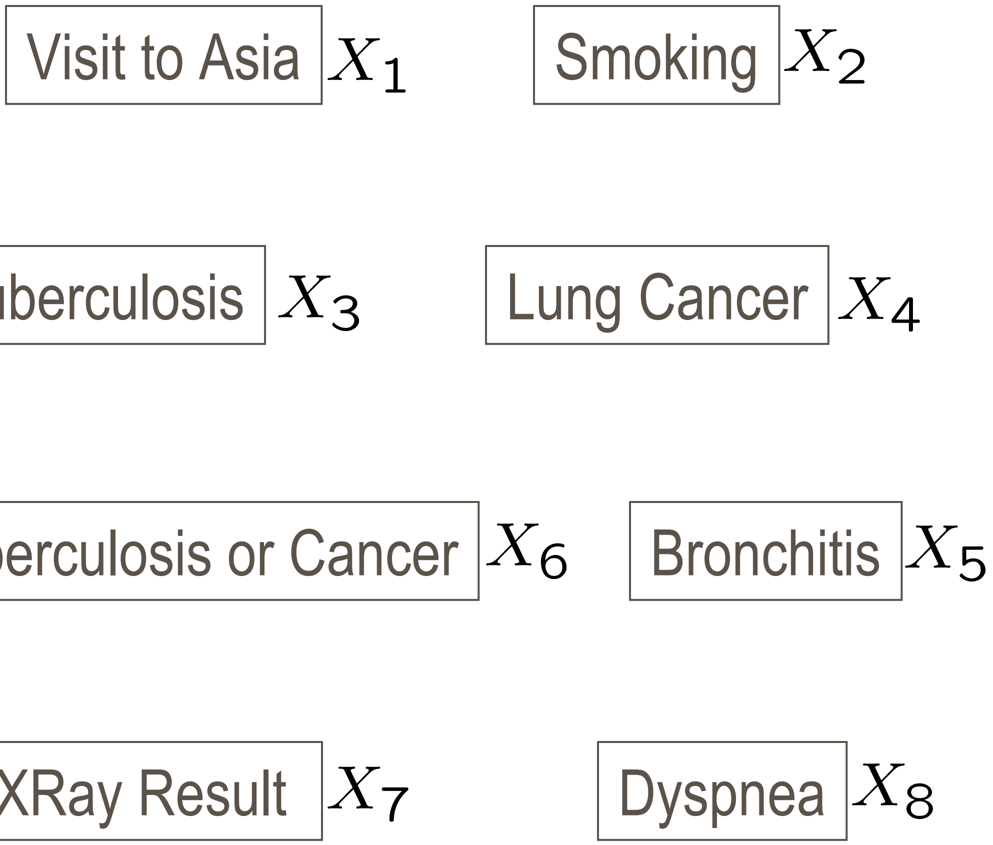
Graphical Models

Adrian Barbu

What is a Graphical Model?

Example: Medical Diagnosis

- Observed data:
  - Symptoms
  - Medical tests
  - Circumstances
- Unknowns:
  - Cause (disease)



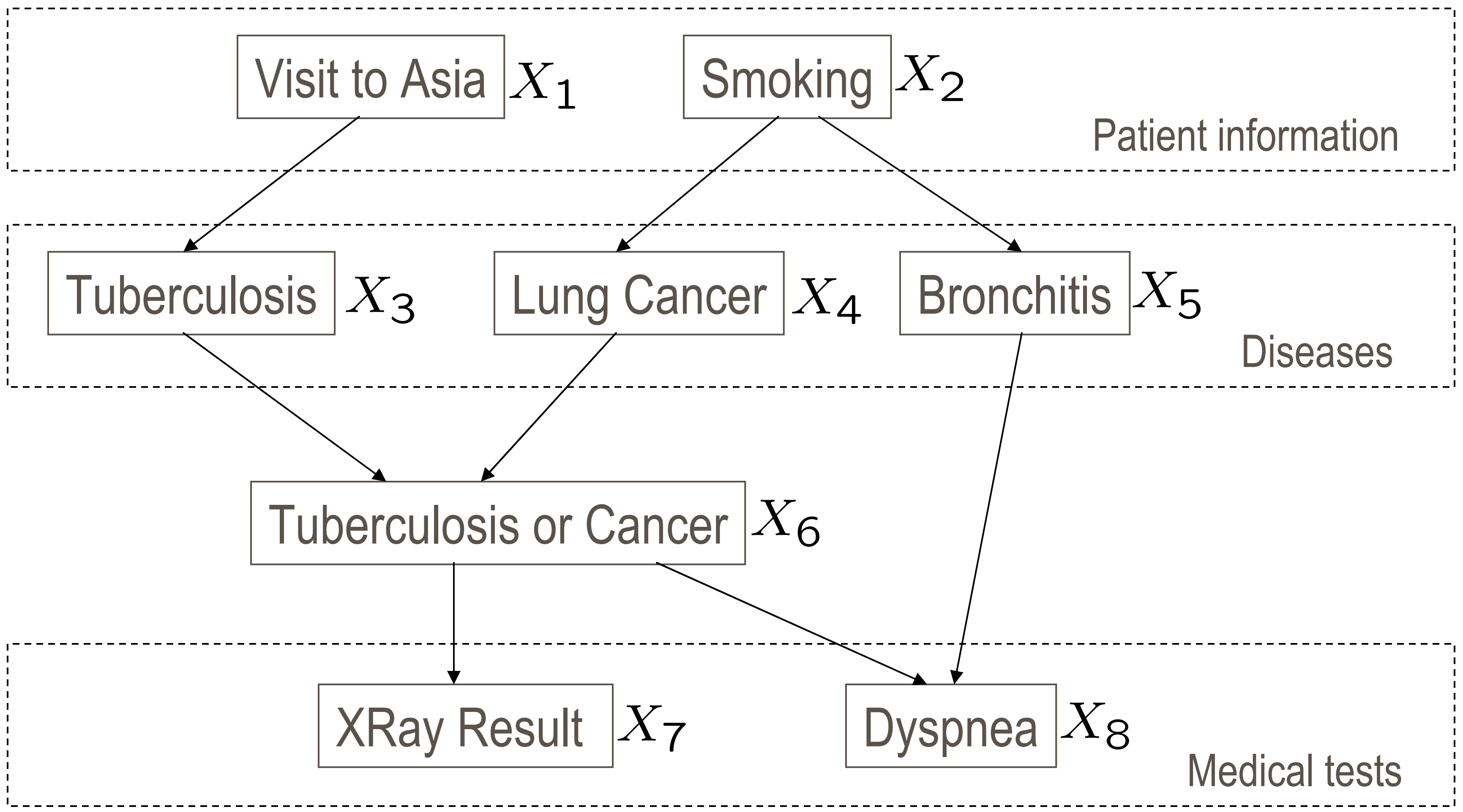
Probabilities

- Want to find most probable disease given the observations
- Can represent whole process by a joint probability

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

  - 2^8 states
  - Not all are needed
  - No medical insight
  - Need too much training data
- Encode domain knowledge into relationships between variables
  - Need less training data
- Inference
  - Find most probable unknown states given observations

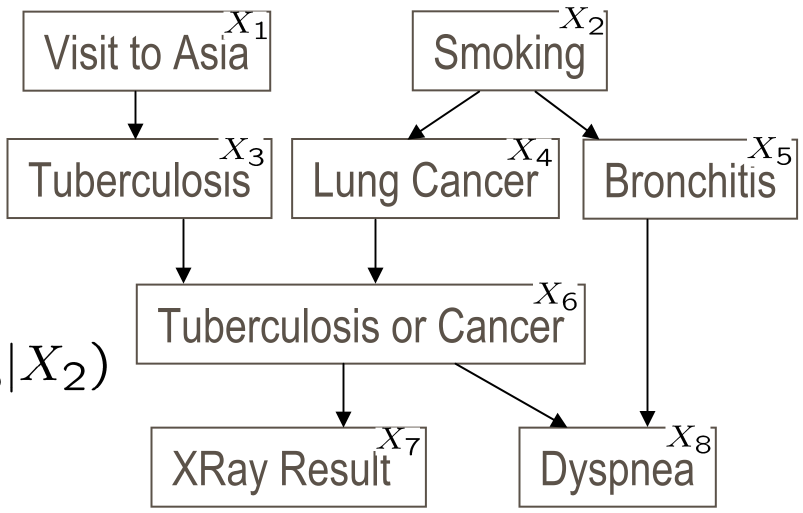
Dependency Graph



Probabilistic Graphical Models

- Representation of the dependencies using a graph
  - Nodes= random variables
  - Edges= dependencies among variables
    - Edge absence= conditional independence
  - Directed: cause→effect
  - Undirected: bidirectional relationship

- Joint probability factors:



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

Advantages of PGM

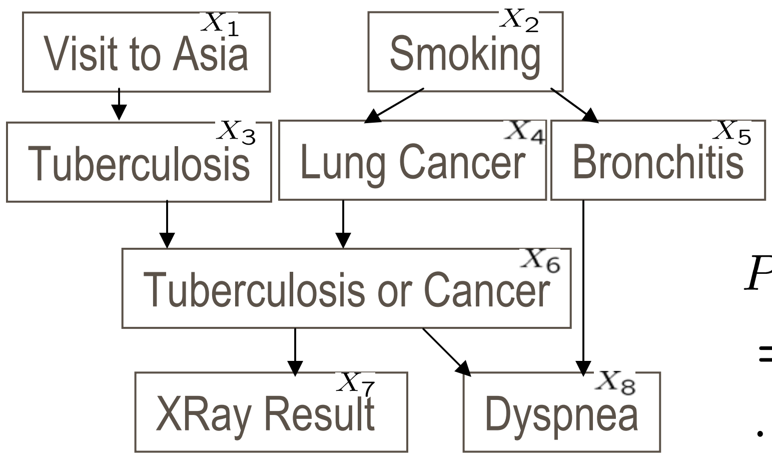
- Why is this good?
  - Each probability involves fewer variables
    - Need fewer samples to train
    - Not so prone to overfitting
  - Total number of bins is smaller
    - Before were 2^8=256 bins
    - Now 2+2+4+4+4+8+4+8=36
    - 8 times fewer bins
  - Easier to find the best solution
    - Explore the graph structure
    - Faster than exhaustive search of all possibilities
  - Incorporation of domain knowledge
    - Avoids impossible solutions

Two types of PGM

- Directed edges
  - Model causality relationships
  - Bayesian Network or Directed Graphical Model
  - DAG=Directed Acyclic Graph

- Undirected edges
  - Model symmetric correlations between variables
  - Markov Random Field or Undirected Graphical Model

Bayesian Network:



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

Factorization Theorem:

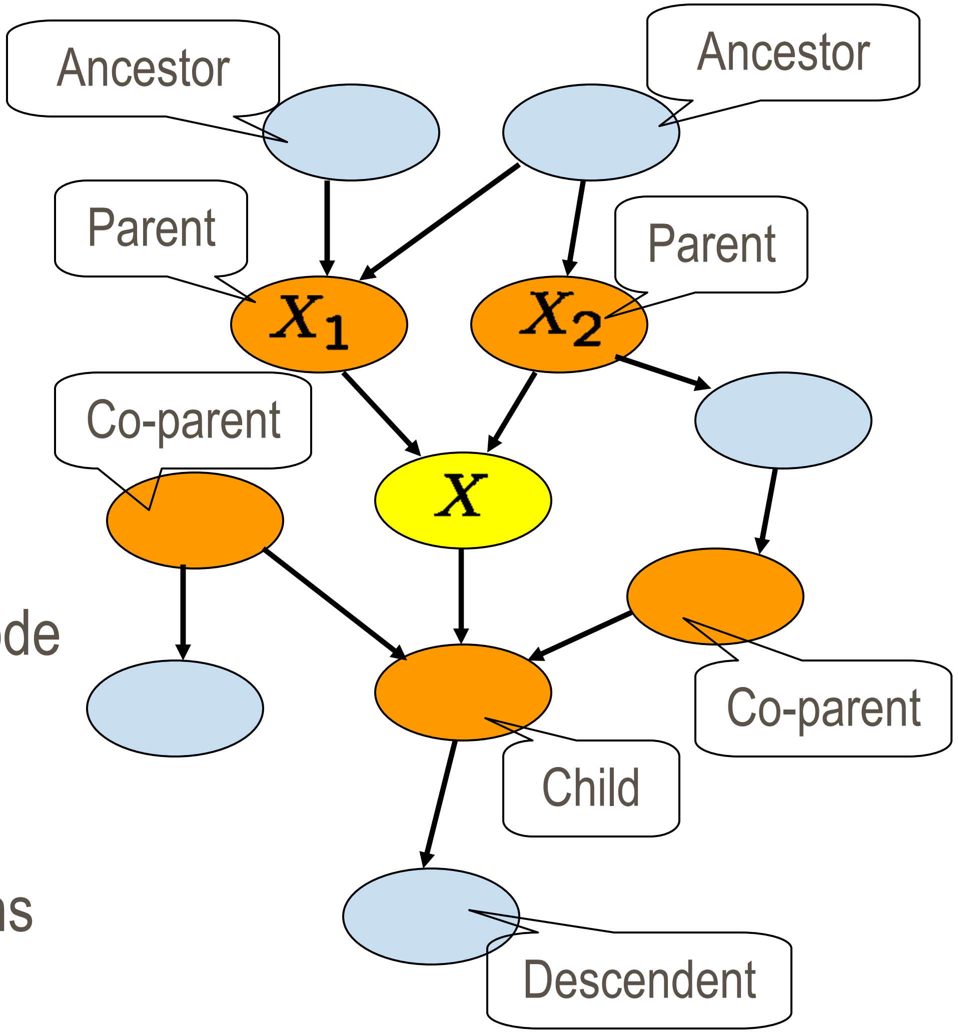
- Given a DAG, the most general form of the probability distribution that is consistent with the graph factors according to “node given its parents”:

$$P(\mathbf{X}) = \prod_{i=1}^{d} P(X_i|\mathbf{X}_{\pi_i})$$

  - $\mathbf{X}_{\pi_i}$ is the set of parents of X_i
  - d = number of nodes (variables) in the graph

Bayesian Network Properties

- Markov Blanket
  - Parents
  - Children
  - Co-parents (parents of the children)
- Conditional Independence
  - Any node is conditional independent of any other node not in its Markov Blanket
- Joint distribution from:
  - Local conditional distributions
  - DAG structure



Conditional Independence

- Common parent
  - Knowing C decouples A and B
  - A and B are conditionally independent given C

$$\begin{tikzcd} & C \arrow[dl] \arrow[dr] & \\ A & & B \end{tikzcd}$$

- Chain
  - Knowing B decouples A and C
  - A and C are conditionally independent given B

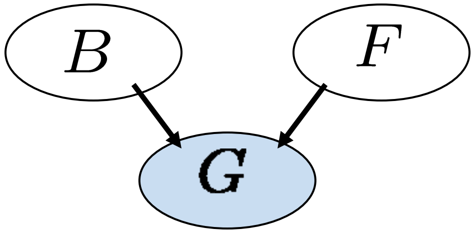
$$\begin{tikzcd} A \arrow[r] & B \arrow[r] & C \end{tikzcd}$$

- Common child
  - A and B are independent
  - Knowing C couples them together
  - A or B can explain-away the observation C

$$\begin{tikzcd} A \arrow[dr] & & B \arrow[dl] \\ & C & \end{tikzcd}$$

Explain Away Example

- Graph nodes:
  - B=battery, F=fuel, G=gauge



- B and F are independent

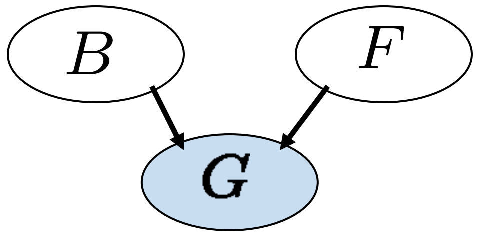
$$P(B=1)=0.9, P(F=1)=0.9$$

- Unreliable fuel gauge:

$$P(G=1|B=1,F=1)=0.8$$
$$P(G=1|B=1,F=0)=0.2$$
$$P(G=1|B=0,F=1)=0.2$$
$$P(G=1|B=0,F=0)=0.1$$

- Suppose we observe $G=0$
- Want $P(F=0|G=0)$

Explain Away Example



$$\begin{tikzcd} B \arrow[dr] & & F \arrow[dl] \\ & G & \end{tikzcd}$$

- Bayes Rule:

$$P(F=0|G=0) = \frac{P(G=0|F=0)P(F=0)}{P(G=0)}$$

- We compute

$$P(G=0) = \sum_B \sum_F P(G=0|B,F)P(B)P(F) = 0.315$$

$$P(G=0|F=0) = \sum_B P(G=0|B,F=0)p(B) = 0.81$$

- So we get

$$P(F=0|G=0) = \frac{0.81 \cdot 0.1}{0.315} = 0.257$$

- If we observe that $B=0$, we get

$$P(F=0|G=0,B=0) = \frac{P(G=0|B=0,F=0)P(F=0)}{\sum_F P(G=0|B=0,F)P(F)} = 0.111$$

- So $B=0$ explained away the $G=0$

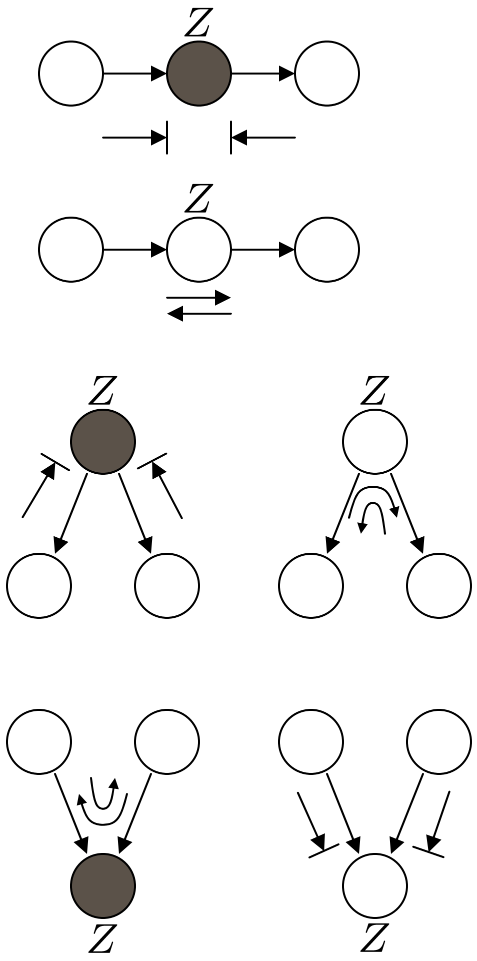
D-separation

Problem

- Find whether some groups of nodes are conditionally independent

Reason

- Can compute each independent group separately
- Faster inference



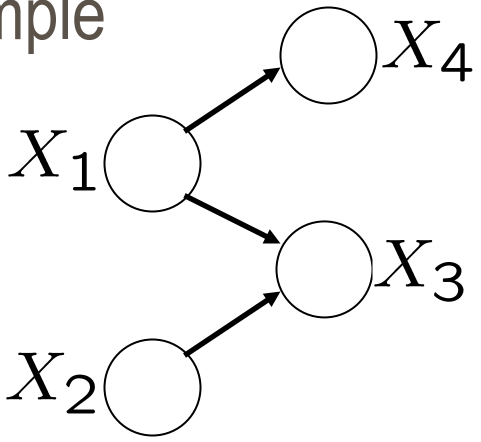
- X is d-separated from Y given Z if one cannot send a "Bayes ball" from X to Y
  - Z can be known (=dark) or not
  - Notation: $\text{dsep}_G(X, Y|Z)$

D-separation and Independence

- Define all independence properties obtained from the d-separation

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

- Example



$$\begin{aligned} I(G) = \{\ & X_1 \perp\!\!\!\perp X_2 \\ & X_2 \perp\!\!\!\perp X_4 \\ & X_2 \perp\!\!\!\perp X_4|X_1 \\ & X_2 \perp\!\!\!\perp X_4|\{X_1, X_3\} \\ & X_3 \perp\!\!\!\perp X_4|X_1 \\ & X_4 \perp\!\!\!\perp \{X_2, X_3\}|X_1\ \} \end{aligned}$$

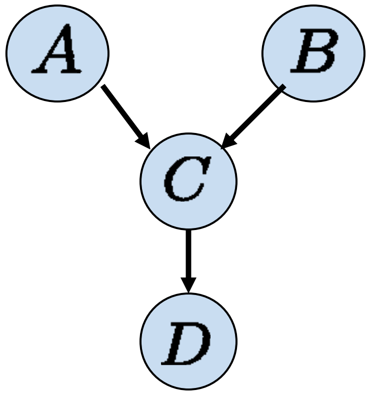
Equivalence Theorem

- A distribution satisfies $I(G)$ if and only if it factors according to the graph G.

Conditional Probability Tables (CPT)

A^0	0.3
A^1	0.7

B^0	0.4
B^1	0.6



	A^0B^0	A^0B^1	A^1B^0	A^1B^1
C^0	0.4	1	0.9	0.8
C^1	0.6	0	0.1	0.2

	C^0	C^1
D^0	0.3	0.5
D^1	0.7	0.5

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

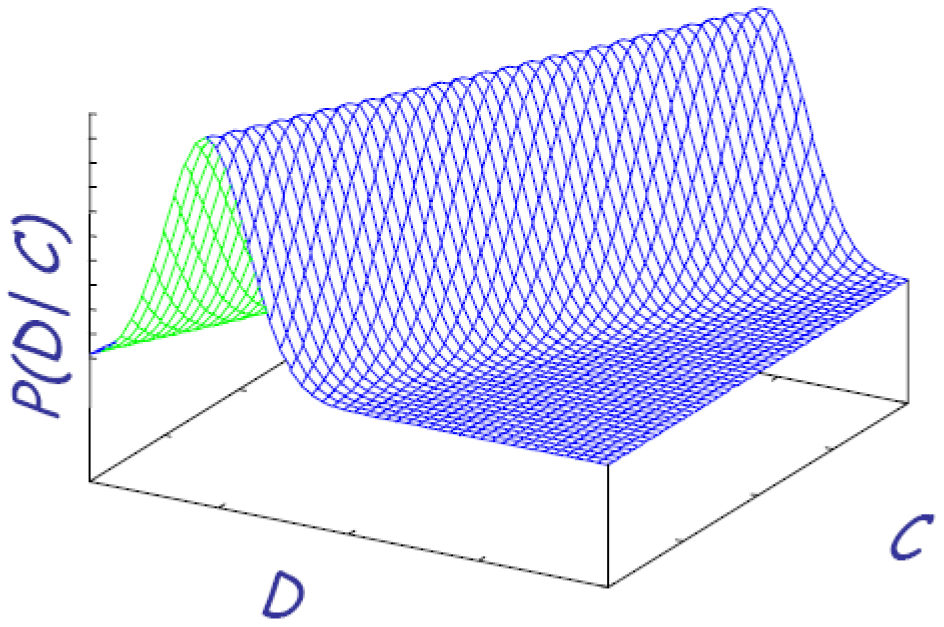
Conditional Probability Density Functions (CPD)

$A \sim \mathcal{N}(\mu_A, \Sigma_A)$ $B \sim \mathcal{N}(\mu_B, \Sigma_B)$

$$\begin{tikzcd}
A \arrow[dr] & & B \arrow[dl] \\
& C \arrow[d] & \\
& D &
\end{tikzcd}$$

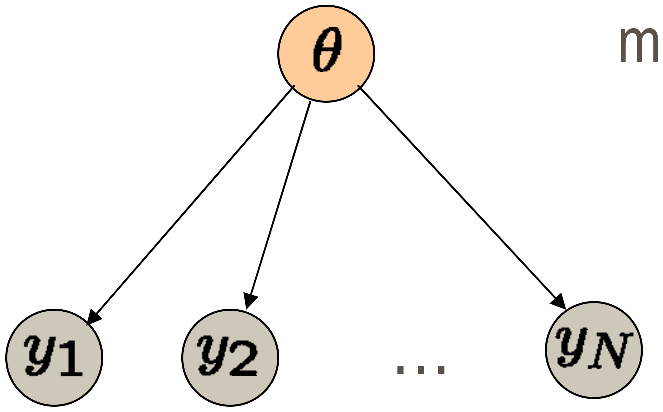
$C \sim \mathcal{N}(A + B, \Sigma_C)$

$D \sim \mathcal{N}(\mu_A + C, \Sigma_D)$



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

Conditionally Independent Observations



model parameters

observations

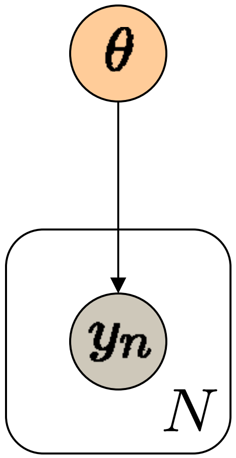


plate notation

- Variables in plate are replicated conditionally independent given model parameters

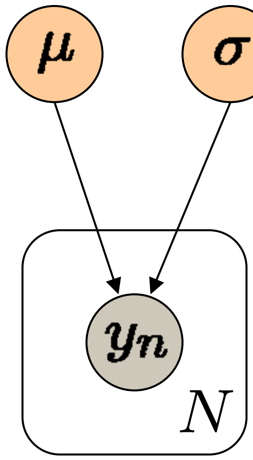
Example: Gaussian Model

- Generative model (Naïve Bayes)
  - Bayes Rule

$$P(\mu,\sigma|y_1,\ldots,y_N)=\frac{P(y_1,\ldots,y_N|\mu,\sigma)P(\mu,\sigma)}{P(y_1,\ldots,y_N)}$$

  - i.i.d assumption
  - Model parameters: μ,σ
  - Likelihood = P(Observations|Parameters):

$$P(y_1,\ldots,y_N|\mu,\sigma)=\prod_{i=1}^{N}P(y_i|\mu,\sigma)=\prod_{i=1}^{N}\frac{1}{\sqrt{2\pi}\sigma}e^{-(y_i-\mu)^2/2\sigma^2}$$

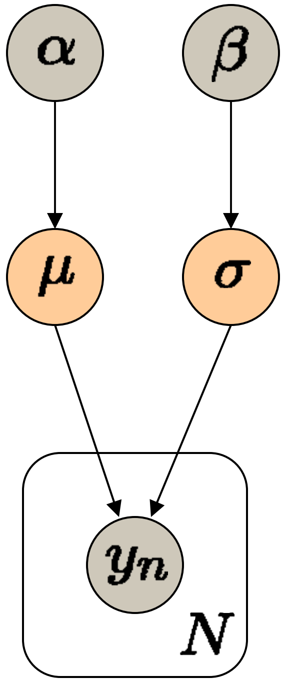


Example: Gaussian Model

- Prior on model parameters
  - Assume parameters independent

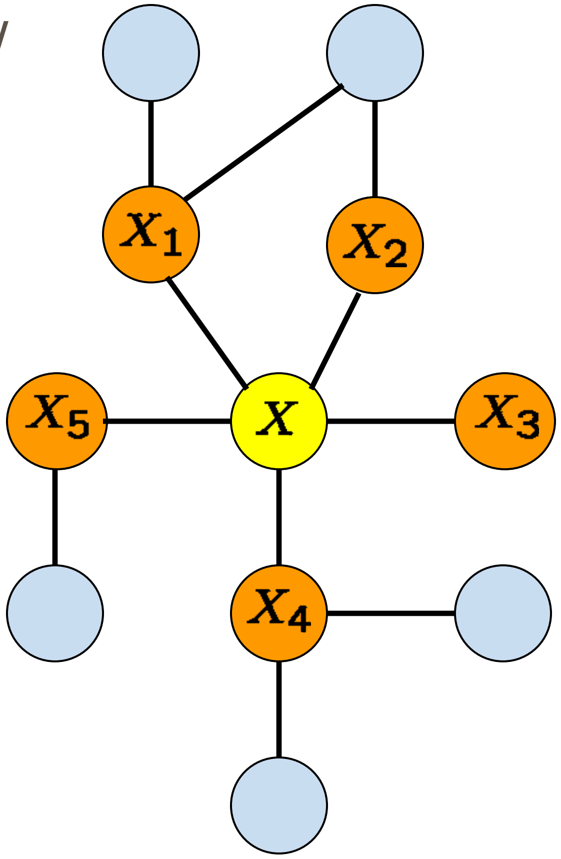
$$P(\mu, \sigma) = P(\mu|\alpha)P(\sigma|\beta)$$

- Possible priors $P(\mu|\alpha)$
  - Parametric:
    - Gaussian
    - Student t distribution
    - …
  - Non-parametric
    - Histogram
    - Parzen windows

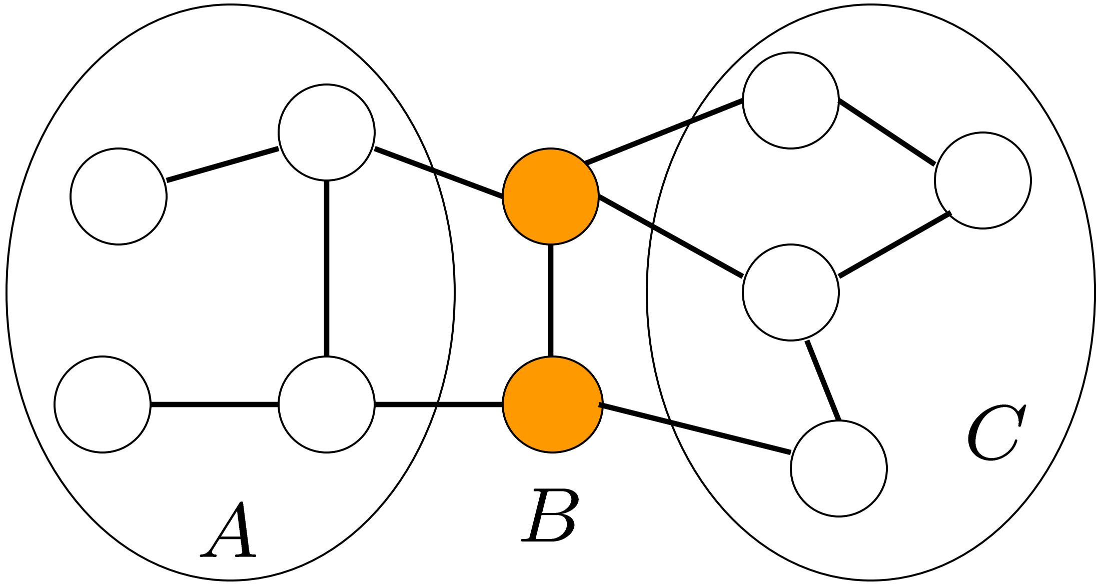


Markov Random Fields

- Undirected Graph:
  - A node is conditionally independent of every other node given its direct neighbors
- Probability Distribution
  - A set of cliques = complete subgraphs
  - A set of potential functions on the cliques
- Pros and cons:
  - Easy to compute the probability
  - Hard to obtain samples from the probability
  - Hard to find max probability configuration



Conditional Independence



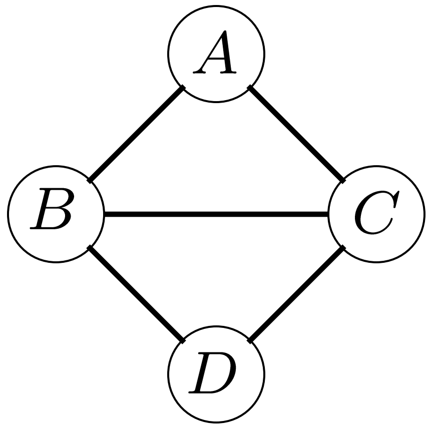
G undirected graph

- B *separates* A and C if every path from A to C passes through a node in B $\text{sep}_G(A, C|B)$
- A probability distribution satisfies the *Global Markov Property* if for any disjoint *A*, *B*, *C*, such that *B* separates *A* and *C* then *A* is independent of *C* given *B:*

$$I(G) = \{A \perp\!\!\!\perp C|B : \text{sep}_G(A, C|B)\}$$

Cliques

- Clique:
  - A complete subgraph G’=(V’,E’) of G=(V,E)
  - Complete means fully connected
- Maximal clique
  - A clique such that there is no other clique that includes it
- Example:
  - Max-cliques:
    - {A,B,C}, {B,C,D}
  - Other cliques:
    - {A,B}, {A,C}, {B,C}
    - {B,D}, {C,D}
    - {A}, {B}, {C}, {D}



Markov Random Field Probability

Given:

- An undirected graph G
- Define a set C of cliques of G
- Define a set of potential functions $\psi_c(\mathbf{x}_c),\ \forall c \in C$
  - Encourage certain configurations

MRF probability:

$$P(\mathbf{x}) = P(x_1, ..., x_N) = \frac{1}{Z}\prod_{c\in C}\psi_c(\mathbf{x}_c)$$

where Z is the partition function

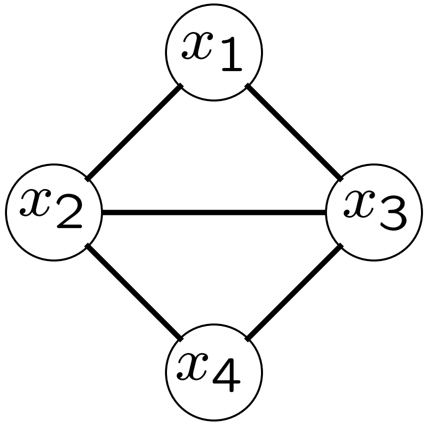
$$Z = \sum_{\mathbf{x}}\prod_{c\in C}\psi_c(\mathbf{x}_c)$$

Example with Max-cliques

- C=all max-cliques

$$P(x_1, ..., x_4) = \frac{1}{Z}\psi_1(x_1, x_2, x_3)\psi_2(x_2, x_3, x_4)$$

$$Z = \sum_{x_1,x_2,x_3,x_4} \psi_1(x_1x_2x_3)\psi_2(x_2x_3x_4)$$

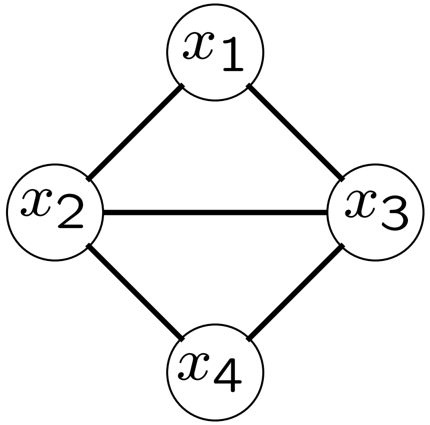


- For discrete $x_1, ..., x_4$, we can represent $P(x_1, ..., x_4)$ with two 3D tables (histograms) instead of one 4D table.
  - Avoid overfitting

Example with Sub-cliques

- C=all 2D cliques

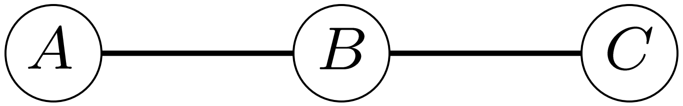
$$P(x_1,...,x_4) = \frac{1}{Z}\psi_{12}(x_1,x_2)\psi_{13}(x_1,x_3)$$
$$\cdot\psi_{23}(x_2,x_3)\psi_{24}(x_2,x_4)\psi_{34}(x_3,x_4)$$



$$Z = \sum_{x_1,x_2,x_3,x_4} \psi_{12}(x_1,x_2)\psi_{13}(x_1,x_3)\psi_{23}(x_2,x_3)\psi_{24}(x_2,x_4)\psi_{34}(x_3,x_4)$$

- For discrete $x_1, ..., x_4$, we can represent $P(x_1, ..., x_4)$ with five 2D tables instead of one 4D table.
  - Avoid overfitting
  - But it might be a weaker model

Clique Potentials



- Clique potentials are not probabilities

$$\begin{aligned} P(A,B,C) &= P(A)P(B|A)P(C|B) \\ &= P(B)P(A|B)P(C|B) \\ &= P(C)P(B|C)P(A|B) \\ &= P(A,B)P(C|B) \end{aligned}$$

- We cannot have only marginal probabilities, e.g. P(A)
- We cannot have only conditional probabilities, e.g. P(B|A)
- Probability = all clique potentials + partition function Z

Exponential Form

- Since all cliques are positive, can use exponential form

$$\psi_c(\mathbf{x}_c) = \exp[-\phi_c(\mathbf{x}_c)]$$

- $\phi_c(\mathbf{x}_c)$ is also called potential
- Obtain exponential form of the probability

$$P(\mathbf{x}) = \frac{1}{Z}\exp[-\sum_{c\in C}\phi_c(\mathbf{x}_c)] = \frac{1}{Z}\exp[-H(\mathbf{x})]$$

- $H(\mathbf{x})$ is called the free energy

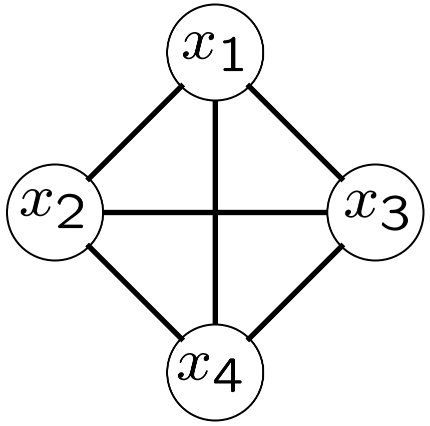
$$H(\mathbf{x}) = \sum_{c\in C}\phi_c(\mathbf{x}_c)$$

- Exponential form
  - In physics is called Boltzmann distribution
  - In statistics, it is called log-linear model

Example: Boltzmann Machine

Boltzmann Machine:

- Binary node values $x_i = \pm 1$
- Fully connected graph G
- Pairwise potentials $\phi_{ij}(x_i, x_j)$
- Probability distribution:



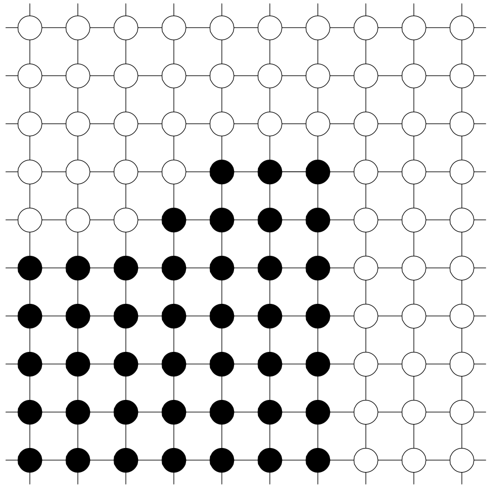
$$P(x_1, ..., x_4) = \frac{1}{Z}\exp[-\sum_{i,j}\phi_{ij}(x_i, x_j)]$$
$$= \frac{1}{Z}\exp[-\sum_{i,j}\theta_{ij}x_i x_j - \sum_i \alpha_i x_i]$$

- Energy:

$$H(\mathbf{x}) = \sum_{i,j}(x_i - \mu_i)\theta_{ij}(x_j - \mu_j) = (\mathbf{x} - \mu)^T\Theta(\mathbf{x} - \mu)$$

Example: Ising/Potts Models

- Nodes on a grid
- Node values (labels):
  - ± 1 for Ising model
  - 1…N>2 for Potts model
- Edges only with the 4 neighbors

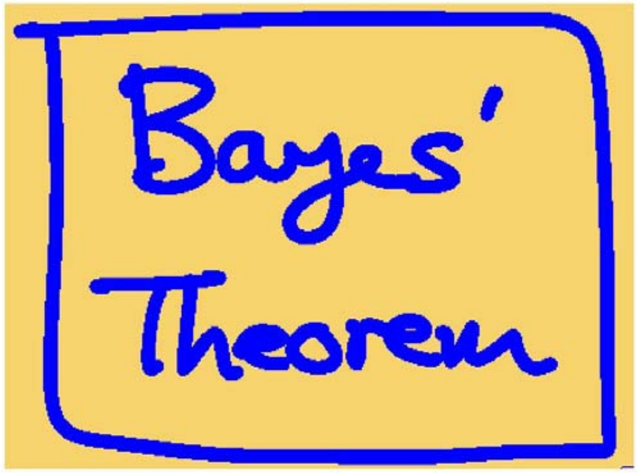


$$P(\mathrm{x}) = \frac{1}{Z}\exp[-\sum_{i,j}\theta_{ij}I(x_i \neq x_j) - \sum_i \alpha_i x_i]$$

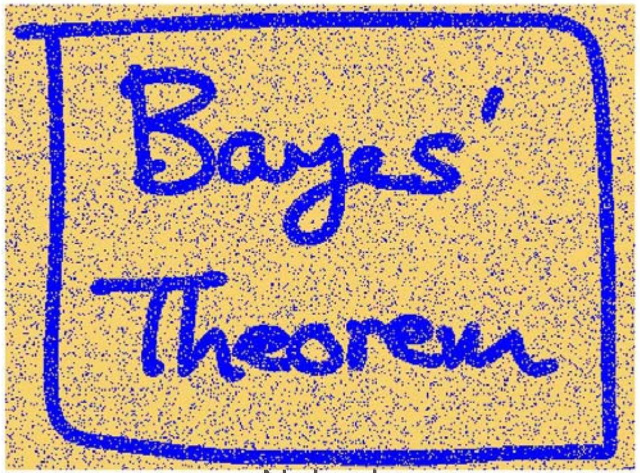
- E.g.
  - Nodes are pixels
  - $\theta_{ij} = \theta > 0$ encourages nearby pixels to have same label
  - α_i form the "external field" (the data term, the likelihood)

Application: Image Denoising

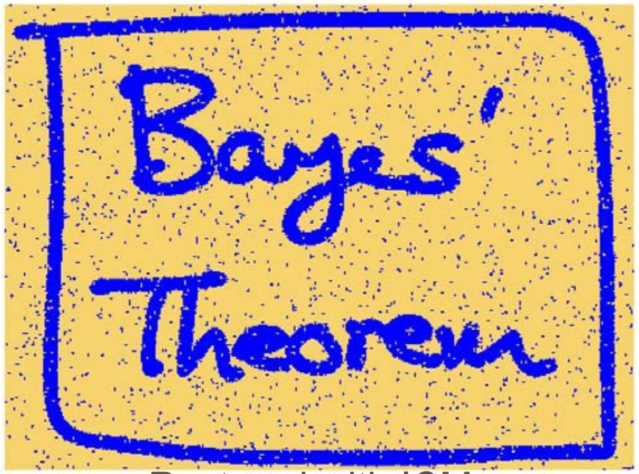
- Ising model $P(\mathbf{x}) = \frac{1}{Z}\exp[-\theta \sum_{i,j} x_i x_j - \alpha \sum_i x_i - \eta \sum_i x_i y_i]$
- Parameters
  - $\theta = 1$
  - $\alpha=0$
  - $\eta=2.1$
- Result depends on:
  - Model
  - Algorithm



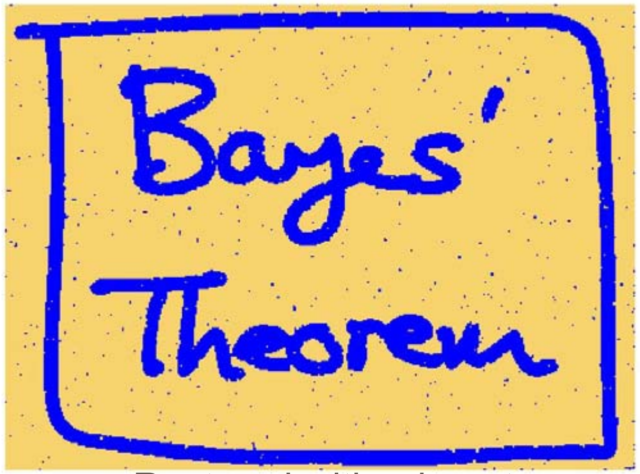
Original image



Noisy image

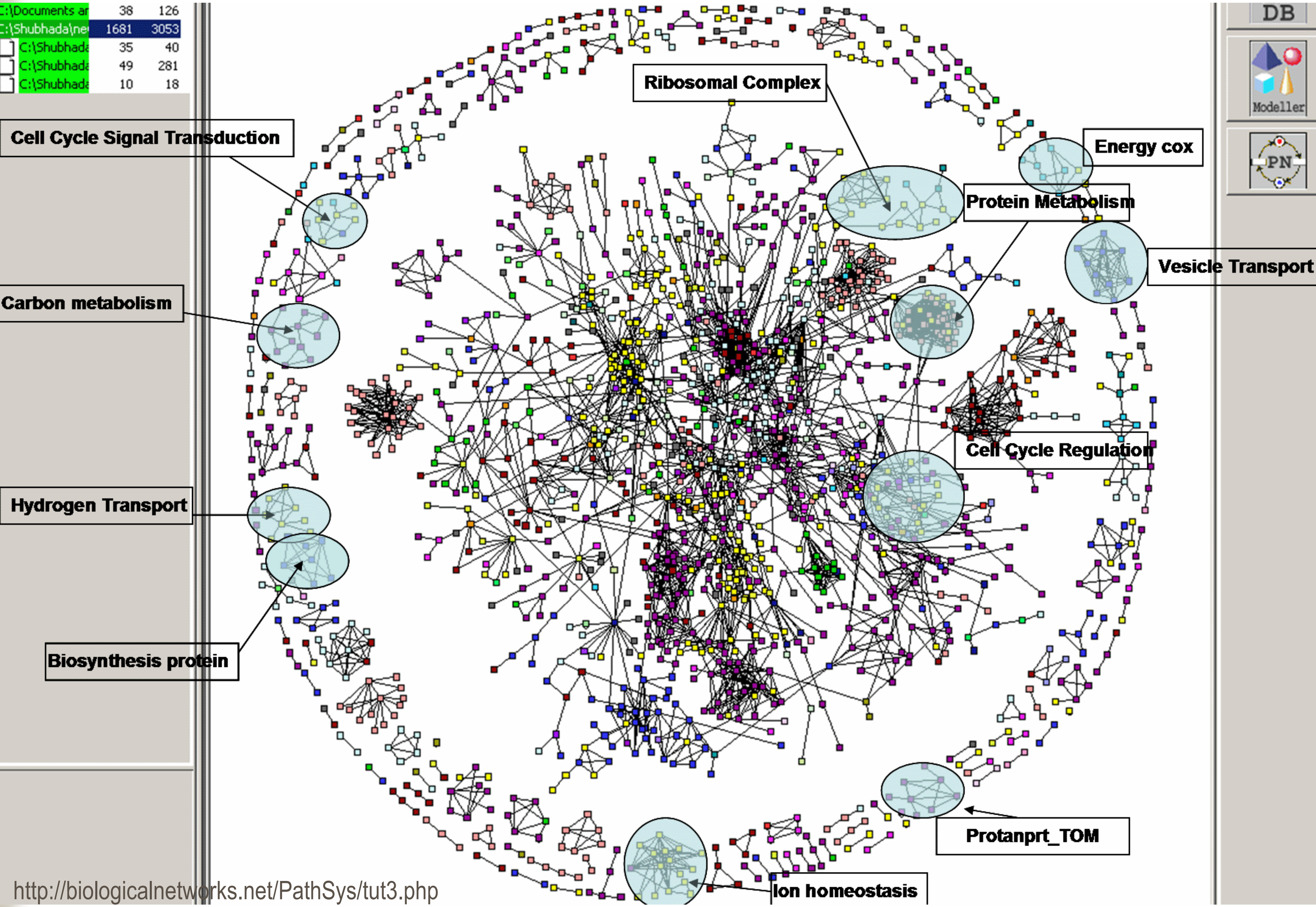


Restored with ICM



Restored with min-cut

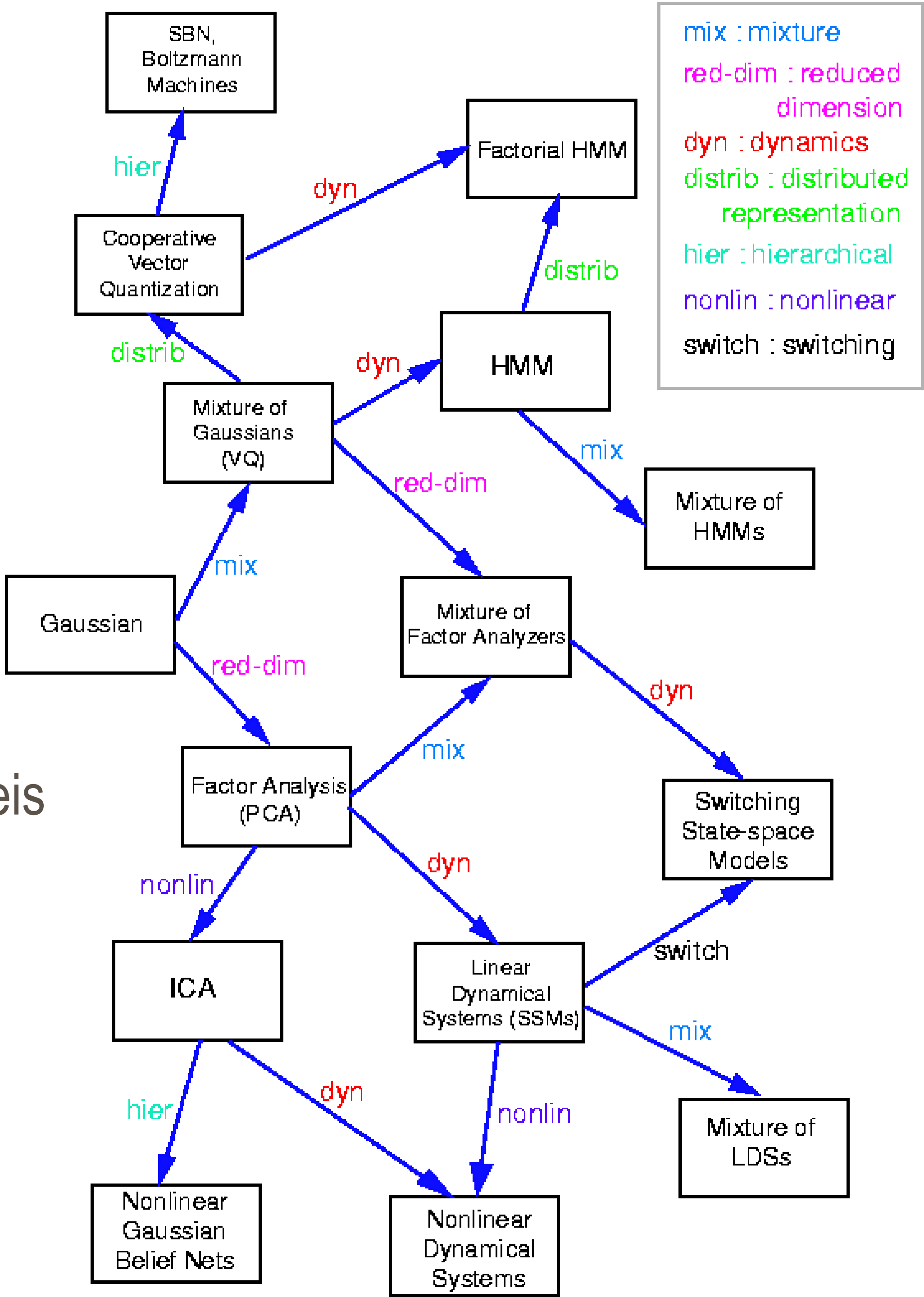
Example: Protein-Protein Interaction Networks



C:\Documents ar	38	126
C:\Shubhada\ne	1681	3053
C:\Shubhada	35	40
C:\Shubhada	49	281
C:\Shubhada	10	18

http://biologicalnetworks.net/PathSys/tut3.php

Genealogy of Graphical Models



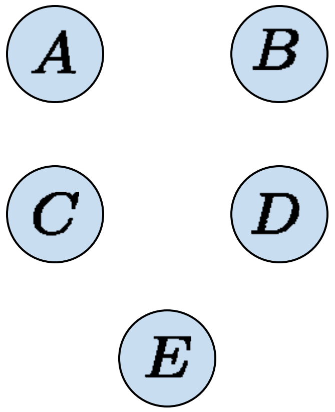
Zoubin Ghahramani and Sam Roweis

http://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html

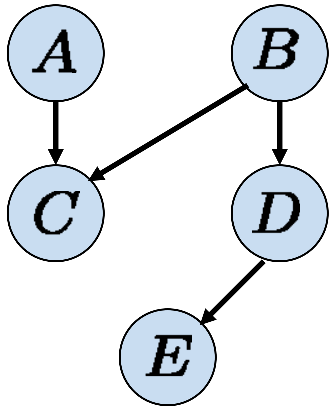
Learning Graphical Models

Task:

- Given training data
- Find “best” DAG and CPD



$$(A, B, C, D, E) = (00110)$$
$$(A, B, C, D, E) = (00010)$$
$$(A, B, C, D, E) = (00111)$$
$$\dots$$
$$(A, B, C, D, E) = (01110)$$



DAG

	A^0B^0	A^0B^1	A^1B^0	A^1B^1
C^0	0.4	1	0.9	0.8
C^1	0.6	0	0.1	0.2

A^0	0.3
A^1	0.7

B^0	0.4
B^1	0.6

Conditional Probability Tables

Bayesian Learning Approach

- Θ= all unknown parameters
  - Graph structure (preferably not)
  - Probabilities
- Bayes Rule $P(\Theta|y_1, ..., y_N) \propto P(y_1, ..., y_N|\Theta)P(\Theta)$
- Find the mean Θ

$$\Theta = \int \Theta P(\Theta|y_1, ..., y_N) d\Theta$$

- Sampling Θ
  - Use MCMC
  - Tricky if graph structure is in Θ
  - Sometimes use uniform P (Θ)

Inference

Graphical Models

- Compact Representation of a probability distribution P

Inference:

- Answer queries about P.
- Examples:
  - Is node X independent on node Y given nodes Z,W ?
  - Probability of X=true if (Y=false and Z=true)=?
  - What is the joint distribution of (X,Y) if Z=false?
  - What is the likelihood when assigning values to all variables?
  - What is the most likely assignment of values to all or a subset the nodes, knowing other nodes?

Query 1: Likelihood

- Some variables have been observed
    - Say the variables are $E = \{x_{k+1}, \dots, X_n\}$
    - They form the evidence
    - Are assigned some value vector e
- Compute probability of the evidence

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

  - Aka likelihood of e
  - Need to integrate all other variables
  - Need to know the partition function (for MRF)

Query 2: Conditional Probability

- Conditional probability distribution of the remaining variables given the evidence

$$P(X|e) = \frac{P(X,e)}{P(e)} = \frac{P(X,e)}{\sum_x P(X=x,e)}$$

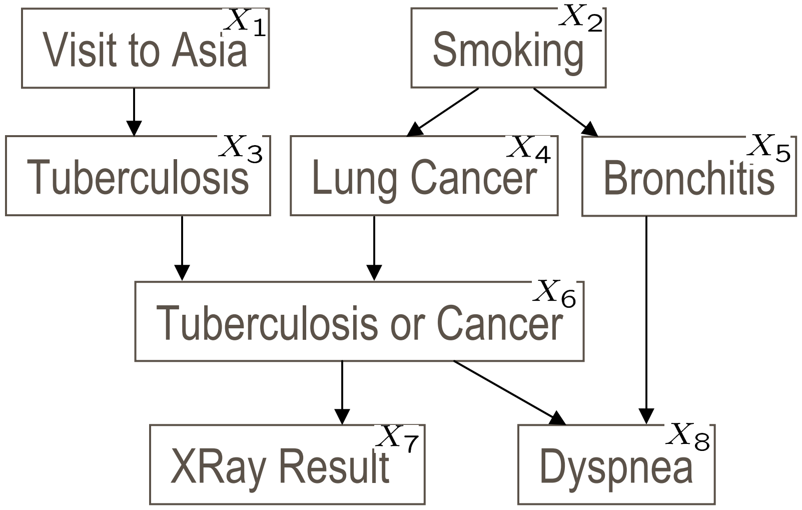
the *a posteriori belief* in X, given evidence e

- If we query a subset Y of the variables X={Y,Z} , we integrate out (don't care about) Z:

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

- integrate out = marginalization

Applications of the A-posteriori Belief



- Prediction:
  - Probability of an outcome given the starting condition
  - The query node is a descendent of the evidence
  - E.g.: Probability of lung cancer, if smoking
- Diagnosis:
  - Probability of disease/fault given symptoms
  - The query node an ancestor of the evidence
  - E.g.: Probability of cancer given X-ray result and dyspneea
- Direction of information flow ≠ direction of edges in GM
  - Inference combines evidence from all parts of the graph

Query 3: Most Probable Assignment

- Find most probable values for variables Y
- Given evidence e
- Ignore the rest Z of variables

$$MPA(Y|e) = \arg\max_y P(y|e) = \arg\max_y \sum_z P(y, z|e)$$

- The maximum a posteriori assignment for Y given e

Applications of MPA

Classification

- Find most likely label, given the evidence

Explanation

- Find the most likely scenario, given the evidence

Observations:

- The MPA of a variable depends on its "context"- the set of variables being jointly queried
- Example:
  - MPA of X = 1
  - MPA of (X, Y) = (0, 0)

X	Y	P(X,Y)
0	0	0.4
0	1	0.05
1	0	0.25
1	1	0.3

Inference Complexity

Theorem:

- For a general GM, computing $P(X = x \mid e)$ is NP-hard.
- We cannot find a general procedure that works efficiently for arbitrary GMs

But

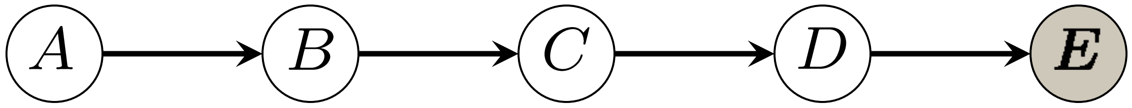
- Approximate (suboptimal) solutions always exist
- For particular families of GMs, there are provably efficient exact procedures

Inference Algorithms

- Exact algorithms
  - The elimination algorithm
  - The junction tree algorithms
  - Efficient for some graphs, otherwise exponential

- Approximate inference techniques
  - Stochastic simulation / sampling methods
  - Markov chain Monte Carlo methods
  - Variational algorithms
  - Belief Propagation
  - Tradeoff speed/accuracy

Variable Elimination on Chains


$$\begin{tikzcd} A \arrow[r] & B \arrow[r] & C \arrow[r] & D \arrow[r] & \boldsymbol{E} \end{tikzcd}$$

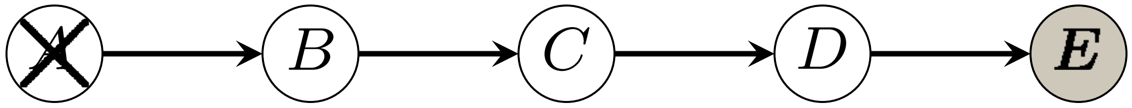
- Say E is observed E=e
- Query: Compute P(e)

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

  - Exponential number of terms
- Using the chain structure we have

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

Variable Elimination on Chains

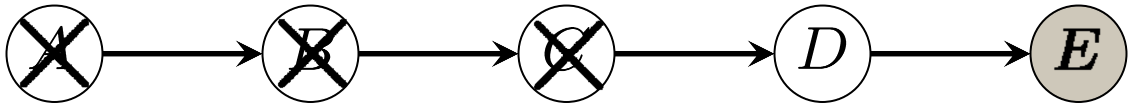


- Actually $\sum_a P(a)P(b|a) = P(b)$
- Compute and memorize all P(b)
- We eliminated A
- We now have

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

- Compute and memorize all $P(c) = \sum_b P(b)P(c|b)$
- We eliminated B

Variable Elimination on Chains



- Repeat the same trick for C and D
  - Compute and memorize all P(c)
  - Then compute and memorize all P(d)
- Now we get what we want

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

- Similar to dynamic programming
  - Save computation by memorization
- Time complexity
  - Each step costs $O(|Val(X_i)| \cdot |Val(X_{i+1})|)$ operations: $O(kn^2)$
  - Brute force= $O(n^k)$ ($n=|Val(X_i)|$, k = chain length)

Variable Elimination on General DGMs

Idea:

- Write the full probability $P(x_1, ..., x_n) = \prod P(x_i|parents(X_i))$
- Integrate out the variables that are not in the query and are not observed

$$P(x_1, e) = \sum_{x_k} ... \sum_{x_2} \prod_i P(x_i|parents(X_i))$$

- Choose a good elimination order
- Iterate
  - Move all irrelevant terms outside of innermost sum
  - Compute innermost sum and memorize its values
  - Insert the new term into the product
- Obtain the final result

$$P(x_1|e) = \frac{P(x_1, e)}{P(e)}$$

Example

- Query: $P(b|h)$
- Full probability

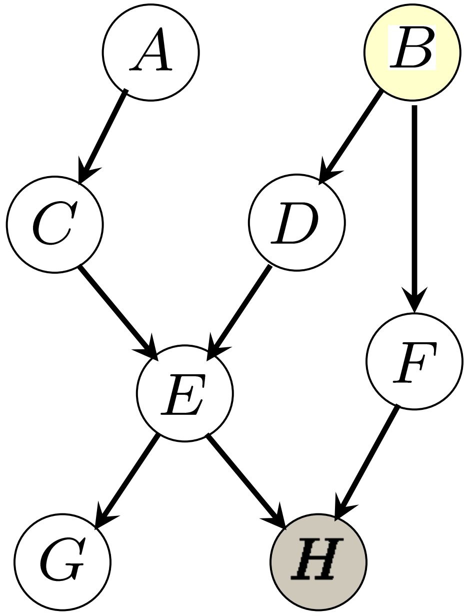
$$P(a,b,c,d,e,f,g,h) = P(a)P(b)P(c|a)$$
$$\cdot P(d|b)P(f|b)P(e|c,d)P(g|e)P(h|e,f)$$

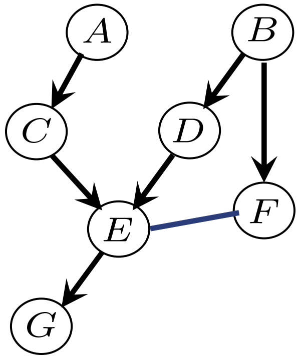
- Need to eliminate A,C,D,E,F,G,H
- Start with H, memorize the marginal

$$m_h(e,f) = P(h_0|e,f)$$

- Obtain

$$P(a,b,c,d,e,f,g,h_0) = P(a)P(b)P(c|a)$$
$$\cdot P(d|b)P(f|b)P(e|c,d)P(g|e)m_h(e,f)$$





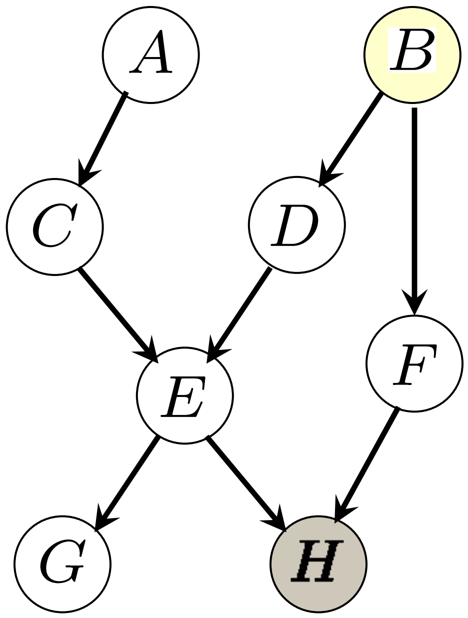
Example

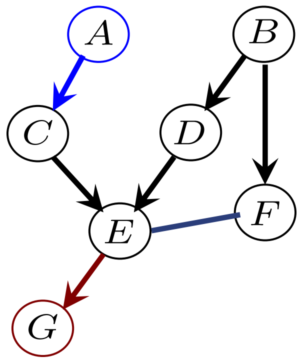
- Memorize $m_g(e) = \sum_g P(g|e)$
- Eliminate G

$$P(a,b,c,d,e,f,h_0) = \sum_g P(a)P(b)P(c|a)$$
$$\cdot P(d|b)P(f|b)P(e|c,d)P(g|e)m_h(e,f)$$
$$= P(a)P(b)P(c|a)P(d|b)P(f|b)$$
$$\cdot P(e|c,d)m_g(e)m_h(e,f)$$

- Memorize $m_a(c) = \sum_a P(c|a)P(a)$
- Eliminate A

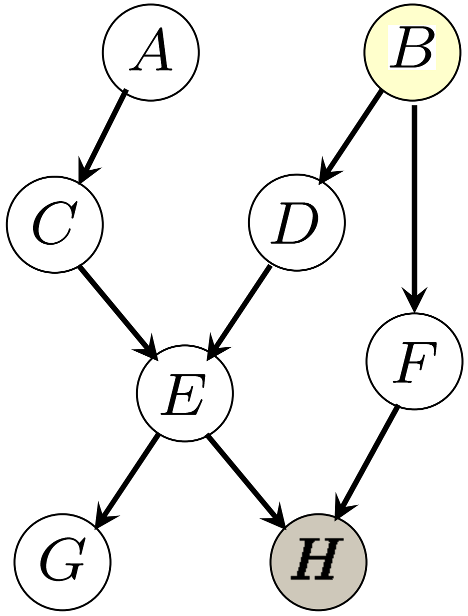
$$P(b,c,d,e,f,h_0) = P(b)P(d|b)P(f|b)$$
$$\cdot P(e|c,d)m_a(c)m_g(e)m_h(e,f)$$





Example

- Memorize $m_f(b,e) = \sum_f P(f|b)m_h(e,f)$

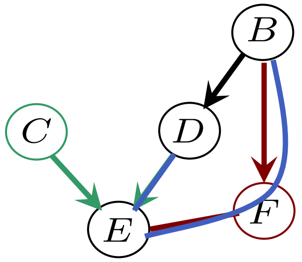


- Eliminate F

$$P(b,c,d,e,h_0) = P(b)P(d|b)P(e|c,d) \cdot m_a(c)m_g(e)m_f(b,e)$$

- Memorize $m_c(d,e) = \sum_c P(e|c,d)m_a(c)$
- Eliminate C

$$P(b,d,e,h_0) = P(b)P(d|b)m_c(d,e)m_g(e)m_f(b,e)$$



Example

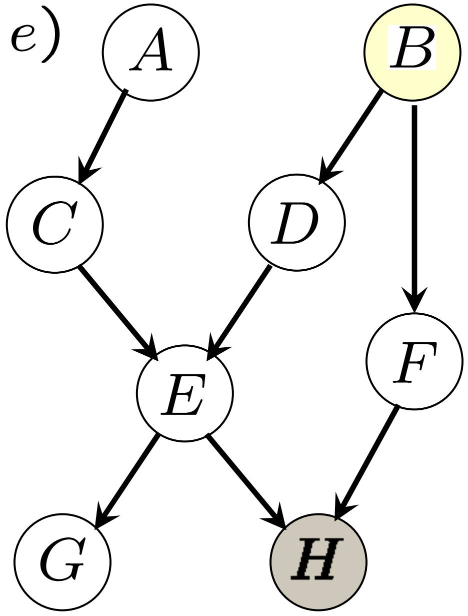
- Memorize $m_e(b,d) = \sum_e m_c(d,e) m_g(e) m_f(b,e)$
- Eliminate E

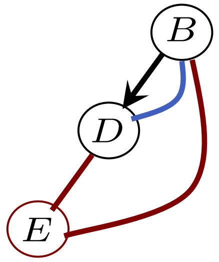
$$P(b,d,h_0) = P(b)P(d|b)m_e(b,d)$$

- Memorize $m_d(b) = \sum_d P(d|b) m_e(b,d)$
- Eliminate D

$$P(b,h_0) = P(b)m_d(b)$$

- Compute $P(h_0) = \sum_b P(b,h_0)$
- Obtain the final result $P(b|h_0) = \dfrac{P(b,h_0)}{P(h_0)}$

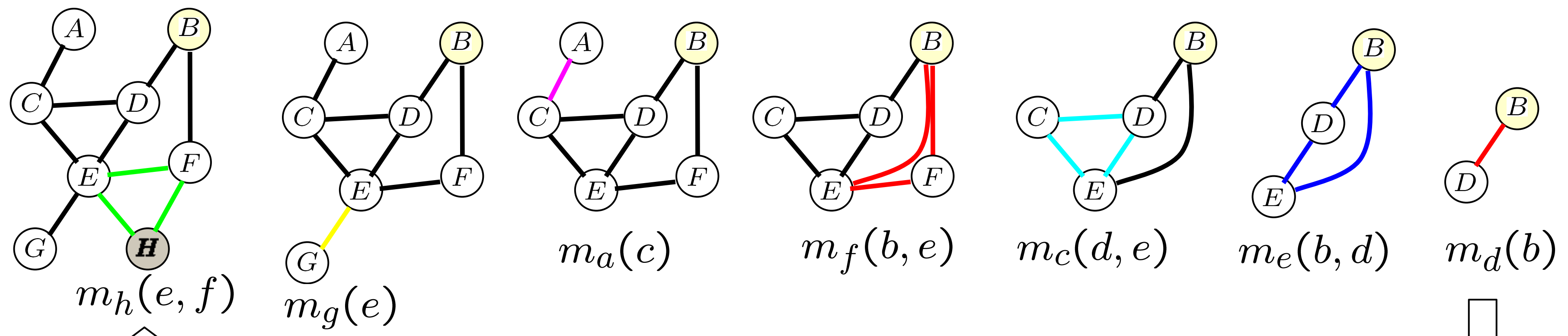




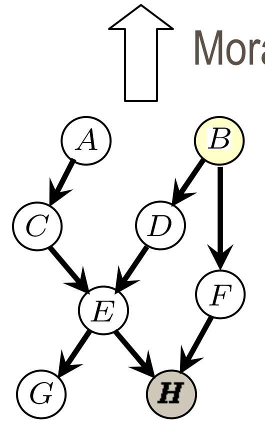
Time Complexity of Variable Elimination

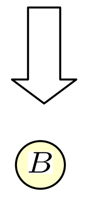
- Look at one elimination step, when we memorize
$$m_x(x_1, ..., x_k) = \sum_x \prod_{i=1}^k m_i(x, x_{c_i})$$
- We compute this for all values of $(x_1, ..., x_k)$
- Memory requirement $\prod_{i=1}^k |Val(x_i)|$
- For each entry, we need $k|Val(x)|$ multiplications and $|Val(x)|$ additions
- Totally
    - $k|Val(x)| \prod_{i=1}^k |Val(x_i)|$ multiplications
    - $|Val(x)| \prod_{i=1}^k |Val(x_i)|$ additions
- Exponential in number of variables in the intermediate factor
    - Prefer elimination orders with few of variables in each factor

Variable Elimination



Moralization=convert to undirected graph by connecting to the Markov Blanket

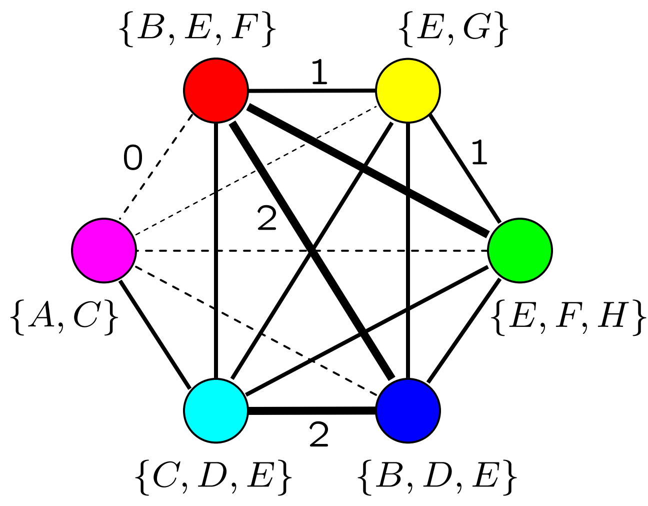




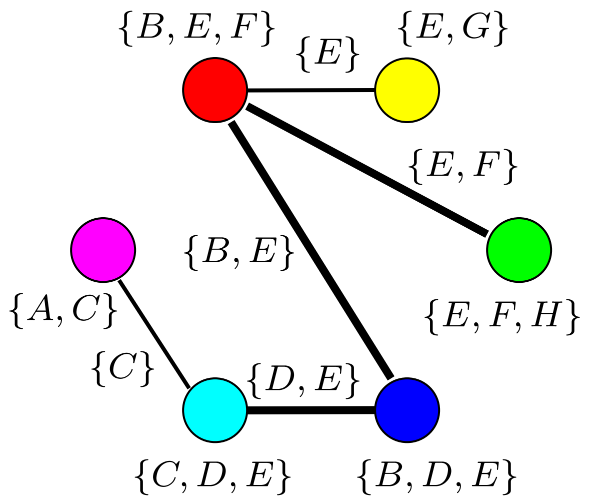
- There are many elimination orders
- Preferable one with small number of variables in each clique (memorized marginal)
- Finding the optimum ordering is NP hard
  - Heuristics can be used
- Also works for undirected GMs

The Junction-Tree

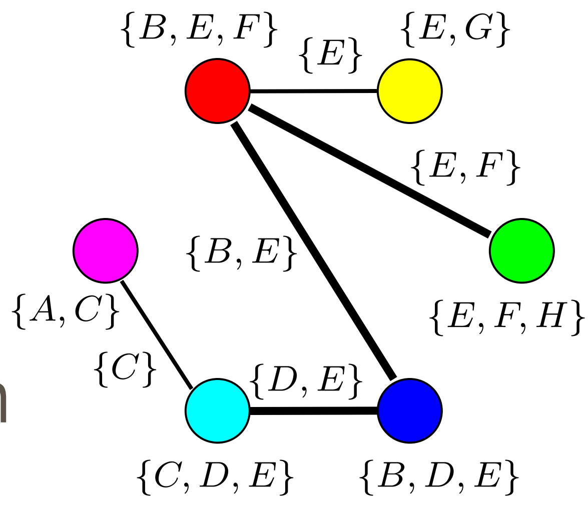
- Works on undirected GM
  - For directed GM, convert them to undirected by moralization



- Build the junction tree:
  1. Choose an ordering of the nodes and use Node Elimination to obtain a set of elimination cliques.
  2. Build a complete cluster graph over the elimination cliques.
  3. Weight each edge {U,V} by |U∩V| and compute a maximum-weight spanning tree.



Properties of the Junction-Tree



- Singly connected: there is exactly one path between each pair of clusters.
- Covering: for each clique C of G there is some cluster node N such that C⊂ N.
- Running intersection: for each pair of clusters B and C containing i, each cluster on the unique path between B and C also contains i.
- Different junction trees are obtained using
  - Different elimination orders
  - Different maximum-weight spanning trees.

Decomposable Probabilities

Definition: A factorized probability

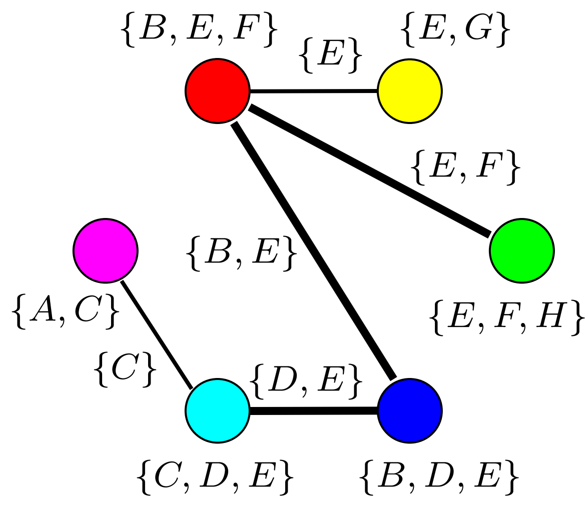
$$P(\mathbf{x}) = P(x_1, ..., x_N) = \frac{1}{Z} \prod_{c \in C} \psi_c(\mathbf{x}_c)$$

is decomposable if there is a junction tree with cluster set C.

- To convert a factorized p to a decomposable probability:
  1. Build a junction tree T for the GM of P.
  2. Initialize the potentials $\psi_c = 1$ for each cluster c∈T.
  3. Multiply each potential ψ of p into the cluster potential ψ_c of one cluster that covers its variables.

Note: This is possible only because of the covering property.

The Junction Tree Algorithm

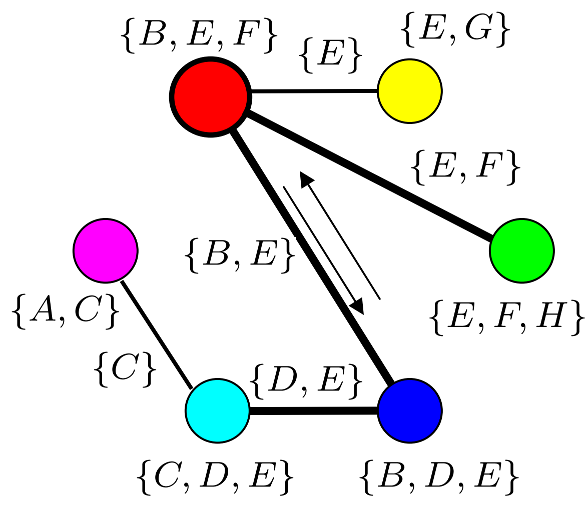


- Input:
  - A decomposable probability
  - The associated junction tree T.
- Output: marginal densities of the cliques

Algorithm:

- Each cluster c∈T knows only its local potential ψ_c and its neighbor clusters.
- Each cluster sends one message (potential function) to each neighbor.
- By combining its local potential with the messages it receives, each cluster is able to compute the marginal density of its variables.
- There are different variants of the algorithm based on what messages are passed

Message Passing



Message passing protocol:

- Cluster B sends a message to a neighbor C only after it has received messages from all neighbors except C.

One version:

- Choose one cluster R to be the root so the junction tree is directed.
- Execute Collect(R) and then Distribute(R)
- Collect(C): For each child B of C,
  1. Recursively call Collect(B)
  2. Pass a message B → C.
- Distribute(C): For each child B of C
  1. Pass a message C → B
  2. Recursively call Distribute(B).

The Shafer–Shenoy Algorithm

- The message from B to C is

$$\mu_{BC}(x_{B\cap C}) = \sum_{x_{B-C}} \psi_B(x) \prod_{A\in\partial B-\{C\}} \mu_{AB}(x_A)$$

  - cluster B computes the product of its local potential ψ_B and the messages from all clusters except C
  - marginalizes out all variables that are not in C
  - sends the result to C.

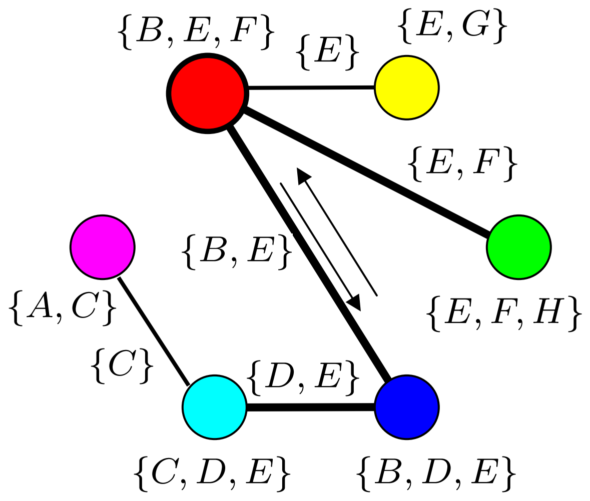
Note: μ_{BC} is well-defined because of the tree structure

- The cluster belief at C is

$$\beta_C(x_C) = \psi_C(x_C) \prod_{B\in\partial C} \mu_{BC}(x_{B\cap C})$$

- After all messages have been passed:

$$\beta_C(x_C) \propto P(x_C)$$



The Hugin Algorithm

- Maintain potential functions for each node C and edge E of the Junction Tree. Initialization:

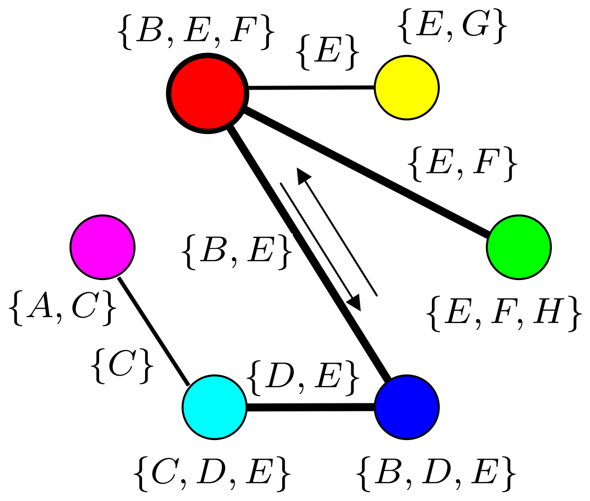
$$\phi_C(x_C) = \psi_C(x_C),$$
$$\phi_E(x_E) = 1$$

- To pass a message from B to C over edge E, update

$$\phi_E^o(x_E) = \phi_E(x_E),$$
$$\phi_E(x_E) = \sum_{x_{B-E}} \phi_B(x_E, x_{B-E}),$$
$$\phi_C(x_C) = \phi_C(x_C)\frac{\phi_E(x_E)}{\phi_E^o(x_E)}$$

- After all messages have been passed:

$$\phi_C(x_C) \propto P(x_C)$$



Complexity of Junction Tree Algorithms

Junction tree algorithms memorize, multiply, and marginalize potentials:

	Tabular	Gaussian
storing ψ_C	$O(k^{\|C\|})$	$O(\|C\|^2)$
storing ϕ_E	$O(k^{\|E\|})$	$O(\|E\|^2)$
updating ϕ_E from B to C	$O(k^{\|B-E\|})$	$O(\|B-E\|^3\|B\|^2)$

- number of messages ~ number of clusters O(|V |).

Thus:

- The time and space complexity is dominated by the size of the largest cluster in the junction tree, named the **width** of the junction tree:
  - When using tables: complexity = exponential in the width.
  - When using Gaussians: complexity = cubic in the width.

Conclusion: Junction Tree

A **generic** exact inference algorithm for any GM

Algorithm

- Construct junction tree: a special **clique tree**
- Propagate probabilities - a message-passing protocol

Output:

- Marginal probabilities of all cliques
- Solves all queries in a single run

Complexity:

- exponential in the size of the maximal clique
- a good elimination order often leads to small maximal clique

Many well-known algorithms are special cases of Junction Trees

- Forward-backward, Kalman filter, Sum-Product ...

Inference Algorithms

- Exact inference:
  - The elimination algorithm
    - Obtains one marginal probability
    - Fast on chains and trees
    - Otherwise NP hard
  - The junction tree algorithms
    - Obtains all marginal probabilities
    - Still NP hard in general
- Approximate inference
  - Stochastic simulation / sampling
  - Markov chain Monte Carlo
  - Variational algorithms

Monte Carlo Algorithms

Overview

- Draw random samples from the desired distribution
  - A stochastic representation of a complex distribution
- Marginals and other expectations can be approximated by sample-based averages

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

- *Asymptotically* exact and easy to apply to arbitrary models

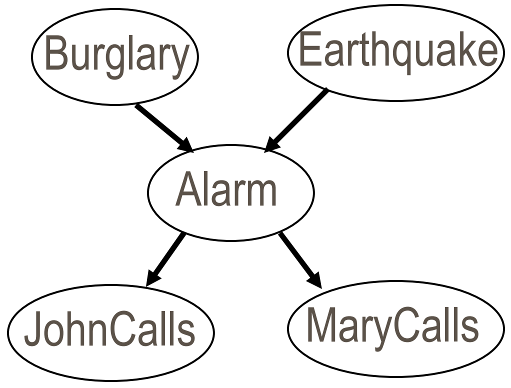
Challenges:

- Sampling from a given distribution
  - Many distributions are hard to sample (e.g. MRF)
- How to make better use of the samples
  - Not all sample are equally useful
- How to know when to stop sampling

Example: Naïve Sampling

B^1	0.001

E^1	0.002



	B^0E^0	B^0E^1	B^1E^0	B^1E^1
A^1	0.001	0.29	0.94	0.95

	A^0	A^1
J^1	0.05	0.9

	A^0	A^1
M^1	0.01	0.7

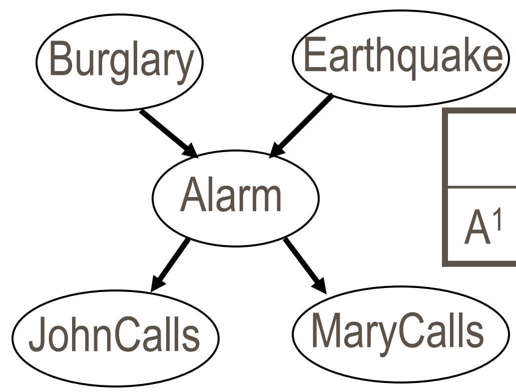
- Sampling: Construct samples according to probabilities given in a Bayesian Network .

Alarm example: (Choose the right sampling sequence)

1) Sample from P(B)=<0.001, 0.999>, say we get B^0.
2) Sample E, say we get E^0
3) Sample $P(A|B^0, E^0)$=<0.001, 0.999>, say we get A^0.

…

Example: Naïve Sampling



B^1	0.001

E^1	0.002

	B^0E^0	B^0E^1	B^1E^0	B^1E^1
A^1	0.001	0.29	0.94	0.95

	A^0	A^1
J^1	0.05	0.9

	A^0	A^1
M^1	0.01	0.7

- By repeating the process we get many samples:
  - $E^0B^0A^0M^0J^0$
  - $E^0B^0A^0M^0J^0$
  - $E^0B^0A^0M^0J^1$
  - $E^0B^0A^0M^0J^0$
  - $E^1B^0A^1M^1J^1$
  - $E^0B^0A^0M^0J^0$

Frequency counting: From the samples we get

$P(J|A^0) = P(J, A^0)/P(A^0) =< 1/5, 4/5 > .$

Problems:

- What if we want P(J|A¹)? We have only one sample

  $P(J|A^1) = P(J, A^1)/P(A^1) =< 0, 1 > .$

- What about $P(J|B^1)$? No samples available

Monte Carlo Methods

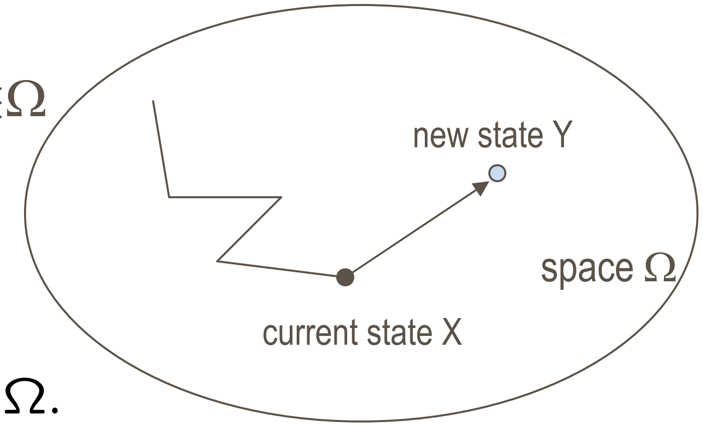
- Direct Sampling
  - Obtain samples from the GM directly.
  - Very difficult to populate a high-dimensional state space
- Rejection Sampling
  - Create samples like direct sampling,
  - Only count samples which are consistent with the given evidence.
- Markov chain Monte Carlo (MCMC)

Markov chain Monte Carlo (MCMC) algorithms

MCMC:

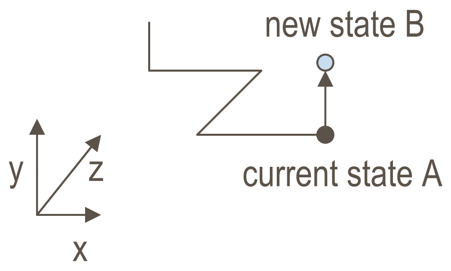
- Iterative stochastic algorithms
- Sample a probability p(X) defined for $X \in \Omega$
- Jump in a new state Y depending on the current state X with probability K(X,Y)
- Reversibility:

$$p(X)K(X,Y) = p(Y)K(Y,X), \ \forall X,Y \in \Omega.$$



Gibbs sampler:

- MCMC algorithm
- Changes one variable X_k at a time by sampling from the marginal probability $p(X_k|X_1,..,X_{k-1},X_{k+1},\dots,X_n)$

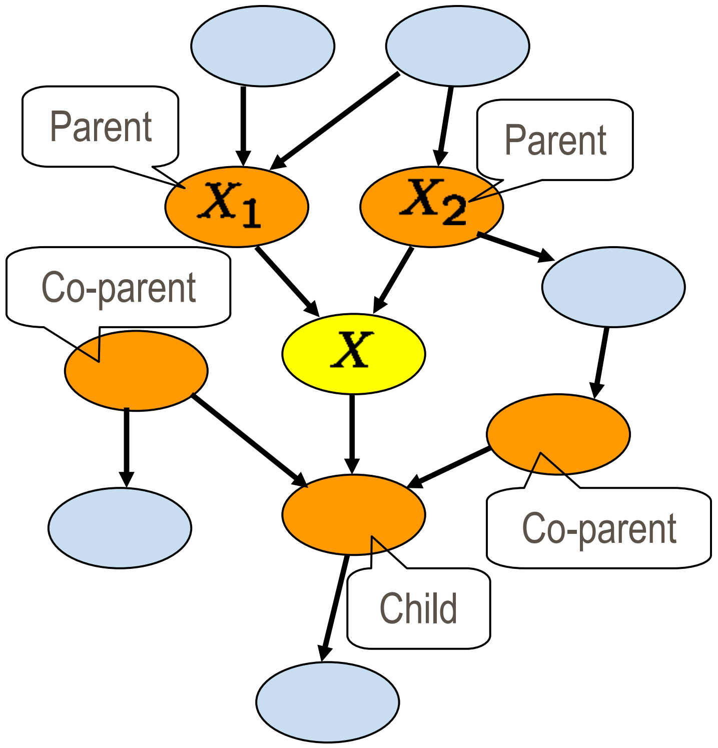


MCMC

Markov-Blanket:

- A node is independent from other nodes, given its parents, children and children‘s parents.

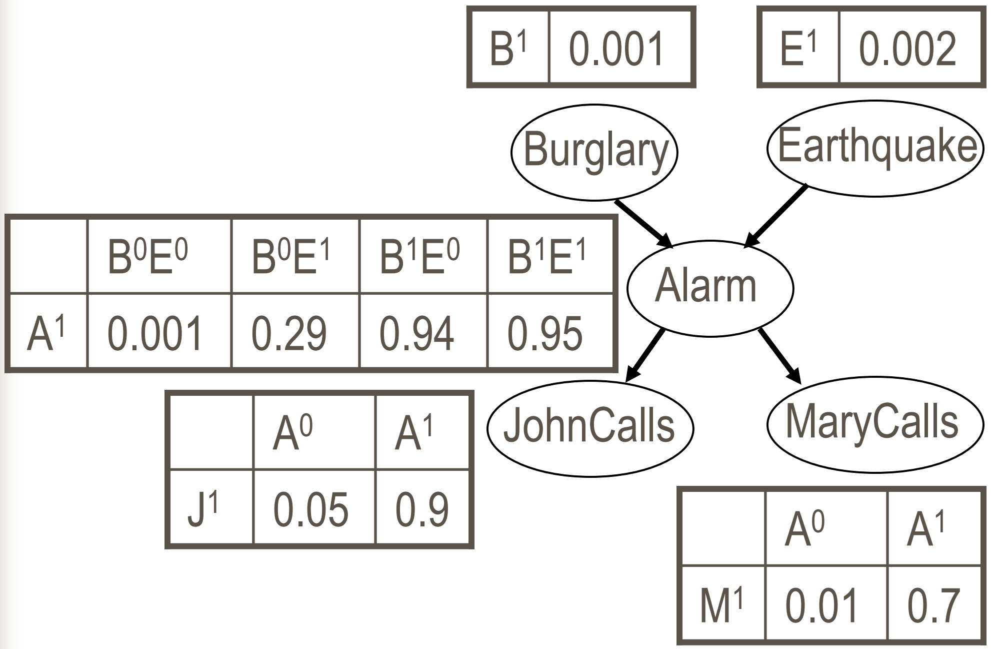
$$p(X_i|X_{-i}) = p(X_i|MB(X_i))$$



Gibbs sampling

- Create a random sample.
- At each step, choose one variable X_i and sample it from $p(X_i|MB(X_i))$ based on the values of the other variables.
- E.g. MB(A)={B, E, J, M}, MB(E)={A, B}

MCMC Example



B^1	0.001

E^1	0.002

	B^0E^0	B^0E^1	B^1E^0	B^1E^1
A^1	0.001	0.29	0.94	0.95

	A^0	A^1
J^1	0.05	0.9

	A^0	A^1
M^1	0.01	0.7

Want to calculate $P(J|B^1,M^1)$

- Evidence is B^1, M^1 (fixed),
- Variables are A, E, J.
- Start from $B^1E^0A^1M^1J^1$
- Randomly choose next variable (say A)
- Sample A from

$$p(A|MB(A)) = p(A|B^1, E^0, M^1, J^1)$$

  say it is 0.
- New state $B^1E^0A^0M^1J^1$
- Choose next random variable (say E), sample

$$E \sim p(E|MB(E)) = p(E|B^1, A^0)$$

...

- Obtain $P(J|B^1,M^1)$ in two ways:
  - Frequency of J=1 during sampling
  - Average of $P(J|A)$ during sampling (faster convergence)

Complexity of MCMC for GM

- Tradeoff speed-accuracy
- It will never reach the true probability, but
  - It will approximate it better and better
  - In an infinite amount of time, it will give the exact solution
- For large and complex graphs, it is much faster than exact inference, since only samples conditionally on the Markov blanket.

References

- M. I. Jordan. Graphical models. *Statistical Science* (Special Issue on Bayesian Statistics), 19, 140-155, 2004.
- http://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html
- http://ai.stanford.edu/~paskin/gm-short-course/lec3.pdf