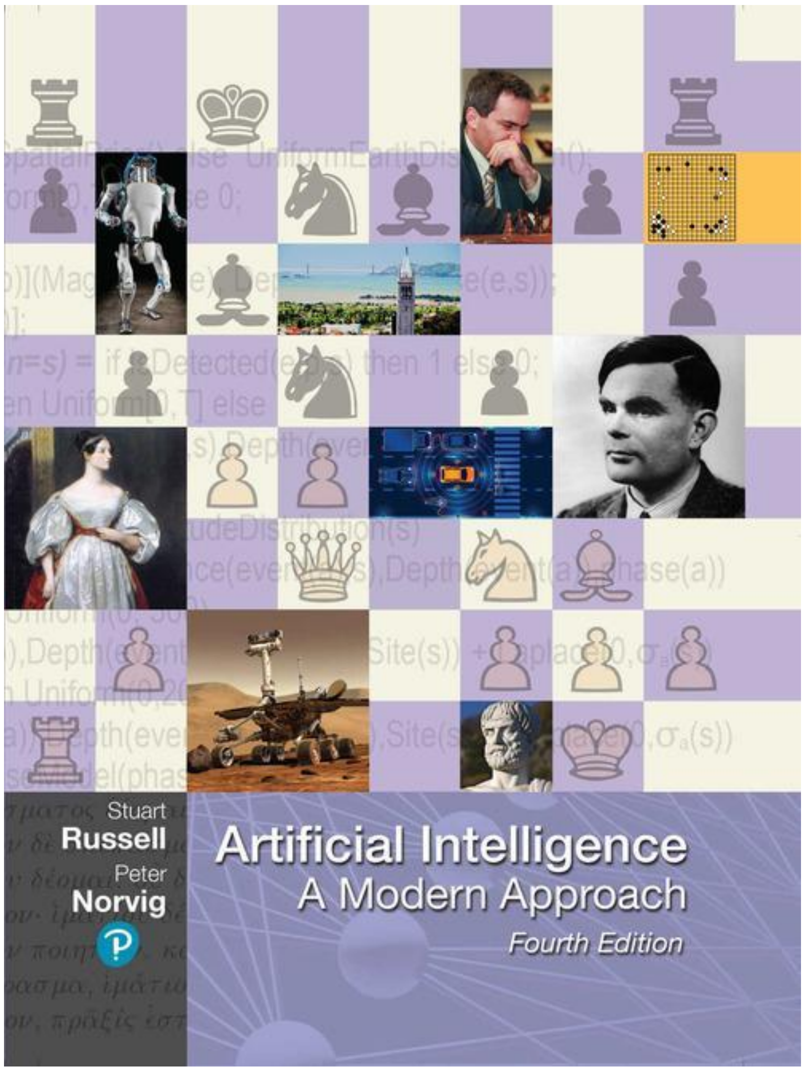
Artificial Intelligence: A Modern Approach

Fourth Edition



Chapter 20

Learning Probabilistic Models

Outline

- Statistical Learning
- Learning with Complete Data
- Learning with Hidden Variables: The EM Algorithm

Full Bayesian learning

View learning as Bayesian updating of a probability distribution over the hypothesis space

H is the hypothesis variable, values $h_1, h_2, \ldots$, prior $\mathbf{P}(H)$

jth observation d_j gives the outcome of random variable D_j
training data $\mathbf{d} = d_1, \ldots, d_N$

Given the data so far, each hypothesis has a posterior probability:

$$P(h_i|\mathbf{d}) = \alpha P(\mathbf{d}|h_i)P(h_i)$$

where $P(\mathbf{d}|h_i)$ is called the likelihood

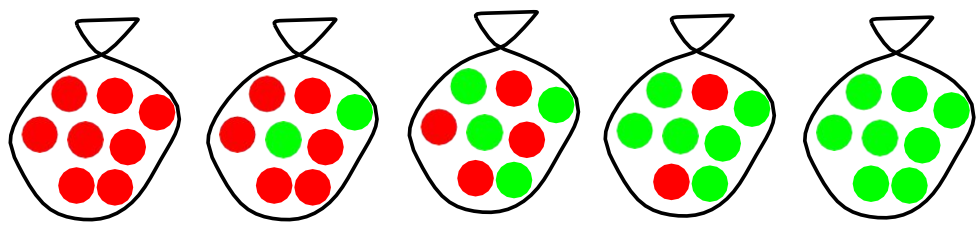
Predictions use a likelihood-weighted average over the hypotheses:

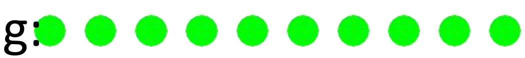
$$\mathbf{P}(X|\mathbf{d}) = \Sigma_i \mathbf{P}(X|\mathbf{d}, h_i)P(h_i|\mathbf{d}) = \Sigma_i \mathbf{P}(X|h_i)P(h_i|\mathbf{d})$$

No need to pick one best-guess hypothesis!

Example

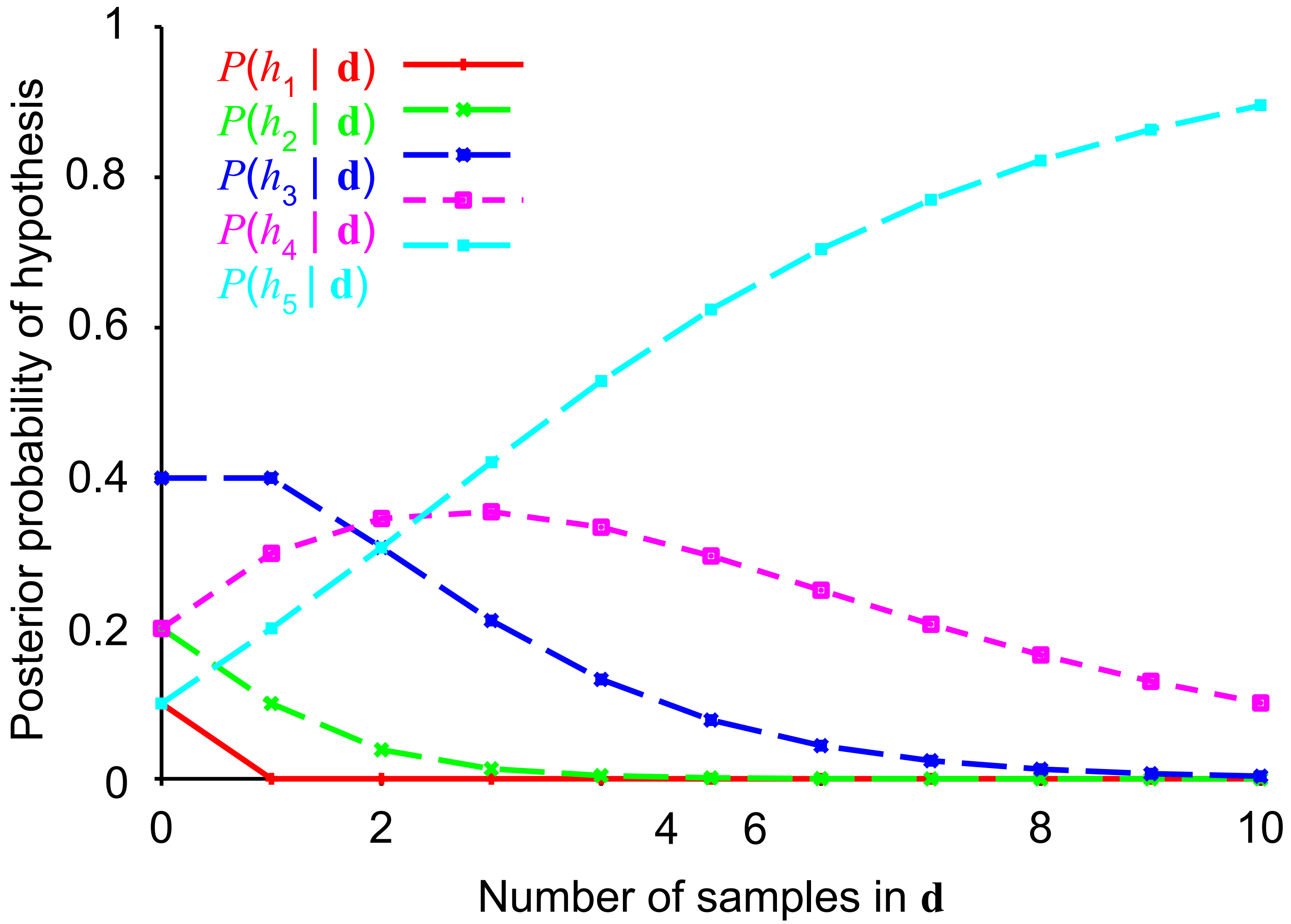
Suppose there are five kinds of bags of candies: 10% are h_1: 100% cherry candies
20% are h_2: 75% cherry candies + 25% lime candies
40% are h_3: 50% cherry candies + 50% lime candies
20% are h_4: 25% cherry candies + 75% lime candies
10% are h_5: 100% lime candies



Then we observe candies drawn from some bag: 

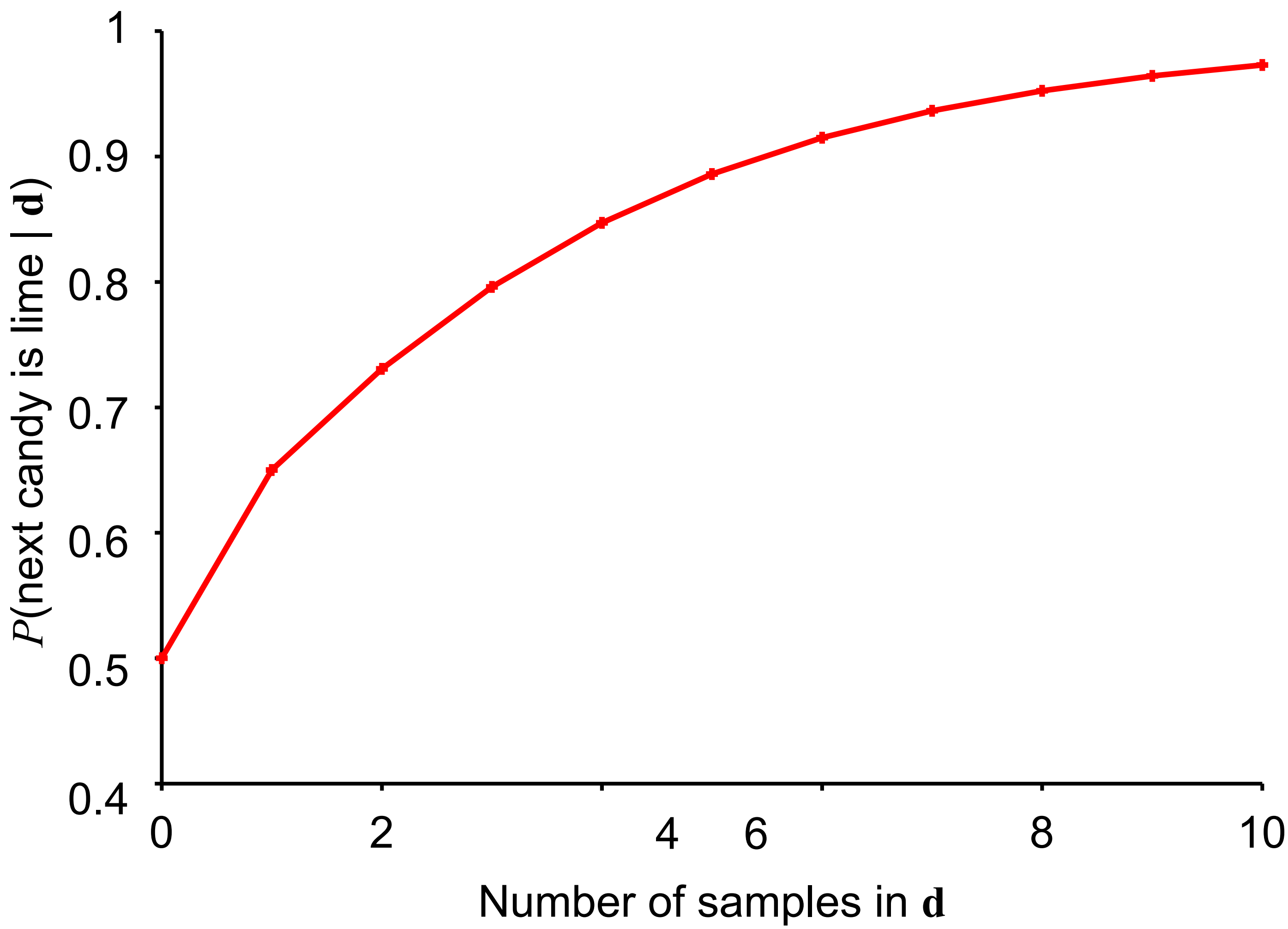
What kind of bag is it? What flavour will the next candy be?

Posterior probability of hypotheses



Pearson

Prediction probability



MAP approximation

Summing over the hypothesis space is often intractable
(e.g., 18,446,744,073,709,551,616 Boolean functions of 6 attributes)

Maximum a posteriori (MAP) learning: choose h_{MAP} maximizing $P(h_i|\mathrm{d})$

I.e., maximize $P(\mathrm{d}|h_i)P(h_i)$ or $\log P(\mathrm{d}|h_i) + \log P(h_i)$

Log terms can be viewed as (negative of)
bits to encode data given hypothesis + bits to encode hypothesis This is the basic idea of minimum description length (MDL) learning

For deterministic hypotheses, $P(\mathrm{d}|h_i)$ is 1 if consistent, 0 otherwise
$\Rightarrow$ MAP = simplest consistent hypothesis (cf. science)

ML approximation

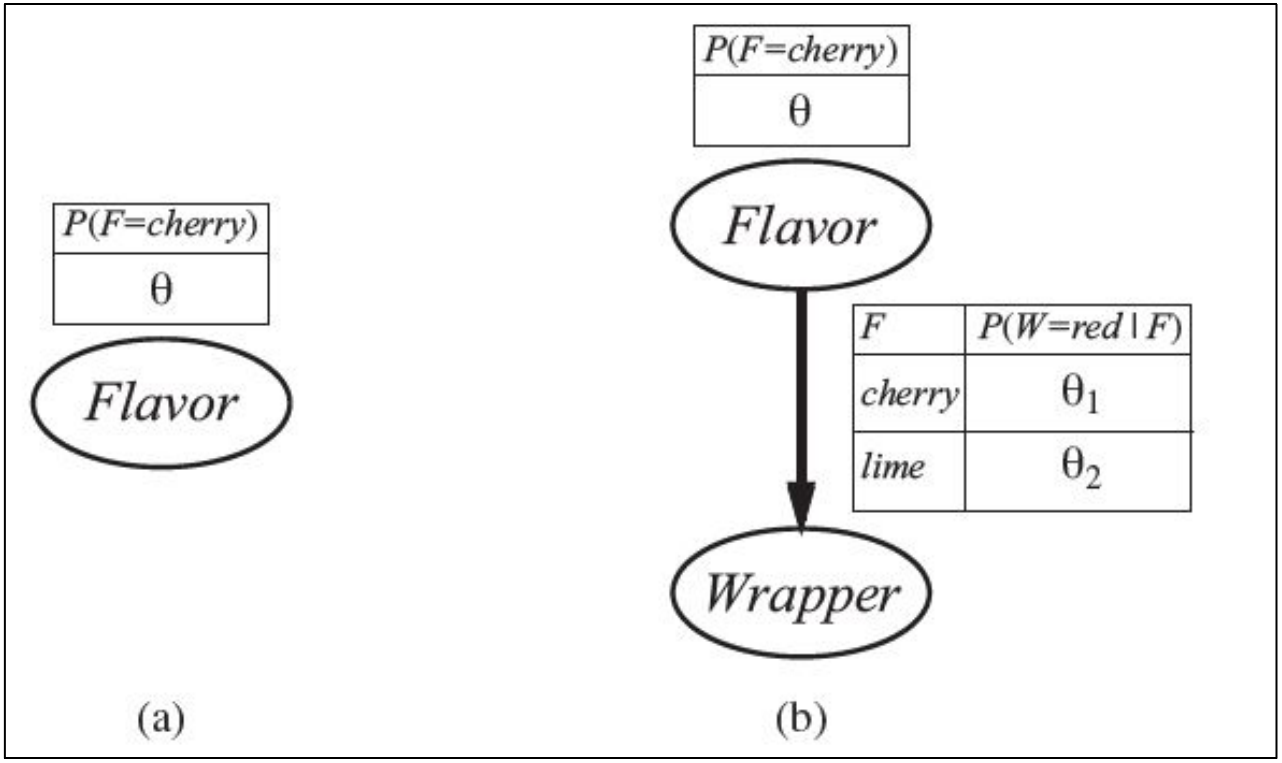
For large data sets, prior becomes irrelevant

Maximum likelihood (ML) learning: choose h_{ML} maximizing $P(\mathrm{d}|h_i)$

I.e., simply get the best fit to the data; identical to MAP for uniform prior (which is reasonable if all hypotheses are of the same complexity)

ML is the "standard" (non-Bayesian) statistical learning method

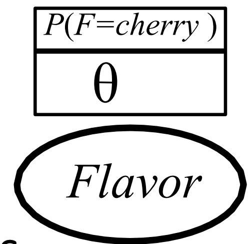
Bayesian Network Model



(a) Bayesian network model for the case of candies with an unknown proportion of cherries and lines.
(b) Model for the case where the wrapper color depends (probabilistically) on the candy flavor.

ML parameter learning in Bayes nets

Bag from a new manufacturer; fraction θ of cherry candies?



Any θ is possible: continuum of hypotheses h_θ

θ is a parameter for this simple (binomial) family of models

Suppose we unwrap N candies, c cherries and $€ = N - c$ limes

These are i.i.d. (independent, identically distributed) observations, so

$$P(\mathrm{d}|h_\theta) = \prod_{j=1}^{N} P(d_j|h_\theta) = \theta^{c} \cdot (1-\theta)^{€}$$

Maximize this w.r.t. θ—which is easier for the log-likelihood:

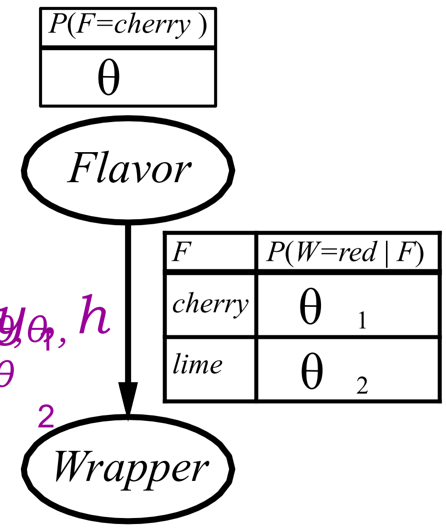
$$L(\mathrm{d}|h_\theta) = \log P(\mathrm{d}|h_\theta) = \sum_{j=1}^{N} \log P(d_j|h_\theta) = c\log\theta + €\log(1-\theta)$$

$$\frac{dL(\mathrm{d}|h_\theta)}{d\theta} = \frac{c}{\theta} - \frac{€}{1-\theta} = 0 \Rightarrow \theta = \frac{c}{c+€} = \frac{c}{N}$$

Seems sensible, but causes problems with 0 counts!

Multiple parameters

Red/green wrapper depends probabilistically on flavor:



Likelihood for, e.g., cherry candy in green wrapper:

$$P(F = cherry\,,\, W = green \mid h_{\theta,\theta_1,\theta_2})$$
$$= P(F = cherry \mid h_{\theta,\theta_1,\theta_2})\,P(W = green \mid F = cherry\,, h_{\theta,\theta_1,\theta_2})$$
$$= \theta \cdot (1 - \theta_1)$$

N candies, r_c red-wrapped cherry candies, etc.:

$$P(\mathrm{d} \mid h_{\theta,\theta_1,\theta_2}) = \theta^c (1-\theta)^\ell \cdot \theta_1^{r_c}(1-\theta_1)^{g_c} \cdot \theta_2^{r_\ell}(1-\theta_2)^{g_\ell}$$

$$L = [c \log \theta + f \log(1-\theta)] + [r_c \log \theta_1 + g_c \log(1-\theta_1)] + [r \log \theta_2 + g \log(1-\theta_2)]$$

Multiple parameters contd.

Derivatives of L contain only the relevant parameter:

$$\frac{\partial L}{\partial \theta} = \frac{c}{\theta} - \frac{\text{€}}{1-\theta} = 0 \Rightarrow \theta = \frac{c}{c+\text{€}}$$

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

$$\frac{\partial L}{\partial \theta_2} = \frac{r_{\text{€}}}{\theta_2} - \frac{g_{\text{€}}}{1-\theta_2} = 0 \Rightarrow \theta_2 = \frac{r_{\text{€}}}{r_{\text{€}}+g_{\text{€}}}$$

With complete data, parameters can be learned separately

Naive Bayes Models

Assuming Boolean variables, the parameters are

$$\theta = P(C = true), \theta_{i1} = P(X_i = true \mid C = true), \theta_{i2} = P(X_i = true \mid C = false).$$

With observed attribute values $x_1, \ldots, x_n$, the probability of each class is given by

$$\mathbf{P}(C \mid x_1, \ldots, x_n) = \alpha\, \mathbf{P}(C) \prod_i \mathbf{P}(x_i \mid C)\ .$$

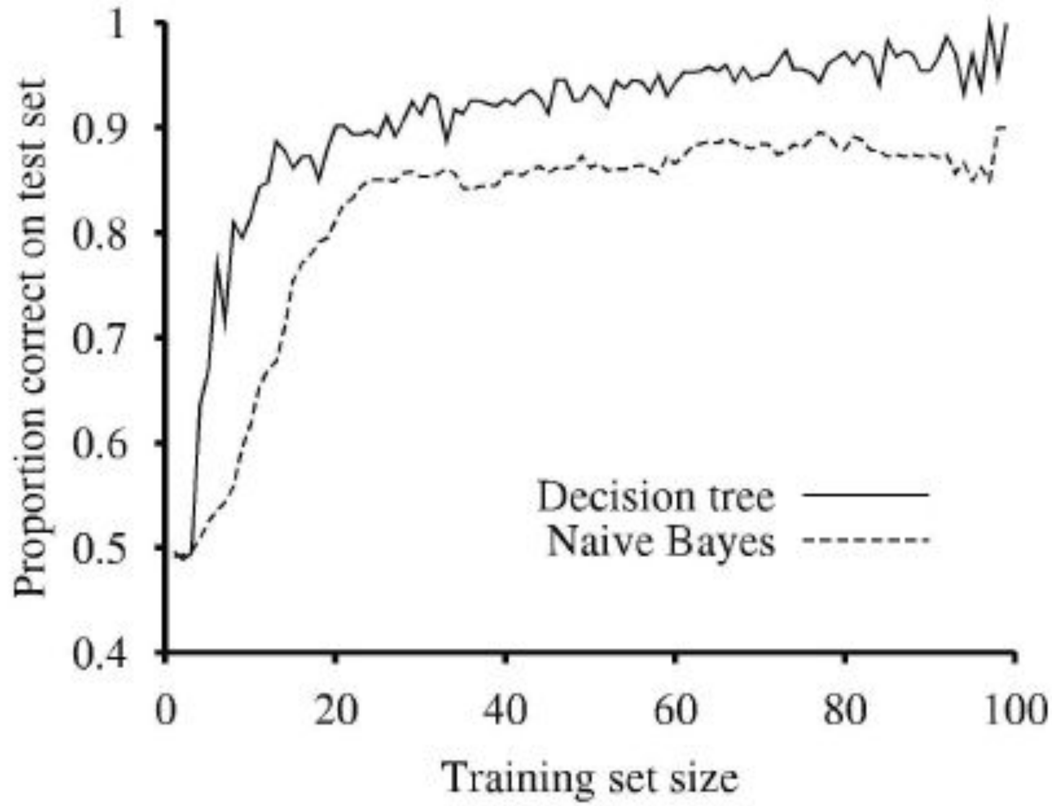
A deterministic prediction can be obtained by choosing the most likely class

The method learns fairly well but not as well as decision-tree learning; this is presumably because the true hypothesis-which is a decision tree-is not representable exactly using a naive Bayes model.

Naive Bayes learning turns out to do surprisingly well in a wide range of applications; the boosted version

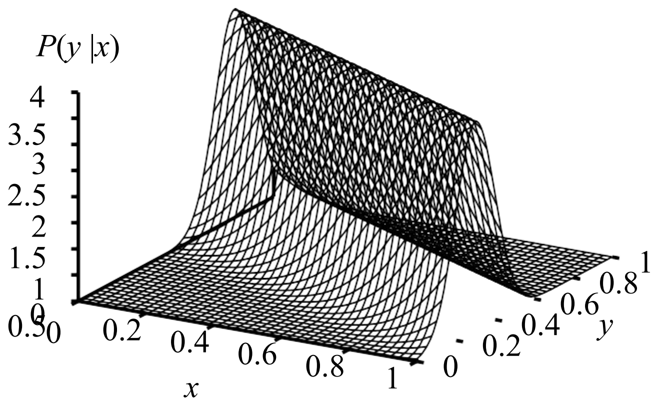
Scales well to large problems with n Boolean attributes there are just $2n + 1$ parameters

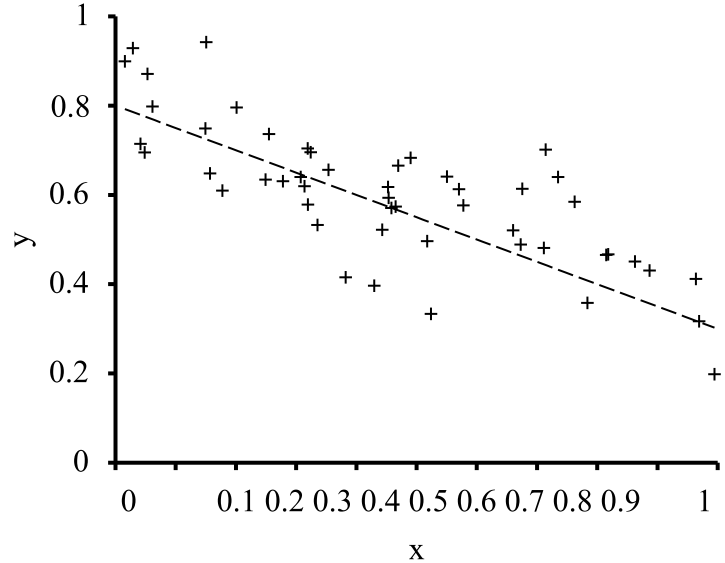
Naive Bayes Models



The learning curve for naive Bayes learning applied to the restaurant problem from chapter 18. Compared with decision tree learning.

Example: linear Gaussian model





Maximizing $P(y|x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y-(\theta_1 x+\theta_2))^2}{2\sigma^2}}$ w.r.t. θ_1, θ_2

= minimizing $E = \sum_{j=1}^{N} (y_j - (\theta_1 x_j + \theta_2))^2$

That is, minimizing the sum of squared errors gives the ML solution for a linear fit assuming Gaussian noise of fixed variance

Naive Bayes Models

Bayesian parameter learning

Hypothesis prior: Bayesian approach to parameter learning starts by defining a prior probability distribution over the possible hypotheses

In the Bayesian view, θ is the (unknown) value of a random variable $\mathbf{\Theta}$ that defines the hypothesis space

The hypothesis prior is just the prior distribution **$\mathbf{P}(\mathbf{\Theta})$.**

Thus, $P(\mathbf{\Theta} = \theta)$ is the prior probability that the bag has a fraction θ of cherry candies.

$P(\theta) = Uniform[0,1](\theta)$, uniform density is part of beta distributions.

Each beta distribution is defined by two **hyperparameters** a and b such that

$$\text{beta}[a, b](\theta) = \alpha\, \theta^{a-1}(1-\theta)^{b-1} ,$$

Naive Bayes Models

Bayesian parameter learning

$$\begin{aligned}
P(\theta \mid D_1 = cherry) &= \alpha\, P(D_1 = cherry \mid \theta) P(\theta) \\
&= \alpha'\, \theta \cdot \text{beta}[a, b](\theta) = \alpha'\, \theta \cdot \theta^{a-1}(1-\theta)^{b-1} \\
&= \alpha'\, \theta^{a}(1-\theta)^{b-1} = \text{beta}[a+1, b](\theta)\ .
\end{aligned}$$

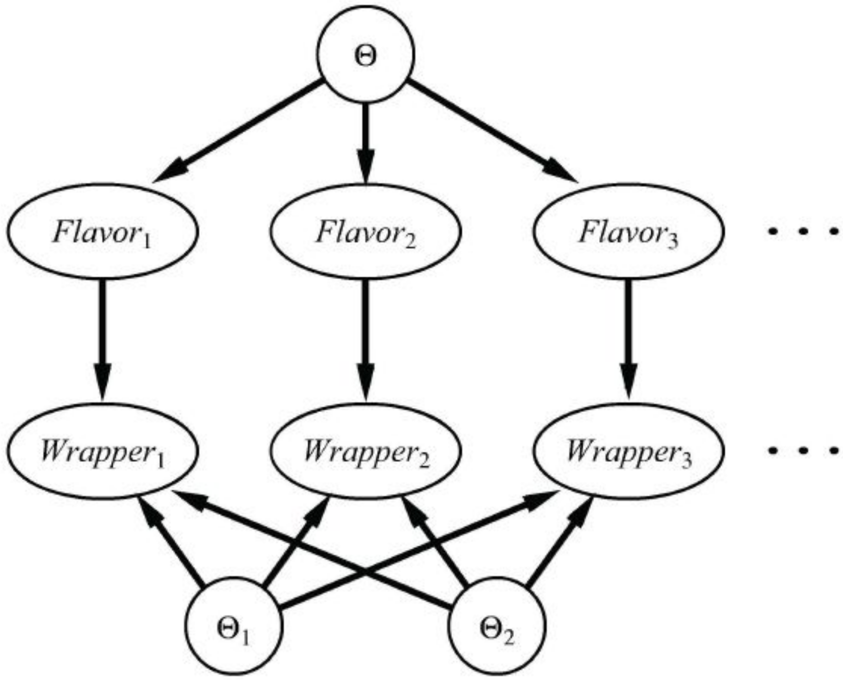
$$\mathbf{P}(\Theta, \Theta_1, \Theta_2) = \mathbf{P}(\Theta)\mathbf{P}(\Theta_1)\mathbf{P}(\Theta_2)\ .$$

$$P(Flavor_i = cherry \mid \Theta = \theta) = \theta\ .$$

$$\begin{aligned}
&P(Wrapper_i = red \mid Flavor_i = cherry, \Theta_1 = \theta_1) = \theta_1 \\
&P(Wrapper_i = red \mid Flavor_i = lime, \Theta_2 = \theta_2) = \theta_2\ .
\end{aligned}$$

Naive Bayes Models

Bayesian parameter learning



A Bayesian network that corresponds to a Bayesian learning process. Posterior distributions for the parameter variables **Θ**, **Θ** 1, and **Θ** 2 can be inferred from their prior distributions and the evidence in the $Flavor_i$ and $Wrapper_i$ variables.

Naive Bayes Models

Density estimation with nonparametric models

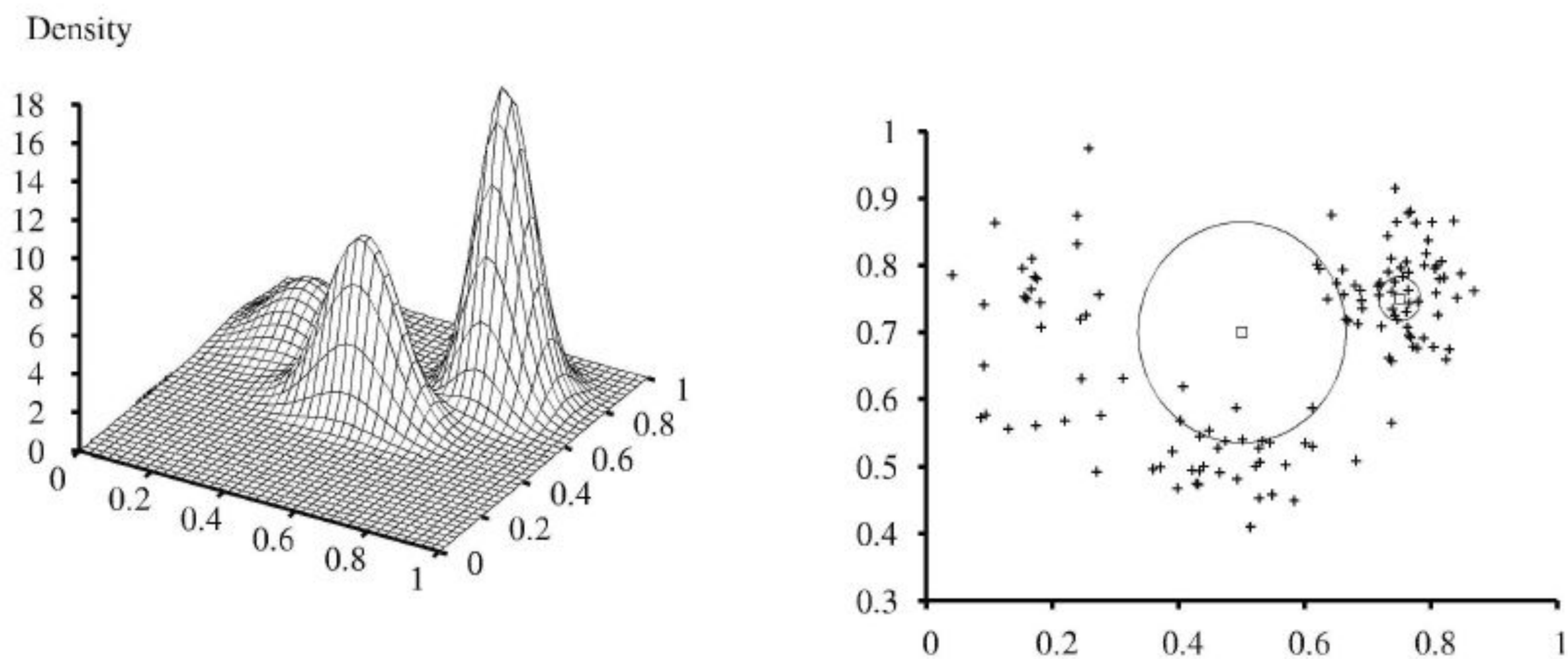
- *k*-nearest neighbors

to estimate the unknown probability density at a query point $\mathbf{x}$, we can simply measure the **density of the data points** in the neighborhood of $\mathbf{x}$.

User kernel functions

$$P(\mathbf{x}) = \frac{1}{N}\sum_{j=1}^{N}\mathcal{K}(\mathbf{x}, \mathbf{x}_j) \ .$$

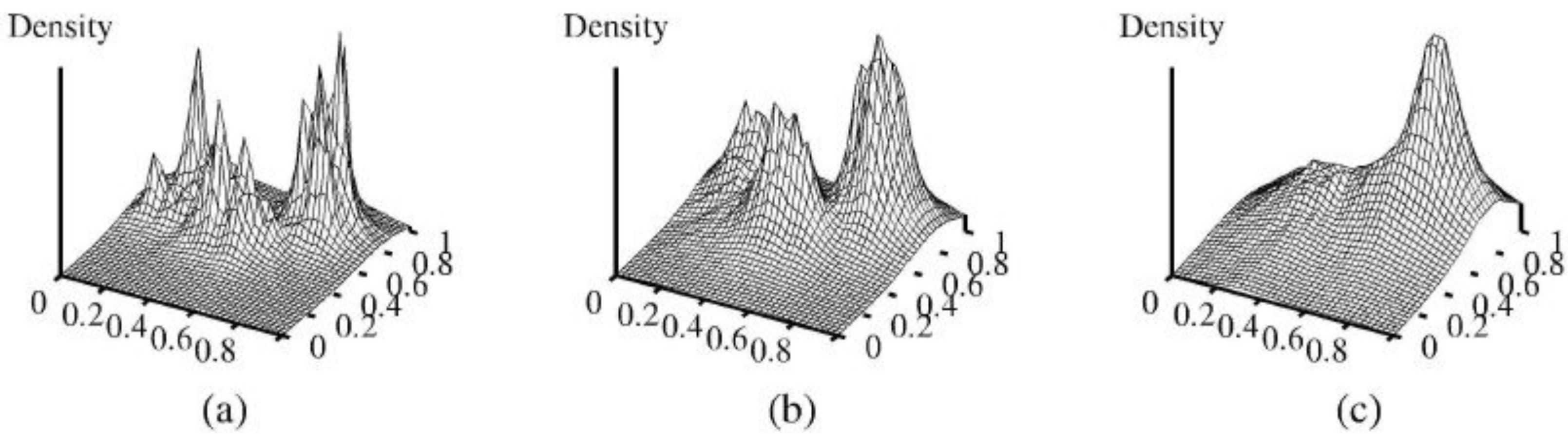
Naive Bayes Models



(a) (b)

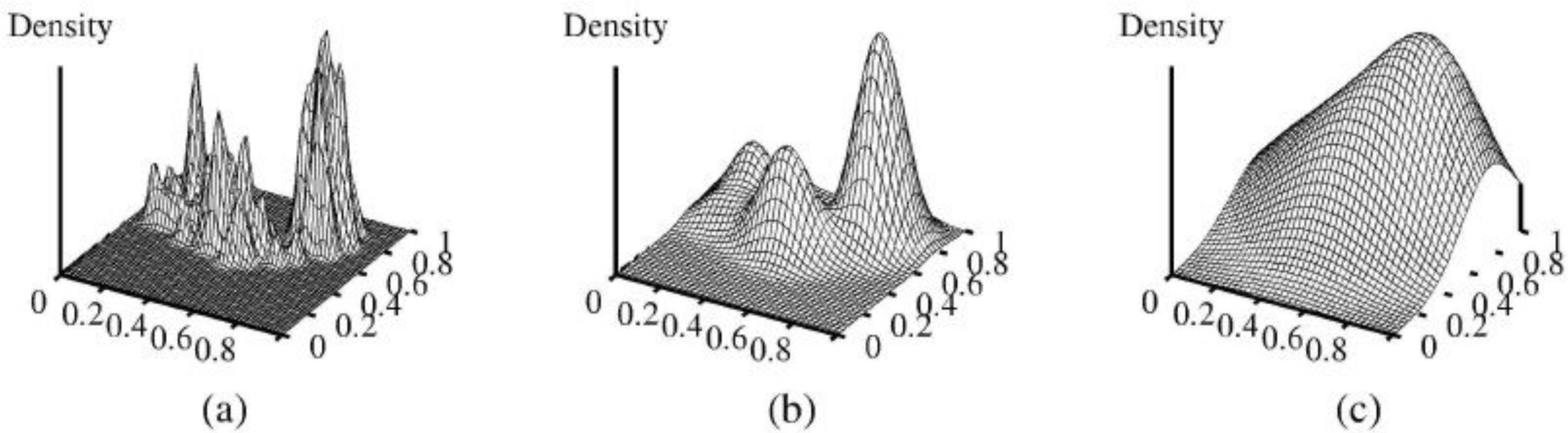
a) A 3D plot of the mixture of Gaussians
b) A 128- point sample of points from the mixture, together with two query points (small squares) and their IO-nearest-neighborhoods (medium and large circles).

Naive Bayes Models



Density estimation using k-nearest-neighbors, for k= 3, 10, and 40 respectively.
k = 3 is too spiky, 40 is too smooth, and 10 is just about right.
The best value for k can be chosen by cross-validation

Naive Bayes Models



Kernel density estimation using Gaussian kernels with w = 0.02, 0.07, and 0.20 respectively. w = 0.07 is about right.

Learning With Hidden Variables: The EM Algorithm

Many real-world problems have **hidden variables** (sometimes called **latent variables)**

Expectation-maximization helps with hidden variables

- infer the probability that each data point belongs to each component.
- refit the components to the data, where each component is fitted to the entire data set with each point weighted by the probability that it belongs to that component.
- The process iterates until convergence

For the **mixture of Gaussians**, initialize the mixture-model parameters arbitrarily and iterate the **E-step & M-step**

$$\begin{aligned}
\boldsymbol{\mu}_i &\leftarrow \sum_j p_{ij}\mathbf{x}_j/n_i \\
\boldsymbol{\Sigma}_i &\leftarrow \sum_j p_{ij}(\mathbf{x}_j-\boldsymbol{\mu}_i)(\mathbf{x}_j-\boldsymbol{\mu}_i)^\top/n_i \\
w_i &\leftarrow n_i/N
\end{aligned}$$

- Observations: the log likelihood for the final learned model slightly *exceeds* that of the original model & EM increases the log likelihood of the data at every iteration.

Pearson

Learning With Hidden Variables: The EM Algorithm

Learning Bayesian networks with hidden variables

Example: a situation in which there are two bags of candies that have been mixed together.

- Candies are described by three features: in addition to the *Flavor* and the *Wrapper,* some candies have a *Hole* in the middle and some do not.
- The distribution of candies in each bag is described by a **naive Bayes** model: the features are independent, given the bag, but the conditional probability distribution for each feature depends on the bag.

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

Learning Bayesian networks with hidden variables

The *expected* count of $\hat{N}(Bag = 1)$ is the sum, over all candies, of the probability that the candy came from bag 1:

$$\theta^{(1)} = \hat{N}(Bag = 1)/N = \sum_{j=1}^{N} P(Bag = 1 \mid flavor_j, wrapper_j, holes_j)/N .$$

using Bayes' rule and applying conditional independence

$$\theta^{(1)} = \frac{1}{N} \sum_{j=1}^{N} \frac{P(flavor_j \mid Bag = 1) P(wrapper_j \mid Bag = 1) P(holes_j \mid Bag = 1) P(Bag = 1)}{\sum_i P(flavor_j \mid Bag = i) P(wrapper_j \mid Bag = i) P(holes_j \mid Bag = i) P(Bag = i)} .$$

The *expected* count of cherry candies from bag 1 is given by

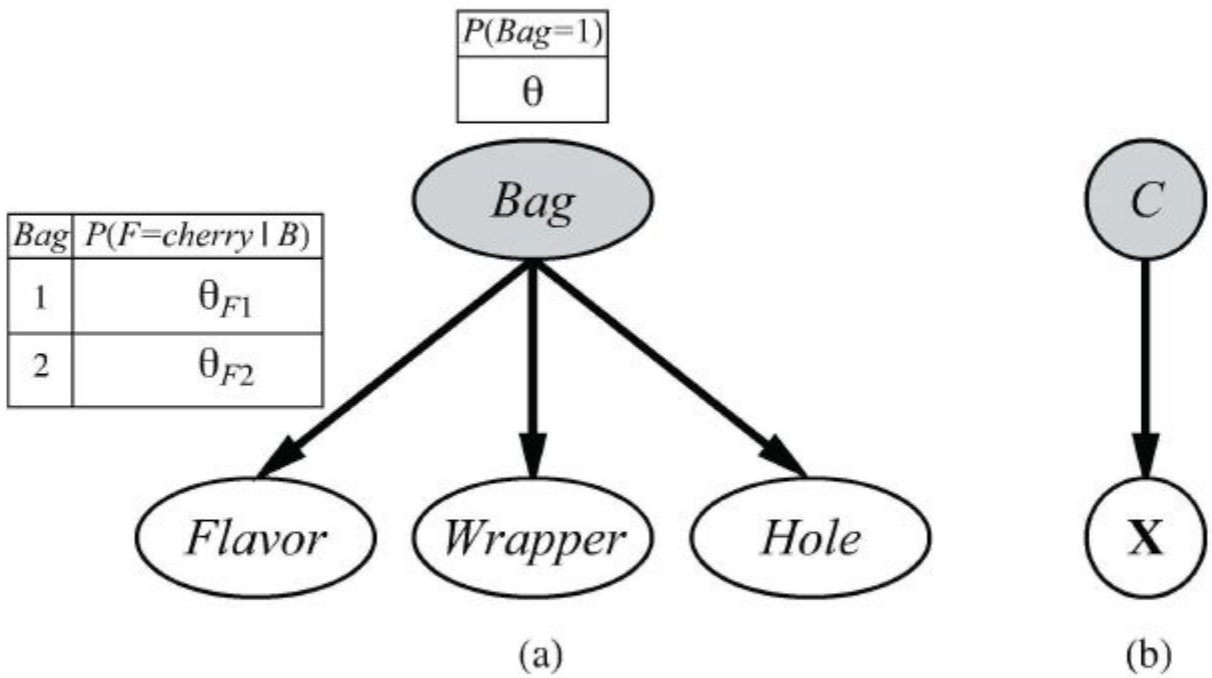
$$\sum_{j: Flavor_j = cherry} P(Bag = 1 \mid Flavor_j = cherry, wrapper_j, holes_j) .$$

The update is given by the normalized expected counts as follows

$$\theta_{ijk} \leftarrow \hat{N}(X_i = x_{ij}, \mathbf{U}_i = \mathbf{u}_{ik}) / \hat{N}(\mathbf{U}_i = \mathbf{u}_{ik}) .$$

Learning With Hidden Variables: The EM Algorithm

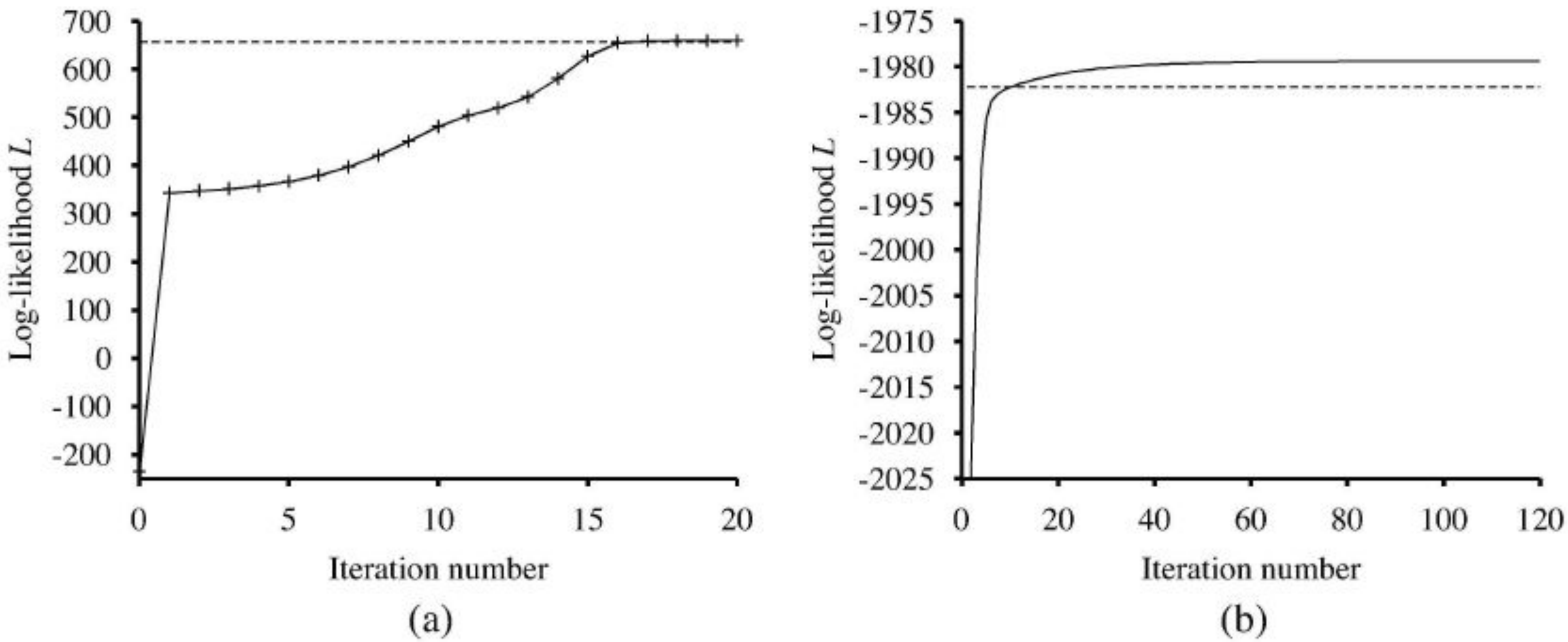
Learning Bayesian networks with hidden variables



(a) A mixture model for candy. The proportions of different flavors, wrappers, presence of holes depend on the bag, which is not observed.

(b) Bayesian network for a Gaussian mixture. The mean and covariance of the observable variables **X** depend on the component *C*.

Learning With Hidden Variables: The EM Algorithm



Graphs showing the log likelihood of the data, L, as a function of the EM iteration (a) Graph for Gaussian mixture model (b) Graph for the Bayesian network

Learning With Hidden Variables: The EM Algorithm

Learning hidden Markov models

One application of EM involves learning the transition probabilities in hidden Markov models **(HMMs).**

A hidden Markov model can be represented by a dynamic Bayes net with a single discrete state variable

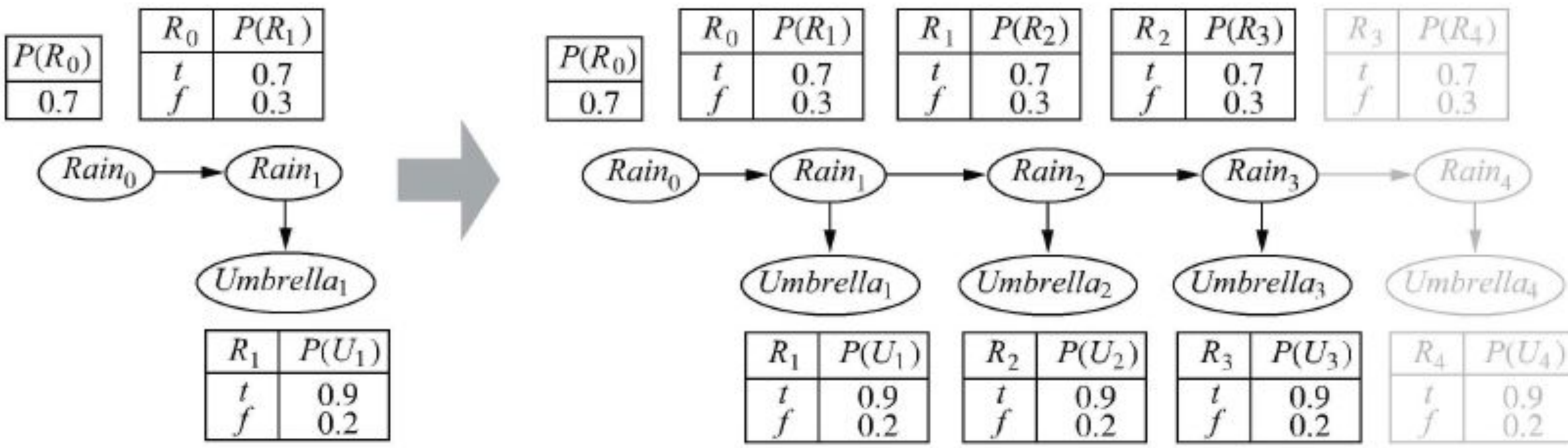
Each data point consists of an observation *sequence* of finite length

transition probability from state i to state *j*,

- calculate the expected proportion of times that the system undergoes a transition to state *j* when in state *i*:

$$\theta_{ij} \leftarrow \sum_t \hat{N}(X_{t+1} = j, X_t = i) / \sum_t \hat{N}(X_t = i) .$$

Learning With Hidden Variables: The EM Algorithm



An unrolled dynamic Bayesian network that represents a hidden Markovmodel

Learning With Hidden Variables: The EM Algorithm

General equation of **EM**

X: all the observed values in all the examples,

Z: all the hidden variables for all the examples,

θ: all the parameters for the probability model

$$\boldsymbol{\theta}^{(i+1)} = \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{\mathbf{z}} P(\mathbf{Z}=\mathbf{z} \mid \mathbf{x}, \boldsymbol{\theta}^{(i)}) L(\mathbf{x}, \mathbf{Z}=\mathbf{z} \mid \boldsymbol{\theta}) .$$

The E-step is the computation of the summation

The M-step is the maximization of this expected log likelihood with respect to the parameters.

Summary

Bayesian learning methods formulate learning as a form of probabilistic inference, using the observations to update a prior distribution over hypotheses.

Maximum a posteriori (MAP) learning selects a single most likely hypothesis given the data.

Maximum-likelihood learning simply selects the hypothesis that maximizes the likelihood of the data; it is equivalent to MAP learning with a uniform prior.

Naive Bayes learning is a particularly effective technique that scales

When some variables are hidden, local maximum likelihood solutions can be found using the EM algorithm

Nonparametric models represent a distribution using the collection of data points.