Learning Probabilistic Models

CMPSCI 383
Nov 22, 2011

Today’s topics

- Full Bayesian Learning
- MAP approximation
- ML approximation
- ML parameter learning in Bayes nets
  - Naïve Bayes Model
  - Linear Gaussian Model
- Bayesian parameter learning
  - Beta family of distributions
  - Conjugate families
- Latent variables
- Expectation Maximization (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:

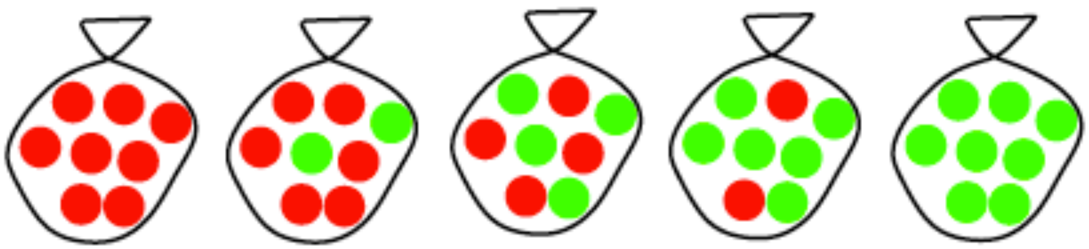
$$\mathbf{P}(X|\mathbf{d}) = \Sigma_i\ \mathbf{P}(X|\mathbf{d}, h_i)P(h_i|\mathbf{d}) = \Sigma_i\ \mathbf{P}(X|h_i)P(h_i|\mathbf{d})$$

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



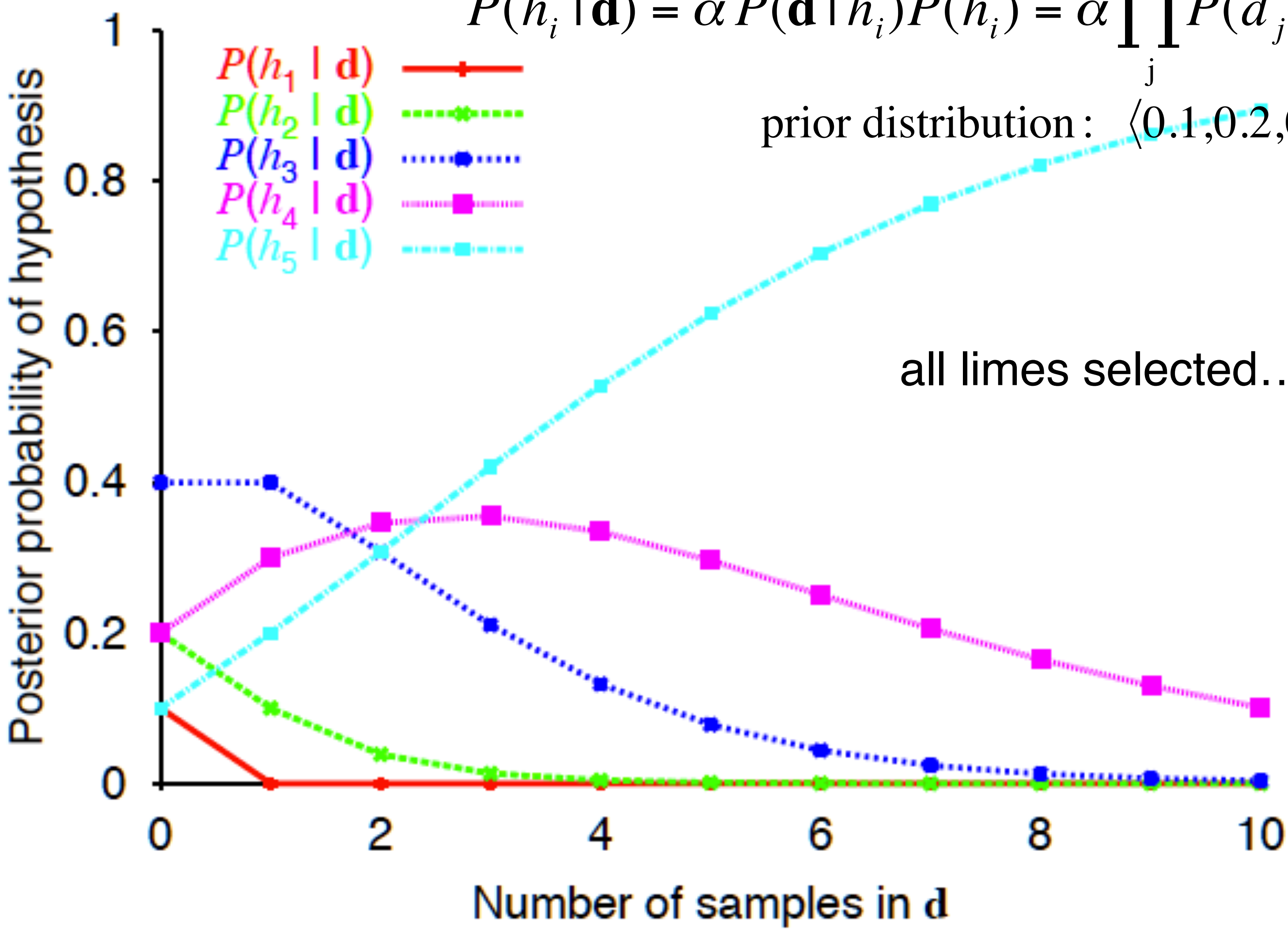
Then we observe candies drawn from some bag: 

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

Posterior Probabilities of the Hypotheses

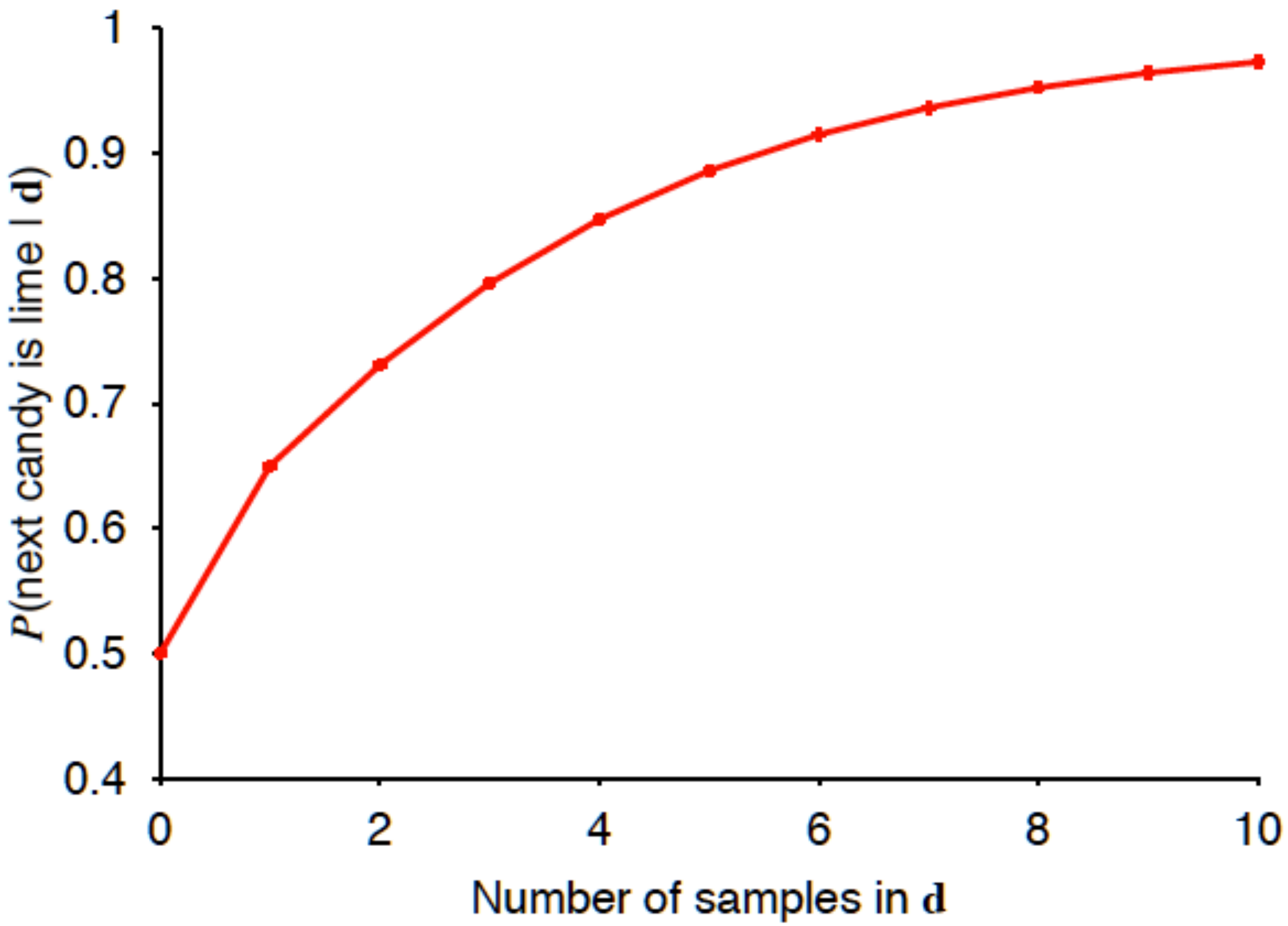
$$P(h_i \mid \mathbf{d}) = \alpha P(\mathbf{d} \mid h_i) P(h_i) = \alpha \prod_j P(d_j \mid h_i) P(h_i)$$

prior distribution : $\langle 0.1, 0.2, 0.4, 0.2, 0.1 \rangle$



all limes selected…

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

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

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

ML approximation

For large data sets, prior becomes irrelevant

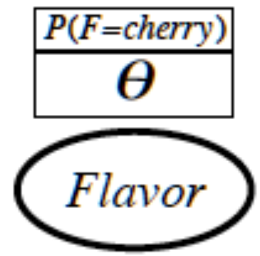
Maximum likelihood (ML) learning: choose h_{ML} maximizing $P(\mathbf{d}|h_i)$

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

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 $\ell = N - c$ limes
These are i.i.d. (independent, identically distributed) observations, so

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

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

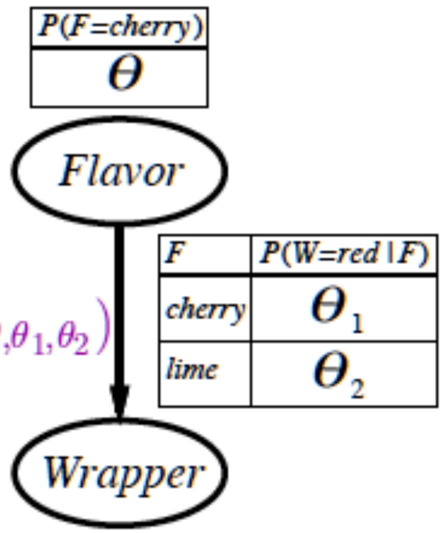
$$L(\mathbf{d}|h_\theta) = \log P(\mathbf{d}|h_\theta) = \sum_{j=1}^{N} \log P(d_j|h_\theta) = c\log\theta + \ell\log(1-\theta)$$

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

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:

$$\begin{aligned} & P(F = cherry, W = green | h_{\theta,\theta_1,\theta_2}) \\ & = P(F = cherry | h_{\theta,\theta_1,\theta_2}) P(W = green | F = cherry, h_{\theta,\theta_1,\theta_2}) \\ & = \theta \cdot (1 - \theta_1) \end{aligned}$$

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

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

$$\begin{aligned} L = \ & [c \log \theta + \ell \log(1-\theta)] \\ + \ & [r_c \log \theta_1 + g_c \log(1-\theta_1)] \\ + \ & [r_\ell \log \theta_2 + g_\ell \log(1-\theta_2)] \end{aligned}$$

Multiple parameters contd.

Derivatives of L contain only the relevant parameter:

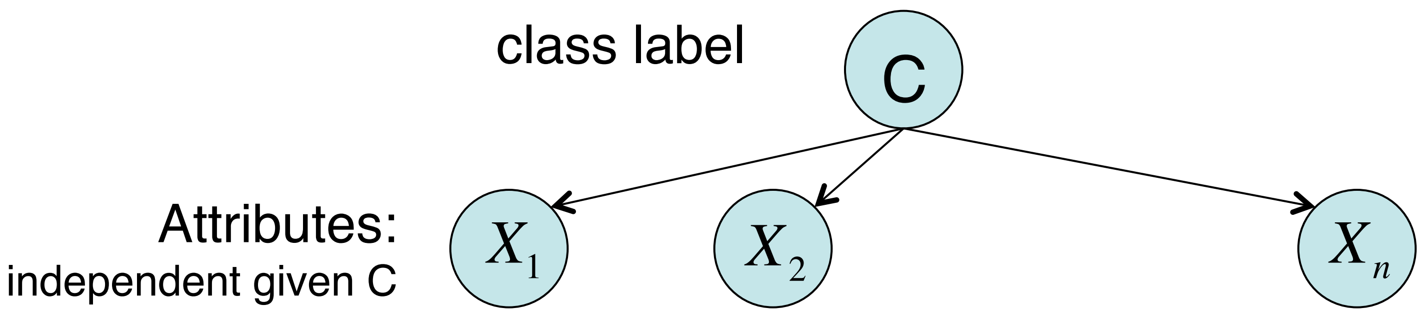
$$\frac{\partial L}{\partial \theta} = \frac{c}{\theta} - \frac{\ell}{1-\theta} = 0 \qquad \Rightarrow \quad \theta = \frac{c}{c+\ell}$$

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

$$\frac{\partial L}{\partial \theta_2} = \frac{r_\ell}{\theta_2} - \frac{g_\ell}{1-\theta_2} = 0 \qquad \Rightarrow \quad \theta_2 = \frac{r_\ell}{r_\ell+g_\ell}$$

With complete data, parameters can be learned separately

Naïve Bayes Model



$$\mathbf{P}(\mathrm{C} \mid \mathrm{x}_1, x_2, \ldots, x_n) = \alpha\, \mathbf{P}(C) \prod_i \mathbf{P}(x_i \mid C)$$

Naïve Bayes Classifier:

$$\mathrm{C}_{\mathrm{NB}} = \mathrm{argmax}_{\mathrm{C} \in \mathrm{lables}} \mathbf{P}(\mathrm{C} \mid \mathrm{x}_1, x_2, \ldots, x_n) = \arg\max \alpha\, \mathbf{P}(C) \prod_i \mathbf{P}(x_i \mid C)$$

Naïve Bayes contd.

$$\mathrm{C_{NB}} = \mathrm{argmax}_{\mathrm{C}\in\mathrm{lables}} \mathbf{P}(\mathrm{C} \mid \mathrm{x}_1, x_2, \ldots, x_n) = \arg\max \alpha \mathbf{P}(C) \prod_i \mathbf{P}(x_i \mid C)$$

Or, taking logs and dropping α :

$$\mathrm{C_{NB}} = \mathrm{argmax}_{\mathrm{C}\in\mathrm{lables}} \log \mathbf{P}(\mathrm{C} \mid \mathrm{x}_1, x_2, \ldots, x_n) = \log \mathbf{P}(C) \prod_i \mathbf{P}(x_i \mid C)$$

$$= \log P(c) + \sum_i \log \mathbf{P}(x_i \mid C)$$

→ a linear classifier

Naïve Bayes vs. decision tree

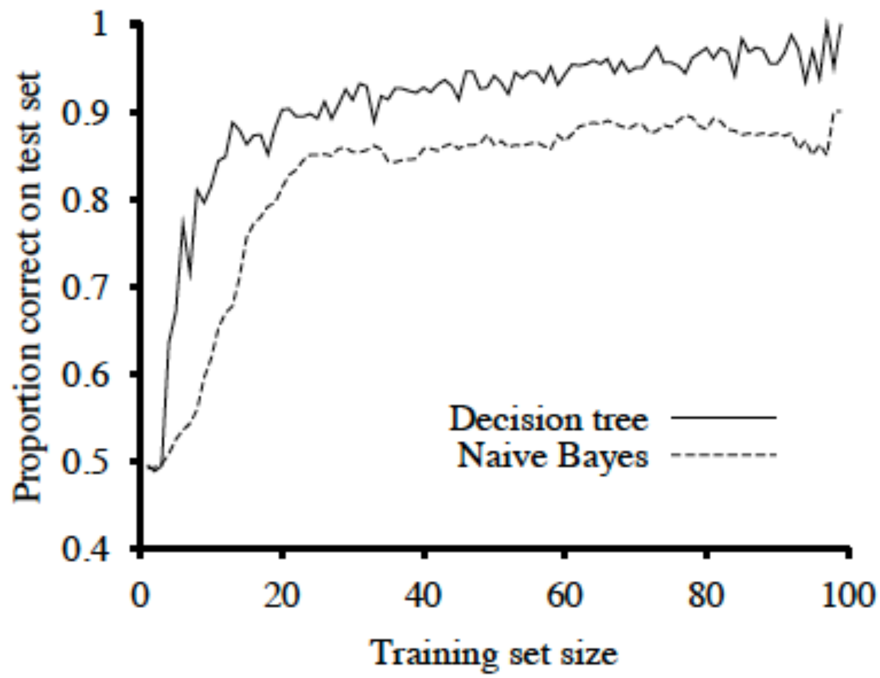
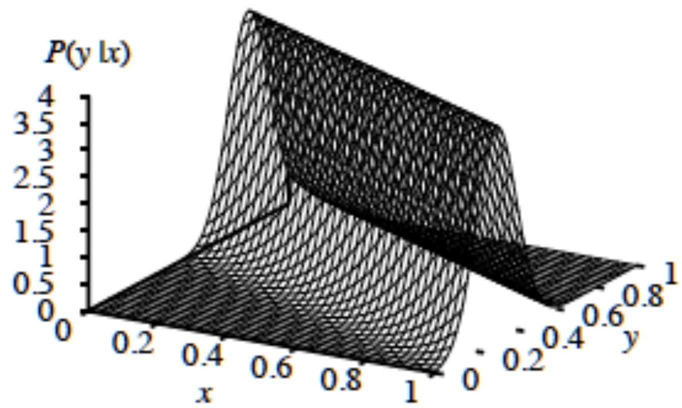
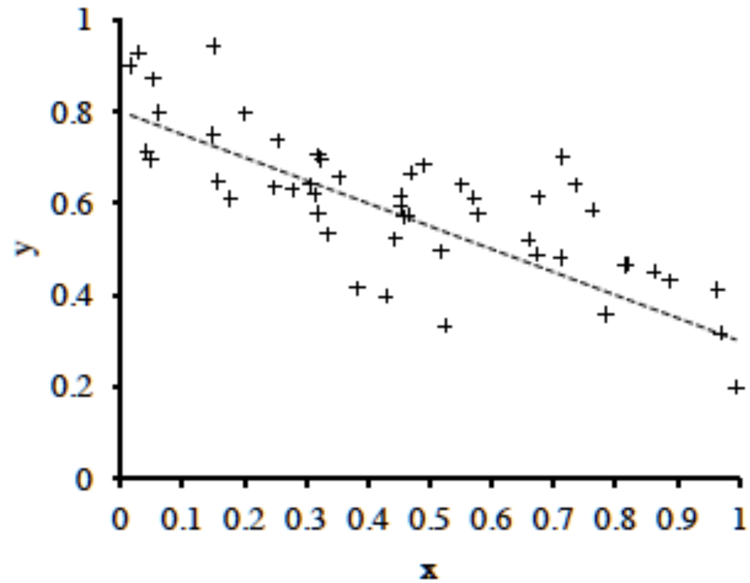


Figure 20.3 **FILES: .** The learning curve for naive Bayes learning applied to the restaurant problem from Chapter 18; the learning curve for decision-tree learning is shown for comparison.

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

Summary so far

Full Bayesian learning gives best possible predictions but is intractable

MAP learning balances complexity with accuracy on training data

Maximum likelihood assumes uniform prior, OK for large data sets

1. Choose a parameterized family of models to describe the data
requires substantial insight and sometimes new models

2. Write down the likelihood of the data as a function of the parameters
may require summing over hidden variables, i.e., inference

3. Write down the derivative of the log likelihood w.r.t. each parameter

4. Find the parameter values such that the derivatives are zero
may be hard/impossible; modern optimization techniques help

Full Bayesian parameter learning

- ML learning is simple but has some problems:
  - e.g., after seeing one sample, the ML estimate is %100 that sample
- Bayesian approach starts with a **hypothesis prior,** which is revised using Bayes rule as more data comes in.
- E.g., consider one unknown parameter θ

We start with a prob. distribution over values of θ : e.g., the prior probability that a bag has a fraction θ of cherries.

Beta family of distributions

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

a and *b* are called **hyperparameters**

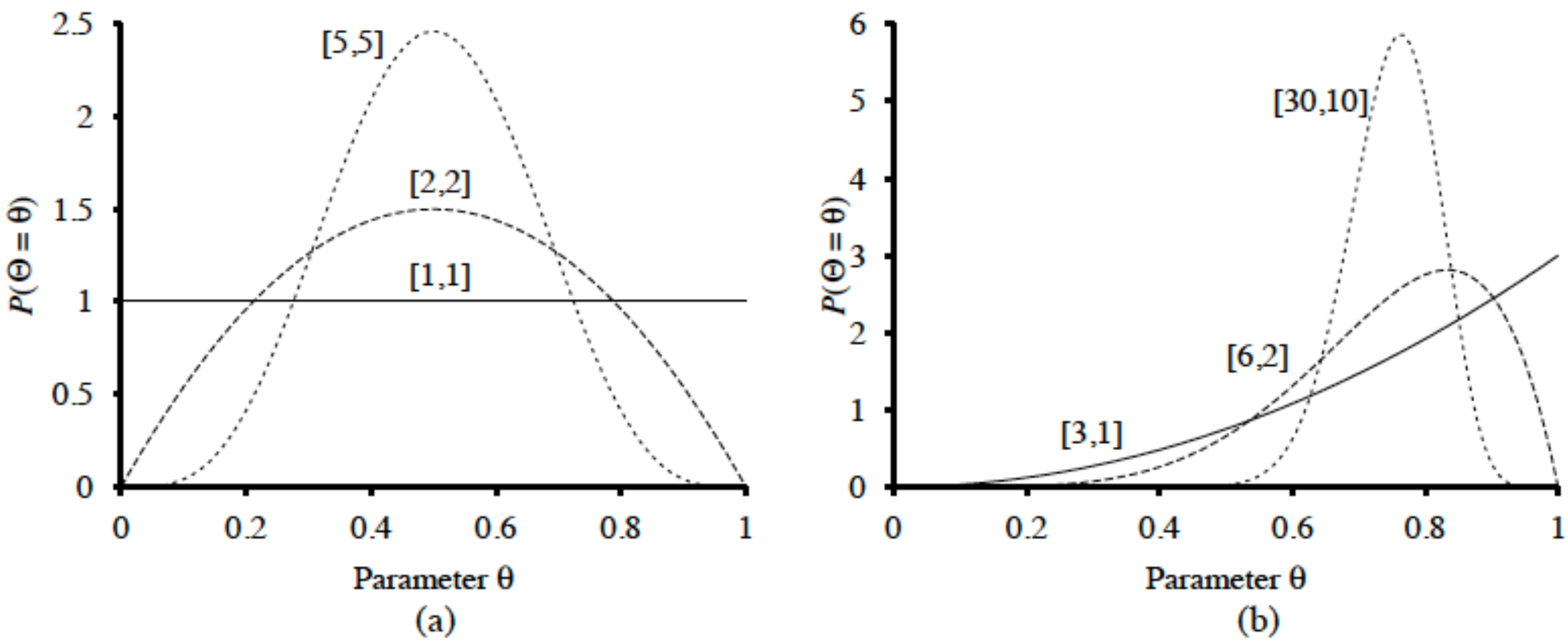


Figure 20.5 **FILES: .** Examples of the beta$[a, b]$ distribution for different values of $[a, b]$.

Conjugate families of distributions

- E.g., the Beta family

Closed under Bayesian updates

$$P(\theta \mid D_1 = cherry) = \alpha\, P(D_1 = cherry \mid \theta) P(\theta)$$

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

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

Nonparametric density estimation

k-nearest-neighbors

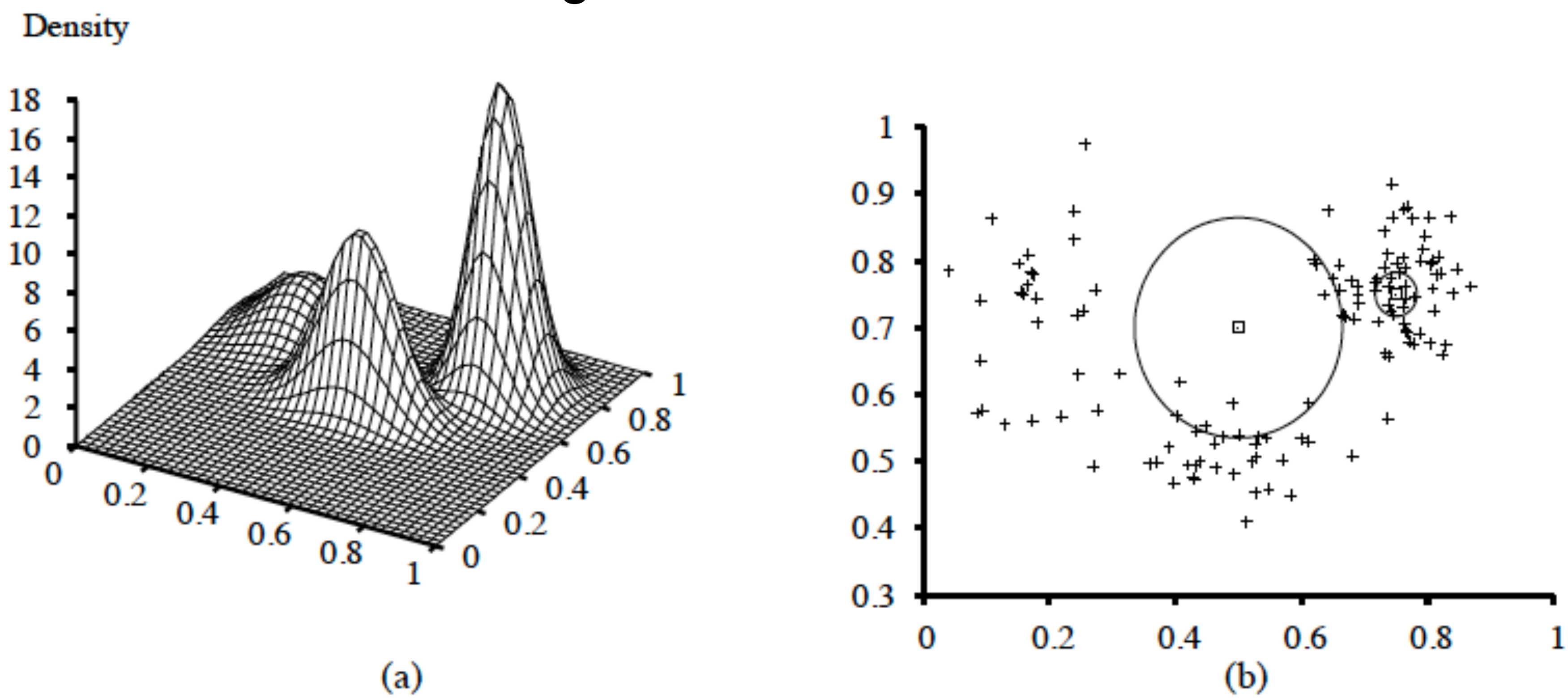


Figure 20.7 **FILES: .** (a) A 3D plot of the mixture of Gaussians from Figure 20.11(a). (b) A 128-point sample of points from the mixture, together with two query points (small squares) and their 10-nearest-neighborhoods (medium and large circles).

Nonparametric density estimation contd.

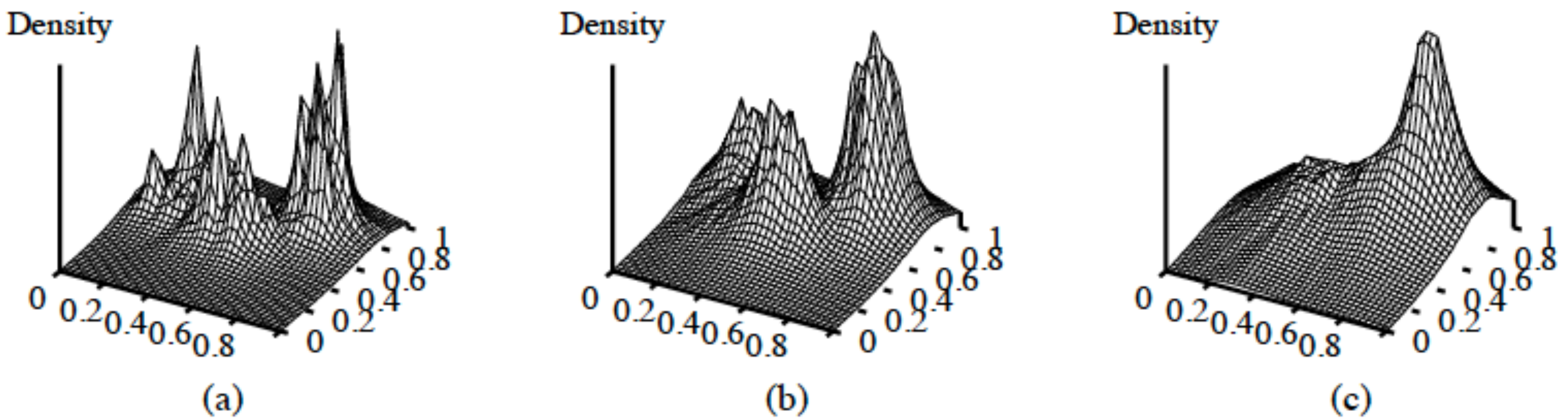


Figure 20.8 **FILES: .** Density estimation using k-nearest-neighbors, applied to the data in Figure 20.7(b), for $k = 3, 10$, and 40 respectively. $k = 3$ is too spiky, 40 is too smooth, and 10 is just about right. The best value for k can be chosen by cross-validation.

Nonparametric density estimation contd.

kernel density estimation

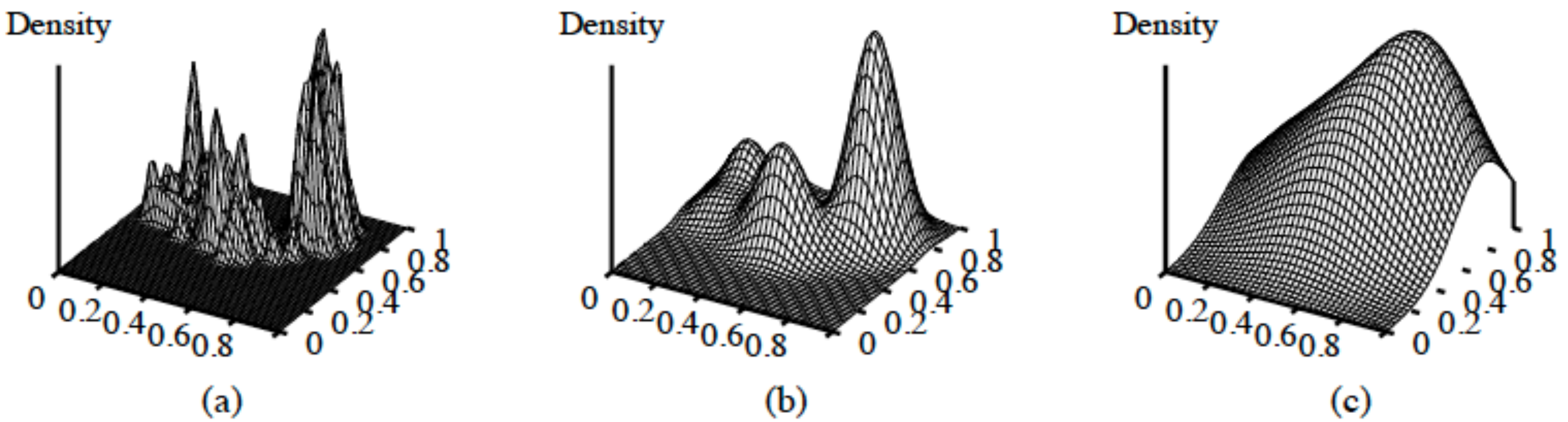


Figure 20.9 **FILES: .** Kernel density estimation for the data in Figure 20.7(b), using Gaussian kernels with $w = 0.02, 0.07$, and 0.20 respectively. $w = 0.07$ is about right.

Latent variables

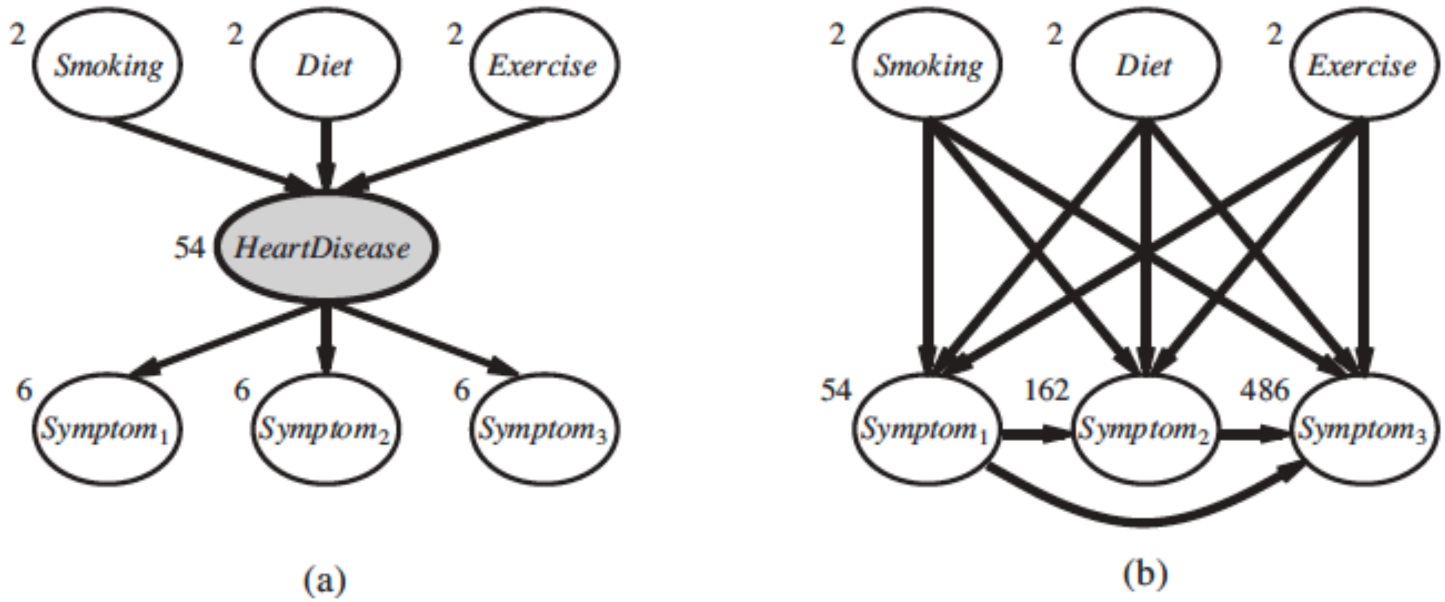


Figure 20.10 FILES: figures/313-heart-disease.eps (Tue Nov 3 16:22:09 2009). (a) A simple diagnostic network for heart disease, which is assumed to be a hidden variable. Each variable has three possible values and is labeled with the number of independent parameters in its conditional distribution; the total number is 78. (b) The equivalent network with *HeartDisease* removed. Note that the symptom variables are no longer conditionally independent given their parents. This network requires 708 parameters.

Expectation Maximization (EM) Algorithm

Clustering with mixture of Gaussians

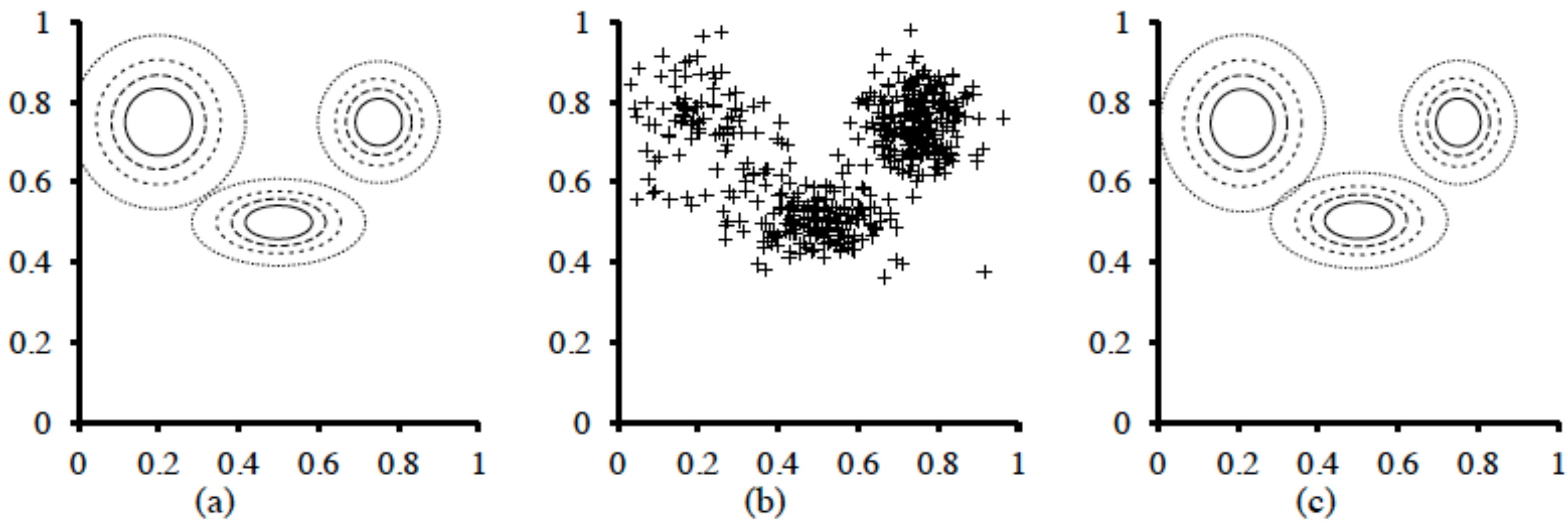


Figure 20.11 **FILES: .** (a) A Gaussian mixture model with three components; the weights (left-to-right) are 0.2, 0.3, and 0.5. (b) 500 data points sampled from the model in (a). (c) The model reconstructed by EM from the data in (b).

Summary

- Full Bayesian Learning
- MAP approximation
- ML approximation
- ML parameter learning in Bayes nets
  - Naïve Bayes Model
  - Linear Gaussian Model
- Bayesian parameter learning
  - Beta family of distributions
  - Conjugate families
- Latent variables
- Very briefly: Expectation Maximization (EM) algorithm

Next Class

- Reinforcement Learning
- Secs. 21.1 – 21.3