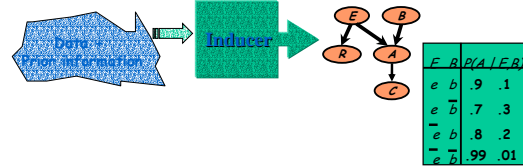




cmsc726: Graphical Models: Learning

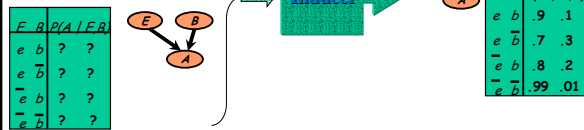
material from: Michael Jordan, Nir Friedman and Daphne Koller

Learning Bayesian networks



Known Structure -- Complete Data

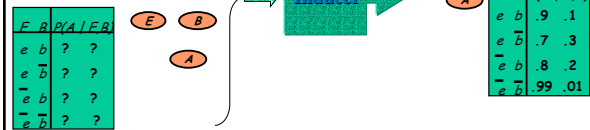
E, B, A
 <Y,N,N>
 <Y,Y,Y>
 <N,N,Y>
 <N,Y,Y>
 .
 <N,Y,Y>



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

Unknown Structure -- Complete Data

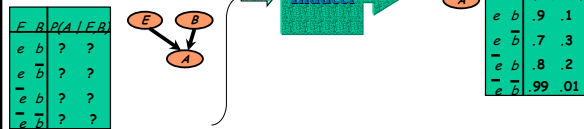
E, B, A
 <Y,N,N>
 <Y,Y,Y>
 <N,N,Y>
 <N,Y,Y>
 .
 <N,Y,Y>



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

Known Structure -- Incomplete Data

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



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

Known Structure / Complete Data

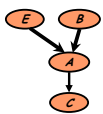
- Given a network structure G
 - And choice of parametric family for $P(X_i|Pa_i)$
- Learn parameters for network

Goal

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

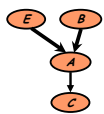
Learning Parameters for a Bayesian Network

- Training data has the form:

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


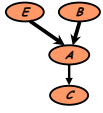
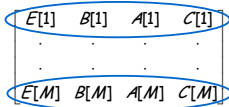
Learning Parameters for a Bayesian Network

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

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


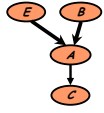
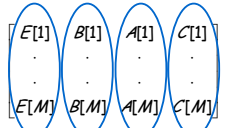
Learning Parameters for a Bayesian Network

- By definition of network, we get

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



Learning Parameters for a Bayesian Network

- Rewriting terms, we get

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



General Bayesian Networks

Generalizing for any Bayesian network:

$$\begin{aligned} \mathcal{L}(\Theta : D) &= \prod_m \mathcal{P}(x_1[m], \dots, x_n[m] : \Theta) \quad \text{i.i.d. samples} \\ &= \prod_m \prod_i \mathcal{P}(x_i[m] | \rho_{a_i}[m] : \Theta_i) \quad \text{Network factorization} \\ &= \prod_i \prod_m \mathcal{P}(x_i[m] | \rho_{a_i}[m] : \Theta_i) \\ &= \prod_i \mathcal{L}_i(\Theta_i : D) \end{aligned}$$

- 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 $1, 2, \dots, K$
- We want to learn the parameters $\theta_1, \theta_2, \dots, \theta_K$

Sufficient statistics:

- N_1, N_2, \dots, N_K - the number of times each outcome is observed

Likelihood function:

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

$$\hat{\theta}_k = \frac{N_k}{\sum_l N_l}$$

Likelihood for Multinomial Networks

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

$$\begin{aligned} L_i(\Theta_i : D) &= \prod_m P(x_i[m] | Pa_i[m] : \Theta_i) \\ &= \prod_{pa_i, m, pa_i[m]=pa_i} P(x_i[m] | pa_i : \Theta_i) \\ &= \prod_{pa_i} \prod_{x_i} P(x_i | pa_i : \Theta_i)^{N(x_i, pa_i)} \\ &= \prod_{pa_i} \prod_{x_i} \theta_{x_i | pa_i}^{N(x_i, pa_i)} \end{aligned}$$

Likelihood for Multinomial Networks

- When we assume that $P(X_i / Pa_i)$ 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

Frequentist Approach:

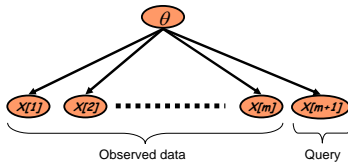
- Assumes there is an unknown but fixed parameter θ
- Estimates θ with some confidence
- Prediction by using the estimated parameter value

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, $P(x[m] | \theta)$, are the parameters in the model
- Prediction is now inference in this network

Bayesian Inference (cont.)

Prediction as **inference** in this network

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

where

$$P(\theta | x[1], \dots, x[M]) = \frac{\text{Likelihood} \cdot \text{Prior}}{\text{Posterior} \cdot \text{Probability of data}}$$

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 $\alpha_1, \dots, \alpha_K$ is defined as

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

Then the posterior has the same form, with hyperparameters

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

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

Dirichlet Priors (cont.)

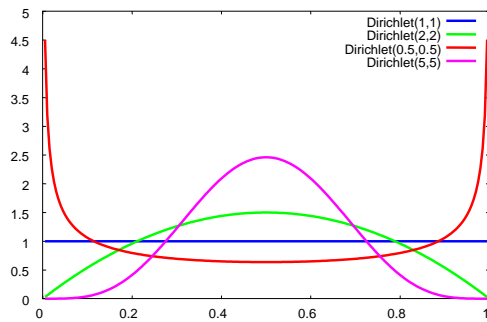
- We can compute the prediction on a new event in closed form:
- If $P(\theta)$ is Dirichlet with hyperparameters $\alpha_1, \dots, \alpha_K$ then

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

- Since the posterior is also Dirichlet, we get

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

Dirichlet Priors -- Example

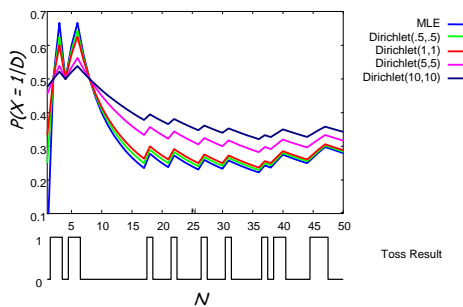


Prior Knowledge

- The hyperparameters $\alpha_1, \dots, \alpha_K$ can be thought of as "imaginary" counts from our prior experience
- Equivalent sample size = $\alpha_1 + \dots + \alpha_K$
- The larger the **equivalent sample size** the more confident we are in our prior

Effect of Priors (cont.)

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



Conjugate Families

- The property that the posterior distribution follows the same parametric form as the prior distribution is called **conjugacy**
 - Dirichlet prior is a **conjugate family** for the multinomial likelihood
- Conjugate families are useful since:
 - For many distributions we can represent them with hyperparameters
 - They allow for sequential update within the same representation
 - In many cases we have closed-form solution for prediction

Bayesian Networks and Bayesian Prediction

Observed data Query Plate notation

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

Bayesian Networks and Bayesian Prediction (Cont.)

Observed data Query Plate notation

- We can also “read” from the network:
Complete data \Rightarrow **posteriors on parameters are independent**

Bayesian Prediction(cont.)

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

Refined model

- Complete data \Rightarrow independent posteriors on $\theta_{y|x=0}$ and $\theta_{y|x=1}$

Bayesian Prediction(cont.)

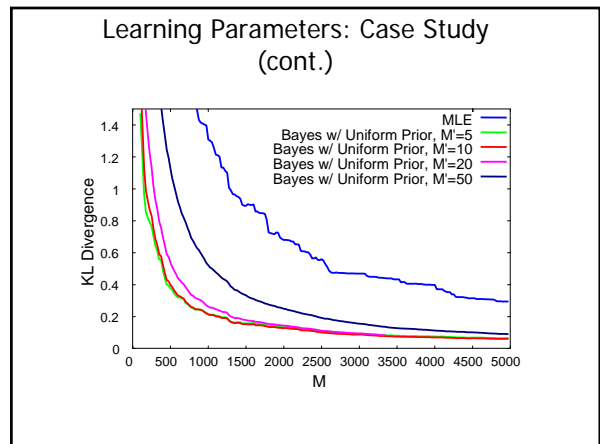
- Given these observations, we can compute the posterior for each multinomial $\theta_{x_i | p_{a_i}}$ independently
 - The posterior is Dirichlet with parameters $\alpha(X_i=1|p_{a_i})+N(X_i=1|p_{a_i}), \dots, \alpha(X_i=k|p_{a_i})+N(X_i=k|p_{a_i})$
- The predictive distribution is then represented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

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

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: Summary

- Estimation relies on **sufficient statistics**

– For multinomial these are of the form $N(x_i, pa_i)$

– Parameter estimation

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

MLE

Bayesian (Dirichlet)

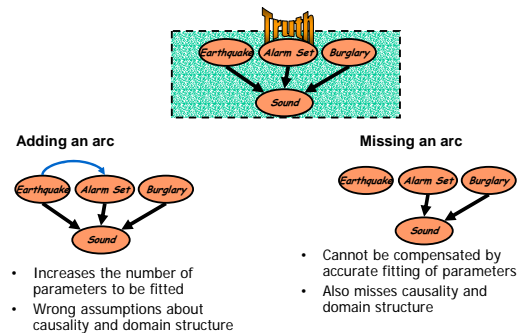
- Bayesian methods also require choice of priors
- Both MLE and Bayesian are asymptotically equivalent and consistent
- Both can be implemented in an **on-line** manner by accumulating sufficient statistics

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?



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

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

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

- 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

Bayesian Inference

- Bayesian Reasoning---compute expectation over unknown G

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

- **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:

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

Diagram labels: **Marginal likelihood** (points to $P(G | D)$), **Prior over structures** (points to $P(G)$), **Probability of Data** (points to $P(D)$)

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

Marginal Likelihood

- By introduction of variables, we have that

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

Likelihood

Prior over parameters

- This integral measures sensitivity to choice of parameters

Marginal Likelihood: Multinomials

For multinomials with Dirichlet prior:

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

$$P(D) = \frac{\Gamma\left(\sum_i \alpha_i\right)}{\Gamma\left(\sum_i (\alpha_i + N_i)\right)} \prod_i \frac{\Gamma(\alpha_i + N_i)}{\Gamma(\alpha_i)}$$

Marginal Likelihood for General Network

The marginal likelihood has the form:

$$P(D | \mathcal{G}) = \prod_i \prod_{pa_i^{\mathcal{G}}} \frac{\Gamma(\alpha(pa_i^{\mathcal{G}}))}{\Gamma(\alpha(pa_i^{\mathcal{G}}) + N(pa_i^{\mathcal{G}}))} \prod_{x_i} \frac{\Gamma(\alpha(x_i, pa_i^{\mathcal{G}}) + N(x_i, pa_i^{\mathcal{G}}))}{\Gamma(\alpha(x_i, pa_i^{\mathcal{G}}))}$$

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

where

- $N(\dots)$ are the counts from the data
- $\alpha(\dots)$ are the hyperparameters for each family **given**
 \mathcal{G}

Priors

- We need: prior counts $\alpha(\dots)$ for each network structure \mathcal{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 $\alpha(x_i, pa_i^{\mathcal{G}}) = M_0 P(x_i, pa_i^{\mathcal{G}} | B_0)$
 - Note that $pa_i^{\mathcal{G}}$ are not the same as the parents of X_i in B_0
 - Compute $P(x_i, pa_i^{\mathcal{G}} | B_0)$ 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 $P(\theta | \mathcal{G})$ is "well-behaved", then

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

Asymptotic Behavior: Consequences

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

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

Asymptotic Behavior

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

- 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
$$\text{Score}(\mathcal{G} : D) = \sum_i \text{Score}(X_i | Pa_i^{\mathcal{G}} : N(X_i, Pa_i))$$
- 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**
 - \mathcal{G} equivalent to $\mathcal{G}' \Rightarrow \text{Score}(\mathcal{G}) = \text{Score}(\mathcal{G}')$

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 $k > 1$

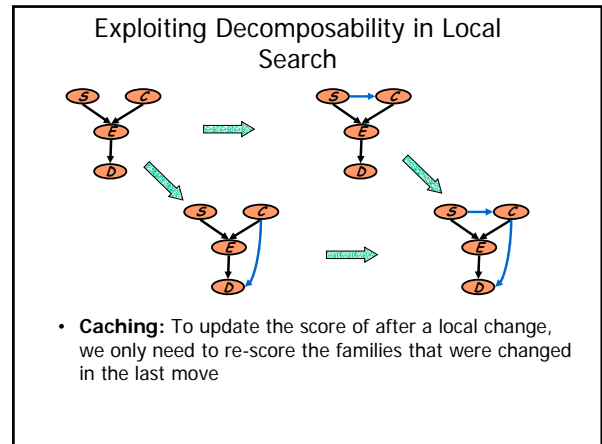
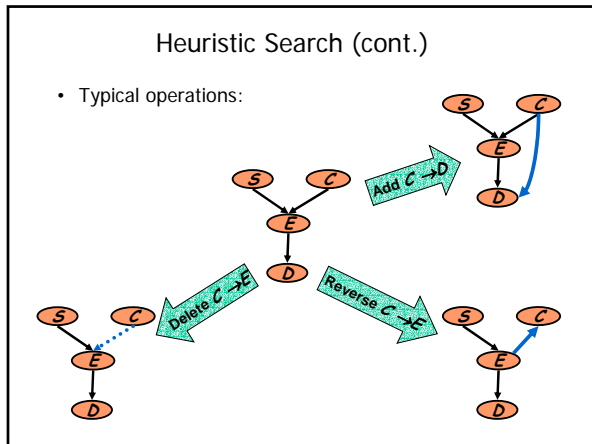
Heuristic Search

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
- ...



- ### 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 stuck 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

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

- 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

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

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 X_i is missing is independent of its actual value given other observed values

Missing Values (cont.)

- 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 omissions)
- We might not be able to learn about the underlying process

| Data | | | Augmented Data | | | | | |
|------|---|---|----------------|---|---|-------|-------|-------|
| X | Y | Z | X | Y | Z | Obs-X | Obs-Y | Obs-Z |
| H | ? | T | H | ? | T | Y | N | Y |
| T | ? | ? | T | ? | ? | Y | N | N |
| H | H | ? | H | H | ? | Y | Y | N |
| H | T | T | H | T | T | Y | Y | Y |
| T | T | H | T | T | H | Y | Y | Y |

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

Hidden: Cluster
Observed: X_1, X_2, \dots, X_n
possible missing values

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

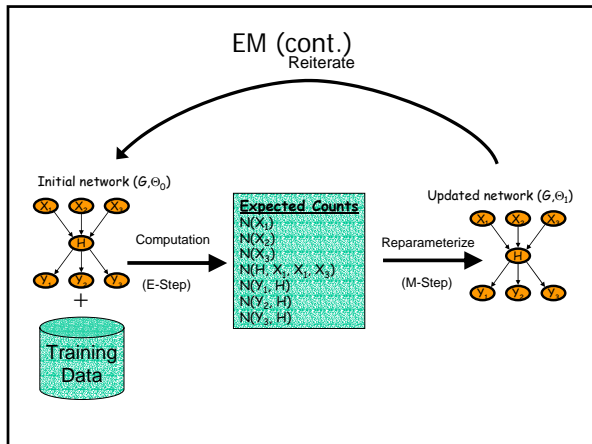
Expectation Maximization (EM)

- 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)

| Data | | | Expected Counts | | |
|------|---|---|-----------------|-----|-----|
| X | Y | Z | X | Y | Z |
| H | ? | T | 7.3 | 0.4 | 1.7 |
| T | ? | ? | 1.7 | 1.6 | |
| H | T | T | | | |
| T | T | H | | | |

Current model: $P(Y=H|X=H,Z=T,\theta) = 0.3$
 Current model: $P(Y=H|X=T,\theta) = 0.4$



EM (cont.)

Formal Guarantees:

- $L(\theta_t; D) \geq L(\theta_{t-1}; D)$
 - Each iteration improves the likelihood
- If $\theta_t = \theta_{t-1}$, then θ_t 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
 - These are exactly the same as for gradient ascent!

Example: EM in clustering

- Consider clustering example

E-Step:

- Compute $P(c[m] | x_1[m], \dots, x_n[m], \theta)$
- This corresponds to "soft" assignment to clusters
- Compute expected statistics:

M-Step

- Re-estimate $P(x_i | c), P(c)$

$$E[N(x_i, c)] = \sum_{m, X_i[m]=x_i} P(c | x_1[m], \dots, x_n[m], \theta)$$

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

Bayesian Inference with Incomplete Data

Recall, Bayesian estimation:

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

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

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

where

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

Assumption:

- Posterior mass is dominated by a MAP parameters

Finding MAP parameters:

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

Stochastic Approximations

Stochastic approximation:

- Sample $\theta_1, \dots, \theta_k$ from $P(\theta/D)$
- Approximate

$$P(\mathbf{x}[M+1] | D) \approx \frac{1}{k} \sum_i P(\mathbf{x}[M+1] | \theta_i)$$

- We can apply Gibbs sampling to perform stochastic simulation in the “meta” network that describes the learning problem

Parameter Learning from Incomplete Data: 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 approximate inference