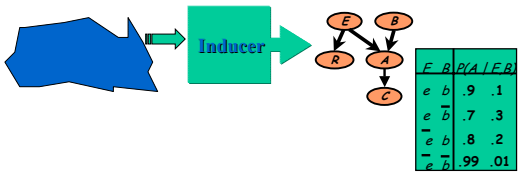
CMSC828G Principles of Data Mining Lecture #16

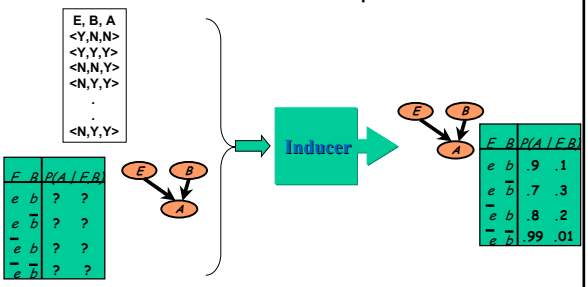
- Readings:
  - handouts
- Today's Lecture:
  - Learning BNs
  - slides courtesy of Nir Friedman
- Upcoming Dates:
  - Project Progress Report due Tuesday 4/9

Learning Bayesian Networks

Learning Bayesian networks

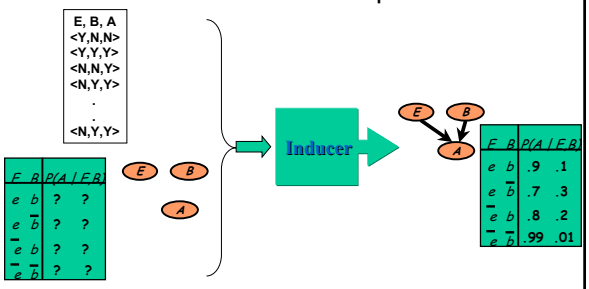


Known Structure -- Complete Data



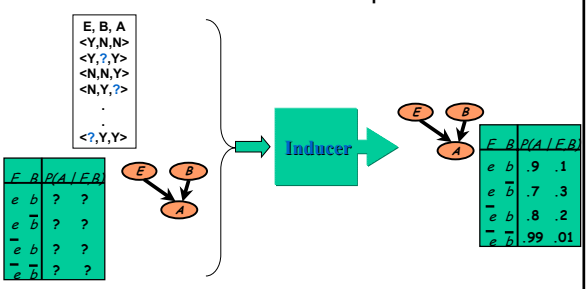
- Network structure is specified
  - Inducer needs to estimate parameters
- Data does not contain missing values

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

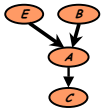
- Given a network structure G
  - And choice of parametric family for $P(X_i|Pa_i)$
- Learn parameters for network

Goal

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

Learning Parameters for a Bayesian Network

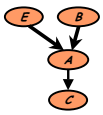
- Training data has the form:



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

Learning Parameters for a Bayesian Network

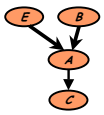
- Since we assume i.i.d. samples, likelihood function is



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

Learning Parameters for a Bayesian Network

- By definition of network, we get



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

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

General Bayesian Networks

Generalizing for any Bayesian network**:**

$$L(\Theta : D) = \prod_m P(x_1[m], \ldots, x_n[m] : \Theta)$$ i.i.d. samples

$$= \prod_m \prod_i P(x_i[m] \mid Pa_i[m] : \Theta_i)$$ Network factorization

$$= \prod_i \prod_m P(x_i[m] \mid Pa_i[m] : \Theta_i)$$

$$= \prod_i L_i(\Theta_i : D)$$

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

General Bayesian Networks (Cont.)

Decomposition

⇒ Independent Estimation Problems

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

Likelihood for Multinomial Networks

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

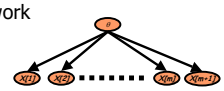
$$L_i(\Theta_i : D) = \prod_{pa_i} \prod_{x_i} \theta_{x_i|pa_i}^{N(x_i, pa_i)}$$

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

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

Bayesian Inference (cont.)

Prediction as **inference** in this network



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

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 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

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

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.)

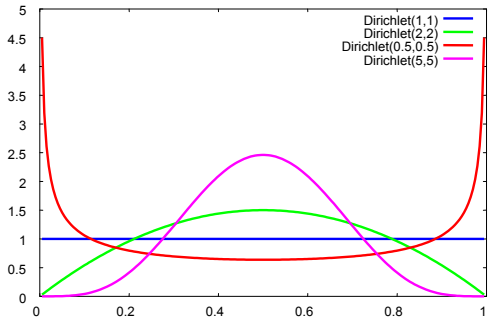
- We can compute the prediction on a new event in closed form:
- If $P(\Theta)$ is Dirichlet with hyperparameters $\alpha_1, \ldots, \alpha_K$ then

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

- Since the posterior is also Dirichlet, we get

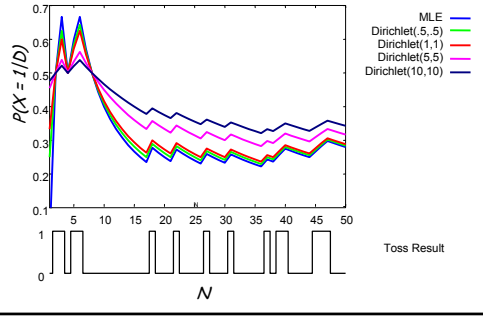
$$P(X[M+1] = k \mid D) = \int \theta_k \, P(\Theta \mid D) d\Theta = \frac{\alpha_k + N_k}{\sum_i (\alpha_i + N_i)}$$

Dirichlet Priors -- Example



Effect of Priors (cont.)

- In real data, Bayesian estimates are less sensitive to noise in the data



Bayesian Prediction(cont.)

- We can compute the posterior for each multinomial $\theta_{X_i|pa_i}$ 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

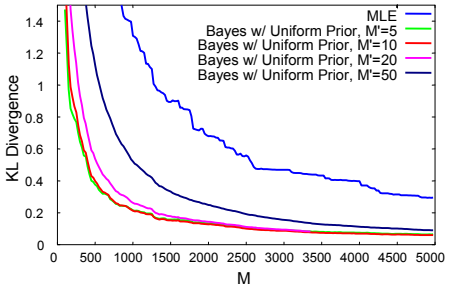
$$\tilde{\theta}_{x_i|pa_i} = \frac{\alpha(x_i, pa_i) + N(x_i, pa_i)}{\alpha(pa_i) + N(pa_i)}$$

Learning Parameters: Case Study (cont.)

Experiment:

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

Learning Parameters: Case Study (cont.)



Learning Parameters: Summary

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

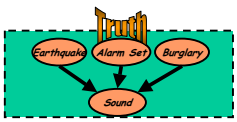
$$\hat{\theta}_{x_i|pa_i} = \frac{N(x_i, pa_i)}{N(pa_i)} \quad \text{MLE}$$

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

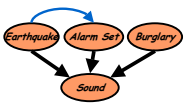
- Bayesian methods also require choice of priors
- Both MLE and Bayesian are asymptotically equivalent and consistent
- Both can be implemented in an **on-line** manner by accumulating sufficient statistics

Learning Structure from Complete Data

Why Struggle for Accurate Structure?

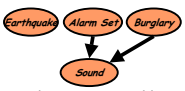


Adding an arc



- 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

- Recall, 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 we assume

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

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

Bayesian Inference

- Bayesian Reasoning---compute expectation over unknown *G*

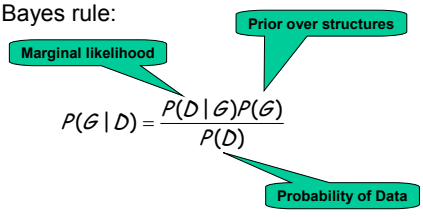
$$P(x[M+1] \mid D) = \sum_G P(x[M+1] \mid D,G)P(G \mid D)$$

- **Assumption**: *G*s are mutually exclusive and exhaustive

- We know how to compute *P(x[M+1]|G,D)*
  - Same as prediction with fixed structure
- How do we compute *P(G|D)*?

Posterior Score

Using Bayes rule:



P(D) is the same for all structures G
Can be ignored when comparing structures

Marginal Likelihood

- By introduction of variables, we have that

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

Likelihood

Prior over parameters

Marginal Likelihood: Binomial case

Assume we observe a sequence of coin tosses….

- By the chain rule we have:

$$P(x[1],\ldots,x[M]) = P(x[1])P(x[2] \mid x[1]) \cdots P(x[M] \mid x[1],\ldots,x[M-1])$$

recall that

$$P(x[m+1] = H \mid x[1],\ldots,x[m]) = \frac{N_H^m + \alpha_H}{m + \alpha}$$

where N^m_H is the number of heads in first m examples.

Marginal Likelihood: Binomials (cont.)

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

We simplify this by using

$$(\alpha)(1+\alpha)\cdots(N-1+\alpha) = \frac{\Gamma(N+\alpha)}{\Gamma(\alpha)}$$

Thus

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

$$\frac{\Gamma(\alpha)}{\Gamma(\alpha + N)} \frac{\Gamma(\alpha_H + N_H)}{\Gamma(\alpha_H)} \frac{\Gamma(\alpha_T + N_T)}{\Gamma(\alpha_T)}$$

Marginal Likelihood: Multinomials

The same argument generalizes to multinomials with Dirichlet prior

- $P(\Theta)$ is Dirichlet with hyperparameters $\alpha_1,\ldots,\alpha_K$
- D is a dataset with sufficient statistics $N_1,\ldots,N_K$

Then

$$P(D) = \frac{\Gamma\left(\sum_\ell \alpha_\ell\right)}{\Gamma\left(\sum_\ell (\alpha_\ell + N_\ell)\right)} \prod_\ell \frac{\Gamma(\alpha_\ell + N_\ell)}{\Gamma(\alpha_\ell)}$$

Marginal Likelihood for General Network

The marginal likelihood has the form:

$$P(D \mid G) = \prod_i \prod_{pa_i^G} \underbrace{\frac{\Gamma\left(\alpha(pa_i^G)\right)}{\Gamma\left(\alpha(pa_i^G) + N(pa_i^G)\right)} \prod_{x_i} \frac{\Gamma(\alpha(x_i, pa_i^G) + N(x_i, pa_i^G))}{\Gamma(\alpha(x_i, pa_i^G))}}$$

Dirichlet Marginal Likelihood
For the sequence of values of X_i when
X_is parents have a particular value

where

- $N(..)$ are the counts from the data
- $\alpha(..)$ are the hyperparameters for each family **given** G

Priors

- We need: prior counts $\alpha(..)$ for each network structure G
- This can be a formidable task
  - There are exponentially many structures…

BDe Score

Possible solution: The **BDe prior**

- Represent prior using two elements M_0, B_0
  - M_0 - **equivalent sample size**
  - B_0 - network representing the prior probability of events

BDe Score

Intuition: M_0 prior examples distributed by B_0

- Set $\alpha(x_i, pa_i^G) = M_0 P(x_i, pa_i^G \mid B_0)$
  - Note that pa_i^G are not the same as the parents of X_i in B_0
  - Compute $P(x_i, pa_i^G \mid B_0)$ using standard inference procedures
- Such priors have desirable theoretical properties
  - Equivalent networks are assigned the same score

Bayesian Score: Asymptotic Behavior

Theorem: If the prior $P(\Theta \mid G)$ is "well-behaved", then

$$\log P(D \mid G) = l(G : D) - \frac{\log M}{2} \dim(G) + O(1)$$

Asymptotic Behavior: Consequences

$$\log P(D \mid G) = l(G : D) - \frac{\log M}{2} \dim(G) + O(1)$$

- Bayesian score is **consistent**
  - As $M \to \infty$ the "true" structure G^* maximizes the score (almost surely)
  - For sufficiently large M, the maximal scoring structures are **equivalent** to G^*
- Observed data eventually overrides prior information
  - Assuming that the prior assigns positive probability to all cases

Asymptotic Behavior

$$\text{Score}(G : D) = l(G : D) - \frac{\log M}{2} \dim(G)$$

- This score can also be justified by the **Minimal Description Length** **(MDL)** principle
- This equation explicitly shows the tradeoff between
  - Fitness to data --- likelihood term
  - Penalty for complexity --- regularization term

Scores -- Summary

- Likelihood, MDL, (log) BDe have the form

$$Score(G : D) = \sum_i Score(X_i \mid Pa_i^G : N(X_i, Pa_i))$$

- BDe requires assessing prior network. It can naturally incorporate prior knowledge and previous experience
- BDe is consistent and asymptotically equivalent (up to a constant) to MDL
- All are **score-equivalent**
  - G equivalent to $G' \Rightarrow Score(G) = Score(G')$

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 $k > 1$

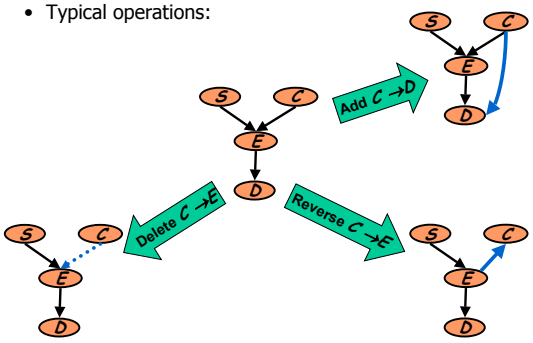
Heuristic Search

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

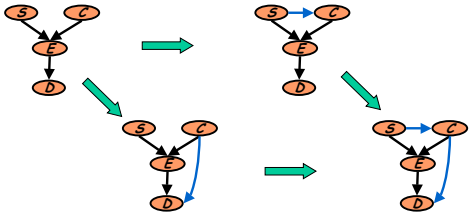
Search techniques:
- Greedy hill-climbing
- Best first search
- Simulated Annealing
- ...

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

Model Selection

- So far, we focused on single model
  - Find best scoring model
  - Use it to predict next example
- Implicit assumption:
  - Best scoring model dominates the weighted sum
- **Pros:**
  - We get a single structure
  - Allows for efficient use in our tasks
- **Cons:**
  - We are committing to the independencies of a particular structure
  - Other structures might be as probable given the data

Model Averaging

- Recall, Bayesian analysis started with

$$P(x[M+1] \mid D) = \sum_{G} P(x[M+1] \mid D, G) P(G \mid D)$$

  - This requires us to average over all possible models

Model Averaging (cont.)

- **Full Averaging**
  - Sum over all structures
  - Usually intractable---there are exponentially many structures
- **Approximate Averaging**
  - Find K largest scoring structures
  - Approximate the sum by averaging over their prediction
  - Weight of each structure determined by the **Bayes Factor**

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

The actual score we compute

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

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

Examples:

- Survey data
- Medical records
  - Not all patients undergo all possible tests

Missing Values (cont.)

Complicating issue:

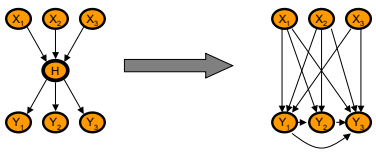
- The fact that a value is missing might be indicative of its value
  - The patient did not undergo X-Ray since she complained about fever and not about broken bones….
- To learn from incomplete data we need the following assumption:

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?

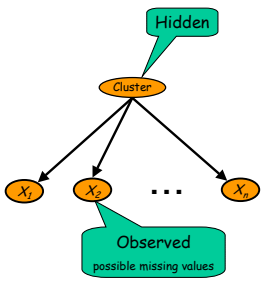


17 parameters

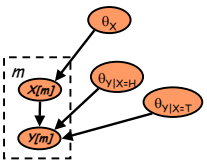
59 parameters

Hidden Variables (cont.)

- Hidden variables also appear in **clustering**
- **Naïve Bayes Model**:
  - Hidden variable assigns class label
  - Observed attributes are independent given the class



Learning Parameters from Incomplete Data

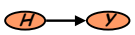


Incomplete data:

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

Learning Parameters from Incomplete Data (cont.).

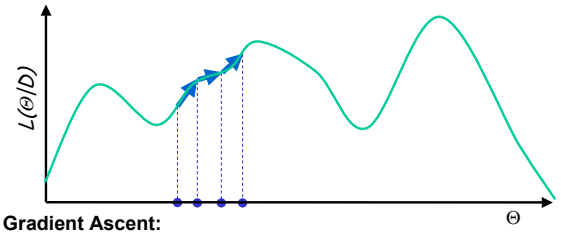
- In the presence of incomplete data, the likelihood can have multiple global maxima
- **Example:**



  - We can rename the values of hidden variable H
  - If H has two values, likelihood has two global maxima
- Similarly, local maxima are also replicated
- Many hidden variables ⇒ a serious problem

MLE from Incomplete Data

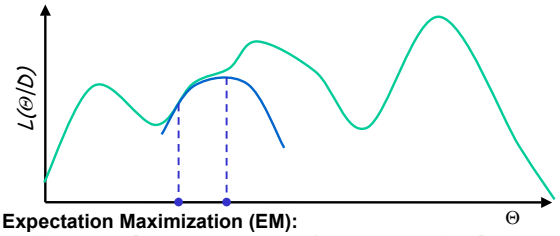
- Finding MLE parameters: **nonlinear optimization** problem



Gradient Ascent:
Follow gradient of likelihood w.r.t. to parameters
Add line search and conjugate gradient methods to get fast convergence

MLE from Incomplete Data

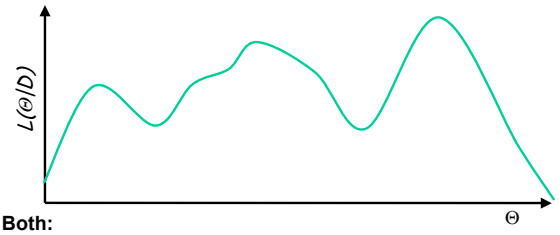
- Finding MLE parameters: **nonlinear optimization** problem



Expectation Maximization (EM):
Use "current point" to construct alternative function (which is "nice")
Guarantee: maximum of new function is better scoring the current point

MLE from Incomplete Data

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



Both:
Find **local** maxima.
Require multiple restarts to find approx. to the global maximum
Require computations in each iteration

Gradient Ascent (cont)

- Putting all together we get

$$\frac{\partial \log P(D \mid \Theta)}{\partial \theta_{x_i, pa_i}} = \sum_m \frac{1}{P(o[m] \mid \Theta)} \frac{\partial P(o[m] \mid \Theta)}{\partial \theta_{x_i, pa_i}}$$

$$= \sum_m \frac{1}{P(o[m] \mid \Theta)} \frac{P(x_i, pa_i, o[m] \mid \Theta)}{\theta_{x_i, pa_i}}$$

$$= \sum_m \frac{P(x_i, pa_i \mid o[m], \Theta)}{\theta_{x_i, pa_i}}$$

Gradient Ascent Summary

Pros:

- Flexible
  - Allows to learn other parameterizations using chain-rule of partial derivatives
- Closely related to methods in neural network training

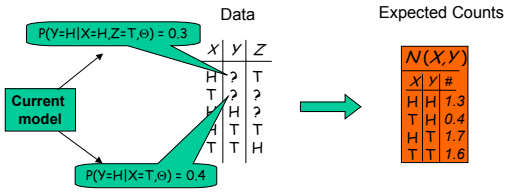
Expectation Maximization (EM)

- A general purpose method for learning from incomplete data

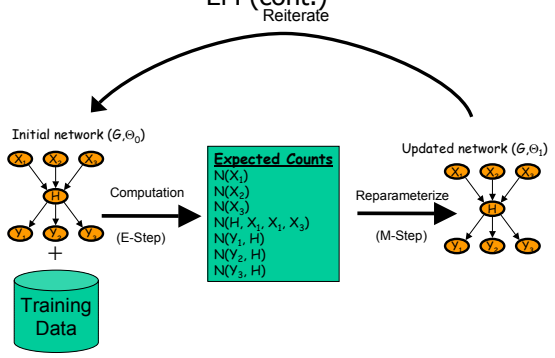
Intuition:

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

Expectation Maximization (EM)



EM (cont.)



EM (cont.)

Formal Guarantees:

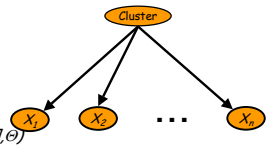
- $L(\Theta_1:D) \geq L(\Theta_0:D)$
  - Each iteration improves the likelihood
- If $\Theta_1 = \Theta_0$, then Θ_0 is a **stationary point** of $L(\Theta:D)$
  - Usually, this means a local maximum

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

Incomplete Data : Structure Scores

MDL

$$MDL(G:D) = I(G:D) - \frac{\log M}{2}\dim(G) - DL(G)$$

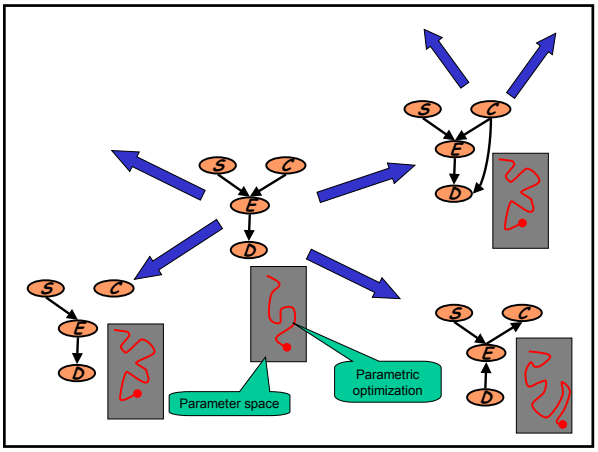
- Use same MDL formula with probability of the data
- Requires finding maximum likelihood parameters
  - Using methods for parameter learning (e.g., EM)
- Theoretical results show that penalty should be adjusted

Incomplete Data : Structure Scores (cont.)

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

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

- We cannot evaluate the marginal likelihood
- We have to resort to approximations:
  - Asymptotic approximations
    - Evaluate score around MAP parameters
    - Need to find MAP parameters (e.g., EM)
  - Stochastic approximations
    - Apply stochastic integration methods
    - Much slower



Problem

Such procedures are computationally expensive!

- Computation of optimal parameters, per candidate, requires non-trivial optimization step
- Spend non-negligible computation on a candidate, even if it is a low scoring one

In practice, such learning procedures are feasible only when we consider small sets of candidate structures

Structural EM

- **Idea:** Use parameters found for previous structures to help evaluate new structures.
- **Scope**: searching over structures over the same set of random variables.

Outline:

- Perform search in (Structure, Parameters) space.
- Use EM-like iterations, using previously best found solution as a basis for finding either:
  - Better scoring parameters --- "parametric" EM step

  or
  - Better scoring structure --- "structural" EM step

Learning Structure from Incomplete Data: Summary

- Hard problem!
- Initial progress:
  - EM-like search techniques
    - 6 CPU years ⇒ 6 CPU hours
- Problems:
  - Escaping local maxima
  - Inducing new variables

References

- *Principles of Data Mining*, Hand, Mannila, Smyth. MIT Press, 2001.
- Nir Friedman's excellent lecture notes, http://www.cs.huji.ac.il/~pmai/