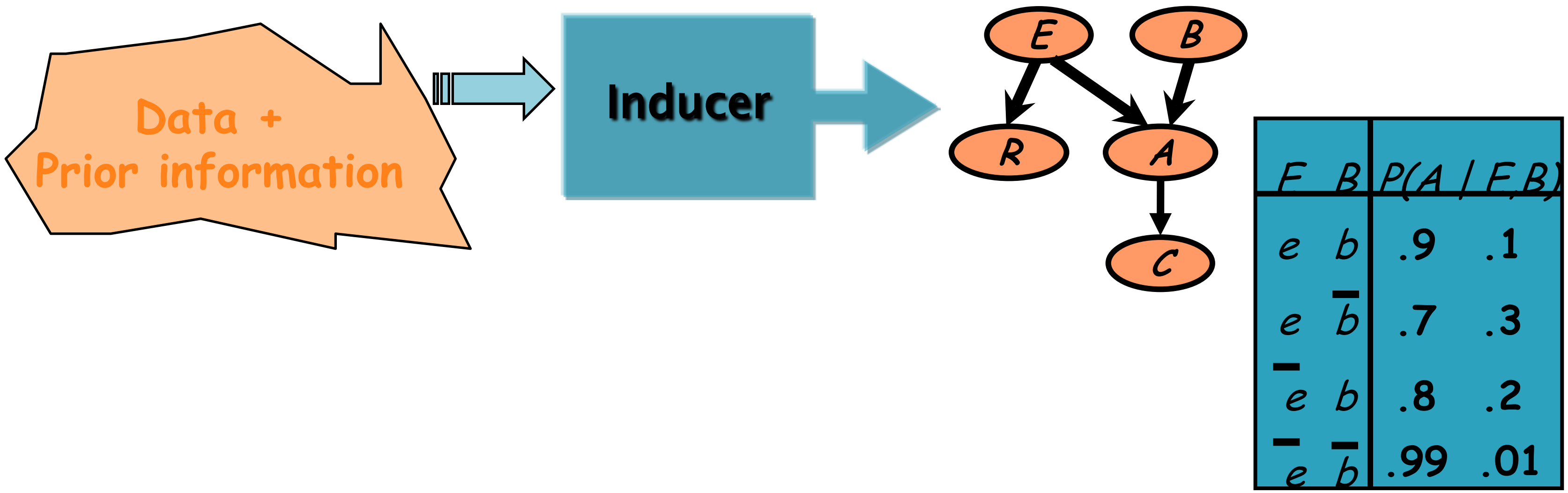
CMSC 726
Lecture 21:Learning Graphical Models

Lise Getoor

November 16, 2010

ACKNOWLEDGEMENTS: The material in this course is a synthesis of materials from many sources, including: Hal Daume III, Mark Drezde, Carlos Guestrin, Andrew Ng, Ben Taskar, Eric Xing, and others. I am very grateful for their generous sharing of insights and materials.

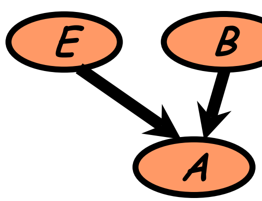
Learning Bayesian networks

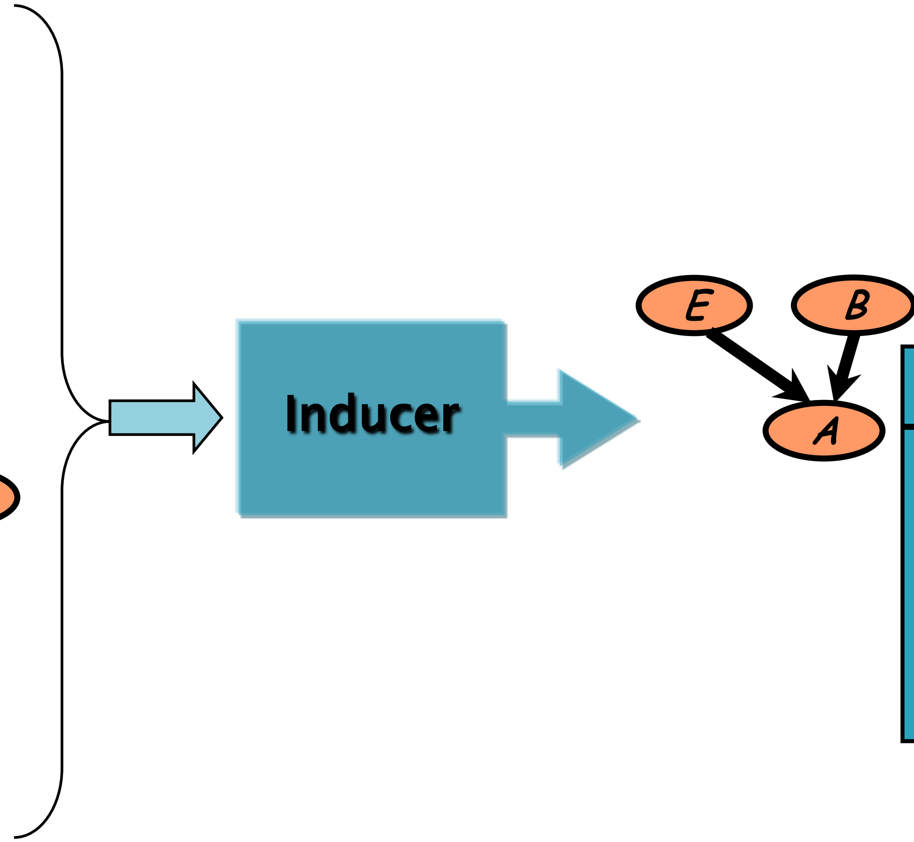


Known Structure -- Complete Data

E, B, A
<Y,N,N>
<Y,Y,Y>
<N,N,Y>
<N,Y,Y>
.
.
<N,Y,Y>

E	B	P(A | E,B)	
e	b	?	?
e	$\bar{b}$	?	?
$\bar{e}$	b	?	?
$\bar{e}$	$\bar{b}$	?	?

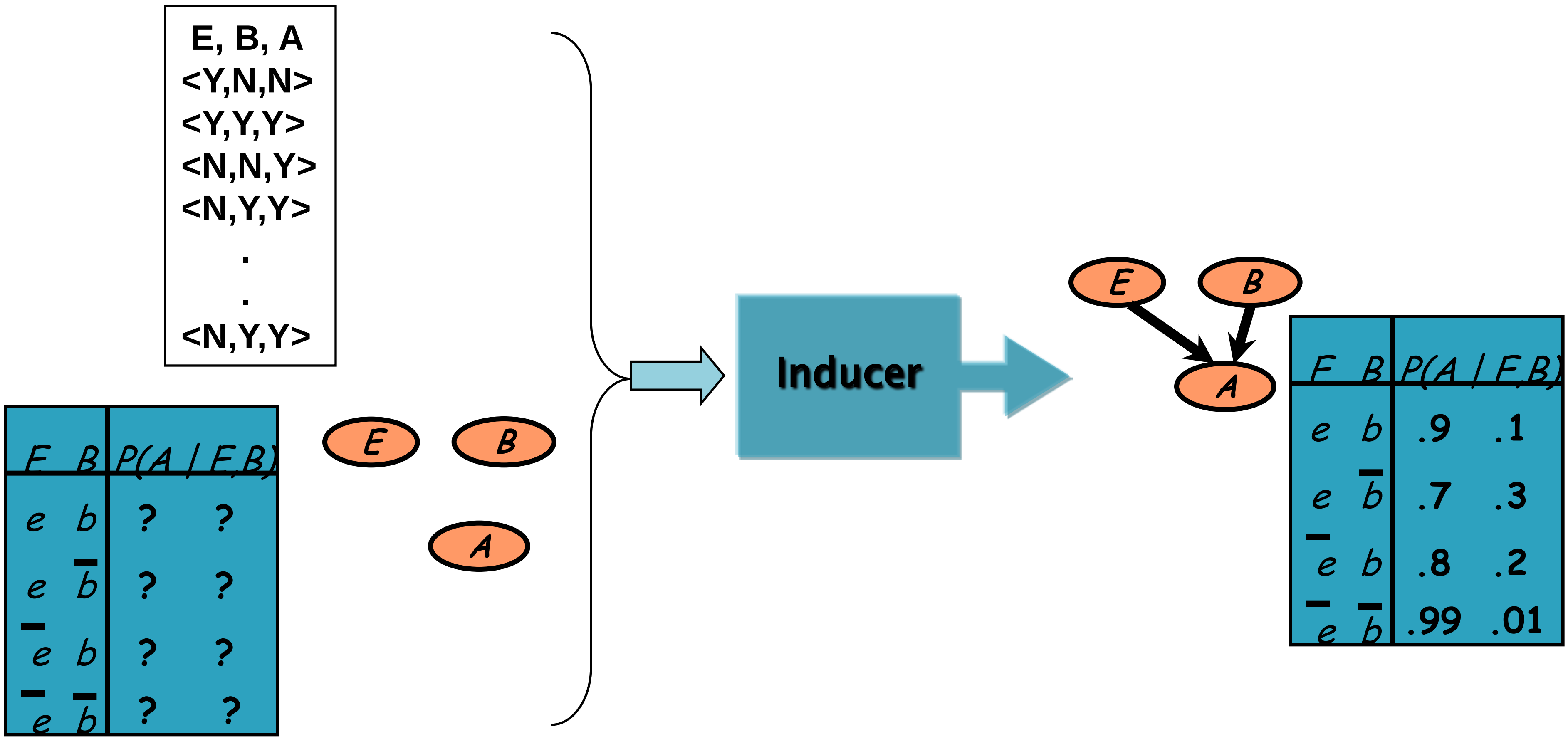




E	B	P(A | E,B)	
e	b	.9	.1
e	$\bar{b}$	.7	.3
$\bar{e}$	b	.8	.2
$\bar{e}$	$\bar{b}$	.99	.01

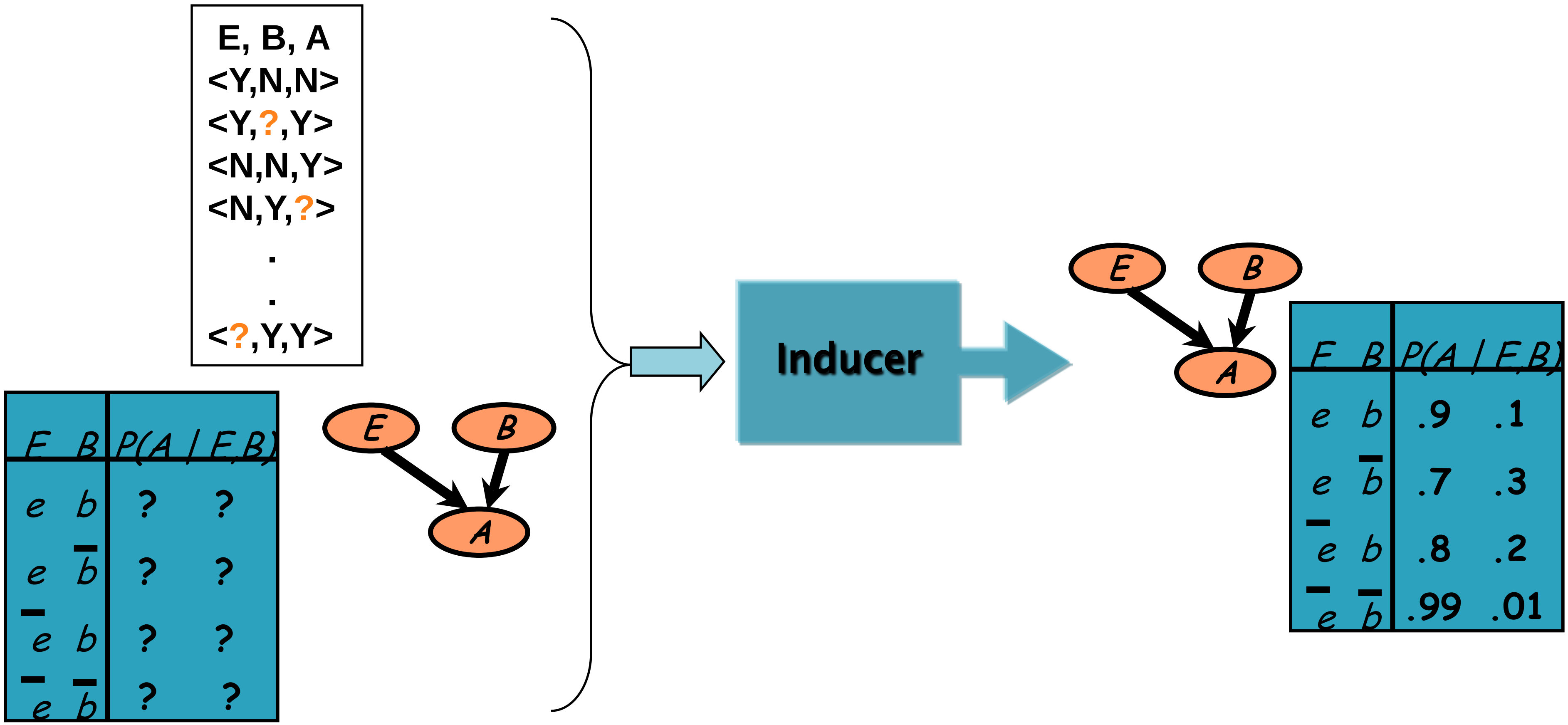
- Network structure is specified
  - Inducer needs to estimate parameters
- Data does not contain missing values

Unknown Structure -- Complete Data



- Network structure is not specified
  - Inducer needs to select arcs & estimate parameters
- Data does not contain missing values

Known Structure -- Incomplete Data



- Network structure is specified
- Data contains missing values
  - We consider assignments to missing values

Known Structure / Complete Data

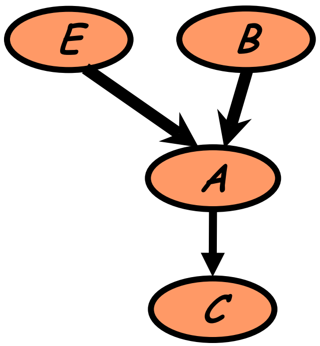
- Given a network structure G
  - And choice of parametric family for $P(X_i|Pa_i)$
- Learn parameters for network

Goal

- Construct a network that is “closest” to probability that generated the data

Learning Parameters for a Bayesian Network

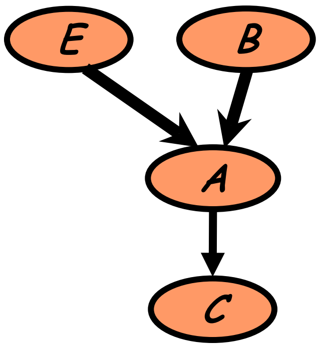
- Training data has the form:



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

Learning Parameters for a Bayesian Network

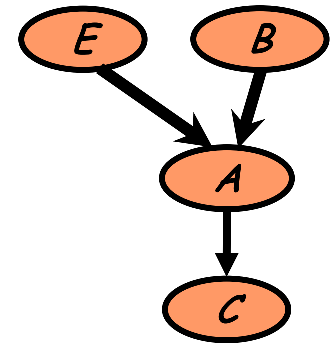
- Since we assume i.i.d. samples, likelihood function is



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

Learning Parameters for a Bayesian Network

- By definition of network, we get



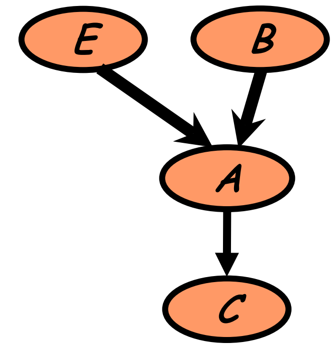
$$L(\Theta : D) = \prod_m P(E[m], B[m], A[m], C[m] : \Theta)$$

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

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

Learning Parameters for a Bayesian Network

- Rewriting terms, we get



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

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

General Bayesian Networks

Generalizing for any Bayesian network:

$$
\begin{aligned}
L(\Theta : D) &= \prod_m P(x_1[m], \ldots, x_n[m] : \Theta) \\
&= \prod_m \prod_i P(x_i[m] \mid Pa_i[m] : \Theta_i) \\
&= \prod_i \prod_m P(x_i[m] \mid Pa_i[m] : \Theta_i) \\
&= \prod_i L_i(\Theta_i : D)
\end{aligned}
$$

i.i.d. samples

Network factorization

- The likelihood **decomposes** according to the structure of the network.

General Bayesian Networks (Cont.)

Decomposition

$\Rightarrow$ Independent Estimation Problems

If the parameters for each family are not related, then they can be estimated independently of each other.

MLE for Multinomials

- For example, suppose X can have the values $1,2,\ldots,K$
- We want to learn the parameters $\theta_1, \theta_2, \ldots, \theta_K$

Sufficient statistics:

- $N_1, N_2, \ldots, N_K$ – the number of times each outcome is observed

Likelihood function: $$L(\theta : D) = \prod_{k=1}^{K} \theta_k^{N_k}$$

MLE: $$\hat{\theta}_k = \frac{N_k}{\sum_{\ell} N_\ell}$$

Likelihood for Multinomial Networks

- When we assume that $P(X_i / Pa_i)$ is multinomial, we get decomposition:

$$L_i(\Theta_i : D) = \prod_{pa_i} \prod_{x_i} \theta_{x_i|pa_i}^{N(x_i, pa_i)}$$

- For each value pa_i of the parents of X_i we get an independent multinomial problem

- The MLE is $\hat{\theta}_{x_i|pa_i} = \dfrac{N(x_i, pa_i)}{N(pa_i)}$

Reminder: Bayesian Inference

Frequentist Approach:

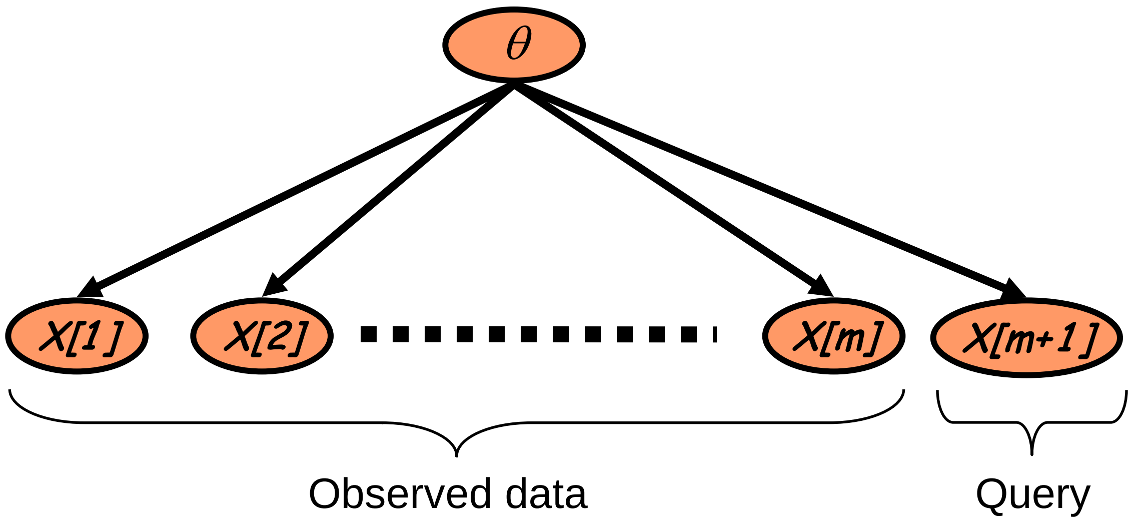
- Assumes there is an unknown but fixed parameter θ
- Estimates θ with some confidence
- Prediction by using the estimated parameter value

Bayesian Approach:

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

Bayesian Inference (cont.)

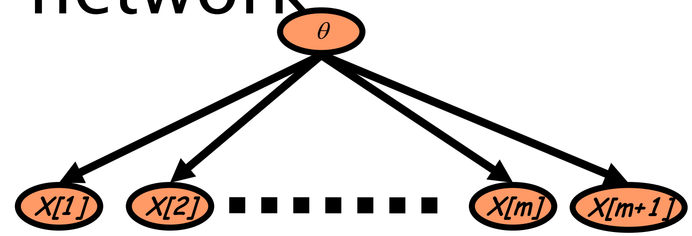
- We can represent our uncertainty about the sampling process using a Bayesian network



- The 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

Bayesian Inference (cont.)

Prediction as **inference** in this network



$$
\begin{aligned}
P(x[M+1] \mid x[1],\ldots,x[M]) &= \int P(x[M+1] \mid \theta, x[1],\ldots,x[M]) P(\theta \mid x[1],\ldots,x[M]) d\theta \\
&= \int P(x[M+1] \mid \theta) P(\theta \mid x[1],\ldots,x[M]) d\theta
\end{aligned}
$$

where

Likelihood

Prior

$$
P(\theta \mid x[1],\ldots x[M]) = \frac{P(x[1],\ldots x[M] \mid \theta) P(\theta)}{P(x[1],\ldots x[M])}
$$

Posterior

Probability of data

Dirichlet Priors

- Recall that the likelihood function for a multinomial is

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

- A **Dirichlet** prior with hyperparameters $\alpha_1,\ldots,\alpha_K$ is defined as

$$P(\Theta) \propto \prod_{k=1}^{K} \theta_k^{\alpha_k - 1}$$ for legal $\theta_1,\ldots,\theta_K$

Then the posterior has the same form, with hyperparameters

$\alpha_1+N_1,\ldots,\alpha_K+N_K$

$$P(\Theta \mid D) \propto P(\Theta)P(D \mid \Theta) \propto \prod_{k=1}^{K} \theta_k^{\alpha_k - 1} \prod_{k=1}^{K} \theta_k^{N_k} = \prod_{k=1}^{K} \theta_k^{\alpha_k + N_k - 1}$$

Dirichlet Priors (cont.)

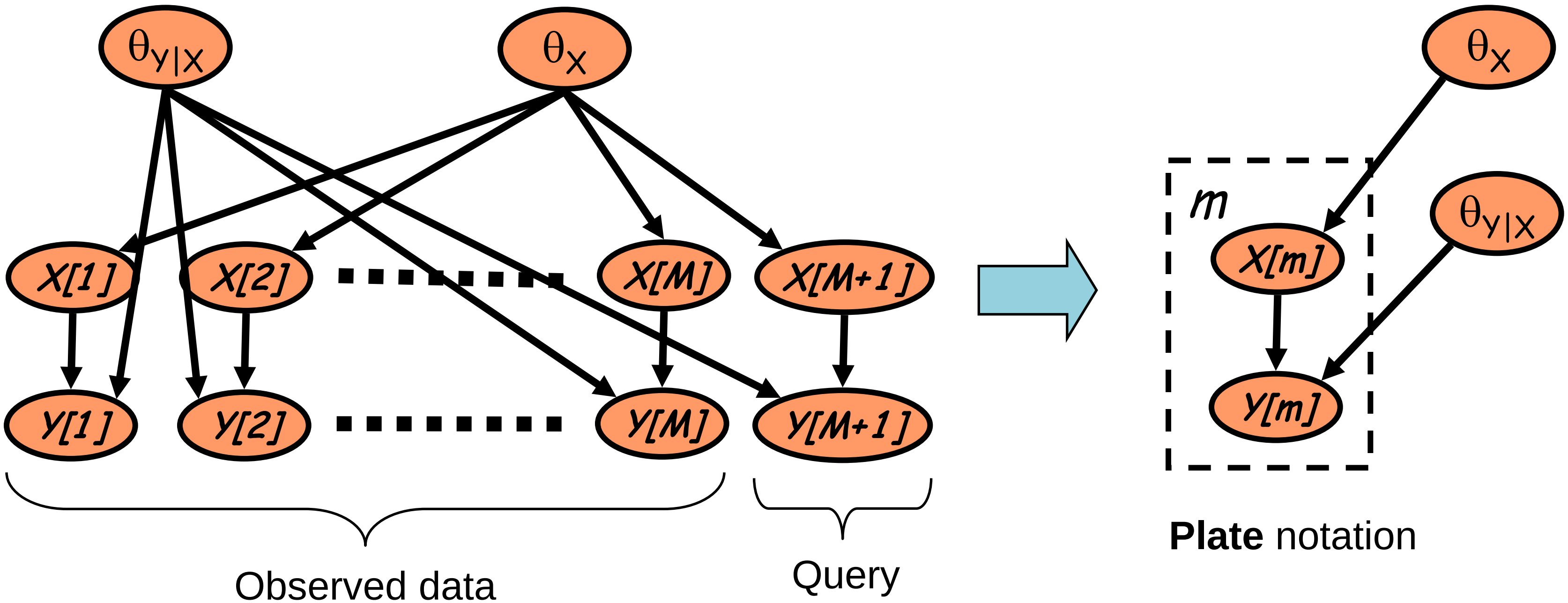
- We can compute the prediction on a new event in closed form:
- If $P(\Theta)$ is Dirichlet with hyperparameters $\alpha_1,\ldots,\alpha_K$ then

$$P(X[1]=k) = \int \theta_k \, P(\Theta) d\Theta = \frac{\alpha_k}{\sum_{\ell} \alpha_{\ell}}$$

- Since the posterior is also Dirichlet, we get

$$P(X[M+1]=k \mid D) = \int \theta_k \, P(\Theta \mid D) d\Theta = \frac{\alpha_k + N_k}{\sum_{\ell} (\alpha_{\ell} + N_{\ell})}$$

Bayesian Networks and Bayesian Prediction



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

Bayesian Prediction(cont.)

- 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

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

Learning Parameters: Summary

- Estimation relies on **sufficient statistics**
  - For multinomial these are of the form $N(x_i, pa_i)$
  - Parameter estimation

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

MLE　　　　Bayesian (Dirichlet)

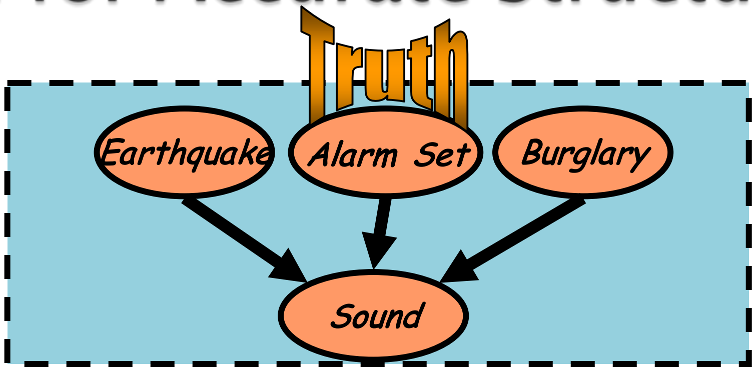
- Bayesian methods also require choice of priors
- Both MLE and Bayesian are asymptotically equivalent and consistent
- Both can be implemented in an **on-line** manner by accumulating sufficient statistics

Learning Structure from Complete Data

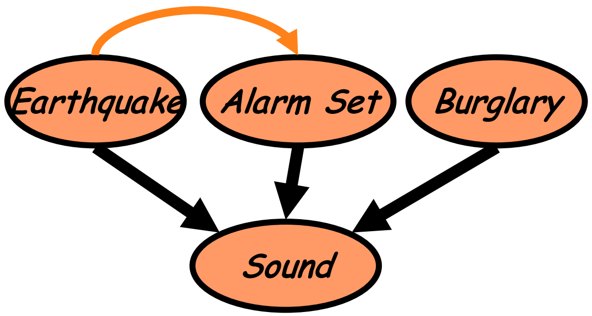
Benefits of Learning Structure

- Efficient learning -- more accurate models with less data
  - Compare: P(A) and P(B) vs. joint P(A,B)
- Discover structural properties of the domain
  - Ordering of events
  - Relevance
- Identifying independencies $\Rightarrow$ faster inference
- Predict effect of actions
  - Involves learning causal relationship among variables

Why Struggle for Accurate Structure?

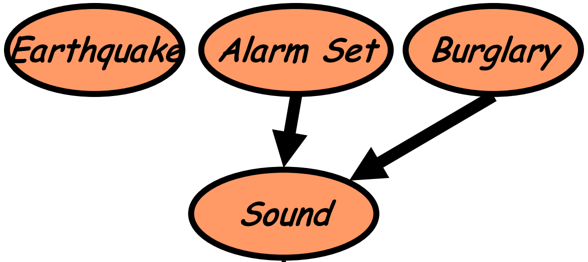


Adding an arc



- Increases the number of parameters to be fitted
- Wrong assumptions about causality and domain structure

Missing an arc



- Cannot be compensated by accurate fitting of parameters
- Also misses causality and domain structure

Approaches to Learning Structure

- Constraint based
  - Perform tests of conditional independence
  - Search for a network that is consistent with the observed dependencies and independencies

- **Pros & Cons**

  \+ Intuitive, follows closely the construction of BNs

  \+ Separates structure learning from the form of the independence tests

  – Sensitive to errors in individual tests

  – Computationally hard

Approaches to Learning Structure

- **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
- **Pros & Cons**
  - \+ Statistically motivated
  - \+ Can make compromises
  - \+ Takes the structure of conditional probabilities into account
  - – Computationally hard

Likelihood Score for Structures

First cut approach:

- Use likelihood function

- The likelihood score for a network structure and parameters is

$$L(G, \Theta_G : D) = \prod_m P(x_1[m], \ldots, x_n[m] : G, \Theta_G)$$

$$= \prod_m \prod_i P(x_i[m] \mid Pa_i^G[m] : G, \Theta_{G,i})$$

- Since we know how to maximize parameters from now on we assume

$$L(G : D) = \max_{\Theta_G} L(G, \Theta_G : D)$$

Likelihood Score for Structure (cont.)

Bad news:

- Adding arcs always helps
  - Maximal score attained by fully connected networks
  - Such networks can **overfit** the data ---
    parameters capture the noise in the data

Avoiding Overfitting

"Classic" issue in learning.

Approaches:

- **Restricting the hypotheses space**
  - Limits the overfitting capability of the learner
  - Example: restrict # of parents or # of parameters
- **Minimum description length**
  - Description length measures complexity
  - Prefer models that compactly describes the training data
- **Bayesian methods**
  - Average over all possible parameter values
  - Use prior knowledge

Posterior Score

Using Bayes rule:

Prior over structures

Marginal likelihood

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

Probability of Data

P(D) is the same for all structures G

Can be ignored when comparing structures

Optimization Problem

Input:

- Training data
- Scoring function (including priors, if needed)
- Set of possible structures
  - Including prior knowledge about structure

Output:

- A network (or networks) that maximize the score

Key Property:

- **Decomposability**: the score of a network is a sum of terms.

Difficulty

Theorem: Finding maximal scoring network structure with at most k parents for each variables is NP-hard for $k > 1$

Heuristic Search

We address the problem by using heuristic search

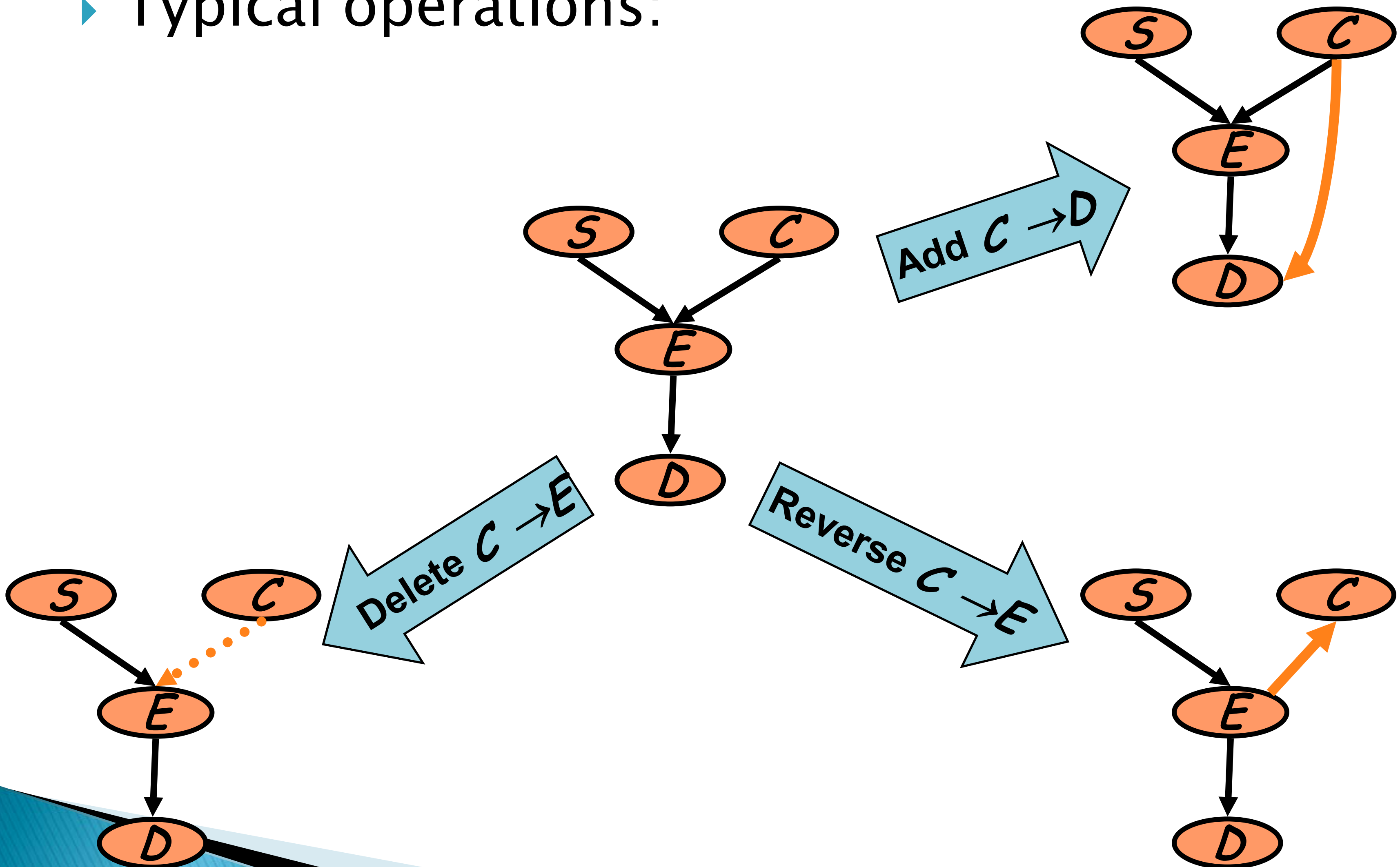
- Define a search space:
  - nodes are possible structures
  - edges denote adjacency of structures
- Traverse this space looking for high-scoring structures

Search techniques:

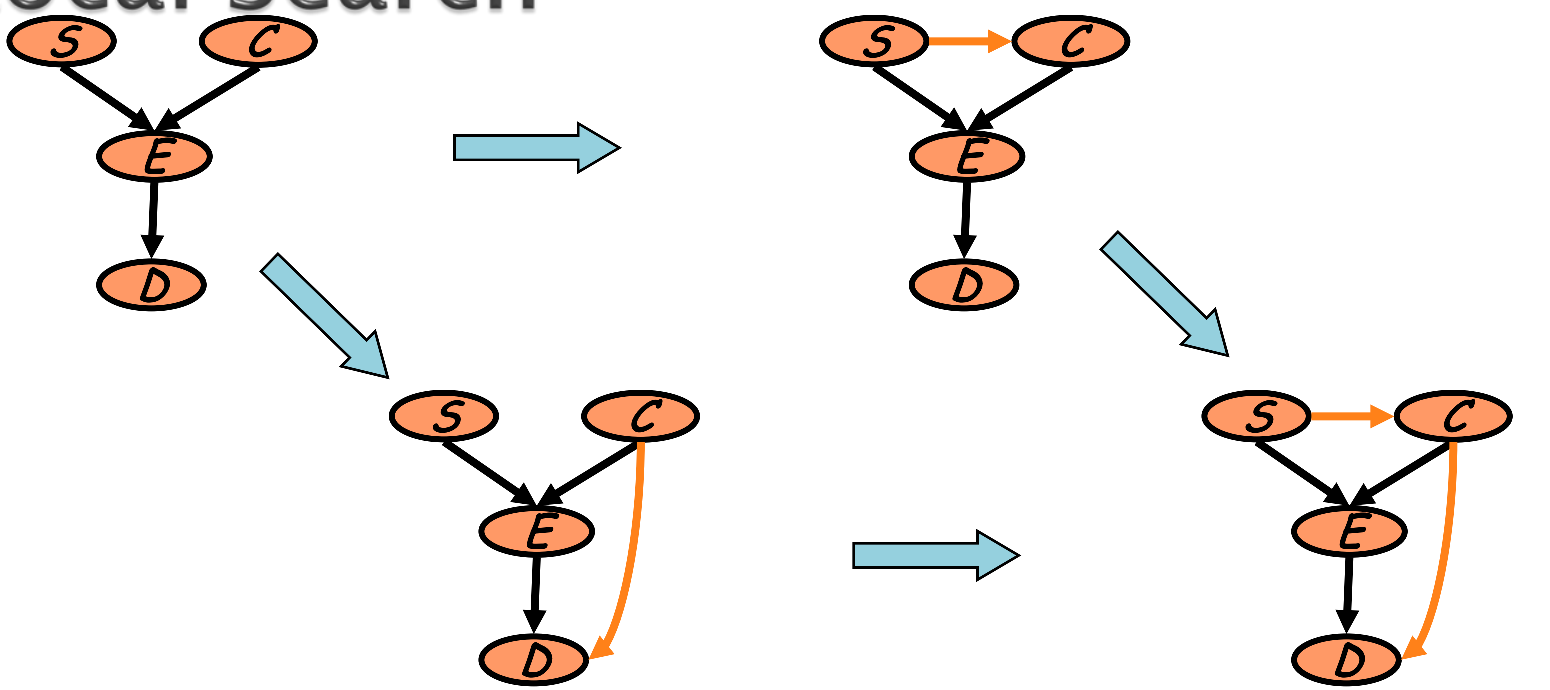
- Greedy hill-climbing
- Best first search
- Simulated Annealing
- ...

Heuristic Search (cont.)

- Typical operations:



Exploiting Decomposability in Local Search



- **Caching:** To update the score of after a local change, we only need to re-score the families that were changed in the last move

Greedy Hill-Climbing

- Simplest heuristic local search
  - Start with a given network
    - empty network
    - best tree
    - a random network
  - At each iteration
    - Evaluate all possible changes
    - Apply change that leads to best improvement in score
    - Reiterate
  - Stop when no modification improves score
- Each step requires evaluating approximately n new changes

Greedy Hill-Climbing: Possible Pitfalls

- Greedy Hill-Climbing can get struck in:
  - **Local Maxima:**
    - All one-edge changes reduce the score
  - **Plateaus:**
    - Some one-edge changes leave the score unchanged
    - Happens because equivalent networks received the same score and are neighbors in the search space
- Both occur during structure search
- Standard heuristics can escape both
  - Random restarts
  - TABU search

Search: Summary

- Discrete optimization problem
- In general, NP-Hard
  - Need to resort to heuristic search
  - In practice, search is relatively fast (~100 vars in ~10 min):
    - Decomposability
    - Sufficient statistics
- Other techniques, model averaging, etc.

- In some cases, we can reduce the search problem to an easy optimization problem
  - Example: learning trees

Incomplete Data

Incomplete Data

Data is often **incomplete**

- Some variables of interest are not assigned value

This phenomena happens when we have

- Missing values
- Hidden variables

Missing Values

Example:

- Medical records – not all patients undergo all possible tests

Complicating issue:

- The fact that a value is missing might be indicative of its value
  - The patient did not undergo X-Ray since she complained about fever and not about broken bones….

To learn from incomplete data we need the following assumption:

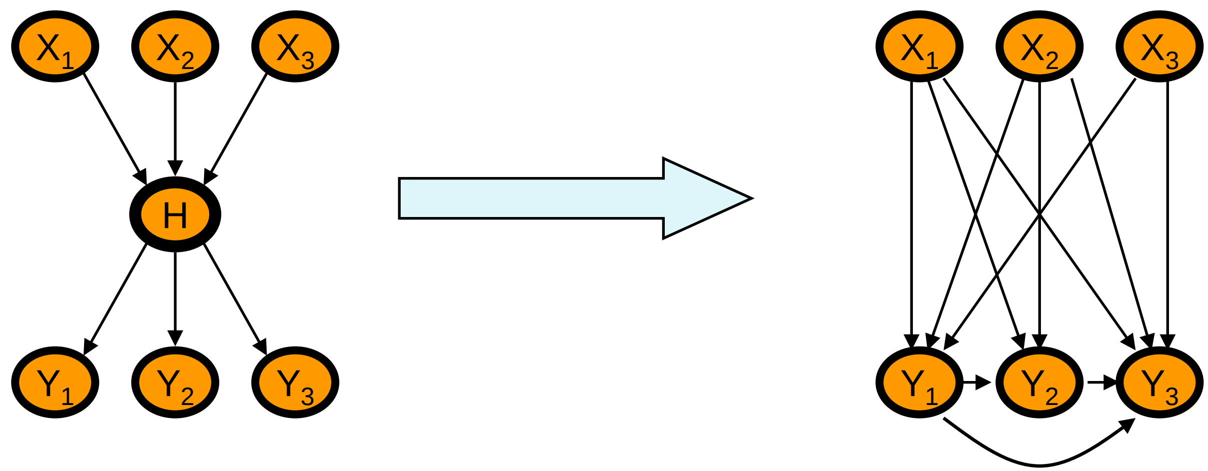
Missing at Random (MAR):

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

Hidden (Latent) Variables

- Attempt to learn a model with variables we never observe
  - In this case, MAR always holds

- Why should we care about unobserved variables?



17 parameters

59 parameters

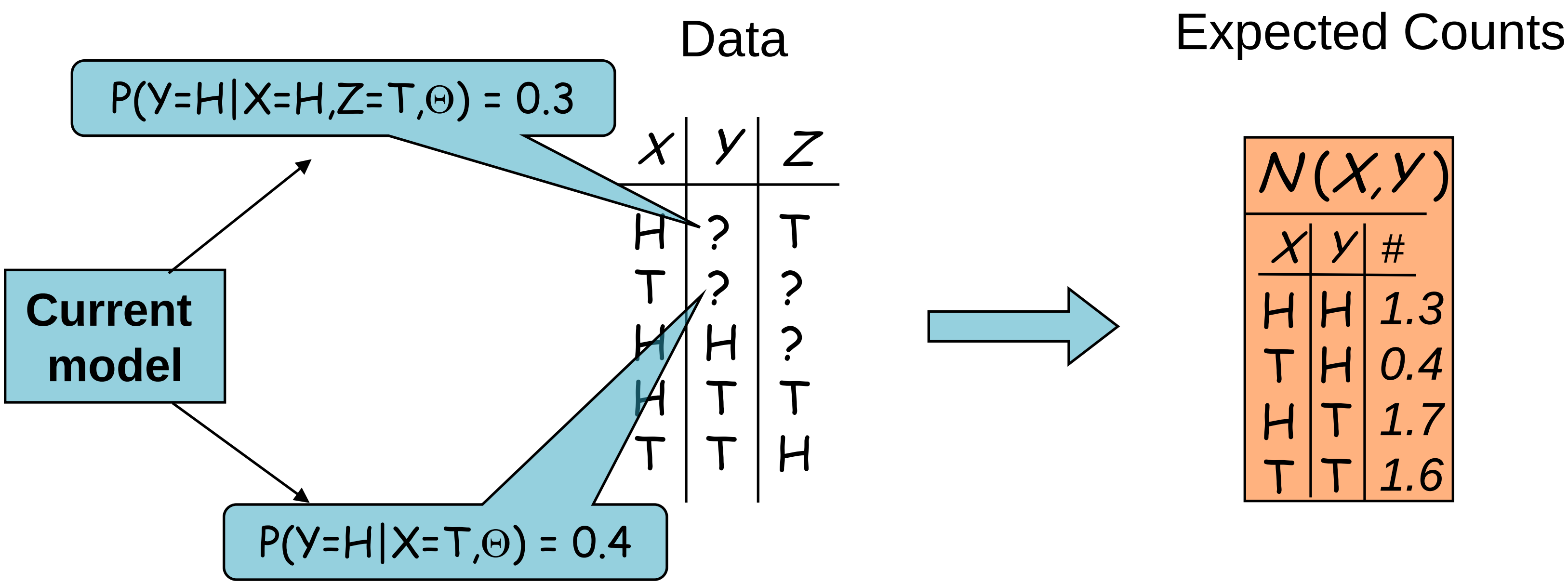
Approach: Expectation Maximization (EM, our old friend)

Recall: general purpose method for learning from incomplete data

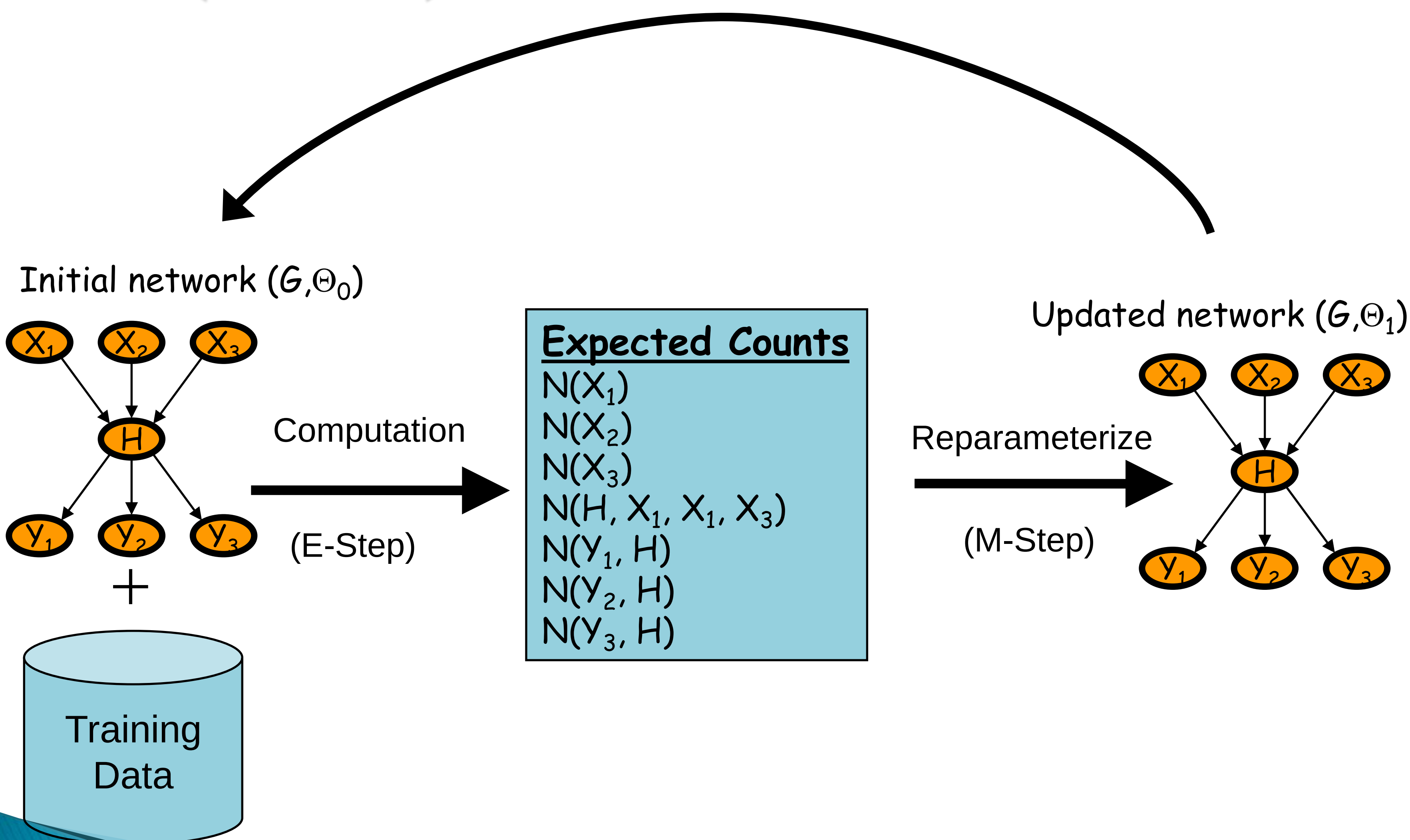
Intuition:

- If we had access to counts, then we can estimate parameters
- However, missing values do not allow us to perform counts
- "Complete" counts using current parameter assignment

Expectation Maximization (EM)



EM (cont.)



Reiterate
Initial network (G,Θ_0)
X_1
X_2
X_3
H
Y_1
Y_2
Y_3
+
Training Data
Computation
(E-Step)
Expected Counts
$N(X_1)$
$N(X_2)$
$N(X_3)$
$N(H, X_1, X_1, X_3)$
$N(Y_1, H)$
$N(Y_2, H)$
$N(Y_3, H)$
Reparameterize
(M-Step)
Updated network (G,Θ_1)
X_1
X_2
X_3
H
Y_1
Y_2
Y_3

EM (cont.)

Formal Guarantees:

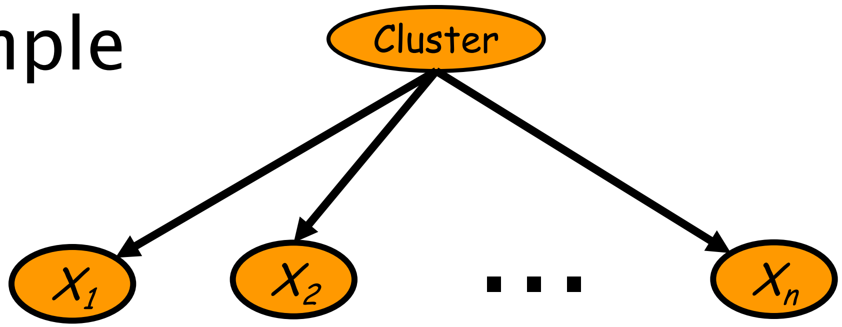
- $L(\Theta_1{:}D) \geq L(\Theta_0{:}D)$
  - Each iteration improves the likelihood
- If $\Theta_1 = \Theta_0$, then Θ_0 is a **stationary point** of $L(\Theta{:}D)$
  - Usually, this means a local maximum

Main cost:

- Computations of expected counts in E-Step
- Requires a computation pass for each instance in training set

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 $P(X_i|C)$, $P(C)$

$$E[N(x_i,c)] = \sum_{m, X_i[m]=x_i} P(c \mid x_1[m],\ldots,x_n[m],\Theta)$$

EM in Practice

Initial parameters:

- Random parameters setting
- “Best” guess from other source

Stopping criteria:

- Small change in likelihood of data
- Small change in parameter values

Avoiding bad local maxima:

- Multiple restarts
- Early “pruning” of unpromising ones

Parameter Learning from Incomplete Data: Summary

- Non-linear optimization problem
- Methods for learning: EM and others
  - 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 approximate inference

Summary

- BN Learning
  - Parameter estimation
  - Structure learning
  - Learning with missing values

- Uses all the tools we've seen so far in class!

Next Time....

- Guest Lecture: Hal Daume III
- Structured Prediction
- Reading: handout from Predicting Structured Data book