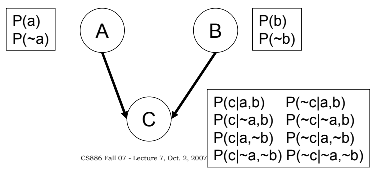
Bayesian Networks

aka belief networks, probabilistic networks

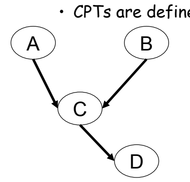
- A BN over variables $\{X_1, X_2, \ldots, X_n\}$ consists of:
  - a DAG whose nodes are the variables
  - a set of CPTs $(\Pr(X_i \mid Parents(X_i))$) for each X_i



Bayesian Networks

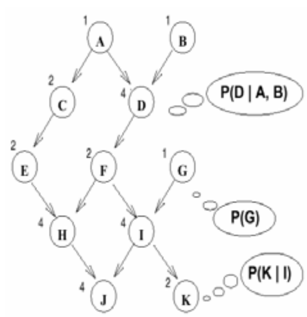
aka belief networks, probabilistic networks

- Key notions
  - parents of a node: Par(X_i)
  - children of node
  - descendents of a node
  - ancestors of a node
  - family: set of nodes consisting of X_i and its parents
    - CPTs are defined over families in the BN



Parents(C)={A,B}
Children(A)={C}
Descendents(B)={C,D}
Ancestors{D}={A,B,C}
Family{C}={C,A,B}

An Example Bayes Net



- A few CPTs are "shown"
- Explicit joint requires 2^{11} -1 =2047 params
- BN requires only 27 parms (the number of entries for each CPT is listed)

Semantics of a Bayes Net

- The structure of the BN means: every X_i is *conditionally independent of all of its nondescendants given its parents*:

$$\Pr(X_i \mid S \cup Par(X_i)) = \Pr(X_i \mid Par(X_i))$$

for any subset $S \subseteq NonDescendants(X_i)$

Semantics of Bayes Nets

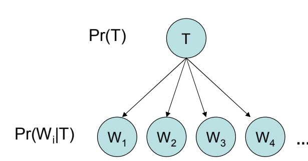
- If we ask for $P(x_1, x_2, \ldots, x_n)$ we obtain
  - assuming an ordering consistent with network
- By the chain rule, we have:

$$P(x_1, x_2, \ldots, x_n)$$
$$= P(x_n \mid x_{n-1}, \ldots, x_1)\, P(x_{n-1} \mid x_{n-2}, \ldots, x_1) \ldots P(x_1)$$
$$= P(x_n \mid Par(x_n))\, P(x_{n-1} \mid Par(x_{n-1})) \ldots P(x_1)$$

- Thus, the joint is recoverable using the parameters (CPTs) specified in an arbitrary BN

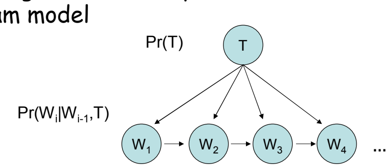
Bayes net example I

- Naïve Bayes model
  - Naïve: because words do not depend on each other.
  - Joint = $\Pr(T)\, \Pi_i \Pr(W_i \mid T)$



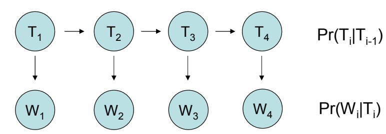
Bayes net example II

- Tree augmented naïve Bayes classifier
  - Allow arcs between the leaves as long as they form a tree
  - E.g., augment naïve Bayes model with bigram model



Bayes net example III

- Hidden Markov model
  - T_i: hidden variable (e.g., tag)
  - W_i: observable variable (e.g., word)



Maximum Likelihood Learning

- ML learning of Bayes net parameters:
  - For $\theta_{V=true,pa(V)=v}$ = Pr(V=true|par(V) = **v**)
  - $\theta_{V=true,pa(V)=v} = \dfrac{\#[V=true,pa(V)=\mathbf{v}]}{\#[V=true,pa(V)=\mathbf{v}] + \#[V=false,pa(V)=\mathbf{v}]}$
  - Assumes all attributes have values…
- What if values of some attributes are missing?

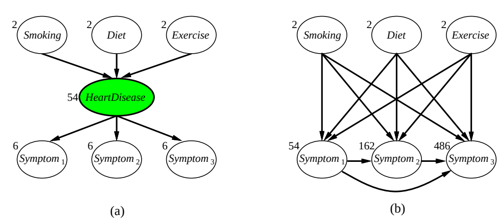
Incomplete data

- But many real-world problems have hidden variables (a.k.a latent variables)
  - Values of some attributes missing
  - Incomplete data → unsupervised learning
- Examples:
  - Part of speech tagging
  - Topic modeling
  - Market segmentation for marketing
  - Medical diagnosis

"Naive" solutions for incomplete data

- Solution #1: Ignore records with missing values
  - But what if all records are missing values (i.e., when a variable is hidden, none of the records have any value for that variable)
- Solution #2: Ignore hidden variables
  - Model may become significantly more complex!

Heart disease example



- a) simpler (i.e., fewer CPT parameters)
- b) complex (i.e., lots of CPT parameters)

“Direct” maximum likelihood

- Solution 3: maximize likelihood directly
  - Let **Z** be hidden and **E** observable
  - h_{ML} = argmax$_h$ P(**e**|h)
    = argmax$_h$ Σ_Z P(**e**,**Z**|h)
    = argmax$_h$ Σ_Z Π_i CPT(V_i)
    = argmax$_h$ log Σ_Z Π_i CPT(V_i)
  - Problem: can't push log past sum to linearize product

Expectation-Maximization (EM)

- Solution #4: EM algorithm
  - Intuition: if we knew the missing values, computing h_{ML} would be trival
- Guess h_{ML}
- Iterate
  - Expectation: based on h_{ML}, compute expectation of the missing values
  - Maximization: based on expected missing values, compute new estimate of h_{ML}

Expectation-Maximization (EM)

- More formally:
  - Approximate maximum likelihood
  - Iteratively compute:

$$h_{i+1} = \underbrace{\text{argmax}_h\ \underbrace{\Sigma_Z\ P(\mathbf{Z}|h_i,\mathbf{e})}_{\text{Expectation}}\ \log P(\mathbf{e},\mathbf{Z}|h)}_{\text{Maximization}}$$

Expectation-Maximization (EM)

- Derivation
  - log P(**e**|h) = log [P(**e**,**Z**|h) / P(**Z**|**e**,h)]
    = log P(**e**,**Z**|h) – log P(**Z**|**e**,h)
    = Σ_Z P(**Z**|**e**,h) log P(**e**,**Z**|h)
    – Σ_Z P(**Z**|**e**,h) log P(**Z**|**e**,h)
    $\geq$ Σ_Z P(**Z**|**e**,h) log P(**e**,**Z**|h)
- EM finds a local maximum of Σ_Z P(Z|e,h) log P(**e**,**Z**|h) which is a lower bound of log P(**e**|h)

Expectation-Maximization (EM)

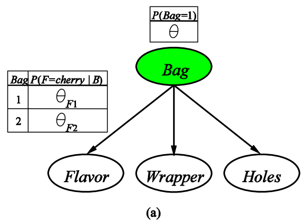
- Log inside sum can linearize product
  - h_{i+1} = argmax$_h$ Σ_Z P(**Z**|h_i,**e**) log P(**e**,**Z**|h)
    = argmax$_h$ Σ_Z P(**Z**|h_i,**e**) log Π_j CPT_j
    = argmax$_h$ Σ_Z P(**Z**|h_i,**e**) Σ_j log CPT_j
- Monotonic improvement of likelihood
  - P(**e**|h_{i+1}) $\geq$ P(**e**|h_i)

Candy Example

- Suppose you buy two bags of candies of unknown type (e.g. flavour ratios)
- You plan to eat sufficiently many candies of each bag to learn their type
- Ignoring your plan, your roommate mixes both bags…
- How can you learn the type of each bag despite being mixed?

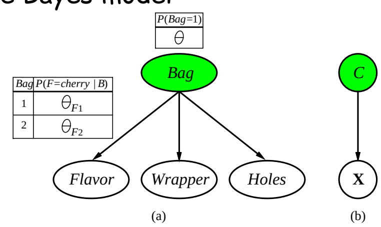
Candy Example

- "Bag" variable is hidden



Unsupervised Clustering

- "Class" variable is hidden
- Naïve Bayes model



Candy Example

- Unknown Parameters:
  - θ_i = P(Bag=i)
  - θ_{Fi} = P(Flavour=cherry|Bag=i)
  - θ_{Wi} = P(Wrapper=red|Bag=i)
  - θ_{Hi} = P(Hole=yes|Bag=i)
- When eating a candy:
  - F, W and H are observable
  - B is hidden

Candy Example

- Let true parameters be:
  - $\theta=0.5$, $\theta_{F1}=\theta_{W1}=\theta_{H1}=0.8$, $\theta_{F2}=\theta_{W2}=\theta_{H2}=0.3$
- After eating 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

Candy Example

- EM algorithm
- Guess h_0:
  - $\theta=0.6$, $\theta_{F1}=\theta_{W1}=\theta_{H1}=0.6$, $\theta_{F2}=\theta_{W2}=\theta_{H2}=0.4$
- Alternate:
  - Expectation: expected # of candies in each bag
  - Maximization: new parameter estimates

Candy Example

- Expectation: expected # of candies in each bag
  - #[Bag=i] = Σ_j P(B=i|f_j,w_j,h_j)
  - Compute P(B=i|f_j,w_j,h_j) by variable elimination (or any other inference alg.)
- Example:
  - #[Bag=1] = 612
  - #[Bag=2] = 388

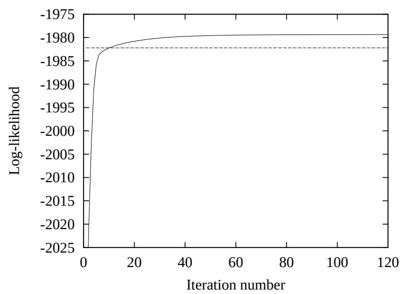
Candy Example

- Maximization: relative frequency of each bag
  - θ_1 = 612/1000 = 0.612
  - θ_2 = 388/1000 = 0.388

Candy Example

- Expectation: expected # of cherry candies in each bag
  - #[B=i,F=cherry] = Σ_j P(B=i|f_j=cherry,w_j,h_j)
  - Compute P(B=i|f_j=cherry,w_j,h_j) by variable elimination (or any other inference alg.)
- Maximization:
  - θ_{F1} = #[B=1,F=cherry] / #[B=1] = 0.668
  - θ_{F2} = #[B=2,F=cherry] / #[B=2] = 0.389

Candy Example



Bayesian networks

- EM algorithm for general Bayes nets
- Expectation:
  - #[V_i=v_{ij},Pa(V_i)=pa_{ik}] = expected frequency
- Maximization:
  - $\theta_{v_{ij},pa_{ik}}$ = #[V_i=v_{ij},Pa(V_i)=pa_{ik}] / #[Pa(V_i)=pa_{ik}]