Learning Bayesian Networks from Incomplete Data
- Parameter Estimation
- Structure Search

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
Missing Values

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….
- Need to specify the mechanism by which data are hidden

Missing at Random (MAR):
- The probability that the value of $X_i$ is missing is independent of its actual value given other observed values

Then we can optimize the likelihood function in the parameters of the distribution $P(X)$ independently of the missing mechanism
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?

Parameter Estimation

- Network structure is specified
- \( x[m] = (o[m], h[m]) \)
- Data set \( D = \{ o[1], \ldots, o[M] \} \)
- Estimate network parameters
- The likelihood function

\[
L(\theta : D) = P(D \mid \theta) = \prod_{m=1}^{M} P(o[m] : \theta)
\]
The Likelihood Function

- Complete data
  \[ P(X[1],Y[1]|\Theta) = P(X[1]|\Theta_X)P(Y[1]|X[1],\Theta_{Y,X}) \]

- Incomplete data

- The complete data likelihood is unimodal
- In the presence of incomplete data, the likelihood can have multiple maxima

The Likelihood Function

- The likelihood function
  \[ L(\theta : D) = \prod_{m=1}^{M} P(o[m]:\theta) = \prod_{m=1}^{M} \sum_{h[m]} P(o[m],h[m]:\theta) \]

- In the presence of incomplete data, the likelihood function lost its unimodality, closed-form representation, and decomposability
MLE from Incomplete Data

- Nonlinear function optimization problem over a high-dimensional space

MLE from Incomplete Data

- Use generic function optimization algorithms, such as gradient ascent
  - Need to compute gradient function

- Expectation Maximization (EM)
  - A more specialized approach for optimizing likelihood functions in the presence of incomplete data
Gradient Ascent

- Main result

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

- Requires computation: \( P(x_i, pa_i | o[m], \Theta) \) for all \( i, m \)

- Use standard BN inference algorithm: join tree algorithm, or approximate algorithms

Expectation Maximization (EM)

- A general purpose method for optimizing likelihood functions in probabilistic models in the presence of incomplete data

Intuition:
- If we had true counts, we could estimate parameters
- But with missing values, counts are unknown
- We “complete” counts using probabilistic inference based on current parameter assignment
- We use completed counts as if real to re-estimate parameters
Expectation Maximization (EM)

Data

Expected Counts

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

$P(Y=H|X=H) = 0.4$

Current model

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

Computation (E-Step)

Reparameterize (M-Step)

Reiterate

Initial network $(\mathcal{G}, \Theta_0)$

Expected Counts

$N(X_1)$

$N(X_2)$

$N(X_3)$

$N(H, X_1, X_1, X_3)$

$N(Y_1, H)$

$N(Y_2, H)$

$N(Y_3, H)$

Training Data

Updated network $(\mathcal{G}, \Theta_1)$
Expectation Maximization (EM)

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

Computational bottleneck:
- Computation of expected counts in E-Step
  \[
  \overline{N}(x_i, pa_i) = \sum_m P(x_i, pa_i \mid o[m], \Theta)
  \]
  - These are exactly the same as for gradient ascent!
  - Use standard BN inference algorithm: join tree algorithm, or approximate algorithms
Example: EM in Clustering

- Consider clustering example
- Bayesian clustering (Autoclass model)

E-Step:
- Compute $P(C[m]|X_1[m],...,X_n[m], \Theta)$
- This corresponds to “soft” assignment to clusters
- Compute expected statistics:

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

M-Step
- Re-estimate $P(X_i/C)$, $P(C)$

Summary: Parameter Learning with Incomplete Data

- Incomplete data makes parameter estimation hard
- Likelihood function
  - Does not have closed form
  - Is multimodal

- Finding max likelihood parameters:
  - EM
  - Gradient ascent
- Both exploit inference procedures for Bayesian networks
Structure Learning

- Structure learning are considerably more complicated in the case of incomplete data
  - the scoring function
  - the search procedure

- Hidden variables
  - introducing new hidden variables?
  - where to introduce the hidden variables?
  - open ended ...

Incomplete Data: Structure Scores

Recall, Bayesian score:

\[ P(G | D) \propto P(G)P(D | G) \]

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

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)
**Naive Approach**

- Perform EM for each candidate graph
  - $G_3$
  - $G_2$
  - $G_1$

- Computationally expensive:
  - Parameter optimization via EM — non-trivial
  - Need to perform EM for all candidate structures
  - Spend time even on poor candidates
  - \(\Rightarrow\) In practice, feasible when considers only a few candidates

**Structural EM**

**Idea:**

- Most of the computation results are discarded
- Construct “quick and dirty” estimates of the change in score
- Use current model to help evaluate new structures

*N Friedman, The Bayesian structural EM algorithm, UAI 1998*
Example: Phylogenetic Reconstruction

**Input:** Biological sequences

- Human: CGTTGC...
- Chimp: CCTAGG...
- Orang: CGAACG...
- ....

**Output:** a phylogeny

An “instance” of evolutionary process

Assumption: positions are independent

Application: Classification

**Generative Models:**
- Bayesian classifiers: model $P(F_1, \ldots, F_d, C)$
- Bayesian network classifiers: learn a BN representation for $P(F_1, \ldots, F_d, C)$
  - Difficult task
- Naïve Bayes classifier
Improving Naïve Bayes

- NBC encodes assumptions of independence that may be unreasonable
- Idea: improve on NBC by weakening the independence assumptions
- Augment the Naïve Bayes structure with additional edges between features that capture correlations among them

Tree Augmented Naïve Bayes (TAN)


- Approximate the dependence among features with a tree structure (imposed on the Naïve Bayes structure)
- The procedure for learning tree BNs can be adapted to learn TAN structures
  - Can be learned in polynomial time
  - Robust parameter estimation
- Experimental studies with UCI repository data sets:
  - TAN classifier dominates NBC,
  - competitive with others like C4.5
Bayesian Network Classifiers

- Discriminative learning:
  - TEEMU ROOS, HANNES WETTIG, PETER GRU’NWALD, On Discriminative Bayesian Network Classifiers and Logistic Regression, Machine Learning, 59, 267–296, 2005

Bayesian Network Classifiers

- Learning unrestricted Bayesian networks?
  - Comparing Bayesian Network Classifiers, Jie Cheng and Russell Greiner, Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI-99), 1999
  - Learning Bayesian Belief Network Classifiers: Algorithms and System, Jie Cheng and Russell Greiner, Proceedings of the Canadian Conference on Artificial Intelligence (CSCSI01) 2001
  - Su and Zhang, Full Bayesian network classifiers, Proceedings of the 23rd international conference on Machine learning (ICML), 2006
Software Packages for BNs

Software Packages for Graphical Models / Bayesian Networks

http://www.cs.ubc.ca/~murphyk/Software/bnsoft.html