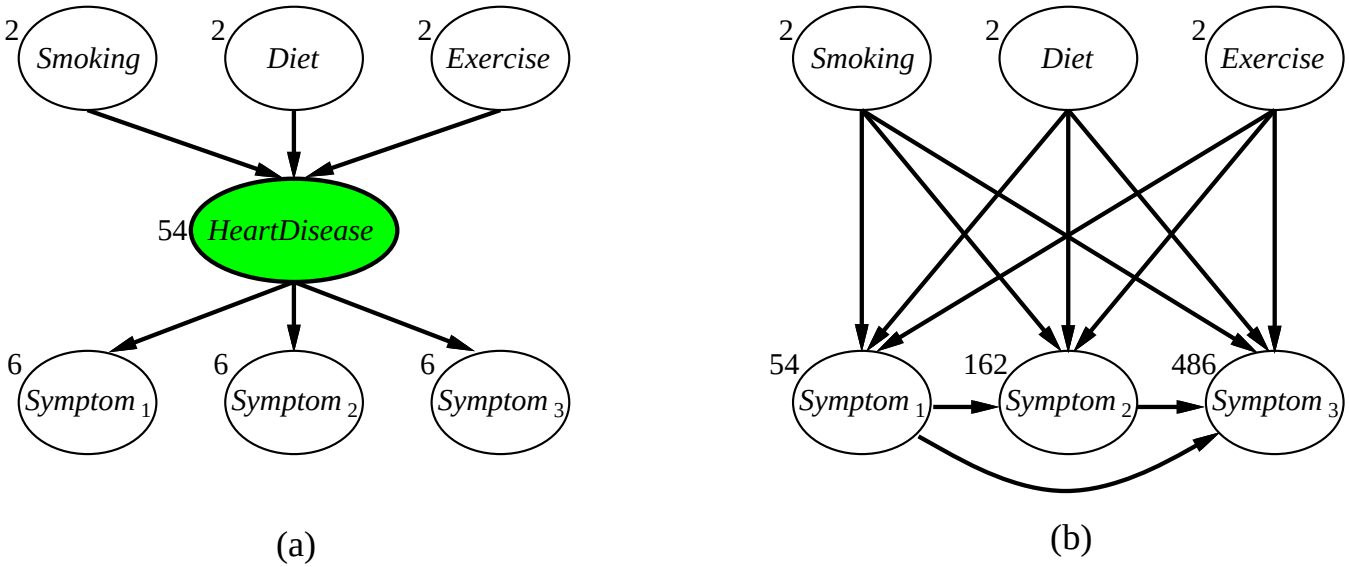
Learning with Hidden Variables

CSci 5512: Artificial Intelligence II

Hidden Variables

- Real world problem have hidden variables
- No training data available on hidden variables
- Model cannot be built without training data
- Inference cannot be done without model
- How to learn models with hidden variables

Example: Diagnostic Network



- Each node has 3 values: none, moderate, severe
- Model with hidden variable has 78 parameters
- Model without hidden variable has 708 parameters
- Hidden variables allow simpler models

Probabilistic Mixture Models

- Consider a mixture model of the form

$$p(\mathbf{x}|\pi,\theta) = \sum_{h=1}^{k} \pi_h p(\mathbf{x}|\theta_h)$$

  - $p(\mathbf{x}|\theta_h)$ is the h^{th} mixing component
  - π_h is the mixing weight
- Mixing component
  - $p(\mathbf{x}|\theta_h)$ is a density function with parameter θ_h
  - Example: Gaussians, Multinomials, Bernoulli
  - Each component corresponds to one cluster
- Mixing weight
  - Forms a probability distribution over components, $\sum_h \pi_h = 1$
  - Relative proportion of each cluster

The Mixture Model Learning Problem

- Given: Data $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$
- Assume:
  - $\mathcal{X}$ is generated by a mixture model
  - k is the number of components in the mixture model
  - $p(\mathbf{x}|\theta_h)$ is from a known (exponential) family
- A point $\mathbf{x}$ is generated as follows
  - Sample component h with probability π_h
  - Sample $\mathbf{x}$ with probability $p(\mathbf{x}|\theta_h)$
- Problem: Find (π, θ) that maximizes

$$p(\mathcal{X}|\pi, \theta) = \prod_{i=1}^{n} p(\mathbf{x}_i|\pi, \theta) \equiv \sum_{i=1}^{n} \log p(\mathbf{x}_i|\pi, \theta)$$

- (π^*, θ^*) best explains the observed data
  - Can be used as a model for the data domain

EM Algorithm for Mixture Models

- Assume $\mathcal{X}$ is generated from a mixture of Gaussians
- Each point $\mathbf{x}$ generated from one component
  - Do not know which component generated $\mathbf{x}$
  - Do not know what the (μ_h, Σ_h) of each component is
- If component assignments were known
  - Can estimate (μ_h, Σ_h) using ML estimates
  - Can estimate π_h using ML estimates
- If parameters $\{\pi_h, (\mu_h, \Sigma_h)\}$ were known
  - Can estimate (probabilistic) component assignments $p(h|\mathbf{x})$
- Learning mixture models
  - Pretend component parameters $\{\pi_h, (\mu_h, \Sigma_h)\}$ are known
  - Estimate component assignments $p(h|\mathbf{x})$ for every point
  - Estimate parameters $\{\pi_h, (\mu_h, \Sigma_h)\}$ based on current assignments
  - Repeat till convergence

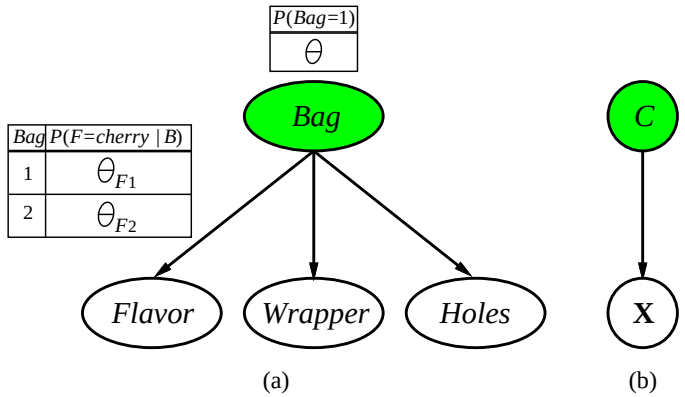
EM Algorithm (Contd.)

- Initialize: Component parameters $\{(\pi_h, (\mu_h, \Sigma_h))\}$
- E-step: Compute $p(h|\mathbf{x}_i)$
  - Bayes Rule $p(h|\mathbf{x}_i) = p_{hi} = \alpha \pi_h p(\mathbf{x}_i|\mu_h, \Sigma_h)$
- M-step: Compute component parameters $p_h = \sum_i p_{hi}$

$$\begin{aligned} \pi_h &= \frac{p_h}{\sum_{h'} p_{h'}} \\ \mu_h &= \sum_i p_{hi} \mathbf{x}_i / p_h \\ \Sigma_h &= \sum_i p_{hi} (\mathbf{x}_i - \mu_h)(\mathbf{x}_i - \mu_h)^T / p_h \end{aligned}$$

- Increases log likelihood at every iteration
- Converges to local minimum (mostly)
- Issues
  - Degenerate solutions can give high likelihood
  - Local minima (or saddle point) may be very bad

Learning Bayesian Networks



- 2 Bags, 2 values for each of the variables
- Sample 1000 candies

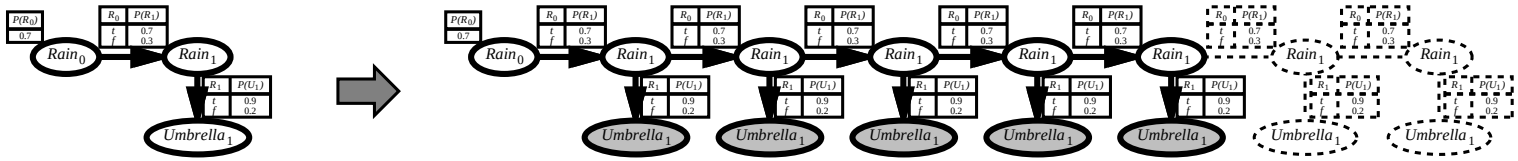
	W=red		W=green	
	H=1	H=0	H=1	H=0
F=cherry	273	93	104	90
F=lime	79	100	94	167

- Estimate the parameters of the Bayesian network
- Estimate which candy came from which bag

Learning Bayesian Networks (Contd.)

- Initialize parameters with random values
- Estimate the probability that a candy came from Bag 1
  - E-step of the EM algorithm
  - Can be solved using exact inference
- After E-step, probabilistic assignments are known
- Estimate the parameters of the network
  - M-step of the EM algorithm
  - ML estimates for discrete distributions

Learning HMMs



- Unrolled dynamic Bayesian network
- Estimate transition probabilities from observation sequences
- The model does not change $\theta_{ijt} = \theta_{ij}$

- Initialize transition probabilities θ_{ij}
- Estimate the number of times a transition happens
  - E-step, can be solved by inference
- Estimate the parameters of the HMM
  - M-step, simple ML estimates
- Baum-Welch algorithm

General Form of EM

- Model has known and hidden components (X, Z)
- The log-likelihood of the complete model $L(X, Z)$
  - But Z is a random variable
  - For a given $\theta^{(t)}$, $P(Z = \mathbf{z}|\mathbf{x}, \theta^{(t)})$ is known
- The EM algorithm computes

$$\theta^{(t+1)} = \text{argmax}_\theta \sum_{\mathbf{z}} P(Z = \mathbf{z}|\mathbf{x}, \theta^{(t)}) L(\mathbf{x}, Z = \mathbf{z}|\theta)$$

- Computing the expectation
  - May be very straightforward, e.g., mixture models
  - May need exact inference algorithms, e.g., Bayes nets
  - May need approximate inference such as Gibbs sampling