Probabilistic Relational Models

—Learning Bayesian Networks from Data

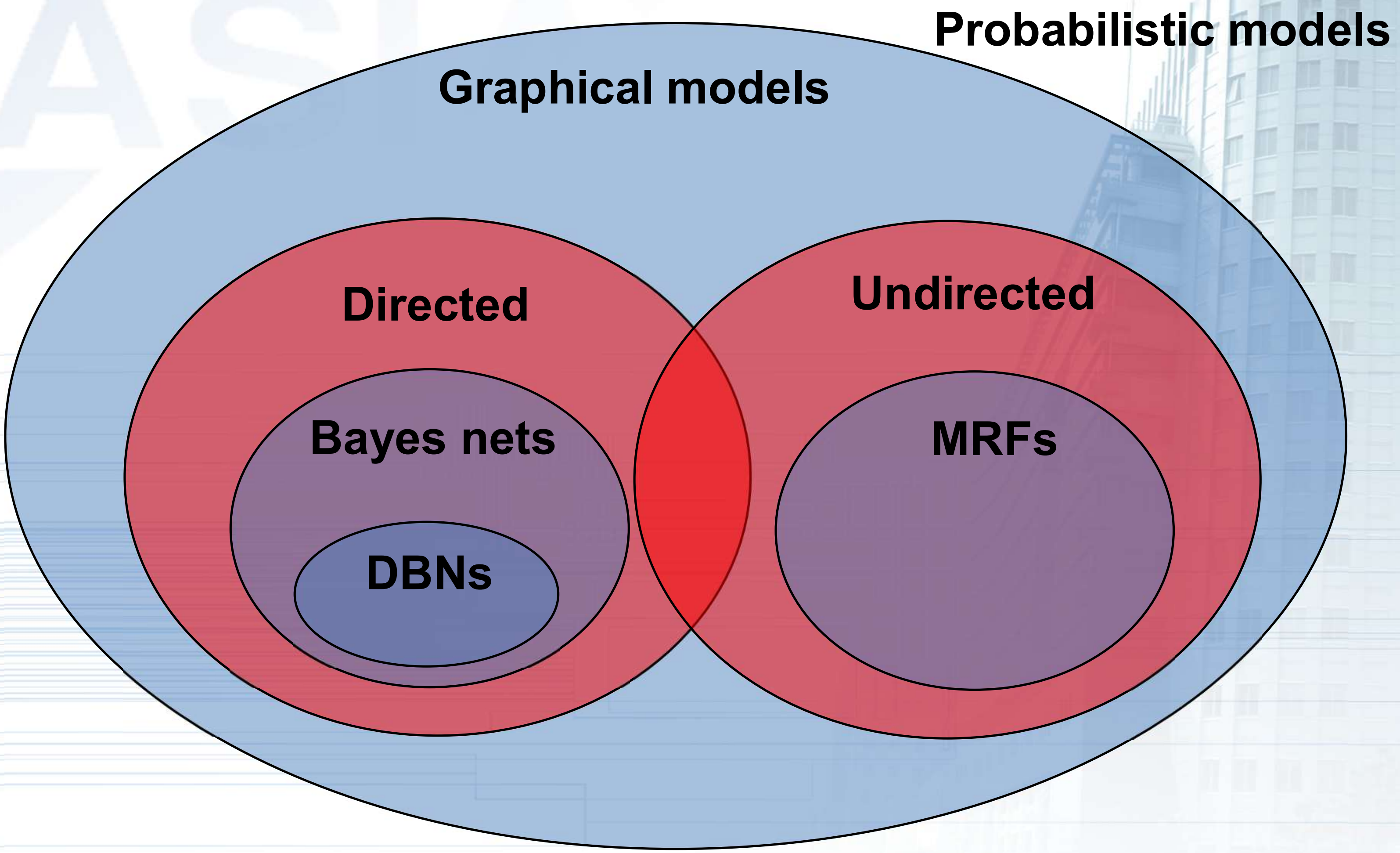
Sun Zhengya Zhang Wensheng

Institute of Automation, Chinese Academy of Sciences

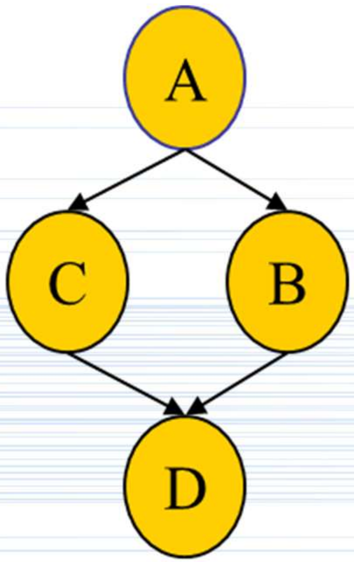
College of AI, University of Chinese Academy of Sciences

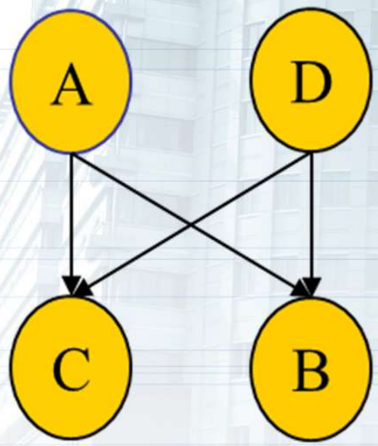
Overview

- **Introduction**
- Parameter Learning
  - Complete Data
  - Incomplete Data
- Structure Learning
  - Complete Data
  - Incomplete Data


Probabilistic models
Graphical models
Directed
Undirected
Bayes nets
MRFs
DBNs

- some structures cannot be represented in a BN
  - Independencies in P: Ind(A;D | B,C), and Ind(B;C | A,D)





Ind(B;C | A,D) does not hold

Ind(A,D) also holds

Bayesian Networks

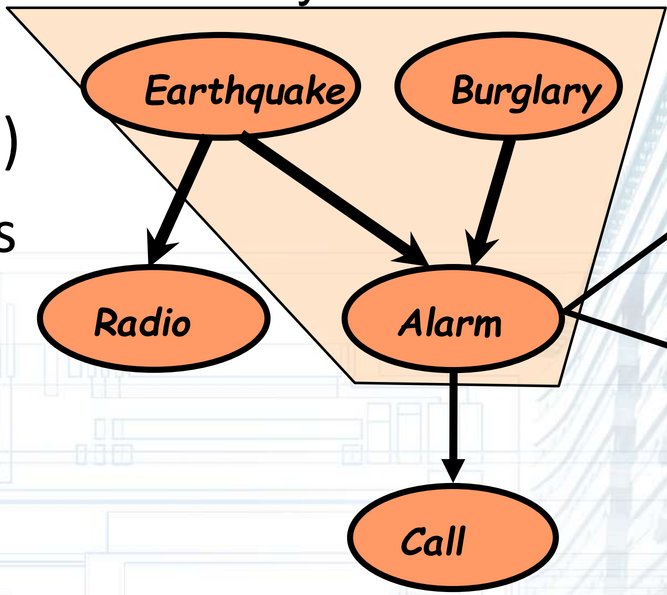
Compact representation of probability distributions via conditional independence

Qualitative part:

Directed acyclic graph (DAG)

- Nodes - random variables
- Edges - direct influence

Family of ***Alarm***



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

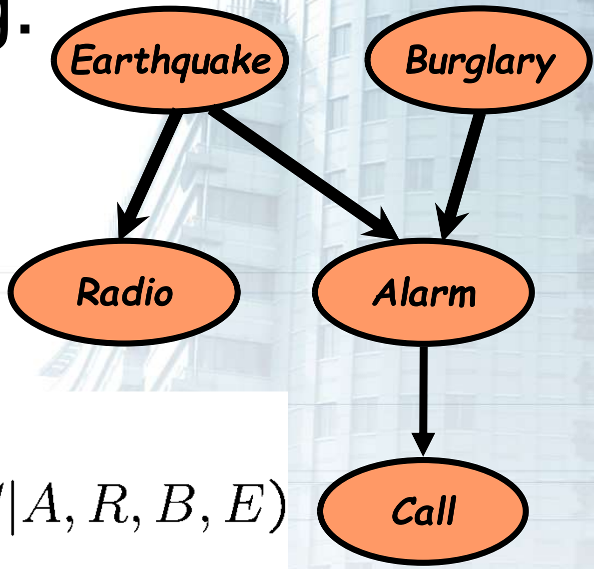
Together:
Define a unique distribution in a factored form

Quantitative part:
Set of conditional probability distributions

$$P(B,E,A,C,R) = P(B)P(E)P(A \mid B,E)P(R \mid E)P(C \mid A)$$

A node is conditionally independent of its ancestors given its parents, e.g.

$$C \perp R, B, E \mid A$$



Hence,

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

Conditional Independence

a is independent of b given c

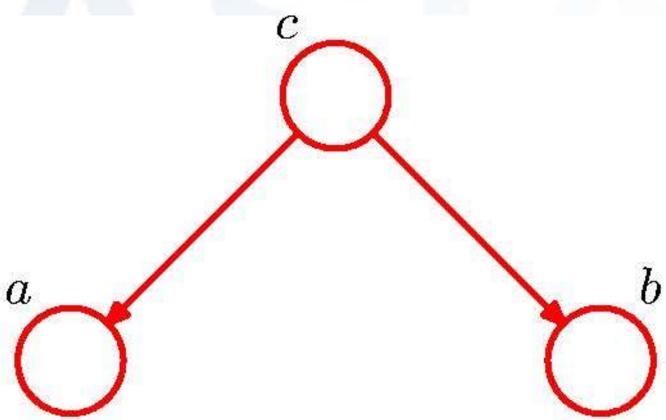
$$p(a|b,c) = p(a|c)$$

Equivalently

$$\begin{aligned} p(a,b|c) &= p(a|b,c)p(b|c) \\ &= p(a|c)p(b|c) \end{aligned}$$

Notation

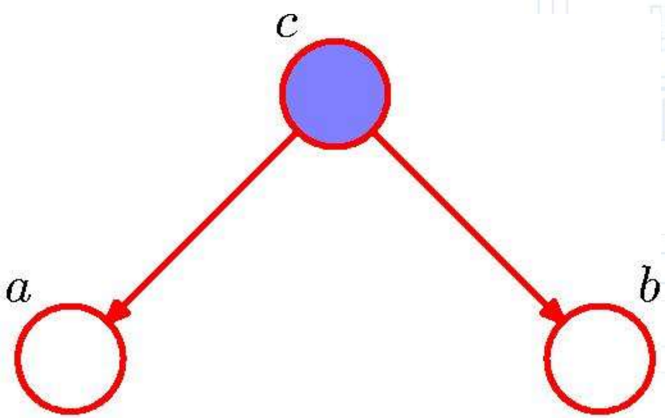
$$a \perp\!\!\!\perp b \mid c$$



$$p(a,b,c) = p(a|c)p(b|c)p(c)$$

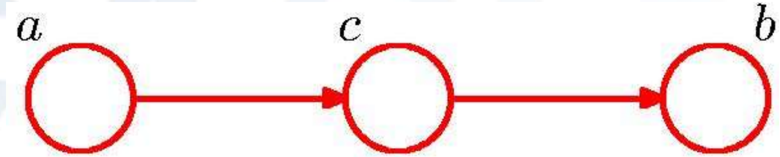
$$p(a,b) = \sum_c p(a|c)p(b|c)p(c)$$

$$a \not\perp\!\!\!\perp b \mid \emptyset$$



$$\begin{aligned} p(a,b|c) &= \frac{p(a,b,c)}{p(c)} \\ &= p(a|c)p(b|c) \end{aligned}$$

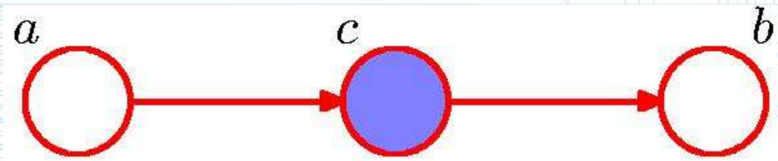
$$a \perp\!\!\!\perp b \mid c$$



$$p(a,b,c) = p(a)p(c|a)p(b|c)$$

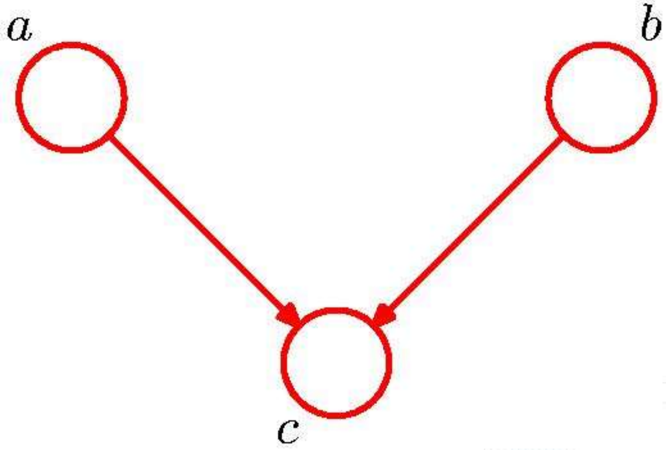
$$p(a,b) = p(a)\sum_c p(c|a)p(b|c) = p(a)p(b|a)$$

$$a \not\perp\!\!\!\perp b \mid \emptyset$$



$$\begin{aligned} p(a,b|c) &= \frac{p(a,b,c)}{p(c)} \\ &= \frac{p(a)p(c|a)p(b|c)}{p(c)} \\ &= p(a|c)p(b|c) \end{aligned}$$

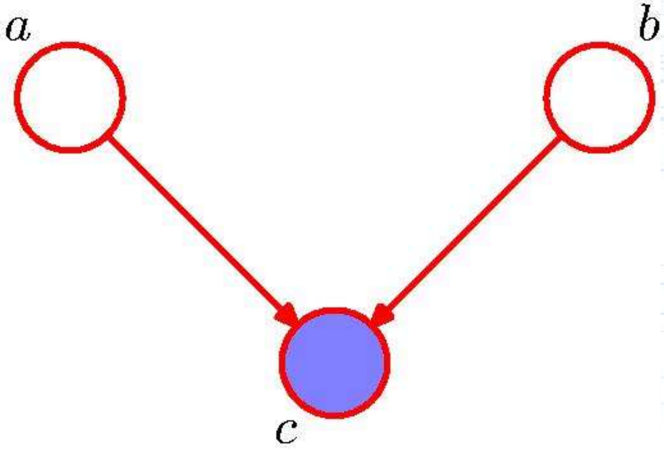
$$a \perp\!\!\!\perp b \mid c$$



$$p(a,b,c) = p(a)p(b)p(c|a,b)$$

$$p(a,b) = p(a)p(b)$$

$$a \perp\!\!\!\perp b \mid \emptyset$$



$$\begin{aligned} p(a,b|c) &= \frac{p(a,b,c)}{p(c)} \\ &= \frac{p(a)p(b)p(c|a,b)}{p(c)} \end{aligned}$$

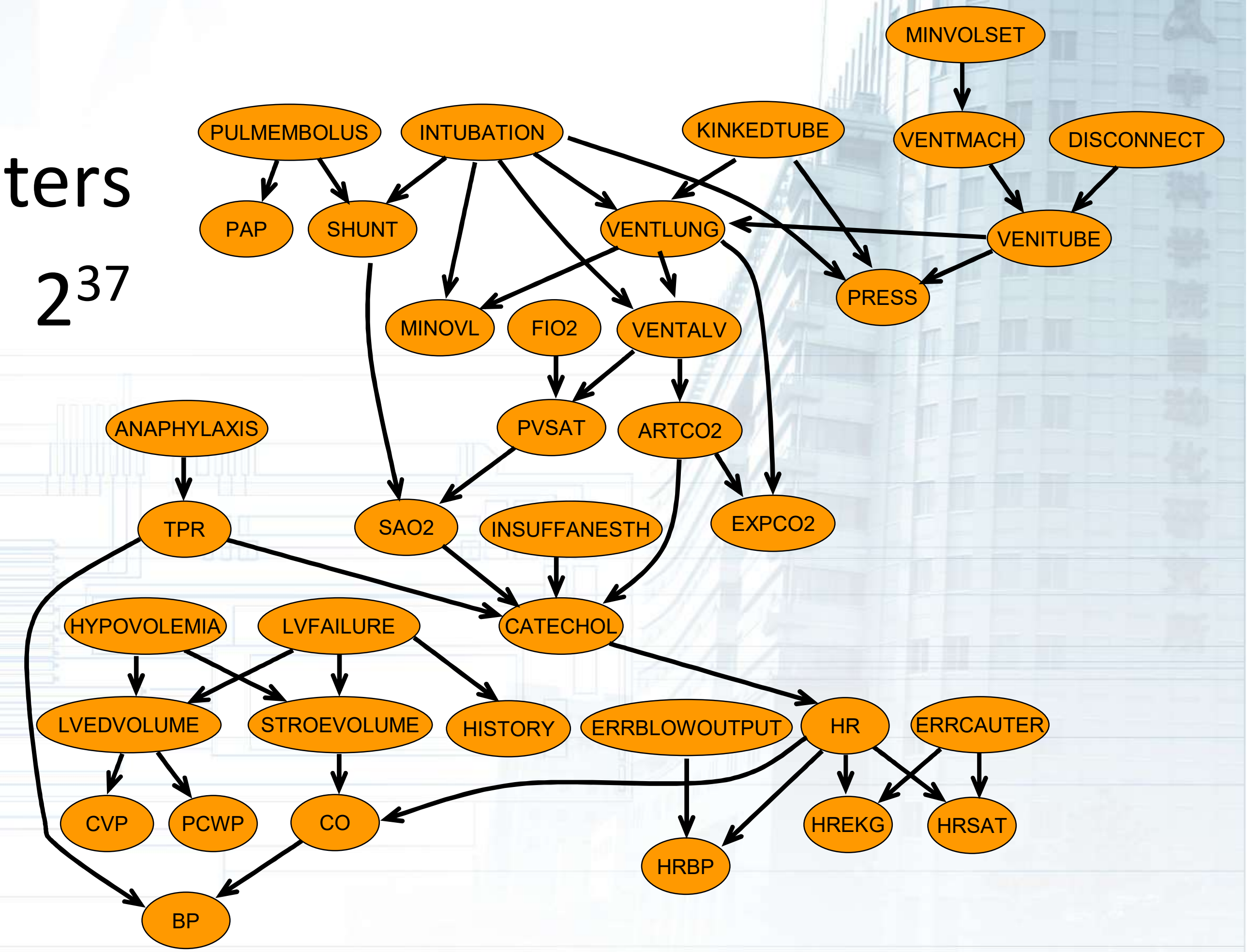
$$a \not\perp\!\!\!\perp b \mid c$$

Example: “ICU Alarm” network

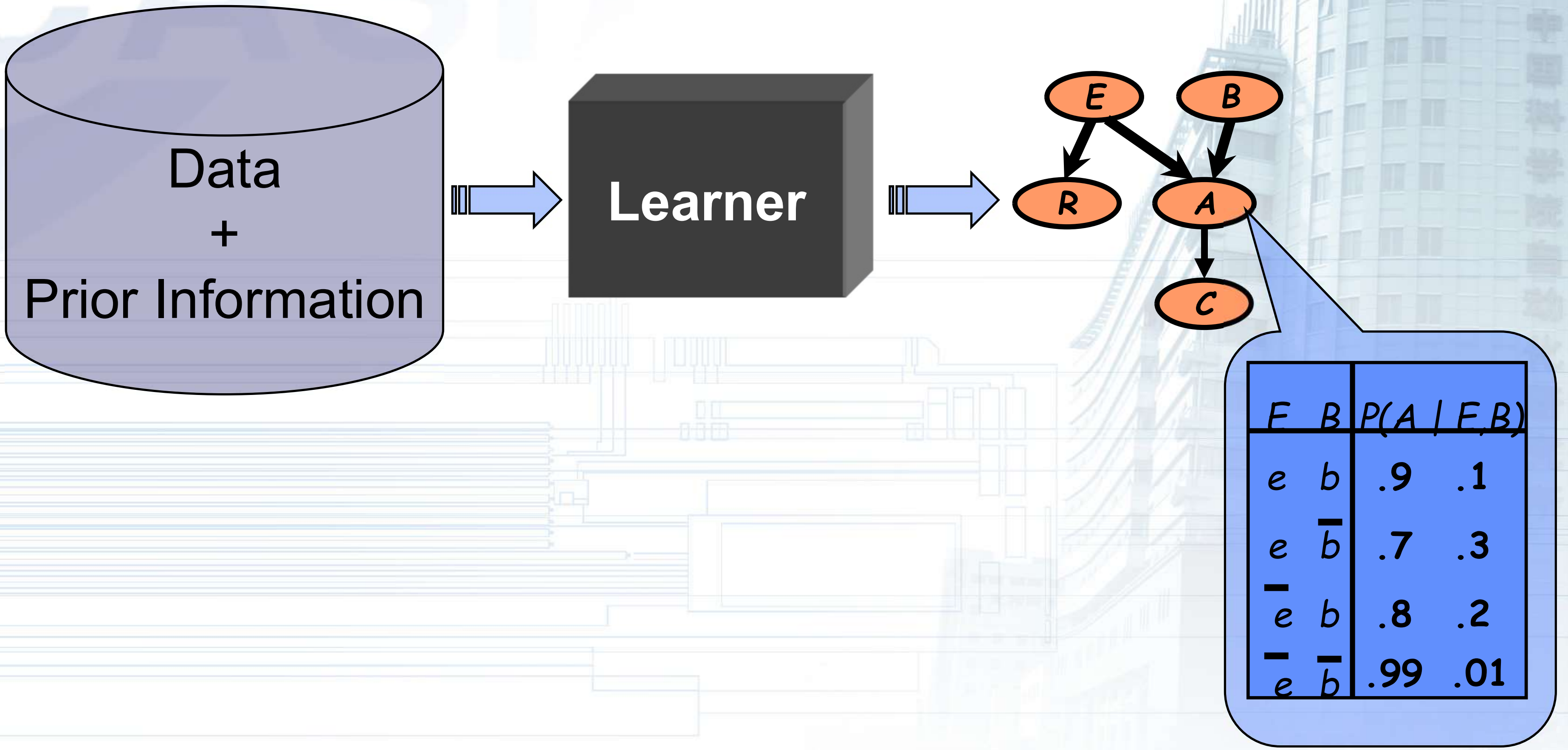
Domain: Monitoring Intensive-Care Patients

- 37 variables
- 509 parameters

  …instead of 2^{37}



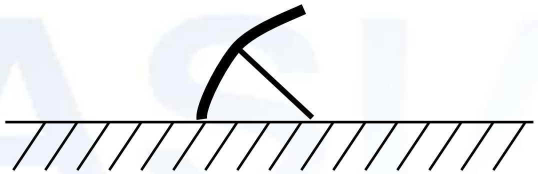
Learning Bayesian networks



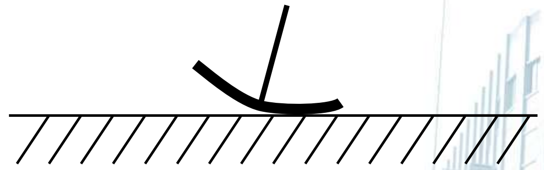
Overview

- Introduction
- **Parameter Learning**
  - **Complete Data**
  - Incomplete Data
- Structure Learning
  - Complete Data
  - Incomplete Data

Example: Binomial Experiment



Head



Tail

When tossed, it can land in one of two positions: *Head* or *Tail*

We denote by θ the (unknown) probability *P(H).*

Estimation task:

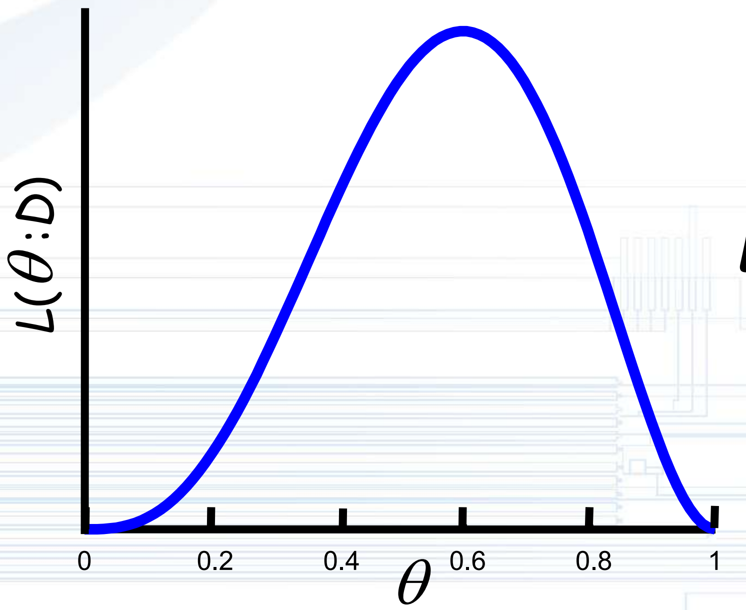
Given a sequence of toss samples *x[1], x[2], …, x[M]* we want to estimate the probabilities $P(H) = \theta$ and $P(T) = 1 - \theta$

Likelihood Function: Multinomials

中国科学院
自动化研究所
INSTITUTE OF AUTOMATION
CHINESE ACADEMY OF SCIENCES

$$L(\theta : D) = P(D \mid \theta) = \prod_m P(x[m] \mid \theta)$$

- The likelihood for the sequence H,T, T, H, H is



$$L(\theta : D) = \theta \cdot (1-\theta) \cdot (1-\theta) \cdot \theta \cdot \theta$$

General case: $$L(\Theta : D) = \prod_{k=1}^{K} \theta_k^{N_k}$$

Count of k^th outcome in D

Probability of k^th outcome

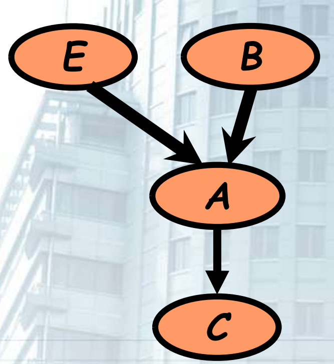
Maximum Likelihood Estimation

- **Consistent**
  - Estimate converges to best possible value as the number of examples grow
- **Asymptotic efficiency**
  - Estimate is as close to the true value as possible given a particular training set
- **Representation invariant**
  - A transformation in the parameter representation does not change the estimated probability distribution

Burglary Example: Parameter Learning

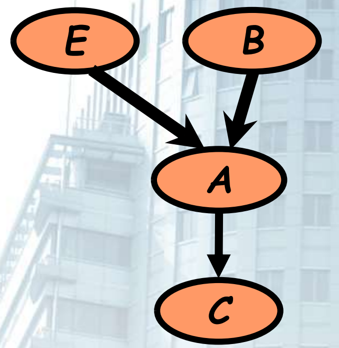
- Training data has the form:

$$D = \begin{bmatrix} E[1] & B[1] & A[1] & C[1] \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ E[M] & B[M] & A[M] & C[M] \end{bmatrix}$$



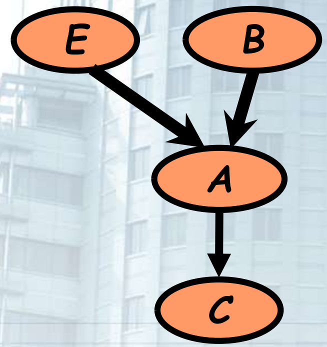
Burglary Example: Likelihood Function

- Assume i.i.d. samples
- Likelihood function is



$$L(\Theta : D) = \prod_m P(E[m], B[m], A[m], C[m] : \Theta)$$

- By definition of network, we get

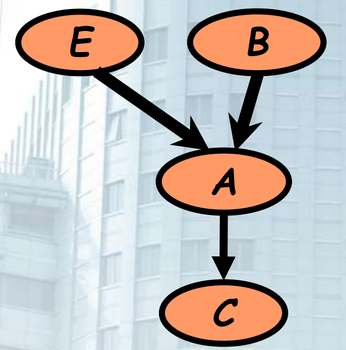


$$L(\Theta : D) = \prod_m P(E[m], B[m], A[m], C[m] : \Theta)$$

$$= \prod_m \begin{pmatrix} P(E[m] : \Theta) \\ P(B[m] : \Theta) \\ P(A[m] \mid B[m], E[m] : \Theta) \\ P(C[m] \mid A[m] : \Theta) \end{pmatrix}$$

$$\begin{bmatrix} E[1] & B[1] & A[1] & C[1] \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ E[M] & B[M] & A[M] & C[M] \end{bmatrix}$$

- Rewriting terms, we get



$$
\begin{aligned}
L(\Theta : D) &= \prod_m P(E[m], B[m], A[m], C[m] : \Theta) \\
&= \prod_m P(E[m] : \Theta) \\
&\quad \prod_m P(B[m] : \Theta) \\
&\quad \prod_m P(A[m] \mid B[m], E[m] : \Theta) \\
&\quad \prod_m P(C[m] \mid A[m] : \Theta)
\end{aligned}
$$

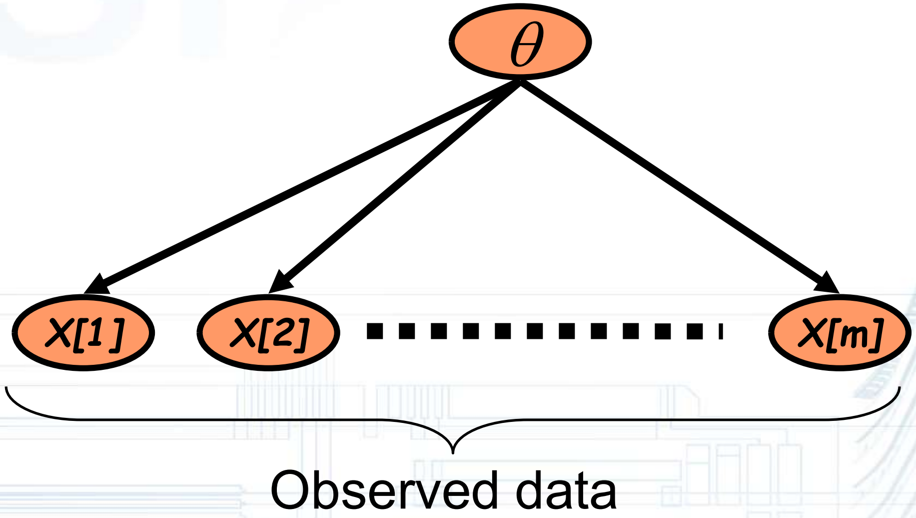
$$
\begin{bmatrix}
E[1] & B[1] & A[1] & C[1] \\
\cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot \\
E[M] & B[M] & A[M] & C[M]
\end{bmatrix}
$$

Bayesian Inference

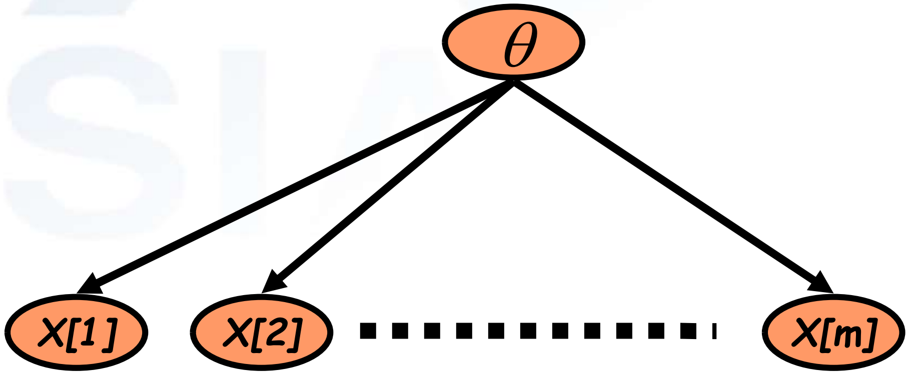
- Represents uncertainty about the unknown parameter
- Uses probability to quantify this uncertainty:
  - Unknown parameters as <u>random variables</u>
- Prediction follows from the rules of probability:
  - Expectation over the unknown parameters

$$\underbrace{P(\theta \mid x[1], \ldots x[M])}_{\text{Posterior}} = \frac{\overbrace{P(x[1], \ldots x[M] \mid \theta)}^{\text{Likelihood}} \overbrace{P(\theta)}^{\text{Prior}}}{\underbrace{P(x[1], \ldots x[M])}_{\text{Probability of data}}}$$

- We can represent our uncertainty about the sampling process using a Bayesian network



  - The observed values of *X* are independent given θ
  - The conditional probabilities, $P(x[m] \mid \theta)$, are the parameters in the model
  - Prediction is now inference in this network



Prediction as **inference** in this network

$$P(x[M+1] \mid x[1],\ldots,x[M])$$

$$= \int P(x[M+1] \mid \theta, x[1],\ldots,x[M]) P(\theta \mid x[1],\ldots,x[M]) d\theta$$

$$= \int P(x[M+1] \mid \theta) P(\theta \mid x[1],\ldots,x[M]) d\theta$$

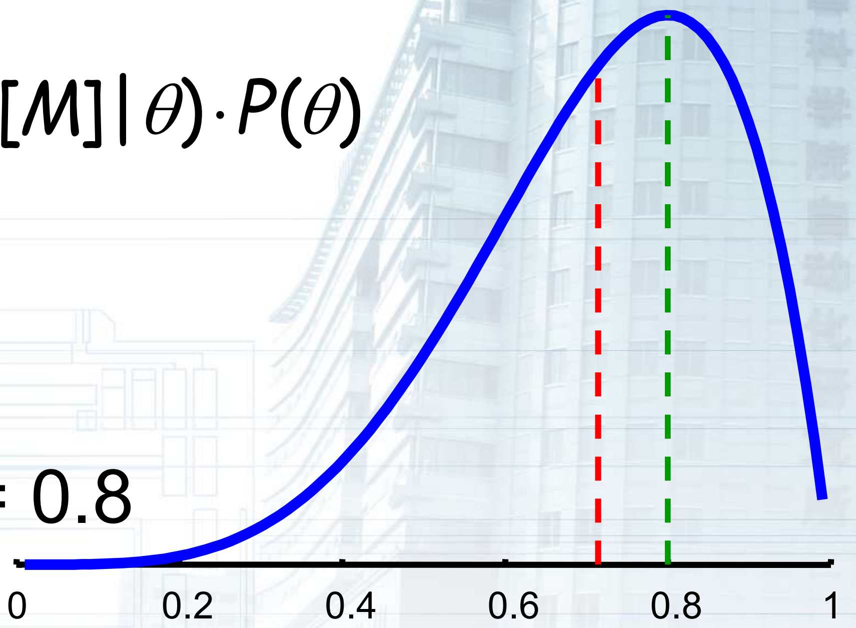
Example: Binomial Data

- Prior: uniform for θ in [0,1]

  $\Rightarrow$ $P(\theta|D) \propto$ the likelihood $L(\theta:D)$

$$P(\theta \mid x[1],\ldots x[M]) \propto P(x[1],\ldots x[M] \mid \theta) \cdot P(\theta)$$

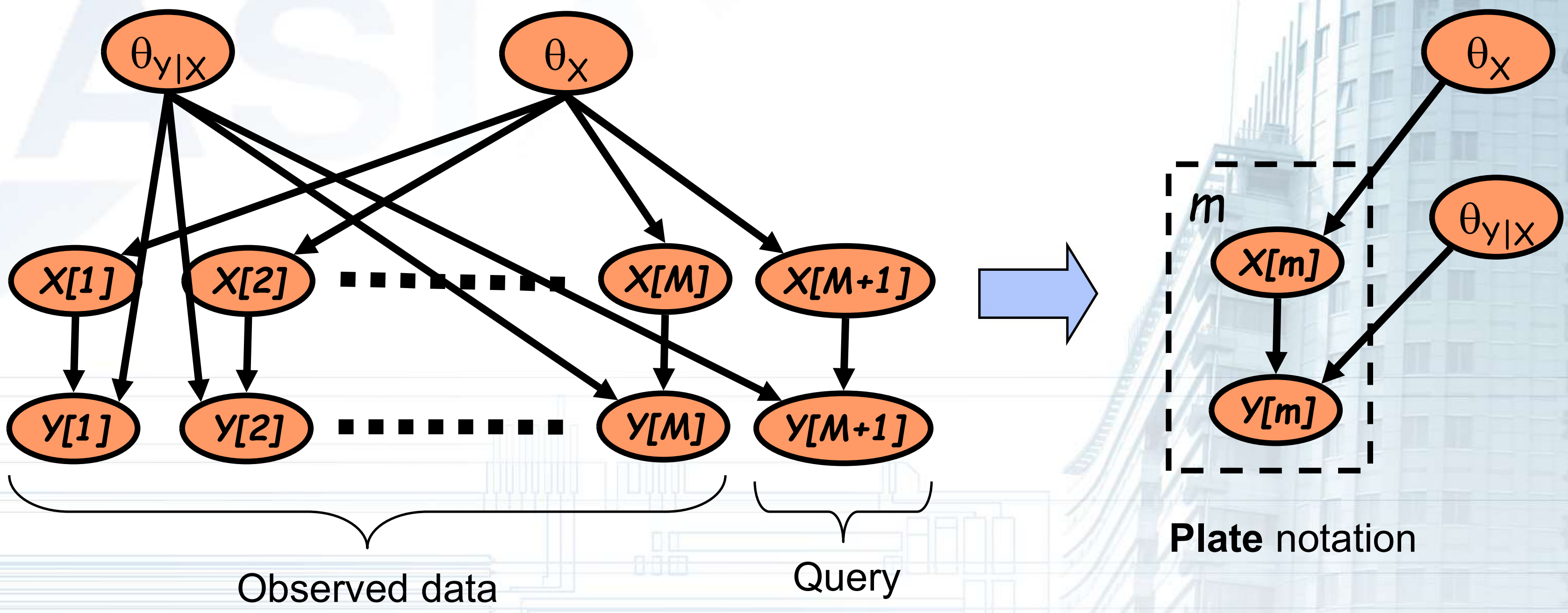
$(N_H, N_T) = (4,1)$



- MLE for $P(X = H)$ is 4/5 = 0.8
- Bayesian prediction is

$$P(x[M+1] = H \mid D) = \int \theta \cdot P(\theta \mid D) d\theta = \frac{5}{7} = 0.7142\ldots$$

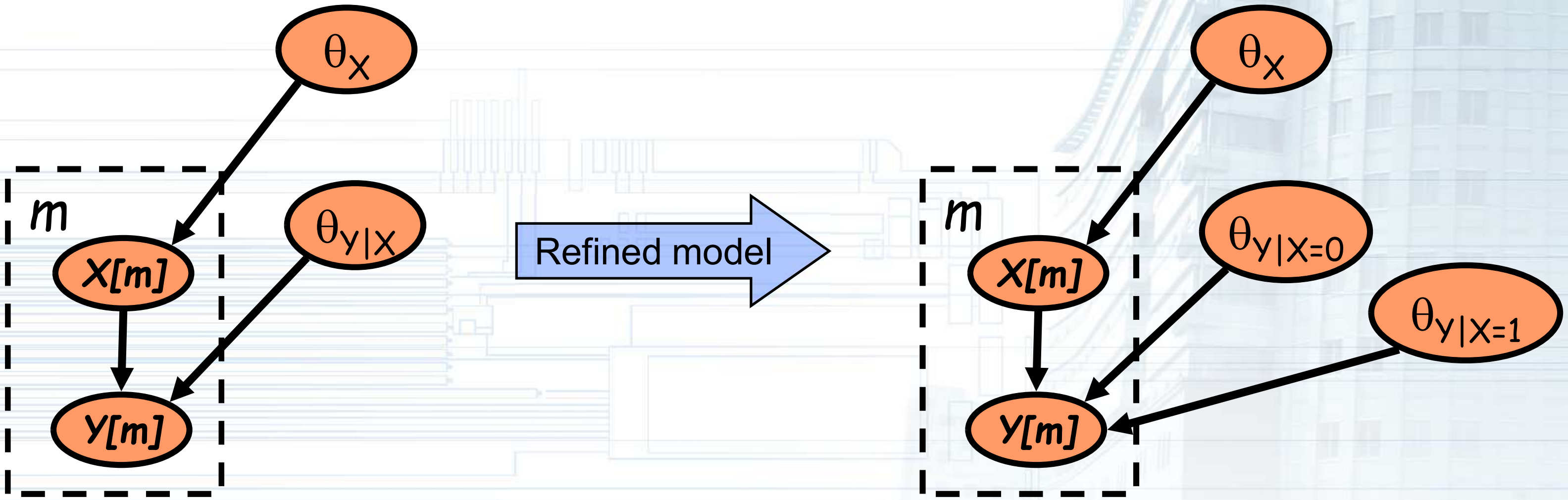
Bayesian Networks and Bayesian Prediction



- Priors for each parameter group are independent
- Data instances are independent given the unknown parameters

Bayesian Prediction

- Since posteriors on parameters for each family are independent, we can compute them separately
- Posteriors for parameters <u>within</u> families are also independent:



- Complete data ⇒ the posteriors on $\theta_{Y|X=0}$ and $\theta_{Y|X=1}$ are independent

- Given these observations, we can compute the posterior for each multinomial $\theta_{X_i \mid pa_i}$ independently
  - The posterior is Dirichlet with parameters $\alpha(X_i{=}1|pa_i)+N(X_i{=}1|pa_i), \ldots, \alpha(X_i{=}k|pa_i)+N(X_i{=}k|pa_i)$
- The predictive distribution is then represented by the parameters

$$\tilde{\theta}_{x_i|pa_i} = \frac{\alpha(x_i, pa_i) + N(x_i, pa_i)}{\alpha(pa_i) + N(pa_i)}$$

The Bayesian analysis just made the assumptions explicit

Assessing Priors for Bayesian Networks

We need the $\alpha(x_i,pa_i)$ *for each node* x_j

- We can use initial parameters Θ_0 as prior information
  - Need also an *equivalent sample size* parameter M_0
  - Then, we let $\alpha(x_i,pa_i) = M_0 \bullet P(x_i,pa_i|\Theta_0)$
- This allows to *update* a network using new data

Learning Parameters: Case Study

- Experiment:
  - Sample a stream of instances from the alarm network
  - Learn parameters using
    - MLE estimator
    - Bayesian estimator with uniform prior with different strengths

Overview

- Introduction
- **Parameter Learning**
  - Complete Data
  - **Incomplete Data**
- Structure Learning
  - Complete Data
  - Incomplete Data

Incomplete Data

Data is often **incomplete**

- Some variables of interest are not assigned values

This phenomenon happens when we have

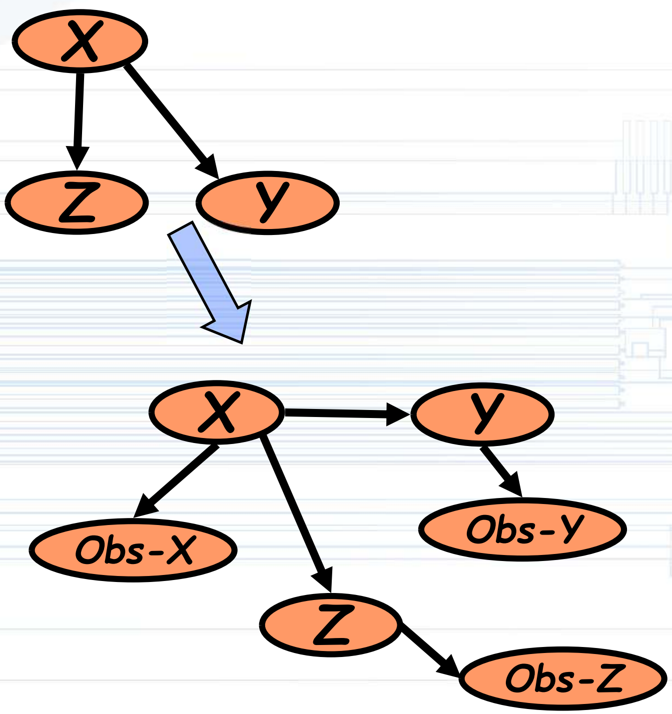
- **Missing values:**
  - Some variables unobserved in some instances
- **Hidden variables:**
  - Some variables are never observed
  - We might not even know they exist

To learn from incomplete data we need the following assumption:

Missing at Random (MAR):

- The probability that the value of X_i is missing is independent of its actual value **given other observed values**

- If MAR assumption does not hold, we can create new variables that ensure that it does
- We now can predict new examples (w/ pattern of ommissions)
- We might not be able to learn about the underlying process



Data

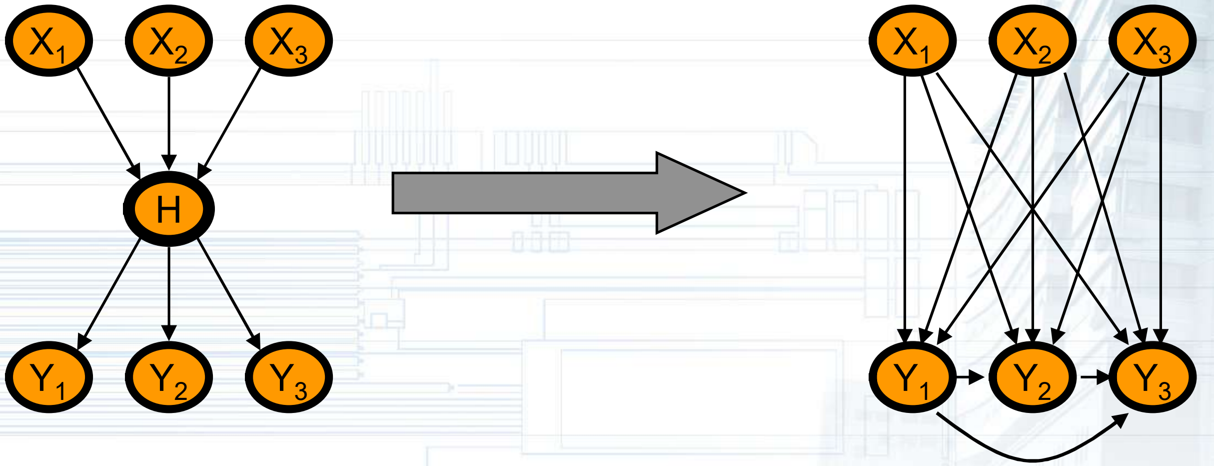
X	Y	Z
H	?	T
T	?	?
H	H	?
H	T	T
T	T	H

Augmented Data

X	Y	Z	Obs-X	Obs-Y	Obs-Z
H	?	T	Y	N	Y
T	?	?	Y	N	N
H	H	?	Y	Y	N
H	T	T	Y	Y	Y
T	T	H	Y	Y	Y

Hidden Variables

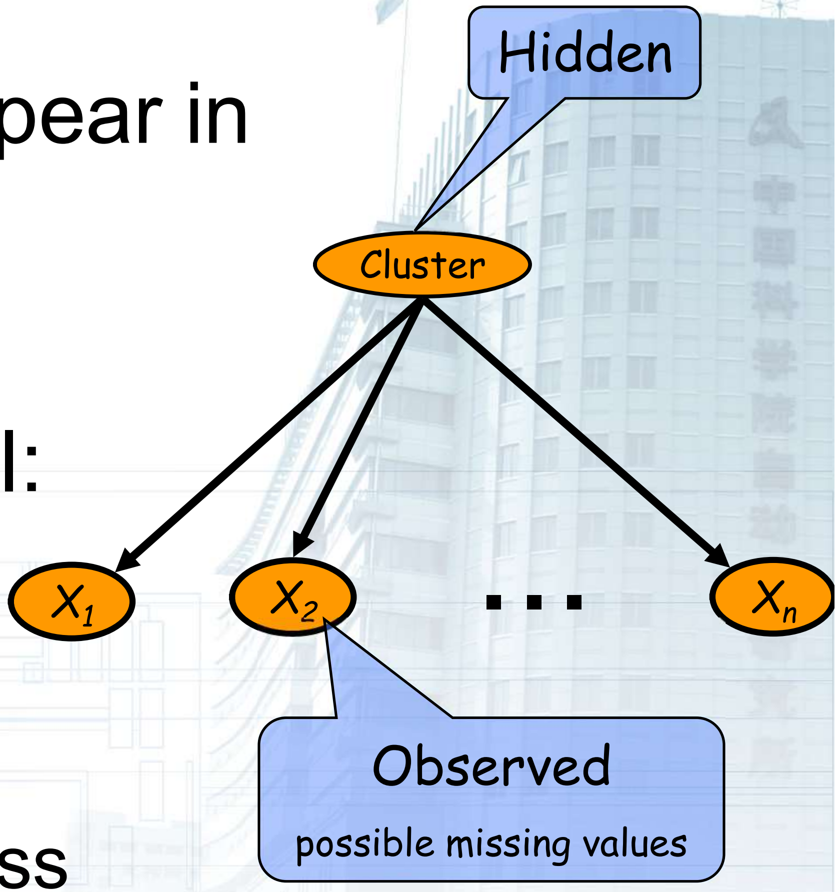
Why should we care about unobserved variables?



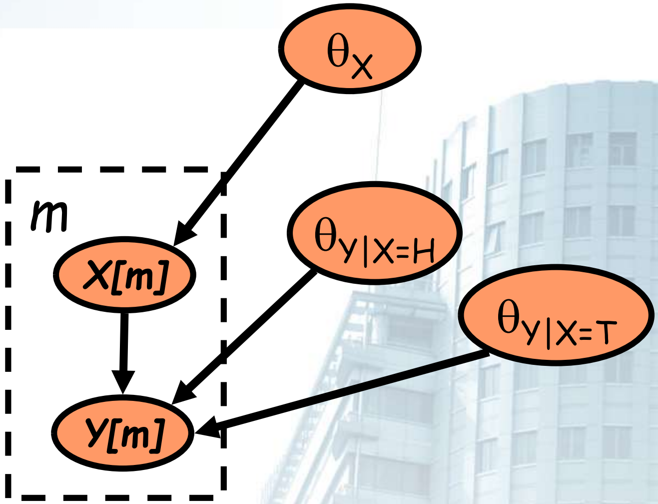
17 parameters **59 parameters**

- Hidden variables also appear in **clustering**

- **Bayesian mixture** model:
  - Hidden variables assigns class labels
  - Observed attributes are independent given the class



Learning Parameters from Incomplete Data



Complete data:

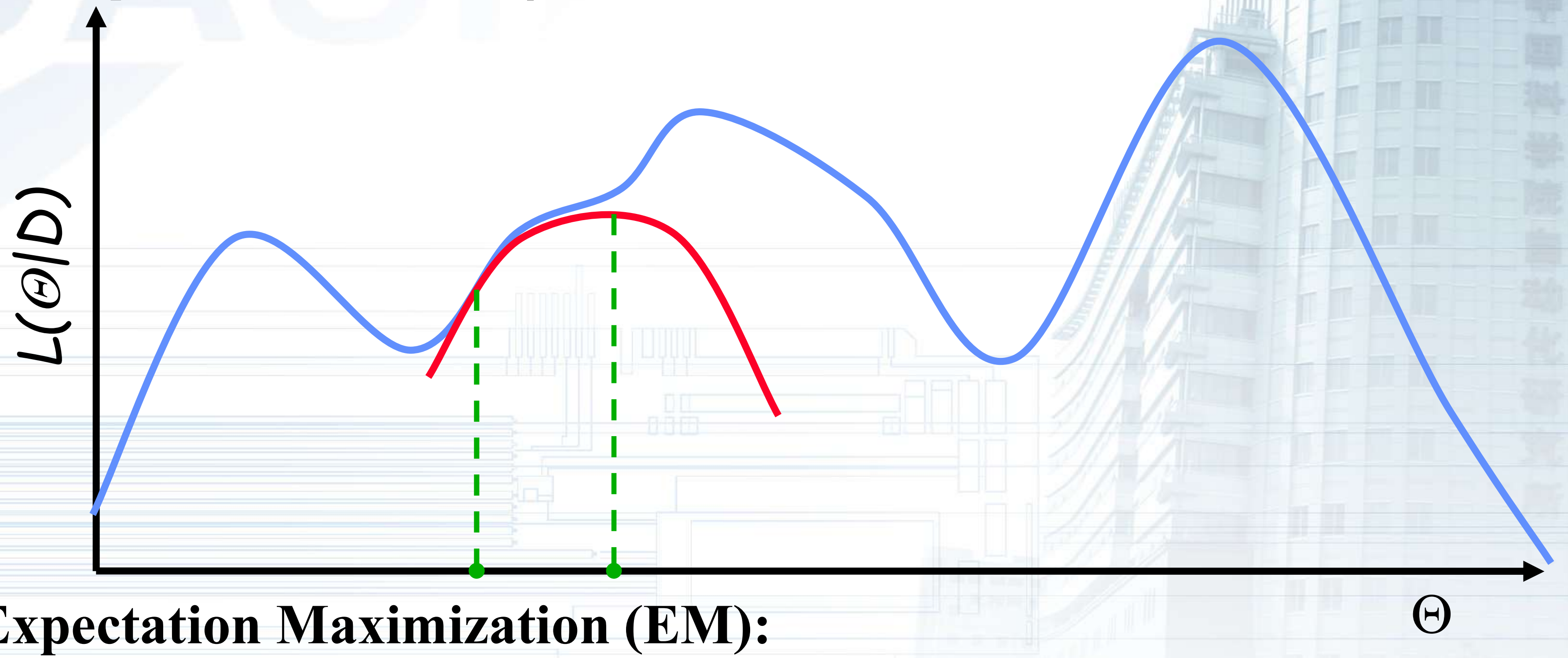
- Independent posteriors for θ_X, $\theta_{Y|X=H}$ and $\theta_{Y|X=T}$

Incomplete data:

- Posteriors can be interdependent
- Consequence:
  - ML parameters can **not** be computed separately for each multinomial
  - Posterior is **not** a product of independent posteriors

MLE from Incomplete Data

- Finding MLE parameters: **nonlinear optimization** problem



Expectation Maximization (EM):

Use "current point" to construct alternative function (which is "nice")

Guaranty: maximum of new function is better scoring the current point

Expectation Maximization

$P(Y=H|X=H,Z=T,\Theta) = 0.3$

Current model

$P(Y=H|X=T,\Theta) = 0.4$

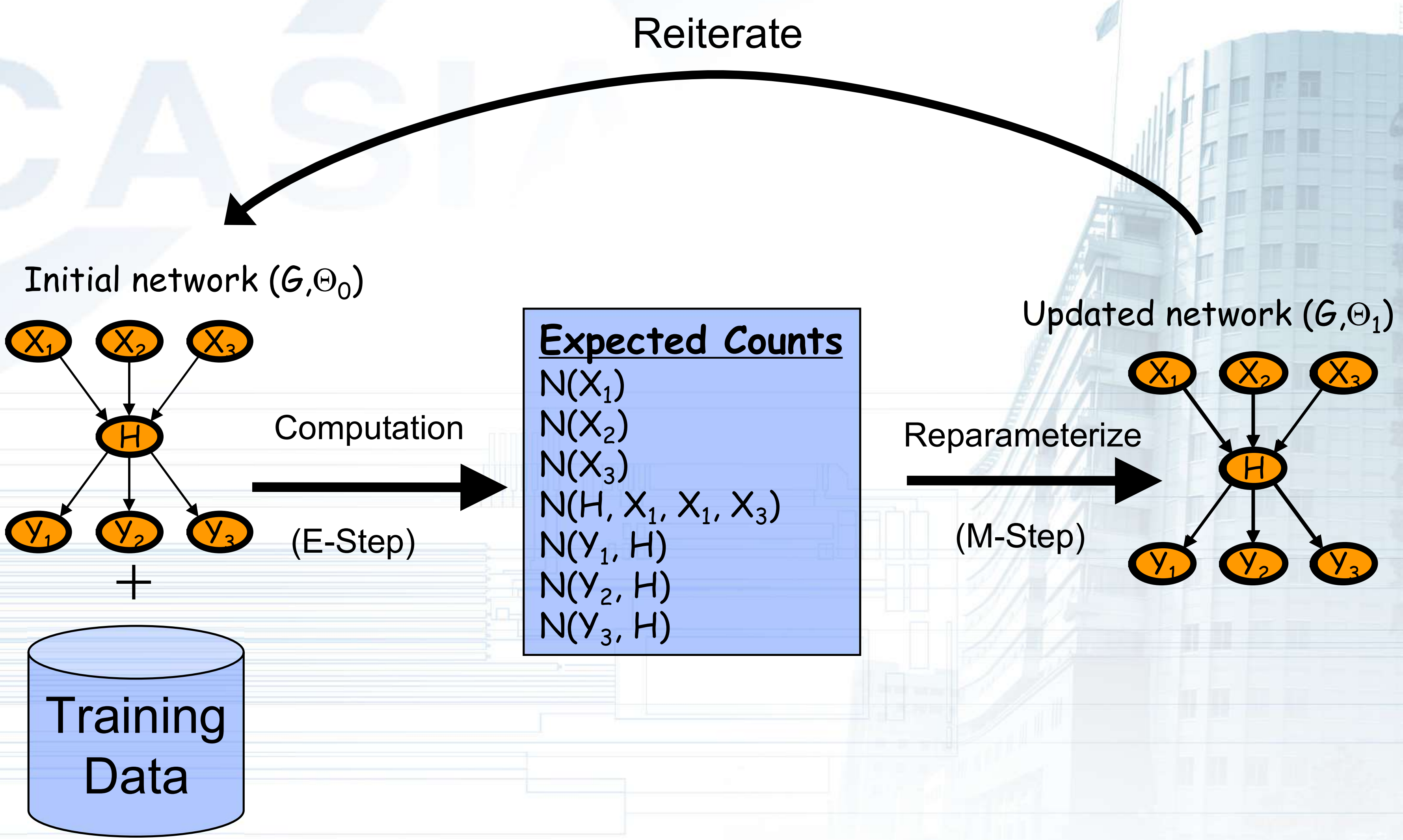
Data

X	Y	Z
H	?	T
T	?	?
H	H	?
H	T	T
T	T	H

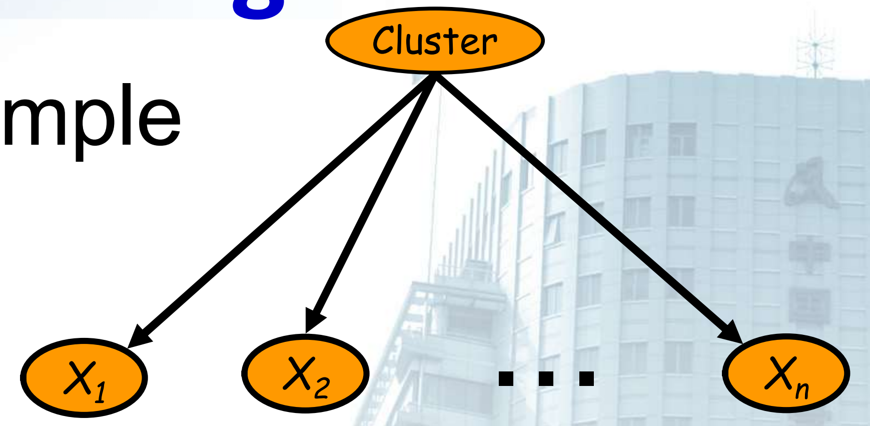
Expected Counts

$N(X,Y)$

X	Y	#
H	H	1.3
T	H	0.4
H	T	1.7
T	T	1.6


Reiterate
Initial network (G,Θ0)
X1 X2 X3
H
Y1 Y2 Y3
+
Training Data
Computation
(E-Step)
Expected Counts
N(X1)
N(X2)
N(X3)
N(H, X1, X1, X3)
N(Y1, H)
N(Y2, H)
N(Y3, H)
Reparameterize
(M-Step)
Updated network (G,Θ1)
X1 X2 X3
H
Y1 Y2 Y3

Example: EM in clustering



- Consider clustering example

E-Step:

– Compute $P(C[m]|X_1[m],\ldots,X_n[m],\Theta)$

– This corresponds to "soft" assignment to clusters

– Compute expected statistics:

M-Step

– Re-estimate $P(X_i|C)$, $P(C)$

Bayesian Inference with Incomplete Data

Recall, Bayesian estimation:

$$P(x[M+1] \mid D) = \int P(x[M+1] \mid \theta) P(\theta \mid D) d\theta$$

Complete data: closed form solution for integral

Incomplete data:

- No sufficient statistics (except the data)
- Posterior does not decompose
- No closed form solution

  Need to use approximations

MAP Approximation

- Simplest approximation: MAP parameters
  - MAP --- **Maximum A-posteriori Probability**

$$P(x[M+1] \mid D) \approx P(x[M+1] \mid \tilde{\theta})$$

where

$$\tilde{\theta} = \arg\max_{\theta} P(\theta \mid D)$$

Assumption:

- Posterior mass is dominated by a MAP parameters

Finding MAP parameters:

- Same techniques as finding ML parameters
- Maximize $P(\theta|D)$ instead of $L(\theta:D)$

Summary

- Non-linear optimization problem
- Methods for learning: EM and Gradient Ascent
  - Exploit inference for learning

Difficulties:

- Exploration of a complex likelihood/posterior
  - More missing data $\Rightarrow$ many more local maxima
  - Cannot represent posterior $\Rightarrow$ must resort to approximations
- Inference
  - Main computational bottleneck for learning
  - Learning large networks
    $\Rightarrow$ exact inference is infeasible
    $\Rightarrow$ resort to stochastic simulation or approximate inference

Overview

- Introduction
- Parameter Learning
  - Complete Data
  - Incomplete Data
- **Structure Learning**
  - **Complete Data**
  - Incomplete Data

Approaches to Learning Structure

- **Constraint based**
  - Perform tests of conditional independence
  - Search for a network that is consistent with the observed dependencies and independencies
- **Score based**
  - Define a score that evaluates how well the (in)dependencies in a structure match the observations
  - Search for a structure that maximizes the score

Likelihood Score for Structure

$$\ell(G : D) = \log L(G : D) = M \sum_i \left(I(X_i ; Pa_i^G) - H(X_i) \right)$$

Mutual information between X_i and its parents

- Larger dependence of X_i on $Pa_i \Rightarrow$ higher score
- Adding arcs always helps
  - $I(X; Y) \leq I(X; \{Y,Z\})$
  - Max score attained by fully connected network
  - Overfitting: A bad idea…

Bayesian Score

Likelihood score: $L(G : D) = P(D \mid G, \hat{\theta}_G)$

Max likelihood params

Bayesian approach:

Deal with uncertainty by assigning probability to all possibilities

$$P(D \mid G) = \int P(D \mid G, \theta) P(\theta \mid G) d\theta$$

Marginal Likelihood

Likelihood

Prior over parameters

$$P(G \mid D) = \frac{P(D \mid G) P(G)}{P(D)}$$

Marginal Likelihood for Networks

The marginal likelihood has the form:

$$P(D \mid G) = \prod_i \prod_{pa_i^G}$$

Dirichlet marginal likelihood for multinomial ***$P(X_i \mid pa_i)$***

$$\frac{\Gamma\left(\alpha(pa_i^G)\right)}{\Gamma\left(\alpha(pa_i^G) + N(pa_i^G)\right)} \prod_{x_i} \frac{\Gamma(\alpha(x_i, pa_i^G) + N(x_i, pa_i^G))}{\Gamma(\alpha(x_i, pa_i^G))}$$

N(..) are counts from the data

α(..) are hyperparameters for each family **given G**

Structure Search as Optimization

Input:

- Training data
- Scoring function
- Set of possible structures

Output:

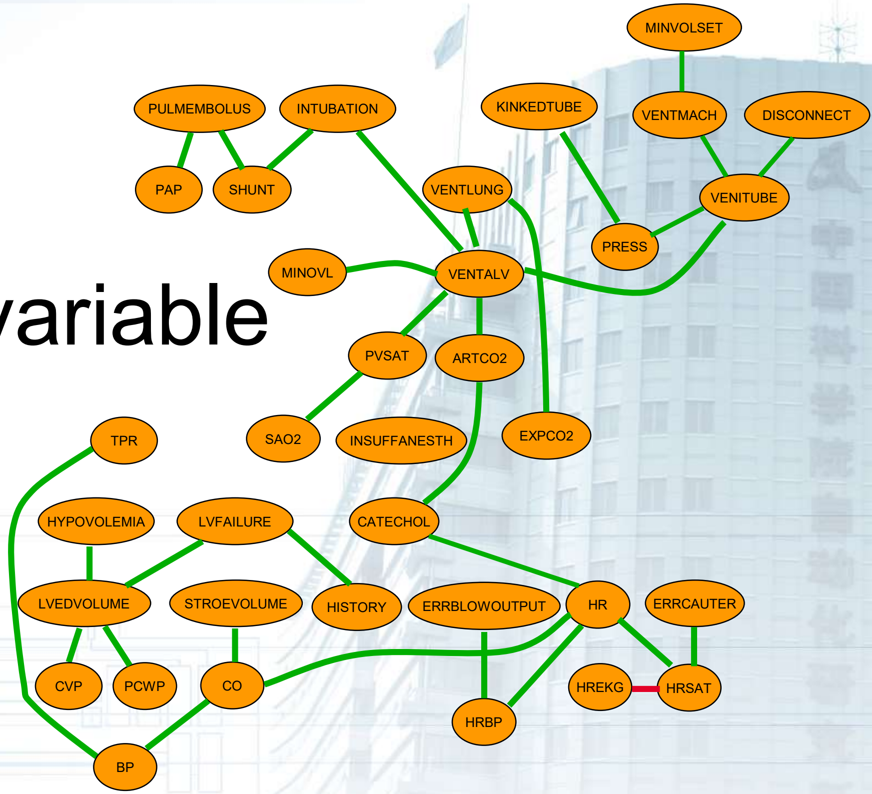
- A network that maximizes the score

Key Computational Property:

Decomposability:

$$\text{score}(G) = \sum \text{score} \,(\text{ family of } X \text{ in } G\ )$$

Tree-Structured Networks



Trees:

- At most one parent per variable

Why trees?

- Elegant math

  ⇒we can solve the optimization problem

- Sparse parameterization

  ⇒avoid overfitting

Learning Trees

- Let $p(i)$ denote parent of X_i
- We can write the Bayesian score as

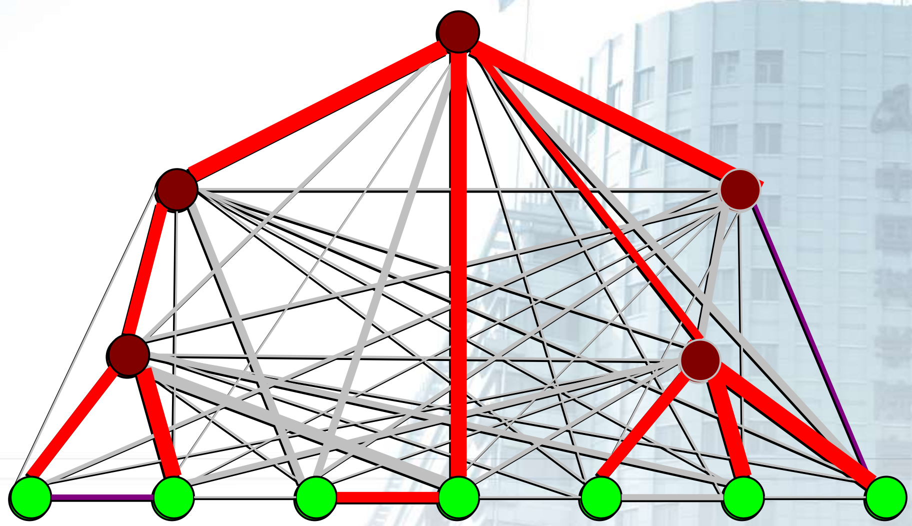
$$Score(G:D) = \sum_i Score(X_i : Pa_i)$$

$$= \sum_i \left(Score(X_i : X_{p(i)}) - Score(X_i) \right) + \sum_i Score(X_i)$$

Improvement over "empty" network

Score of "empty" network

Score = sum of edge scores + constant



- Set $w(j \rightarrow i) = Score(X_j \rightarrow X_i) - Score(X_i)$
- Find tree (or forest) with maximal weight
  - Standard max spanning tree algorithm — $O(n^2 \log n)$

Theorem: This procedure finds tree with max score

Summary

- Discrete optimization problem
- In some cases, optimization problem is easy
  - Example: learning trees
- In general, NP-Hard
  - Need to resort to heuristic search
  - In practice, search is relatively fast (~100 vars in ~2-5 min):
    - Decomposability
    - Sufficient statistics
  - Adding randomness to search is critical

Overview

- Introduction
- Parameter Learning
  - Complete Data
  - Incomplete Data
- **Structure Learning**
  - Complete Data
  - **Incomplete Data**

Incomplete Data: Structure Scores

Recall, Bayesian score:

$$P(G \mid D) \propto P(G)P(D \mid G)$$

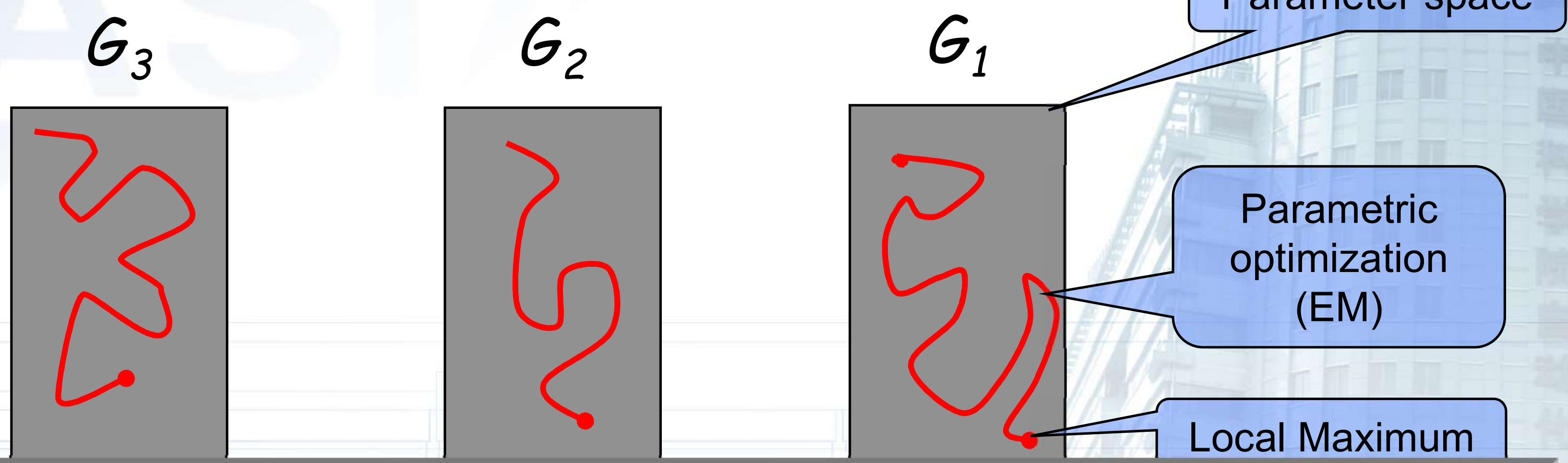
$$= P(G)\boxed{\int P(D \mid G, \Theta)P(\Theta \mid G)d\theta}$$

With incomplete data:

- Cannot evaluate marginal likelihood in closed form
- We have to resort to **approximations**:
  - Evaluate score around MAP parameters
  - Need to find MAP parameters (e.g., EM)

Naïve Approach

Perform EM for each candidate graph



Computationally expensive:

- Parameter optimization via EM — non-trivial
- Need to perform EM for all candidate structures
- Spend time even on poor candidates

⇒In practice, considers only a few candidates

Structural EM

Recall, in complete data we had

– Decomposition $\Rightarrow$ efficient search

Idea:

- Instead of optimizing the real score…
- Find **decomposable** alternative score
- Such that maximizing new score

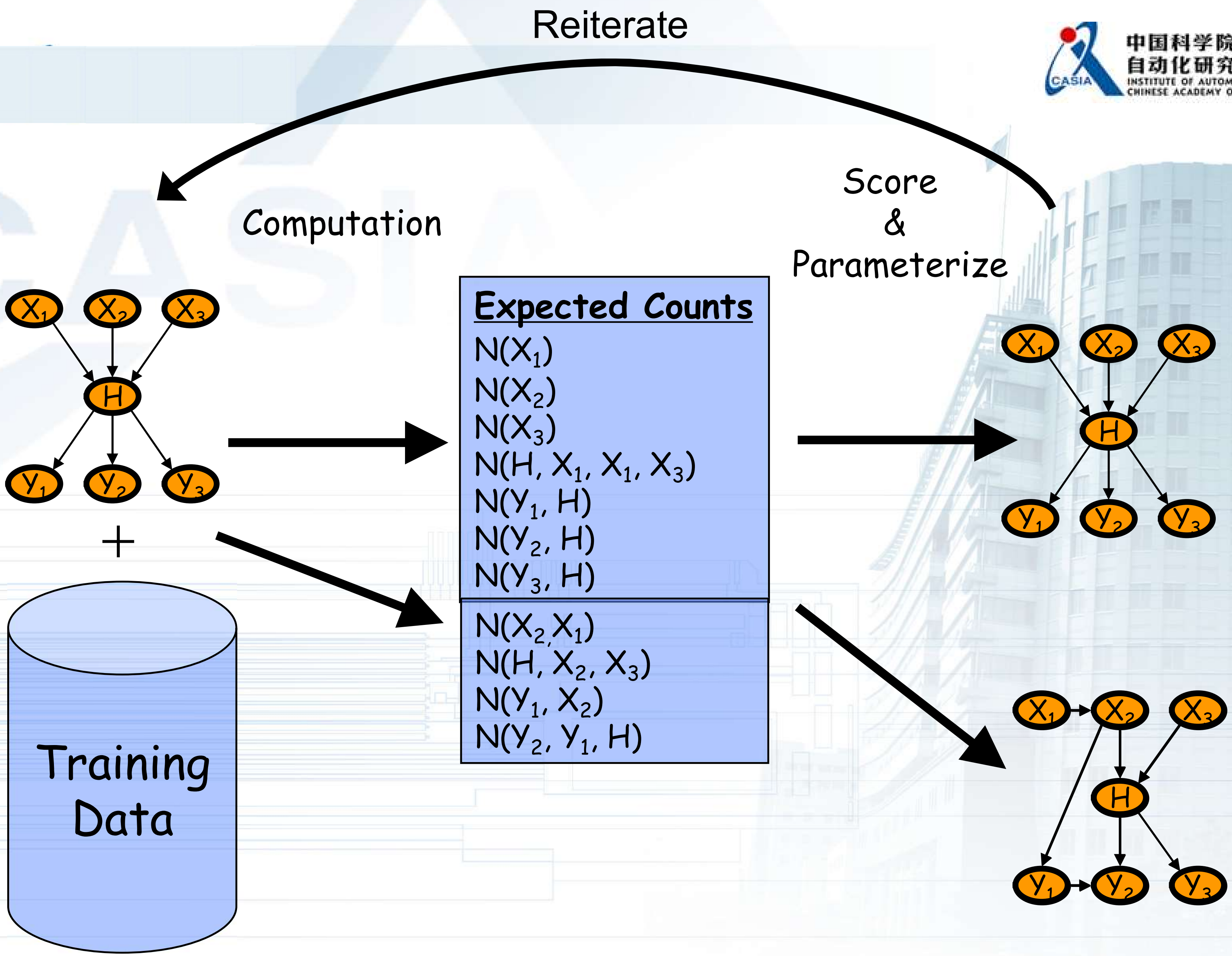
  $\Rightarrow$ improvement in real score

Idea:

- Use current model to help evaluate new structures

Outline:

- Perform search in (Structure, Parameters) space
- At each iteration, use current model for finding either:
  - Better scoring parameters: “parametric” EM step or
  - Better scoring structure: “structural” EM step



Reiterate
Computation
Score
&
Parameterize
Expected Counts
$N(X_1)$
$N(X_2)$
$N(X_3)$
$N(H, X_1, X_1, X_3)$
$N(Y_1, H)$
$N(Y_2, H)$
$N(Y_3, H)$
$N(X_2, X_1)$
$N(H, X_2, X_3)$
$N(Y_1, X_2)$
$N(Y_2, Y_1, H)$
Training Data
X_1 X_2 X_3
H
Y_1 Y_2 Y_3
+

Example: Phylogenetic Reconstruction

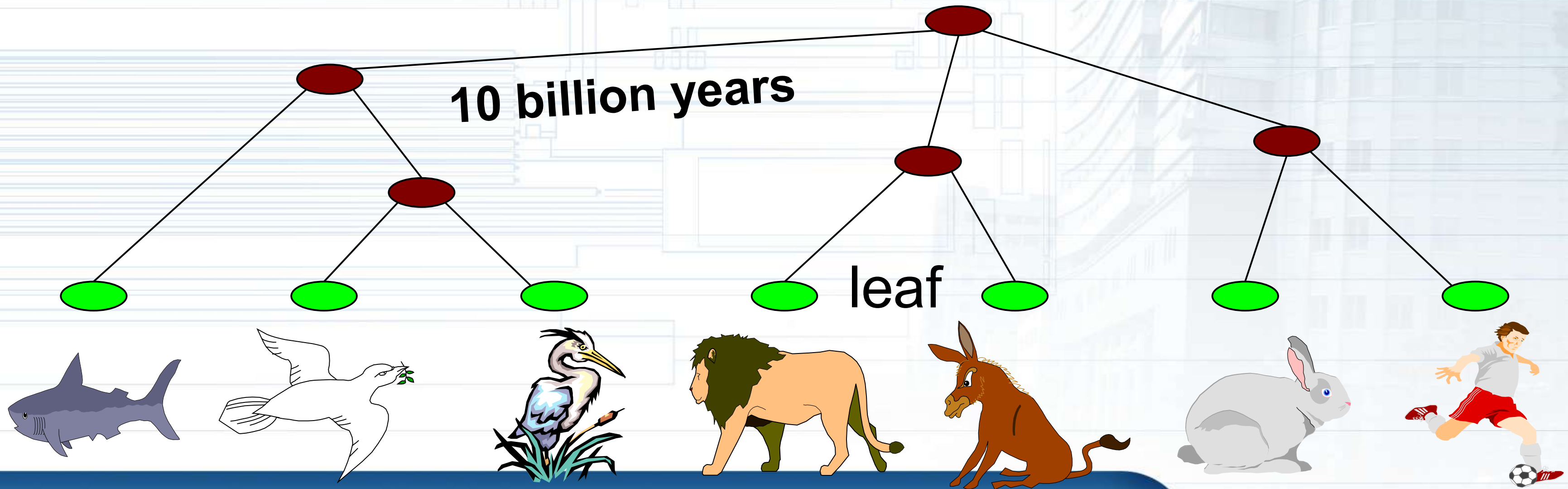
Input: Biological sequences

Human	CGTTGC…
Chimp	CCTAGG…
Orang	CGAACG…
….	

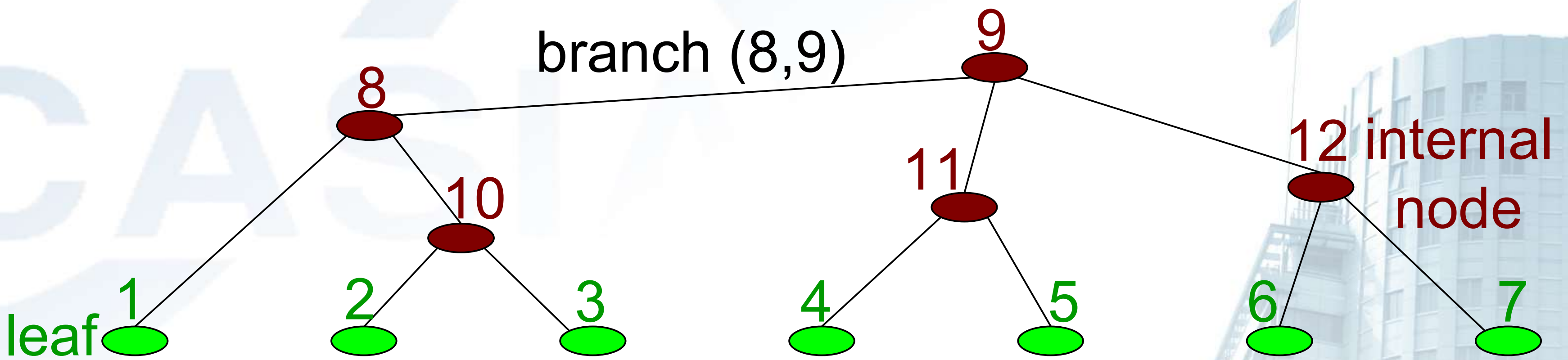
An "instance" of evolutionary process

Assumption: positions are independent

Output: a phylogeny



Phylogenetic Model



Topology: bifurcating

Observed species – $1 \ldots N$

Ancestral species – $N+1 \ldots 2N\text{-}2$

Lengths $t = \{t_{i,j}\}$ for each branch (i,j)

Evolutionary model:

P(A changes to T| 10 billion yrs)

Phylogenetic Tree as a Bayes Net

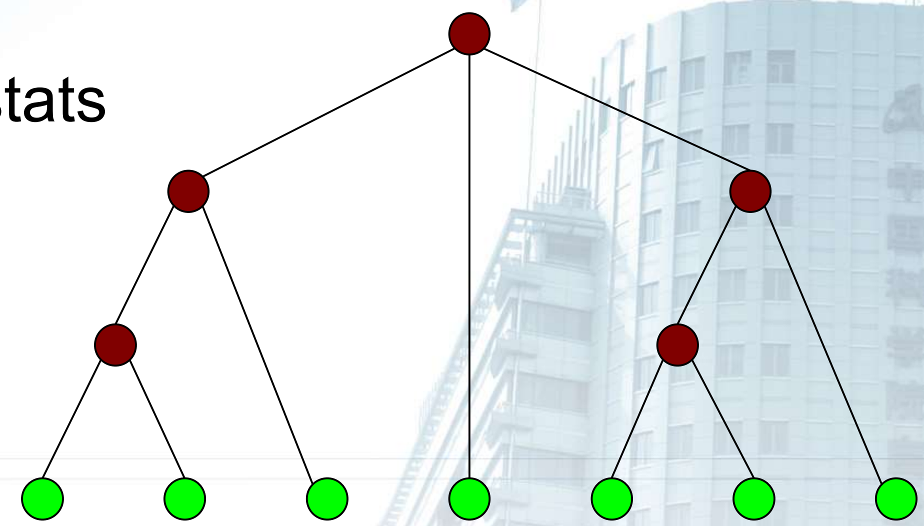
- Variables: Letter at each position for each species
  - Current day species – observed
  - Ancestral species - hidden
- BN Structure: Tree topology
- BN Parameters: Branch lengths (time spans)

Main problem: Learn topology

If ancestral were observed

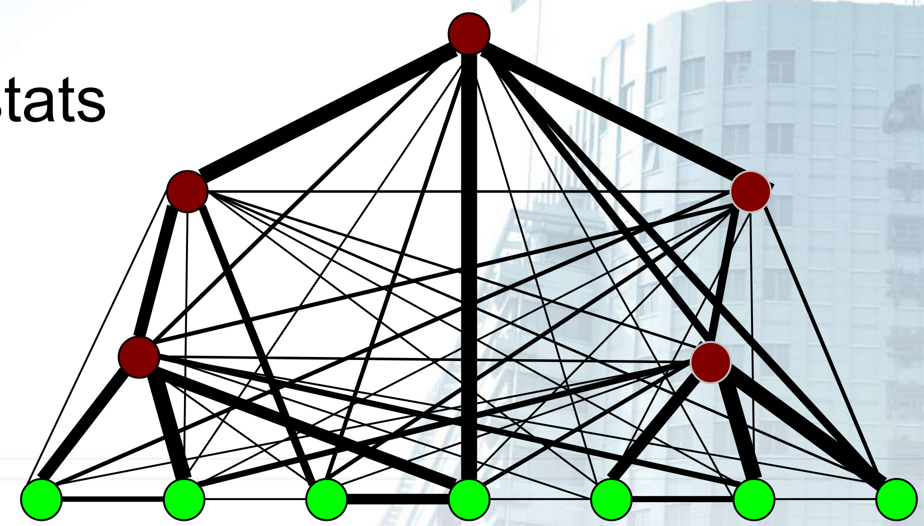
$\Rightarrow$ easy learning problem (learning trees)

- Compute expected pairwise stats
- Weights: Branch scores



Original Tree $(T^0, \boldsymbol{t}^0)$

- Compute expected pairwise stats
- Weights: Branch scores
- Find: $T' = \arg\max_T \sum_{(i,j)\in T} w_{i,j}$



Pairwise weights

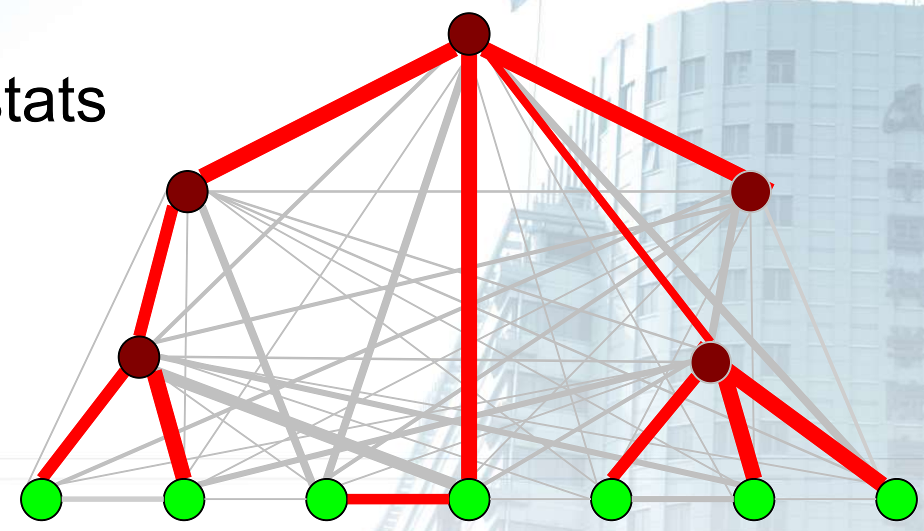
O(N^2) pairwise statistics suffice to evaluate all trees

→ Compute expected pairwise stats

→ Weights: Branch scores

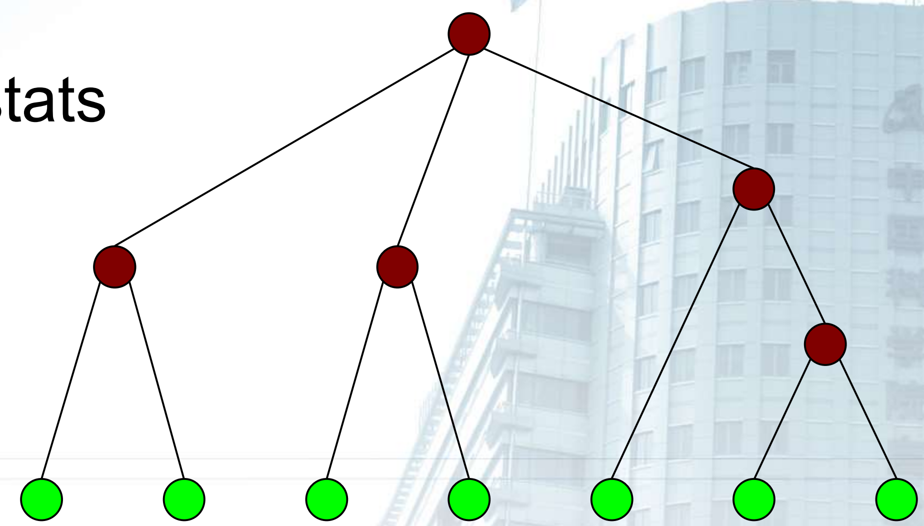
→ Find: $T' = \arg\max_T \sum_{(i,j)\in T} w_{i,j}$

→ Construct bifurcation T_1



Max. Spanning Tree

- Compute expected pairwise stats
- Weights: Branch scores
- Find: $T' = \arg\max_T \sum_{(i,j)\in T} w_{i,j}$
- Construct bifurcation T_1
- Theorem: $L(T_1,t_1) \geq L(T_0,t_0)$



Repeat until convergence…

Thank you very much for your presence