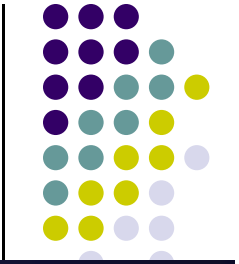
$$p(y_1) \sim \text{Multinomial}(\pi_1, \pi_2, \ldots, \pi_M).$$

- Emission probabilities associated with each state

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

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

Supervised ML estimation



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

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

  - **Define:**

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

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

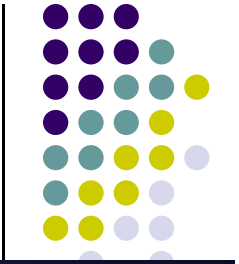
  - **We can show that the maximum likelihood parameters θ are:**

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

$$b_{ik}^{ML} = \frac{\#(i \to k)}{\#(i \to \bullet)} = \frac{\sum_n \sum_{t=1}^{T} y_{n,t}^{i} x_{n,t}^{k}}{\sum_n \sum_{t=1}^{T} y_{n,t}^{i}} = \frac{B_{ik}}{\sum_{k'} B_{ik'}}$$

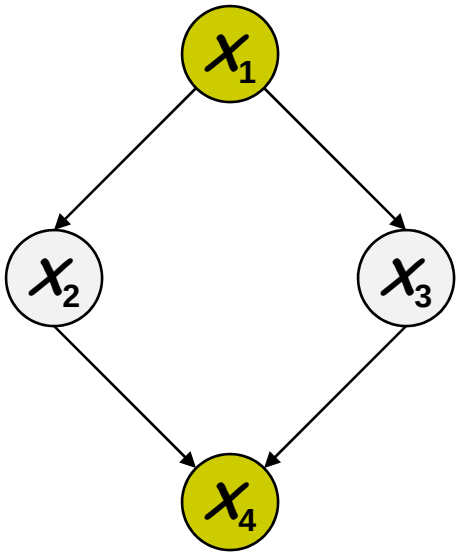
  - **If y is continuous, we can treat $\{(x_{n,t}, y_{n,t}) : t = 1:T, n = 1:N\}$ as $N \times T$ observations of, e.g., a Gaussian, and apply learning rules for Gaussian …**

What if some nodes are not observed?



- Consider the distribution defined by the directed acyclic GM:

$$p(x \mid \theta) = p(x_1 \mid \theta_1)\, p(x_2 \mid x_1, \theta_1)\, p(x_3 \mid x_1, \theta_3)\, p(x_4 \mid x_2, x_3, \theta_1)$$

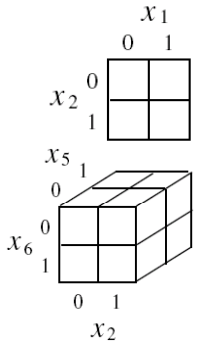


- Need to compute $p(x_H|x_V)$ → inference

MLE for BNs with tabular CPDs

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

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

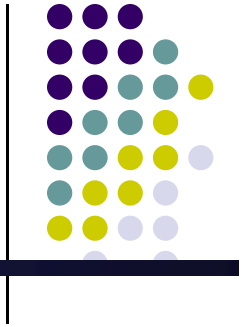


  - Note that in case of multiple parents, $\mathbf{X}_{\pi_i}$ will have a composite state, and the CPD will be a high-dimensional table
  - The sufficient statistics are counts of family configurations

$$n_{ijk} \overset{\text{def}}{=} \sum_n \left\langle x_{n,i}^j x_{n,\pi_i}^k \right\rangle$$

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

Summary



- A GM*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 $p(M \mid D)$, 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*.