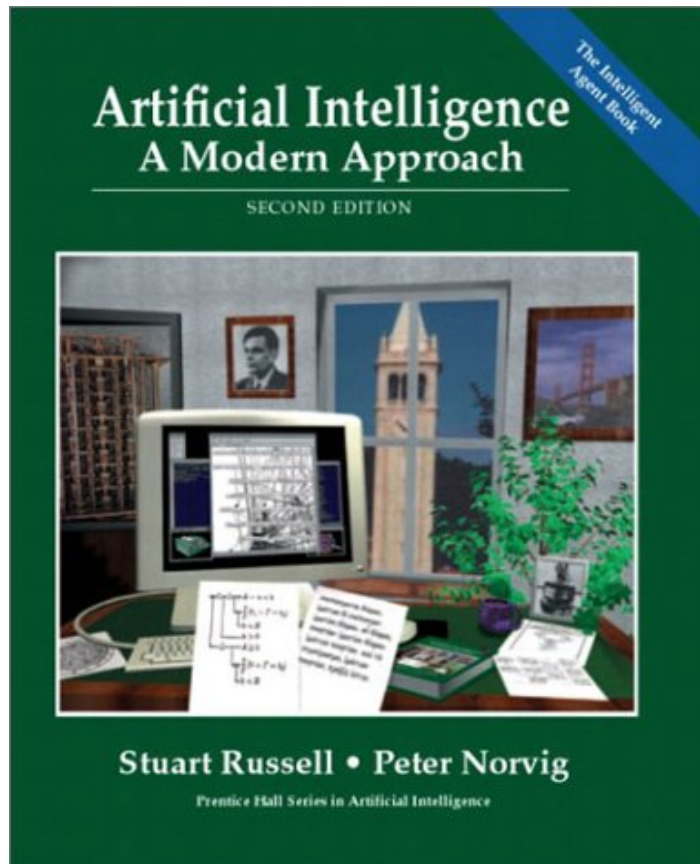


Bayesian Learning and Learning Bayesian Networks



Chapter 20
some slides by
Cristina Conati

Overview

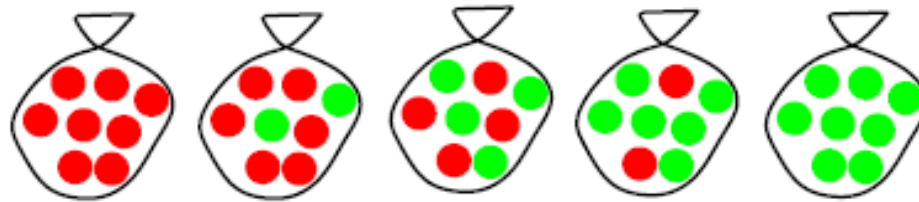
- Full Bayesian Learning
- MAP learning
- Maximum Likelihood Learning
- Learning Bayesian Networks
 - Fully observable
 - With hidden (unobservable) variables


Full Bayesian Learning

- In the learning methods we have seen so far, the idea was always to find the best model that could explain some observations
- In contrast, full Bayesian learning sees learning as Bayesian updating of a probability distribution over the hypothesis space, given data
 - H is the hypothesis variable
 - Possible hypotheses (values of H) h_1, \dots, h_n
 - $P(H)$ = prior probability distribution over hypothesis space
- j_{th} observation d_j gives the outcome of random variable D_j
 - training data $\mathbf{d} = d_1, \dots, d_k$

Example

- Suppose we have 5 types of candy bags
 - 10% are 100% cherry candies (h_{100})
 - 20% are 75% cherry + 25% lime candies (h_{75})
 - 40% are 50% cherry + 50% lime candies (h_{50})
 - 20% are 25% cherry + 75% lime candies (h_{25})
 - 10% are 100% lime candies (h_0)



- Then we observe candies drawn from some bag 
- Let's call θ the parameter that defines the fraction of cherry candy in a bag, and h_θ the corresponding hypothesis
- Which of the five kinds of bag has generated my 10 observations? $P(h_\theta | \mathbf{d})$.
- What flavour will the next candy be? Prediction $\mathbf{P}(X|\mathbf{d})$

Full Bayesian Learning

- Given the data so far, each hypothesis h_i has a posterior probability:
 - $P(h_i | \mathbf{d}) = \alpha P(\mathbf{d} | h_i) P(h_i)$ (**Bayes theorem**)
 - where $P(\mathbf{d} | h_i)$ is called the *likelihood* of the data under each hypothesis
- Predictions over a new entity X are a weighted average over the prediction of each hypothesis:

- $$\begin{aligned} \mathbf{P}(X|\mathbf{d}) &= \\ &= \sum_i \mathbf{P}(X, h_i | \mathbf{d}) \\ &= \sum_i \mathbf{P}(X | h_i, \mathbf{d}) P(h_i | \mathbf{d}) \\ &= \sum_i \mathbf{P}(X | h_i) P(h_i | \mathbf{d}) \\ &\sim \sum_i \mathbf{P}(X | h_i) P(\mathbf{d} | h_i) P(h_i) \end{aligned}$$

The data does not add anything to a prediction given an hp

- The weights are given by the data likelihood and prior of each h
- No need to pick one best-guess hypothesis!

$$\mathbf{P}(Y|X) = \frac{\mathbf{P}(X|Y)\mathbf{P}(Y)}{\mathbf{P}(X)} = \alpha\mathbf{P}(X|Y)\mathbf{P}(Y)$$

Example

➤ If we re-wrap each candy and return it to the bag, our 10 observations are independent and identically distributed, *i.i.d.*, so

- $P(\mathbf{d} | h_\theta) = \prod_{j=1}^{10} P(d_j | h_\theta)$ for $j=1, \dots, 10$

➤ For a given h_θ , the value of $P(d_j | h_\theta)$ is

- $P(d_j = \text{cherry} | h_\theta) = \theta$; $P(d_j = \text{lime} | h_\theta) = (1-\theta)$

➤ And given N observations, of which c are cherry and $l = N-c$ lime

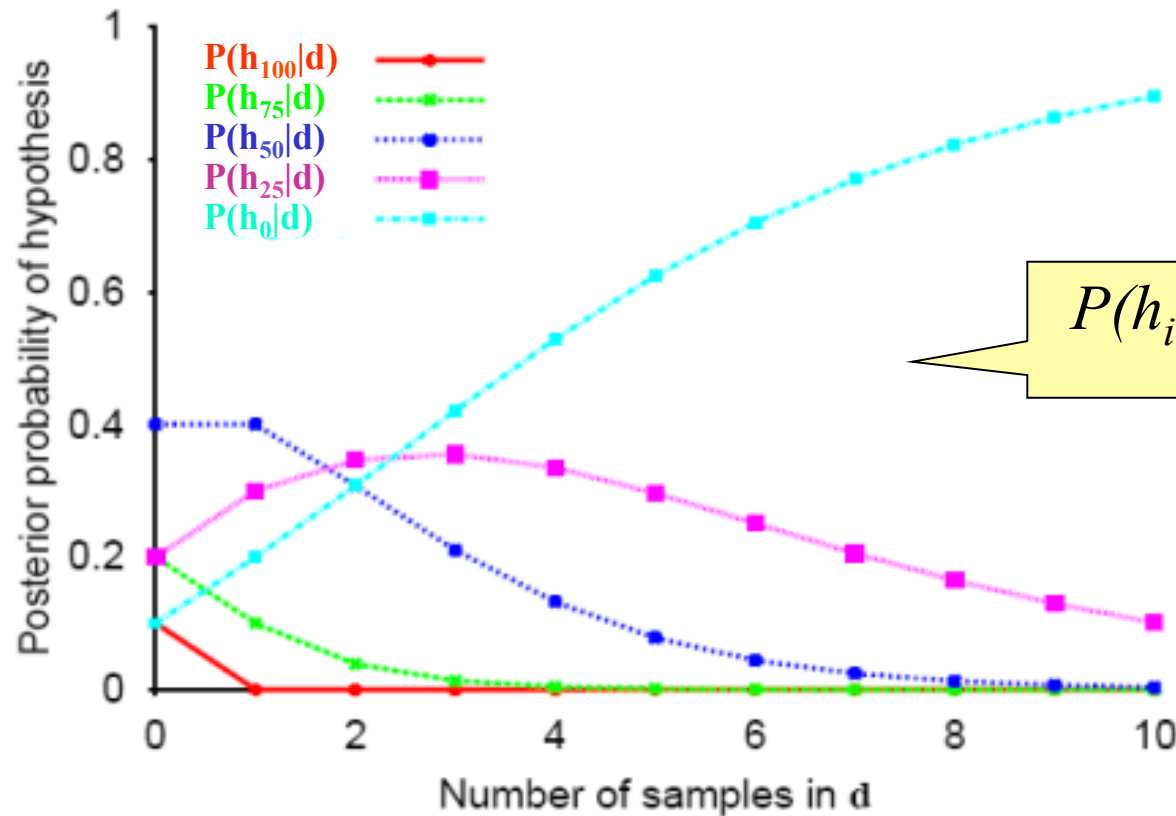
$$P(\mathbf{d} | h_\theta) = \prod_{j=1}^c \theta \prod_{j=1}^l (1-\theta) = \theta^c (1-\theta)^l$$

- **Binomial distribution**: probability of # of successes in a sequence of N independent trials with binary outcome, each of which yields success with probability θ .

➤ For instance, after observing 3 lime candies in a row:

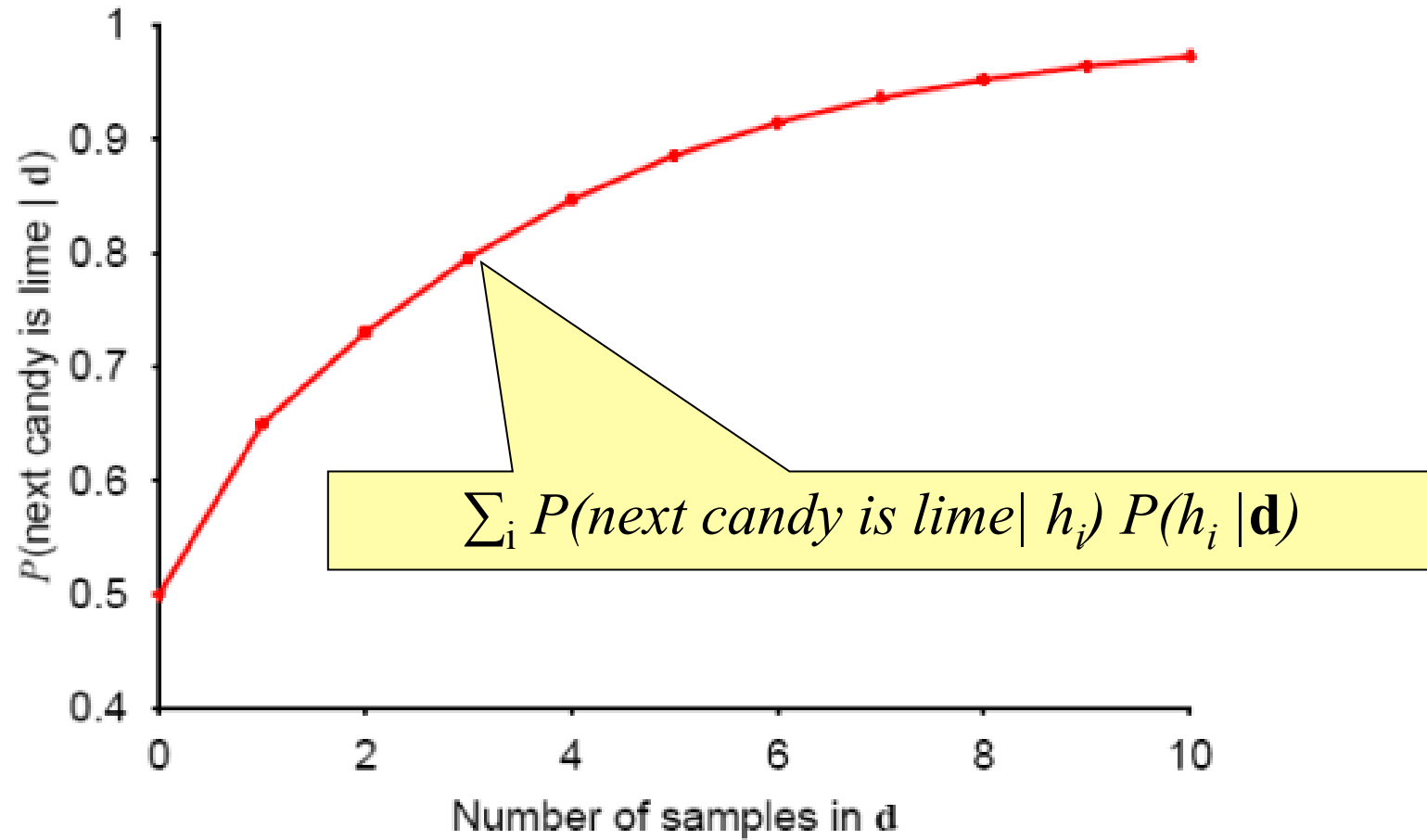
- $P([\text{lime}, \text{lime}, \text{lime}] | h_{.50}) = 0.5^3$ because the probability of seeing lime for each observation is 0.5 under this hypotheses

All-limes: Posterior Probability of H



- Initially, the h_p with higher priors dominate (h_{50} with prior = 0.4)
- As data comes in, the finally best hypothesis (h_0) starts dominating, as the probability of seeing this data given the other hypotheses gets increasingly smaller
 - After seeing three lime candies in a row, the probability that the bag is the all-lime one starts taking off

Prediction Probability



- The probability that the next candy is lime increases with the probability that the bag is an all-lime one

Overview

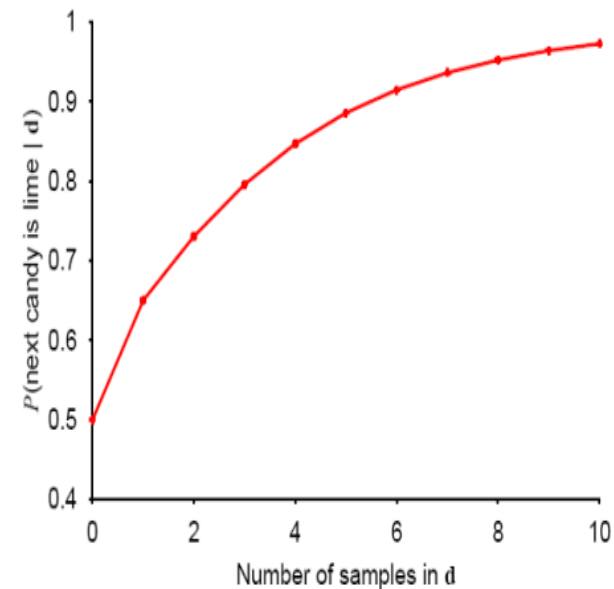
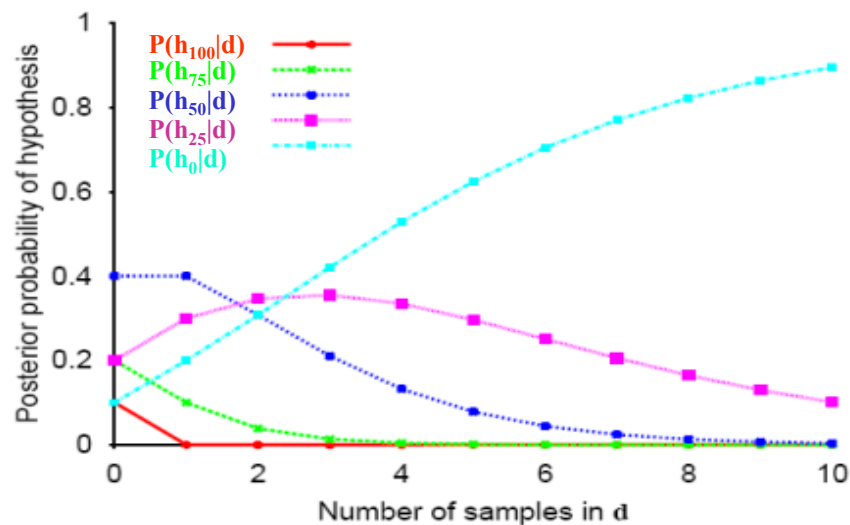
- Full Bayesian Learning
- MAP learning
- Maximum Likelihood Learning
- Learning Bayesian Networks
 - Fully observable
 - With hidden (unobservable) variables

MAP approximation

- Full Bayesian learning seems like a very safe bet, but unfortunately it does not work well in practice
 - Summing over the hypothesis space is often intractable (e.g., 18,446,744,073,709,551,616 Boolean functions of 6 attributes)
- Very common approximation: Maximum a posterior (MAP) learning:
 - Instead of doing prediction by considering all possible hypotheses , as in
 - $\mathbf{P}(X|\mathbf{d}) = \sum_i \mathbf{P}(X| h_i) P(h_i | \mathbf{d})$
 - Make predictions based on h_{MAP} that maximises $P(h_i | \mathbf{d})$
 - I.e., maximize $P(\mathbf{d}| h_i) P(h_i)$
 - $\mathbf{P}(X|\mathbf{d}) \sim \mathbf{P}(X| h_{\text{MAP}})$

MAP approximation

- MAP is a good approximation when $P(X | \mathbf{d}) \approx P(X | h_{\text{MAP}})$
 - In our example, h_{MAP} is the all-lime bag after only 3 candies, predicting that the next candy will be lime with $p = 1$
 - the bayesian learner gave a prediction of 0.8, safer after seeing only 3 candies



Bias

- As more data arrive, MAP and Bayesian prediction become closer, as MAP's competing hypotheses become less likely
- Often easier to find MAP (optimization problem) than deal with a large summation problem
- $P(H)$ plays an important role in both MAP and Full Bayesian Learning
 - Defines the *learning bias*, i.e. which hypotheses are favoured
- Used to define a tradeoff between model complexity and its ability to fit the data
 - More complex models can explain the data better => higher $P(\mathbf{d} | h_i)$
danger of overfitting
 - But they are less likely a priori because there are more of them than simpler model => lower $P(h_i)$
 - I.e. common learning bias is to penalize complexity

Overview

- Full Bayesian Learning
- MAP learning
- Maximun Likelihood Learning
- Learning Bayesian Networks
 - Fully observable
 - With hidden (unobservable) variables

Maximum Likelihood (ML) Learning

- Further simplification over full Bayesian and MAP learning
 - Assume uniform priors over the space of hypotheses
 - MAP learning (maximize $P(\mathbf{d} | h_j) P(h_j)$) reduces to maximize $P(\mathbf{d} | h_j)$
- When is ML appropriate?

Maximum Likelihood (ML) Learning

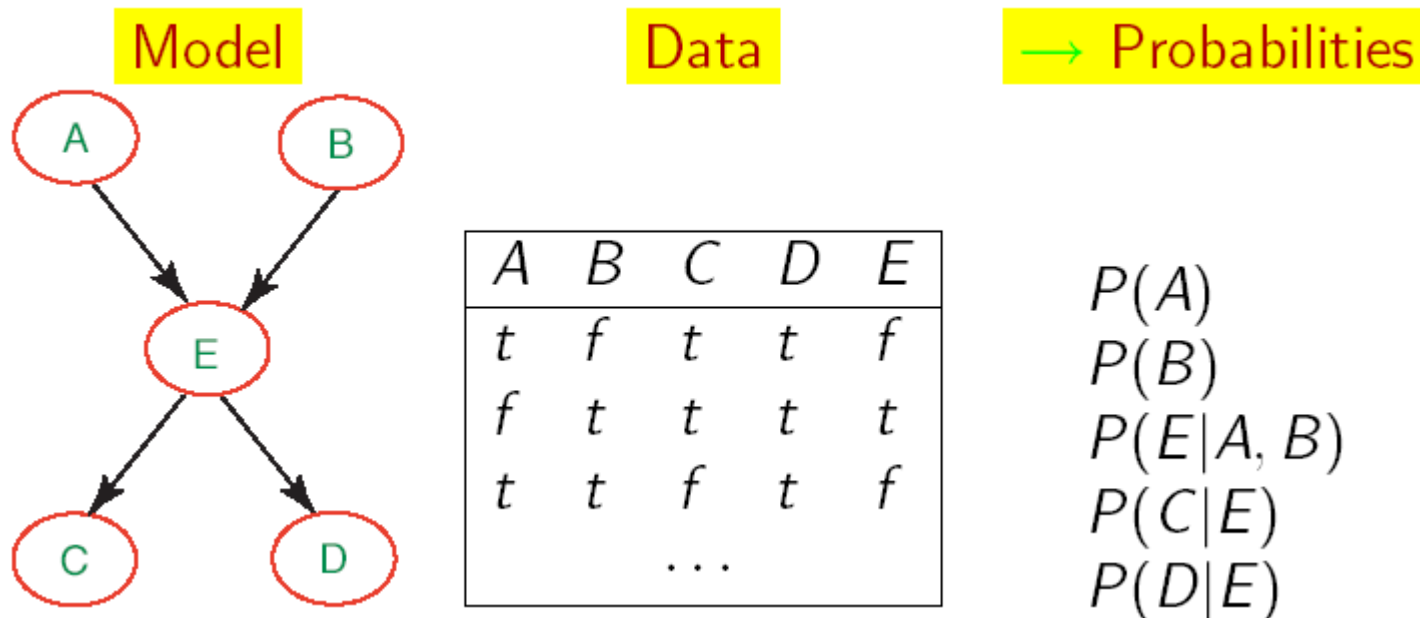
- Further simplification over Full Bayesian and MAP learning
 - Assume uniform prior over the space of hypotheses
 - MAP learning (maximize $P(\mathbf{d} | h_i) P(h_i)$) reduces to maximize $P(\mathbf{d} | h_i)$
- When is ML appropriate?
 - Used in statistics as the standard (non-bayesian) statistical learning method by those who distrust subjective nature of hypotheses priors
 - When the competing hypotheses are indeed equally likely (e.g. have same complexity)
 - With very large datasets, for which $P(\mathbf{d} | h_i)$ tends to overcome the influence of $P(h_i)$

Overview

- Full Bayesian Learning
- MAP learning
- Maximum Likelihood Learning
- Learning Bayesian Networks
 - Fully observable (complete data)
 - With hidden (unobservable) variables

Learning BNets: Complete Data

- We will start by applying ML to the simplest type of BNets learning:
 - known structure
 - Data containing observations for all variables
 - ✓ All variables are observable, no missing data
- The only thing that we need to learn are the network's parameters



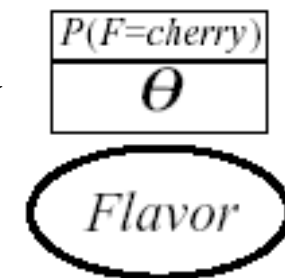
ML learning: example

➤ Back to the candy example:

- New candy manufacturer that does not provide data on the probability of fraction θ of cherry candy in its bags
- Any θ is possible: continuum of hypotheses h_θ
- Reasonable to assume that all θ are equally likely (we have no evidence of the contrary): uniform distribution $P(h_\theta)$
- θ is a parameter for this simple family of models, that we need to learn

➤ Simple network to represent this problem

- *Flavor* represents the event of drawing a cherry vs. lime candy from the bag
- $P(F=cherry)$, or $P(cherry)$ for brevity, is equivalent to the fraction θ of cherry candies in the bag



➤ We want to infer θ by unwrapping N candies from the bag

ML learning: example (cont' d)

- Unwrap N candies, c cherries and $l = N - c$ lime (and return each candy in the bag after observing flavor)
- As we saw earlier, this is described by a binomial distribution
 - $P(\mathbf{d} | h_\theta) = \prod_j P(d_j | h_\theta) = \theta^c (1 - \theta)^l$
- With ML we want to find θ that maximizes this expression, or equivalently its *log likelihood* (L)
 - $L(P(\mathbf{d} | h_\theta))$
 - $= \log (\prod_j P(d_j | h_\theta))$
 - $= \log (\theta^c (1 - \theta)^l)$
 - $= c \log \theta + l \log(1 - \theta)$

ML learning: example (cont' d)

- To maximise, we differentiate $L(P(\mathbf{d} | h_\theta))$ with respect to θ and set the result to 0

$$\begin{aligned} & \frac{\partial(c \log \theta + \ell \log(1 - \theta))}{\partial \theta} \\ &= \frac{c}{\theta} - \frac{\ell}{1 - \theta} \\ &= \frac{c}{\theta} - \frac{N - c}{1 - \theta} = 0 \end{aligned}$$

- Doing the math gives

$$\theta = \frac{c}{N}$$

Frequencies as Priors

- So this says that the proportion of cherries in the bag is equal to the proportion (frequency) of cherries in the data

- Now we have justified why this approach provides a reasonable estimate of node priors

General ML procedure

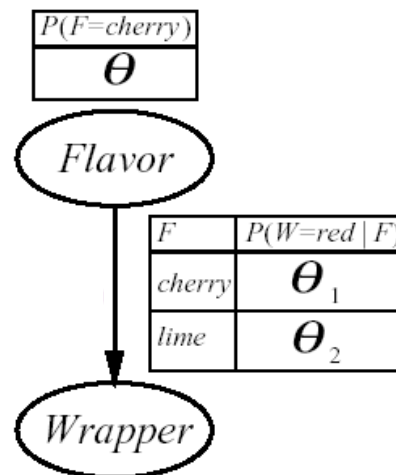
- Express the likelihood of the data as a function of the parameters to be learned
- Take the derivative of the log likelihood with respect of each parameter
- Find the parameter value that makes the derivative equal to 0
- The last step can be computationally very expensive in real-world learning tasks

More complex example

- The manufacturer chooses the color of the wrapper probabilistically for each candy based on flavor, following an unknown distribution
 - If the flavour is *cherry*, it chooses a red wrapper with probability θ_1
 - If the flavour is *lime*, it chooses a red wrapper with probability θ_2
- The Bayesian network for this problem includes 3 parameters to be learned
 - $\theta \theta_1 \theta_2$

More complex example

- The manufacturer chooses the color of the wrapper probabilistically for each candy based on flavor, following an unknown distribution
 - If the flavour is *cherry*, it chooses a red wrapper with probability θ_1
 - If the flavour is *lime*, it chooses a red wrapper with probability θ_2
- The Bayesian network for this problem includes 3 parameters to be learned
 - $\theta \theta_1 \theta_2$



Another example (cont' d)

➤ $P(W=\text{green}, F = \text{cherry} | h_{\theta_1\theta_2}) = (*)$

$$= P(W=\text{green} | F = \text{cherry}, h_{\theta_1\theta_2}) P(F = \text{cherry} | h_{\theta_1\theta_2})$$

$$= \theta (1 - \theta_1)$$

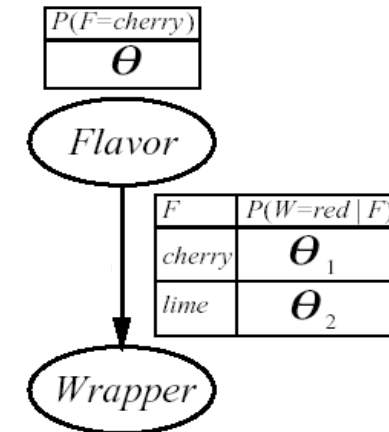
➤ We unwrap N candies

- c are cherry and l are lime
- r^c cherry with red wrapper, g^c cherry with green wrapper
- r^l lime with red wrapper, g^l lime with green wrapper
- every trial is a combination of wrapper and candy flavor similar to event (*) above, so

➤ $P(\mathbf{d} | h_{\theta_1\theta_2})$

$$= \prod_j P(d_j | h_{\theta_1\theta_2})$$

$$= \theta^c (1 - \theta)^l (\theta_1)^{r^c} (1 - \theta_1)^{g^c} (\theta_2)^{r^l} (1 - \theta_2)^{g^l}$$



Another example (cont' d)

➤ I want to maximize the log of this expression

- $c \log \theta + l \log(1 - \theta) + r^c \log \theta_1 + g^c \log(1 - \theta_1) + r^l \log \theta_2 + g^l \log(1 - \theta_2)$

➤ Take derivative with respect of each of $\theta, \theta_1, \theta_2$

- The terms not containing the derivation variable disappear

$$\frac{\partial L}{\partial \theta} = \frac{c}{\theta} - \frac{l}{1 - \theta} = 0 \quad \Rightarrow \quad \theta = \frac{c}{c + l}$$

$$\frac{\partial L}{\partial \theta_1} = \frac{r_c}{\theta_1} - \frac{g_c}{1 - \theta_1} = 0 \quad \Rightarrow \quad \theta_1 = \frac{r_c}{r_c + g_c}$$

$$\frac{\partial L}{\partial \theta_2} = \frac{r_l}{\theta_2} - \frac{g_l}{1 - \theta_2} = 0 \quad \Rightarrow \quad \theta_2 = \frac{r_l}{r_l + g_l}$$

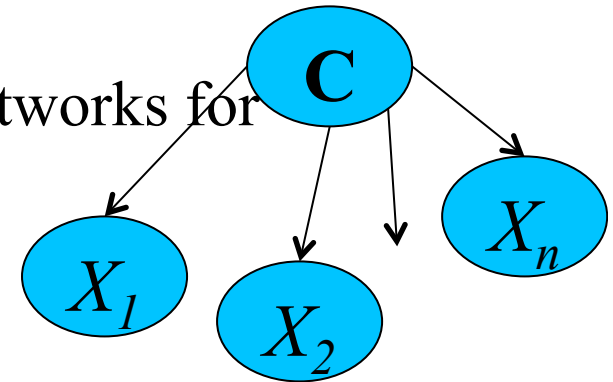
ML parameter learning in Bayes nets

- Frequencies again!
- This process generalizes to every fully observable Bnet.
- With complete data and ML approach:
 - Parameters learning decomposes into a separate learning problem for each parameter (CPT), because of the log likelihood step
 - Each parameter is given by the frequency of the desired child value given the relevant parents values

Very Popular Application

- Naïve Bayes models: very simple Bayesian networks for classification

- *Class* variable (to be predicted) is the root node
- Attribute variables X_i (observations) are the leaves



- Naïve because it assumes that the attributes are conditionally independent of each other given the class

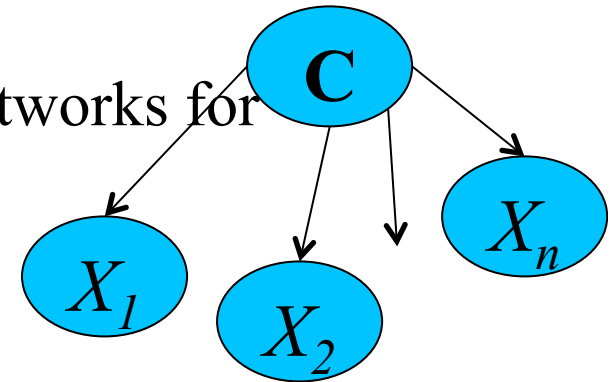
$$P(C|x_1, x_2, \dots, x_n) = \frac{P(C, x_1, x_2, \dots, x_n)}{P(x_1, x_2, \dots, x_n)} = \alpha P(C) \prod_i P(x_i | C)$$

- Deterministic prediction can be obtained by picking the most likely class
- Scales up really well: with n boolean attributes we just need.....

Very Popular Application

- Naïve Bayes models: very simple Bayesian networks for classification

- *Class* variable (to be predicted) is the root node
- Attribute variables X_i (observations) are the leaves



- Naïve because it assumes that the attributes are conditionally independent of each other given the class

$$P(C|x_1, x_2, \dots, x_n) = \frac{P(C, x_1, x_2, \dots, x_n)}{P(x_1, x_2, \dots, x_n)} = \alpha P(C) \prod_i P(x_i | C)$$

- Deterministic prediction can be obtained by picking the most likely class
- Scales up really well: with n boolean attributes we just need $2n+1$ parameters

Problem with ML parameter learning

- With small datasets, some of the frequencies may be 0 just because we have not observed the relevant data
- Generates very strong incorrect predictions:
 - Common fix: initialize the count of every relevant event to 1 before counting the observations

Probability from Experts

- As we mentioned in previous lectures, an alternative to learning probabilities from data is to get them from experts
- Problems
 - Experts may be reluctant to commit to specific probabilities that cannot be refined
 - How to represent the confidence in a given estimate
 - Getting the experts and their time in the first place
- One promising approach is to *leverage both sources* when they are available
 - Get initial estimates from experts
 - Refine them with data

Combining Experts and Data

- Get the expert to express her belief on event A as the pair $\langle n, m \rangle$
 - i.e. how many observations of A they have seen (or expect to see) in m trials
- Combine the pair with actual data
 - If A is observed, increment both n and m
 - If $\neg A$ is observed, increment m alone
- The absolute values in the pair can be used to express the expert's level of confidence in her estimate
 - Small values (e.g., $\langle 2, 3 \rangle$) represent low confidence
 - The larger the values, the higher the confidence



WHY?

Combining Experts and Data

- Get the expert to express her belief on event A as the pair

$\langle n, m \rangle$

i.e. how many observations of A they have seen (or expect to see) in m trials

- Combine the pair with actual data

- If A is observed, increment both n and m
- If $\neg A$ is observed, increment m alone

- The absolute values in the pair can be used to express the expert's level of confidence in her estimate

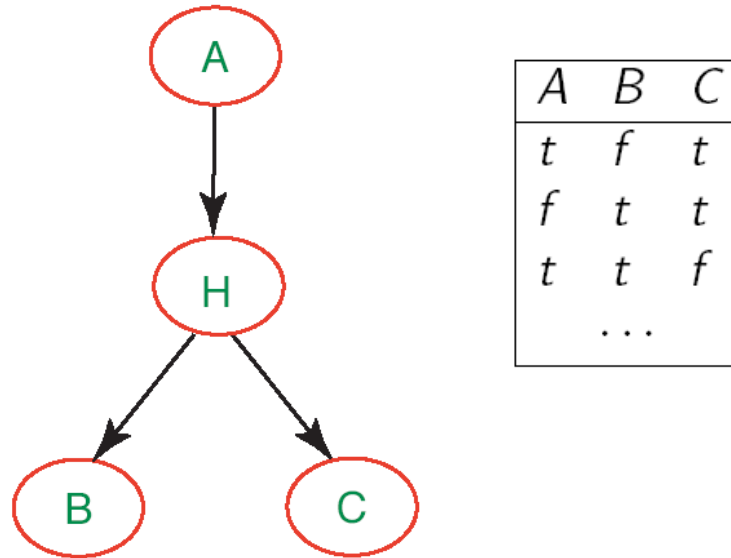
- Small values (e.g., $\langle 2, 3 \rangle$) represent low confidence, as they are quickly dominated by data
- The larger the values, the higher the confidence as it takes more and more data to dominate the initial estimate (e.g. $\langle 2000, 3000 \rangle$)

Overview

- Full Bayesian Learning
- MAP learning
- Maximum Likelihood Learning
- Learning Bayesian Networks
 - Fully observable (complete data)
 - With hidden (unobservable) variables

Learning Parameters with Hidden Variables

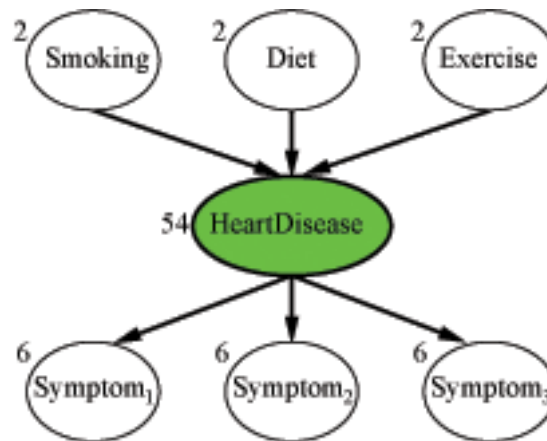
- So far we have assumed that we can collect data on all variables in the network
- What if this is not true, i.e. the network has *hidden variables*?



- Clearly we can't use the frequency approach, because we are missing all the counts involving H

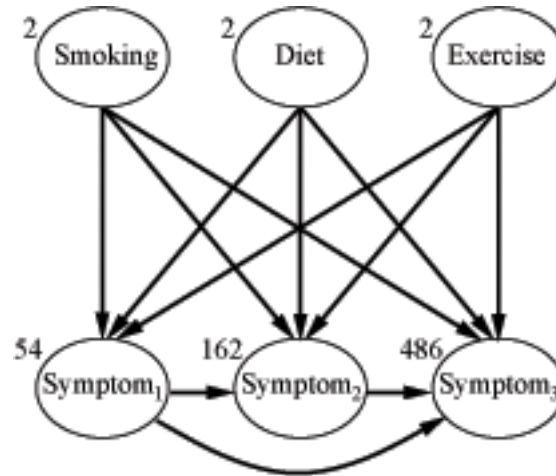
Quick Fix

- Get rid of the hidden variables.
- It may work in the simple network given earlier, but what about the following one?



- Each variable has 3 values (low, moderate, high)
- the numbers by the nodes represent how many parameters need to be specified for the CPT of that node
- 78 probabilities to be specified overall

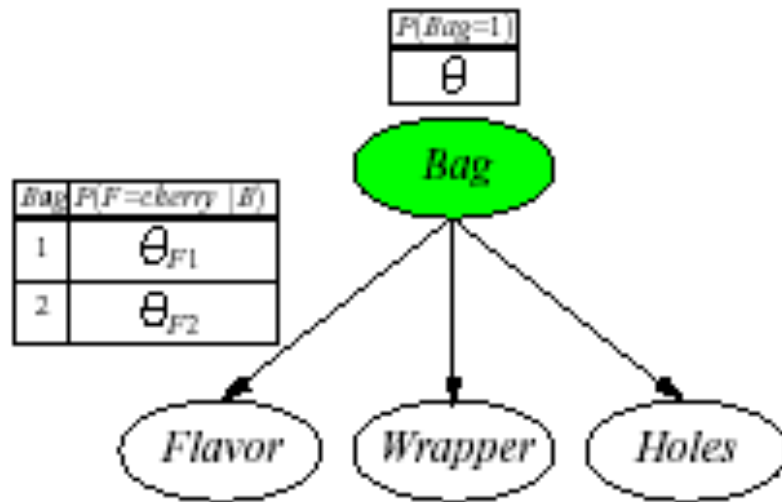
Not Necessarily a Good Fix



- The symptom variables are no longer conditionally independent given their parents
 - Many more links, and many more probabilities to be specified: 708 overall
 - Need much more data to properly learn the network

Example: The cherry/lime candy world again

- Two bags of candies (1 and 2) have been mixed together
- Candies are described by 3 features: Flavor and Wrapper as before, plus Hole (whether they have a hole in the middle)
- Candies' features depend probabilistically from the bag they originally came from
- We want to predict for each candy, which was its original bag, from its features: Naïve Bayes model



$$\theta = P(\text{Bag} = 1)$$

$$\theta_{Fj} = P(\text{Flavor} = \text{cherry} | \text{Bag} = j)$$

$$\theta_{Wj} = P(\text{Wrapper} = \text{red} | \text{Bag} = j)$$

$$\theta_{Hj} = P(\text{Hole} = \text{yes} | \text{Bag} = j)$$

$$j = 1, 2$$

Expectation-Maximization (EM)

- If we keep the hidden variables, and want to learn the network parameters from data, we have a form of *unsupervised learning*
 - The data do not include information on the true nature of each data point
- Expectation-Maximization
 - General algorithm for learning model parameters from incomplete data
 - We'll see how it works on learning parameters for Bnets with discrete variables

Bayesian learning: Bayes' rule

- Given some model space (set of hypotheses h_i) and evidence (data D):
 - $P(h_i|D) = \alpha P(D|h_i) P(h_i)$
- We assume that observations are independent of each other, given a model (hypothesis), so:
 - $P(h_i|D) = \alpha \prod_j P(d_j|h_i) P(h_i)$
- To predict the value of some unknown quantity, X (e.g., the class label for a future observation):

$$\bullet P(X|D) = \sum_i P(X|D, h_i) P(h_i|D) = \sum_i P(X|h_i) P(h_i|D)$$

These are equal by our independence assumption

Bayesian learning

- We can apply Bayesian learning in three basic ways:
 - **BMA (Bayesian Model Averaging):** Don't just choose one hypothesis; instead, make predictions based on the weighted average of all hypotheses (or some set of best hypotheses)
 - **MAP (Maximum A Posteriori) hypothesis:** Choose the hypothesis with the highest *a posteriori* probability, given the data
 - **MLE (Maximum Likelihood Estimate):** Assume that all hypotheses are equally likely *a priori*; then the best hypothesis is just the one that maximizes the likelihood (i.e., the probability of the data given the hypothesis)
- **MDL (Minimum Description Length) principle:** Use some encoding to model the complexity of the hypothesis, and the fit of the data to the hypothesis, then minimize the overall description length of $h_i + D$

Parameter estimation

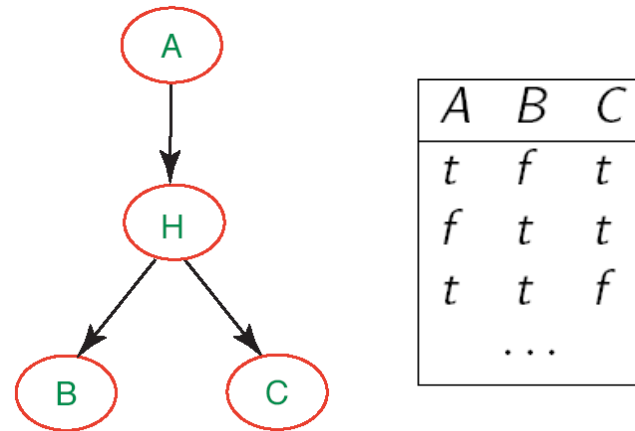
- Assume known structure
- Goal: estimate BN parameters θ
 - entries in local probability models, $P(X \mid \text{Parents}(X))$
- A parameterization θ is good if it is likely to generate the observed data:

$$\text{Score}(\theta) = P(D \mid \theta) = \prod_m P(x[m] \mid \theta)$$

i.i.d. samples

- Maximum Likelihood Estimation (MLE) Principle: Choose θ^* so as to maximize *Score*

EM: general idea



➤ If we had data for all the variables in the network, we could learn the parameters by using ML (or MAP) models

- Frequencies of the relevant events as we saw in previous examples

➤ If we had the parameters in the network, we could estimate the posterior probability of any event, including the hidden variables

$$P(H|A,B,C)$$

EM: General Idea

- The algorithm starts from “invented” (e.g., randomly generated) information to solve the learning problem, i.e.
 - Determine the network parameters
- It then refines this initial guess by cycling through two basic steps
 - Expectation (E): update the data with predictions generated via the current model
 - Maximization (M): given the updated data, update the model parameters using the Maximum Likelihood (ML) approach
 - ✓ This is the same step that we described when learning parameters for fully observable networks

EM: How it Works on Naive Bayes

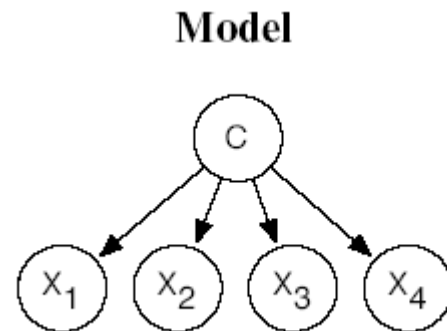
➤ Consider the following data,

- N examples with Boolean attributes X_1, X_2, X_3, X_4

Data			
X_1	X_2	X_3	X_4
t	f	t	t
f	t	t	f
f	f	t	t
	...		

➤ which we want to categorize in one of three possible values of class $C = \{1,2,3\}$

➤ We use a Naive Bayes classifier with hidden variable C

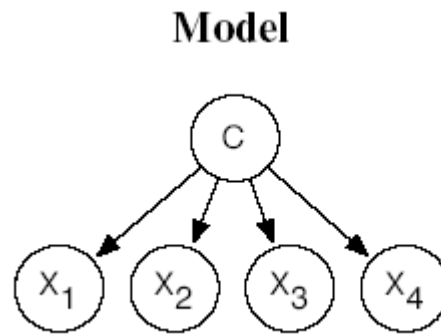


→ **Probabilities**

- $P(C)$?
- $P(X_1|C)$?
- $P(X_2|C)$?
- $P(X_3|C)$?
- $P(X_4|C)$?

EM: Initialization

- The algorithm starts from “invented” (e.g., randomly generated) information to solve the learning problem, i.e.
 - Determine the network parameters

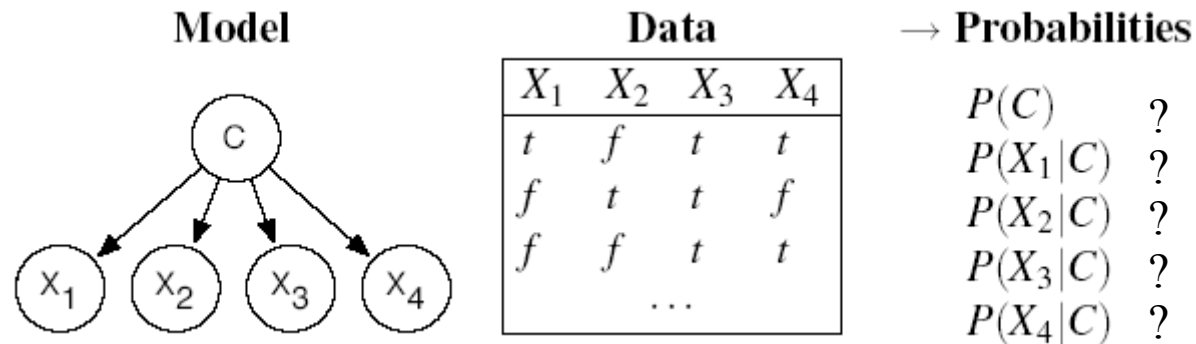


→ **Probabilities**

$P(C)$?
 $P(X_1|C)$?
 $P(X_2|C)$?
 $P(X_3|C)$?
 $P(X_4|C)$?

**Define
arbitrary
parameters**

EM: Expectation Step (Get Expected Counts)



➤ What would we need to learn the network parameters using ML approach?

- for $P(C) = \text{Count}(\text{datapoints with } C=i) / \text{Count}(\text{all datapoints}) \quad i=1,2,3$
- for $P(X_h|C) = \text{Count}(\text{datapoints with } X_h = \text{val}_k \text{ and } C=i) / \text{Count}(\text{data with } C=i)$
for all values val_k of X_h and $i=1,2,3$

EM: Expectation Step (Get Expected Counts)

- We only have $Count(\text{all datapoints}) = N$.
- We approximate all other necessary counts with *expected* counts derived from the model with “invented” parameters
- Expected count $\hat{N}(C = i)$ is the sum, over all N examples in my dataset, of the probability that each example is in category i

$$\begin{aligned}\hat{N}(C = i) &= \sum_{j=1}^N P(C = i \mid \text{attributes of example } e_j) \\ &= \sum_{j=1}^N P(C = i \mid x1_j, x2_j, x3_j, x4_j)\end{aligned}$$

EM: Expectation Step (Get Expected Counts)

- How do we get the necessary probabilities from the model?

$$\begin{aligned}\hat{N}(C = i) &= \sum_{j=1}^N P(C = i \mid \text{attributes of example } e_j) \\ &= \sum_{j=1}^N P(C = i \mid x1_j, x2_j, x3_j, x4_j)\end{aligned}$$

- Easy with a Naïve bayes network

$$\begin{aligned}P(C = i \mid x1_j, x2_j, x3_j, x4_j) &= \frac{P(C = i, x1_j, x2_j, x3_j, x4_j)}{P(x1_j, x2_j, x3_j, x4_j)} \\ &= \frac{P(x1_j \mid C = i) \dots P(x4_j \mid C = i) P(C = i)}{P(x1_j, x2_j, x3_j, x4_j)}\end{aligned}$$

Also available from Naïve Bayes. You do the necessary transformations

Naïve bayes “invented parameters”

EM: Expectation Step (Get Expected Counts)

- By a similar process we obtain the expected counts of examples with attribute $X_h = val_k$ and belonging to category i .
- These are needed *later* for estimating $\mathbf{P}(X_h | C)$:

$$\mathbf{P}(X_h | C) = \frac{\text{Exp.Counts}(\text{examples with } X_h = val_k \text{ and } C = i)}{\text{Exp.Counts}(\text{examples with } C = i)} = \frac{\hat{N}(X_h = val_k, C = i)}{\hat{N}(C = i)}$$

- for all values val_k of X_h and $i=1,2,3$

- For instance

$$\hat{N}(X_1 = t, C = 1) = \sum_{e_j \text{ with } X_1 = t} \mathbf{P}(C = i | x1_j = t, x2_j, x3_j, x4_j)$$

Again, get these probabilities from model with current parameters

EM: General Idea

- **The algorithm starts from “invented” (e.g., randomly generated) information to solve the learning problem, i.e.**
 - **the network parameters**
- **It then refines this initial guess by cycling through two basic steps**
 - **Expectation (E): compute expected counts based on the generated via the current model**
 - **Maximization (M): given the expected counts, update the model parameters using the Maximum Likelihood (ML) approach**
 - ✓ **This is the same step that we described when learning parameters for fully observable networks**

Maximization Step: (Refining Parameters)

- Now we can refine the network parameters by applying ML to the expected counts

$$P(C = i) = \frac{\hat{N}(C = i)}{N}$$

$$P(X_j = \text{val}_k \mid C = i) = \frac{\hat{N}(X_j = \text{val}_k \mid C = i)}{\hat{N}(C = i)}$$

- for all values val_k of X_j and $i=1,2,3$

EM Cycle

- Ready to start the E-step again

Expected Counts
("Augmented data")

X_1	X_2	X_3	X_4	C	count
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
t	f	t	t	1	
t	f	t	t	2	
t	f	t	t	3	
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots

M-step

Probabilities

$P(C)$
 $P(X_1|C)$
 $P(X_2|C)$
 $P(X_3|C)$
 $P(X_4|C)$

E-step

Procedure EM(X, D, k)

Inputs: X set of features $X = \{X_1, \dots, X_n\}$; D data set on features $\{X_1, \dots, X_n\}$; k number of classes

Output: $P(C)$, $P(X_i|C)$ for each $i \in \{1:n\}$, where $C = \{1, \dots, k\}$.

Local

real array $A[X_1, \dots, X_n, C]$

real array $P[C]$

real arrays $M_i[X_i, C]$ for each $i \in \{1:n\}$

real arrays $P_i[X_i, C]$ for each $i \in \{1:n\}$

$s \leftarrow$ number of tuples in D

Assign $P[C]$, $P_i[X_i, C]$ arbitrarily

repeat

// E Step

for each assignment $\langle X_1=v_1, \dots, X_n=v_n \rangle \in D$ **do**

let $m \leftarrow |\langle X_1=v_1, \dots, X_n=v_n \rangle \in D|$

for each $c \in \{1:k\}$ **do**

$A[v_1, \dots, v_n, c] \leftarrow m \times P(C=c | X_1=v_1, \dots, X_n=v_n)$

end for each

end for each

// M Step

for each $i \in \{1:n\}$ **do**

$M_i[X_i, C] = \sum_{X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_n} A[X_1, \dots, X_n, C]$

$P_i[X_i, C] = (M_i[X_i, C]) / (\sum_C M_i[X_i, C])$

end for each

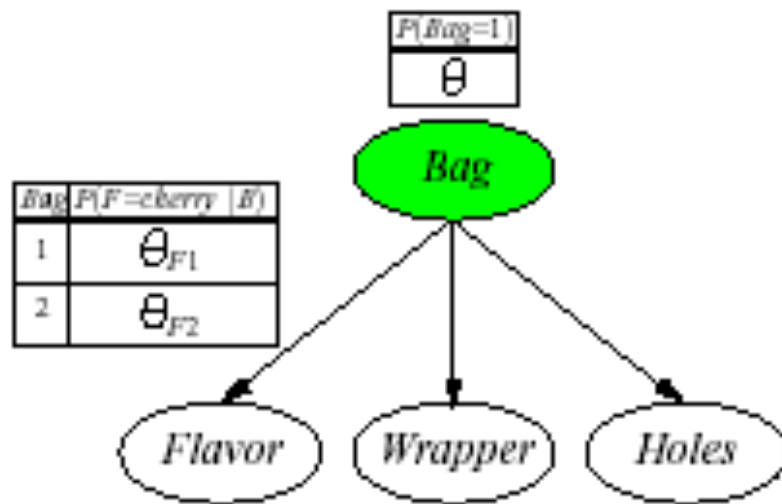
$P[C] = \sum_{X_1, \dots, X_n} A[X_1, \dots, X_n, C] / s$

until probabilities do not change significantly

end procedure

Example: Back to the cherry/lime candy world.

- Two bags of candies (1 and 2) have been mixed together
- Candies are described by 3 features: Flavor and Wrapper as before, plus Hole (whether they have a hole in the middle)
- Candies' features depend probabilistically from the bag they originally came from
- We want to predict for each candy, which was its original bag, from its features: Naïve Bayes model



$$\theta = P(\text{Bag} = 1)$$

$$\theta_{Fj} = P(\text{Flavor} = \text{cherry} | \text{Bag} = j)$$

$$\theta_{Wj} = P(\text{Wrapper} = \text{red} | \text{Bag} = j)$$

$$\theta_{Hj} = P(\text{Hole} = \text{yes} | \text{Bag} = j)$$

$$j = 1, 2$$

Data

➤ Assume that the true parameters are

- $\theta = 0.5$;
- $\theta_{F1} = \theta_{W1} = \theta_{H1} = 0.8$;
- $\theta_{F2} = \theta_{W2} = \theta_{H2} = 0.3$;

➤ The following counts are “generated” from $\mathbf{P}(C, F, W, H)$
($N = 1000$)

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

➤ We want to re-learn the true parameters using EM

EM: Initialization

- Assign arbitrary initial parameters
 - Usually done randomly; here we select numbers convenient for computation

$$\theta^{(0)} = 0.6;$$

$$\theta_{F1}^{(0)} = \theta_{W1}^{(0)} = \theta_{H1}^{(0)} = 0.6;$$

$$\theta_{F2}^{(0)} = \theta_{W2}^{(0)} = \theta_{H2}^{(0)} = 0.4$$

- We'll work through one cycle of EM to compute $\theta^{(1)}$.

E-step

- First, we need the expected count of candies from Bag 1,
- Sum of the probabilities that each of the N data points comes from bag 1
 - Be $flavor_j, wrapper_j, hole_j$ the values of the corresponding attributes for the j^{th} datapoint

$$\begin{aligned}\hat{N}(\text{Bag} = 1) &= \sum_{j=1}^N P(\text{Bag} = 1 | flavor_j, wrapper_j, whole_j) = \\ &= \sum_{j=1}^N \frac{P(flavor_j, wrapper_j, hole_j | \text{Bag} = 1) P(\text{Bag} = 1)}{P(flavor_j, wrapper_j, hole_j)} \\ &= \sum_{j=1}^N \frac{P(flavor_j | \text{Bag} = 1) P(wrapper_j | \text{Bag} = 1) P(hole_j | \text{Bag} = 1) P(\text{Bag} = 1)}{\sum_i P(flavor_j | \text{Bag} = i) P(wrapper_j | \text{Bag} = i) P(hole_j | \text{Bag} = i) P(\text{Bag} = i)}\end{aligned}$$

E-step

$$\sum_{j=1}^N \frac{P(\text{flavor}_j | \text{Bag} = 1) P(\text{wrapper}_j | \text{Bag} = 1) P(\text{hole}_j | \text{Bag} = 1) P(\text{Bag} = 1)}{\sum_i P(\text{flavor}_j | \text{Bag} = i) P(\text{wrapper}_j | \text{Bag} = i) P(\text{hole}_j | \text{Bag} = i) P(\text{Bag} = i)}$$

- This summation can be broken down into the 8 candy groups in the data table.
 - For instance the sum over the 273 cherry candies with red wrap and hole (first entry in the data table) gives

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

$$= 273 \frac{\theta_{F1}^{(0)} \theta_{W1}^{(0)} \theta_{H1}^{(0)} \theta^{(0)}}{\theta_{F1}^{(0)} \theta_{W1}^{(0)} \theta_{H1}^{(0)} \theta^{(0)} + \theta_{F2}^{(0)} \theta_{W2}^{(0)} \theta_{H2}^{(0)} (1 - \theta^{(0)})} =$$

$$273 \frac{0.6^4}{0.6^4 + 0.4^4} = 273 \frac{0.1296}{0.1552} = 227.97$$

$$\theta^{(0)} = 0.6;$$

$$\theta_{F1}^{(0)} = \theta_{W1}^{(0)} = \theta_{H1}^{(0)} = 0.6;$$

$$\theta_{F2}^{(0)} = \theta_{W2}^{(0)} = \theta_{H2}^{(0)} = 0.4$$

M-step

- If we do compute the sums over the other 7 candy groups we get

$$\hat{N}(\text{Bag} = 1) = 612.4$$

- At this point, we can perform the M-step to refine θ , by taking the expected frequency of the data points that come from Bag 1

$$\theta(1) = \frac{\hat{N}(\text{Bag} = 1)}{N} = 0.6124$$

One More Parameter

- If we want to do the same for parameter θ_{F1}
- E-step: compute the expected count of cherry candies from Bag 1

$$\hat{N}(Bag = 1 \wedge Flavor = cherry) = \sum_{j: Flavor_j = cherry} P(Bag = 1 | Flavor_j = cherry, wrapper_j, hole_j)$$

- Can compute the above value from the Naïve model as we did earlier
- TRY AS AN EXERCISE
- M-step: refine θ_{F1} by computing the corresponding expected frequencies

$$\theta_{F1}^{(1)} = \frac{\hat{N}(Bag = 1 \wedge Flavor = cherry)}{\hat{N}(Bag = 1)}$$

Learning Performance

- After a complete cycle through all the parameters, we get

$$\theta^{(1)} = 0.6124;$$

$$\theta_{F1}^{(1)} = 0.6684; \quad \theta_{W1}^{(1)} = 0.6483; \quad \theta_{H1}^{(1)} = 0.658;$$

$$\theta_{F2}^{(1)} = 0.3887; \quad \theta_{W2}^{(1)} = 0.3817; \quad \theta_{H2}^{(1)} = 0.3827;$$

- For any set of parameters, I can compute the log likelihood as we did in the previous class
- It can be seen that the log likelihood increases with each EM iteration (see textbook)
- EM tends to get stuck in local maxima, so it is often combined with gradient-based techniques in the last phase of learning

Learning Performance

- After a complete cycle through all the parameters, we get

$$\theta^{(1)} = 0.6124;$$

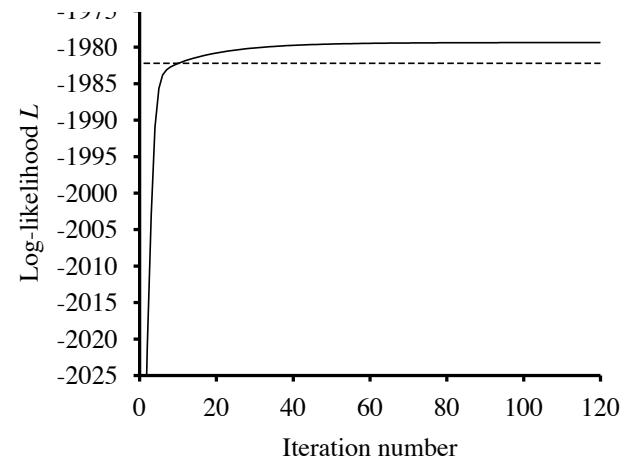
$$\theta_{F1}^{(1)} = 0.6684; \quad \theta_{W1}^{(1)} = 0.6483; \quad \theta_{H1}^{(1)} = 0.658;$$

$$\theta_{F2}^{(1)} = 0.3887; \quad \theta_{W2}^{(1)} = 0.3817; \quad \theta_{H2}^{(1)} = 0.3827;$$

- For any set of parameters, I can compute the log likelihood as we did in the previous class

$$P(\mathbf{d} | h_{\theta^{(i)}_{F1} \theta^{(i)}_{W1} \theta^{(i)}_{H1} \theta^{(i)}_{F2} \theta^{(i)}_{W2} \theta^{(i)}_{H2}}) = \prod_{j=1}^{1000} P(d_j | h_{\theta^{(i)}_{F1} \theta^{(i)}_{W1} \theta^{(i)}_{H1} \theta^{(i)}_{F2} \theta^{(i)}_{W2} \theta^{(i)}_{H2}})$$

- It can be shown that the log likelihood increases with each EM iteration, surpassing even the likelihood of the original model after only 3 iterations



EM: Discussion

➤ For more complex Bnets the algorithm is basically the same

- In general, I may need to compute the conditional probability parameter for each variable X_i given its parents Pa_i
- $\theta_{ijk} = P(X_i = x_{ij} | Pa_i = pa_{ik})$

$$\theta_{ijk} = \frac{\hat{N}(X_i = x_{ij}, Pa_i = pa_{ik})}{\hat{N}(Pa_i = pa_{ik})}$$

➤ The expected counts are computed by summing over the examples, after having computed all the necessary $P(X_i = x_{ij}, Pa_i = pa_{ik})$ using any Bnet inference algorithm

➤ The inference can be intractable, in which case there are variations of EM that use sampling algorithms for the E-Step

EM: Discussion

- The algorithm is sensitive to “degenerated” local maxima due to extreme configurations
 - e.g., data with outliers can generate categories that include only 1 outlier each because these models have the highest log likelihoods
 - Possible solution: re-introduce priors over the learning hypothesis and use the MAP version of EM