CS 559: Machine Learning Fundamentals and Applications 4th Set of Notes

Instructor: Philippos Mordohai
Webpage: www.cs.stevens.edu/~mordohai
E-mail: Philippos.Mordohai@stevens.edu
Office: Lieb 215

Overview

- Parameter Estimation
  - Frequentist or Maximum Likelihood approach (cont.)
  - Bayesian approach (Barber Ch. 8 and DHS Ch. 3)
- Cross-validation
- Overfitting
- Naïve Bayes Classifier
- Non-parametric Techniques

MLE Classifier Example

Data

- Pima Indians Diabetes Database
  - http://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes
  - Number of Instances: 768
  - Number of Attributes: 8 plus class
  - Class Distribution: (class value 1 is interpreted as "tested positive for diabetes")
  - Class Value Number of instances

Class Value	Number of instances
0	500
1	268

Data

Attributes: (all numeric-valued)

1. Number of times pregnant
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml) 6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)
9. Class variable (0 or 1)

Simple MLE Classifier

```
data = dlmread('pima-indians-diabetes.data');

data = reshape(data,[],9);

% use randperm to re-order data.
% ignore if not using Matlab
rp = randperm(length(data));
data=data(rp,:);

train_data = data(1:length(data)/2,:);
test_data = data(length(data)/2+1:end,:);
```

```
% pick a feature
active_feat = 3;

% training
mean1 =
  mean(train_data(train_data(:,9)==0,active_feat))
mean2 =
  mean(train_data(train_data(:,9)==1,active_feat))

var1 = var(train_data(train_data(:,9)==0,active_feat))
var2 = var(train_data(train_data(:,9)==1,active_feat))

prior1tmp = length(train_data(train_data(:,9)==0));
prior2tmp = length(train_data(train_data(:,9)==1));

prior1 = prior1tmp/(prior1tmp+prior2tmp)
prior2 = prior2tmp/(prior1tmp+prior2tmp)
```

```
% testing
correct=0;
wrong=0;

for i=1:length(test_data)
    lklhood1 = exp(-(test_data(i,active_feat)-mean1)^2/(2*var1))
   /sqrt(var1);
    lklhood2 = exp(-(test_data(i,active_feat)-mean2)^2/(2*var2));
   /sqrt(var2);

    post1 = lklhood1*prior1;
    post2 = lklhood2*prior2;

    if(post1 > post2 && test_data(i,9) == 0)
        correct = correct+1;
    elseif(post1 < post2 && test_data(i,9) == 1)
        correct = correct+1;
    else
        wrong = wrong+1;
    end
end
```

Training/Test Split

- Randomly split dataset into two parts:
  - Training data
  - Test data
- Use training data to optimize parameters
- Evaluate error using test data

Training/Test Split

- How many points in each set?
- Very hard question
  - Too few points in training set, learned classifier is bad
  - Too few points in test set, classifier evaluation is insufficient
- Cross-validation
- Leave-one-out cross-validation
- Bootstrapping

Cross-Validation

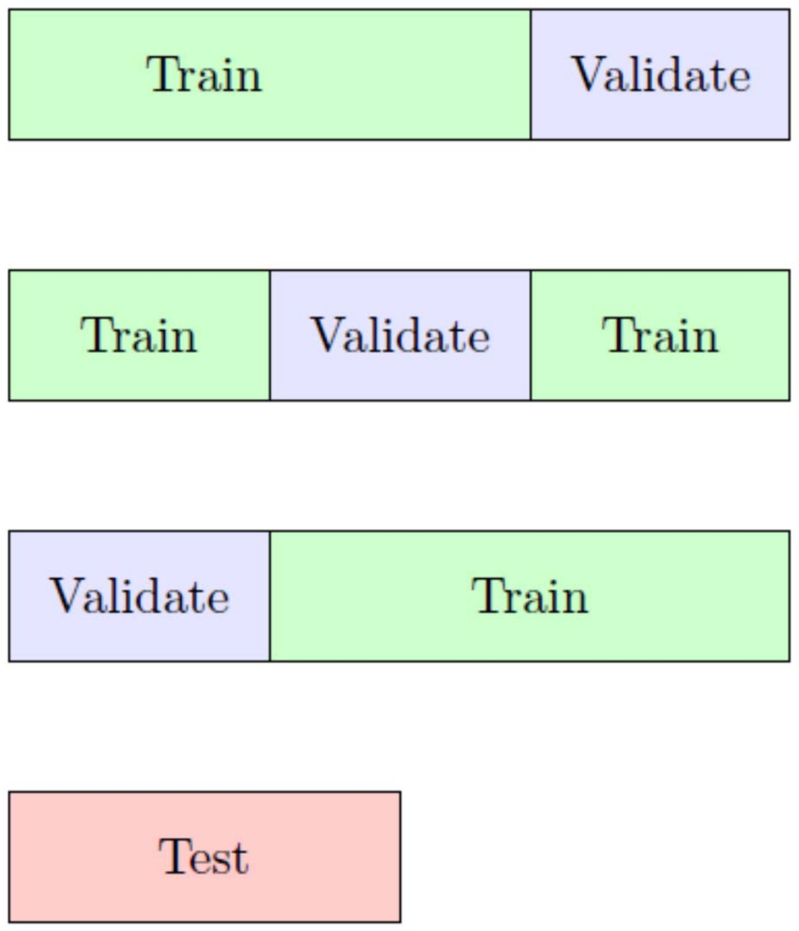
- In practice
- Available data => training and validation
- Train on the training data
- Test on the validation data
- k-fold cross validation:
  - Data randomly separated into k groups
  - Each time k−1 groups used for training and one as testing

Cross Validation and Test Accuracy

- If we select parameters so that CV is highest:
  - Does CV represent future test accuracy?
  - Slightly different
- If we have enough parameters, we can achieve 100% CV as well
  - e.g. more parameters than # of training data
- But test accuracy may be different
- So split available data with class labels, into:
  - training
  - validation
  - testing

Cross Validation and Test Accuracy

- Using CV on training + validation
- Classify test data with the best parameters from CV



Overfitting

- Prediction error: probability of test pattern not in class with max posterior (true)
- Training error: probability of test pattern not in class with max posterior (estimated)
- Classifier optimized w.r.t. training error
  - Training error: optimistically biased estimate of prediction error

Overfitting

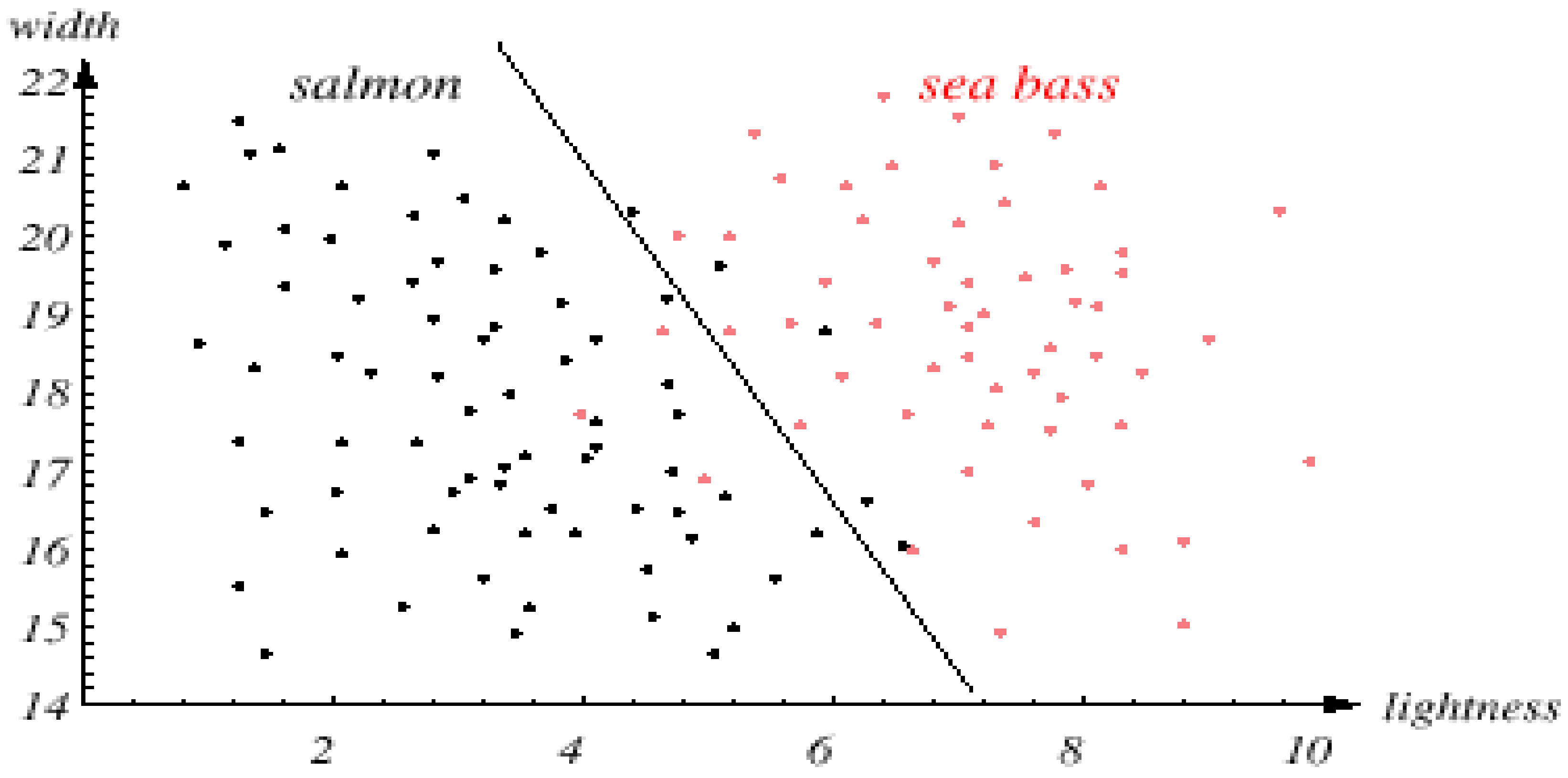
Overfitting: a learning algorithm overfits the training data if it outputs a solution **w** when another solution **w'** exists such that:

$$\text{error}_{\text{train}}(w) < \text{error}_{\text{train}}(w')$$

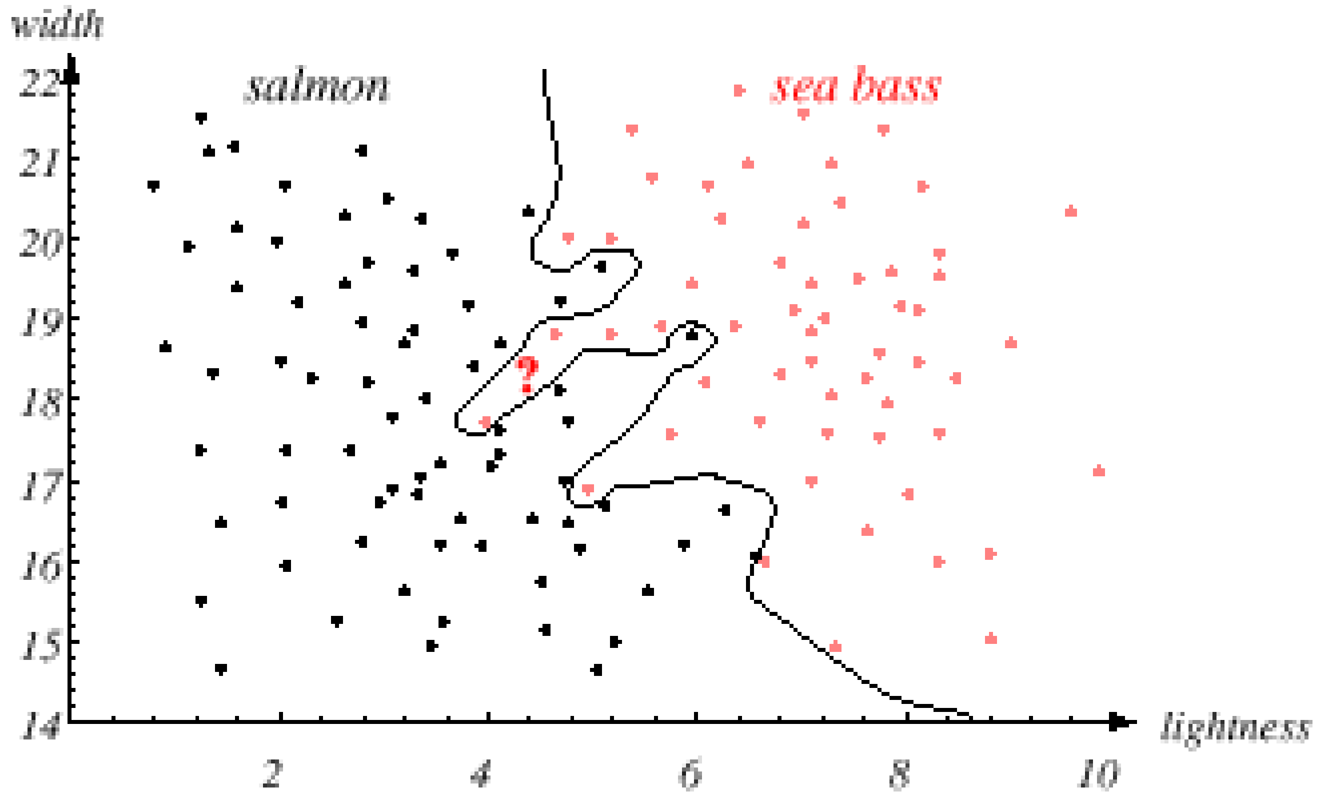
AND

$$\text{error}_{\text{true}}(w') < \text{error}_{\text{true}}(w)$$

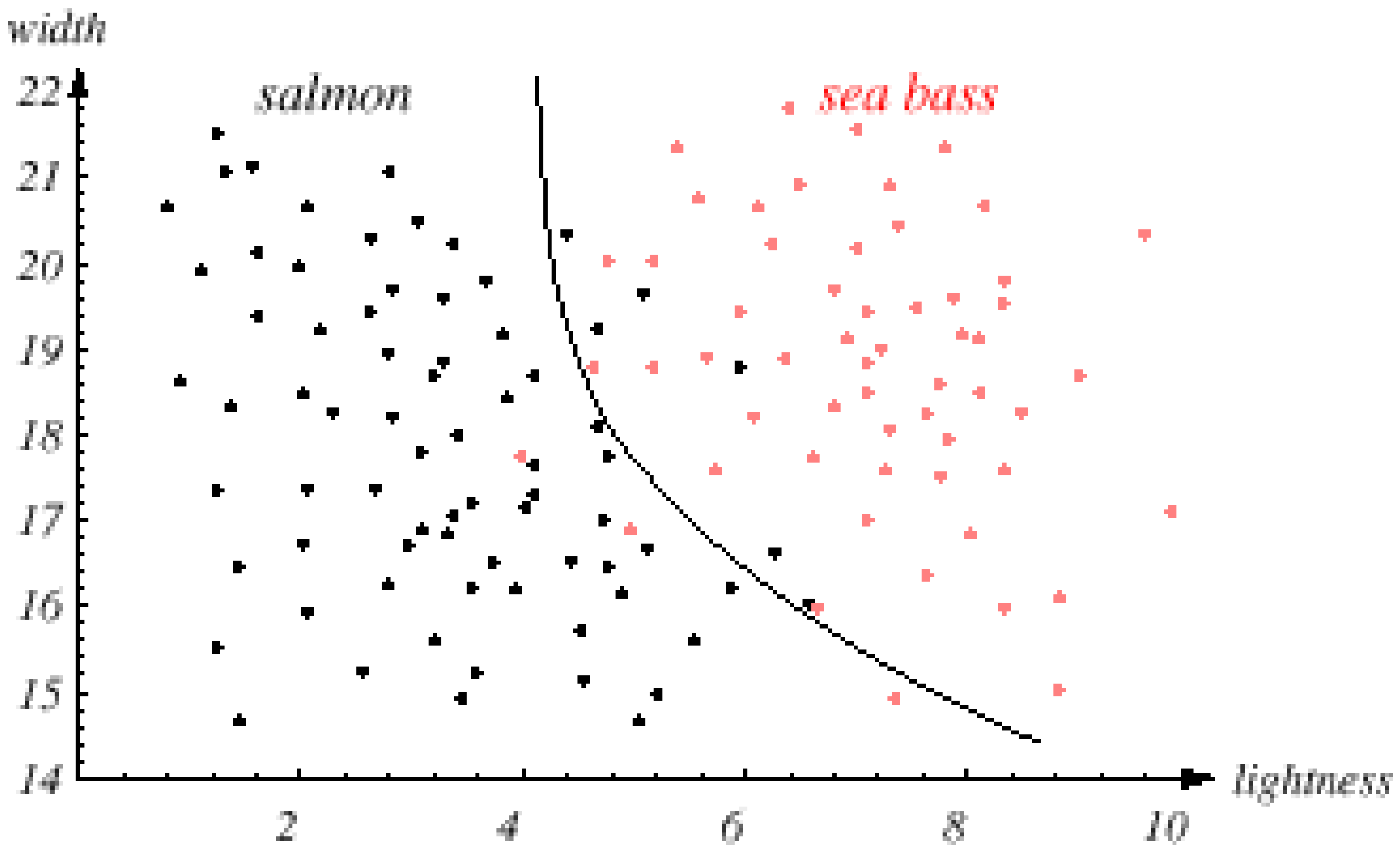
Fish Classifier from DHS Ch. 1



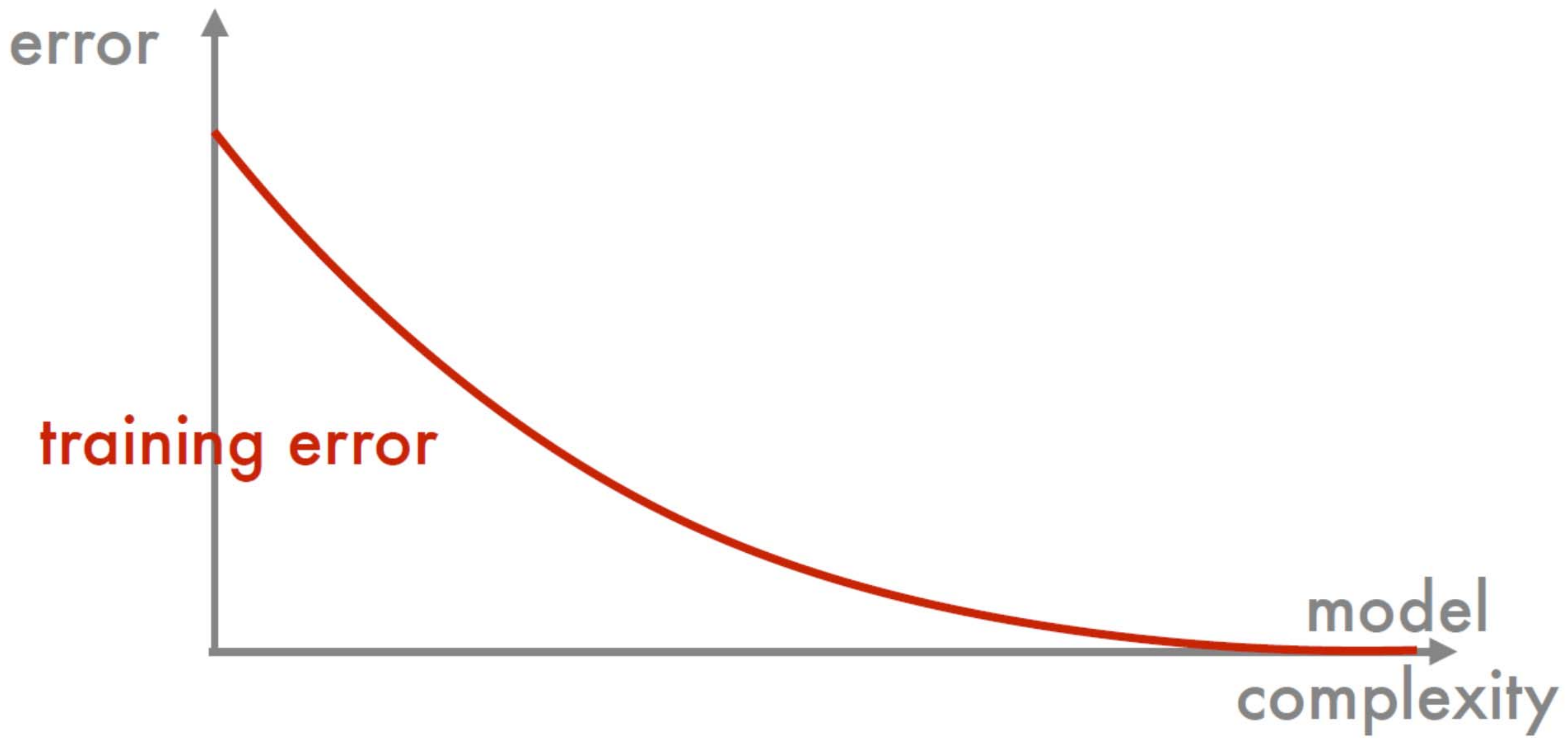
Minimum Training Error



Final Decision Boundary

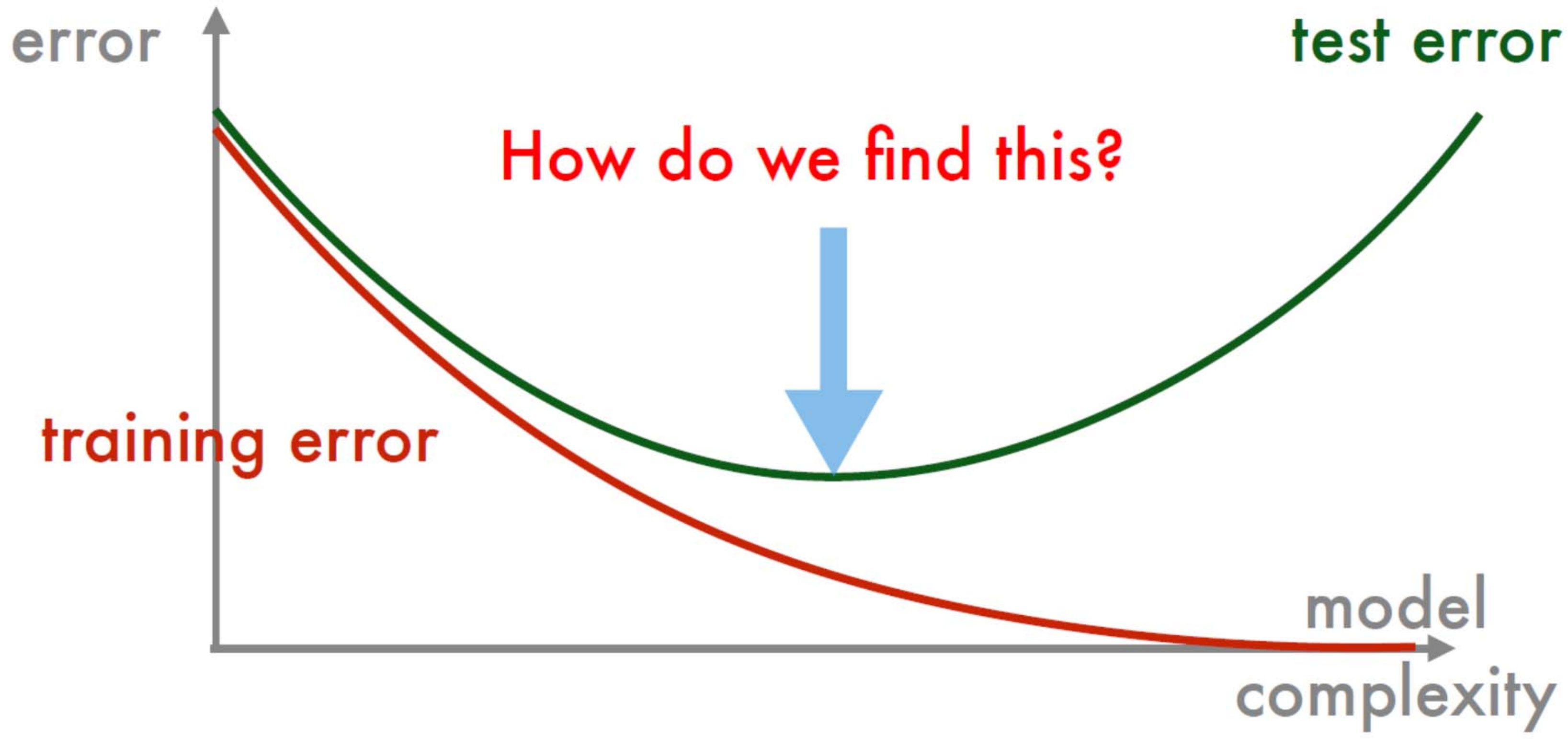


Typical Behavior



Slide credit: A. Smola

Typical Behavior



Slide credit: A. Smola

Bayesian Parameter Estimation

- Gaussian Case
- General Estimation

Bayesian Estimation

- In MLE θ was assumed fixed

- In BE θ is a random variable
- Suppose we have some idea of the range where the parameters θ should be
  - Shouldn’t we utilize this prior knowledge in hope that it will lead to better parameter estimation?

Bayesian Estimation

- Let θ be a random variable with prior distribution P(θ)
  - This is the key difference between ML and Bayesian parameter estimation
  - This allows us to use a prior to express the uncertainty present before seeing the data

  - Frequentist approach does not account for uncertainty in θ (see bootstrap for more on this, however)

Motivation

- As in MLE, suppose $p(x|\theta)$ is completely specified if θ is given
- But now θ is a random variable with prior $p(\theta)$
  - Unlike MLE case, $p(x|\theta)$ is a conditional density
- After we observe the data D, using Bayes rule we can compute the posterior $p(\theta|D)$

Motivation

- Recall that for the MAP classifier we find the class ω_i that maximizes the posterior $p(\omega|D)$
- By analogy, a reasonable estimate of θ is the one that maximizes the posterior $p(\theta|D)$
- But θ is not our final goal, our final goal is the unknown $p(x)$
- Therefore a better thing to do is to maximize $p(x|D)$, this is as close as we can come to the unknown $p(x)$!

Parameter Distribution

- Assumptions:
  - p(x) is unknown, but has known parametric form
  - Parameter vector θ is unknown
  - p(x| θ) is completely known
  - Prior density p(θ) is known
- Observation of samples provides posterior density p(θ|D)
  - Hopefully peaked around true value of θ
- Treat each class separately and drop subscripts

- Converted problem of learning probability density function to learning parameter vector
- Goal: compute p(x|D) as best possible estimate of p(x)

$$\mathrm{p}(\mathrm{x} \mid \mathrm{D}) = \int \mathrm{p}(\mathrm{x}, \theta \mid D)\, d\theta$$

$$\mathrm{p}(\mathrm{x} \mid \mathrm{D}) = \int \mathrm{p}(\mathrm{x} \mid \theta, D)\, p(\theta \mid D)\, d\theta = \int \mathrm{p}(\mathrm{x} \mid \theta)\, p(\theta \mid D)\, d\theta$$

p(x) is completely known given θ, independent of samples in D

$$\mathrm{p}(\mathrm{x} \mid \mathrm{D}) = \int \mathrm{p}(\mathrm{x} \mid \theta, D)\, p(\theta \mid D)\, d\theta = \int \mathrm{p}(\mathrm{x} \mid \theta)\, p(\theta \mid D)\, d\theta$$

- Links class-conditional density p(x|D) to posterior density p(θ|D)

Bayesian Parameter Estimation: Gaussian Case

Goal: Estimate θ using the a-posteriori density $P(\theta \mid D)$

– The univariate case: $p(\mu \mid D)$
  μ is the only unknown parameter

$$p(x \mid \mu) \sim N(\mu, \sigma^2)$$

$$p(\mu) \sim N(\mu_0, \sigma_0^2)$$

μ_0 and σ_0 are known
μ_0 is best guess for μ, σ_0 is uncertainty of guess

$$p(\mu \mid \mathrm{D}) = \frac{p(\mathrm{D} \mid \mu)\, p(\mu)}{\int p(\mathrm{D} \mid \mu)\, p(\mu)\, d\mu} \qquad (1)$$

$$= \alpha \prod_{k=1}^{k=n} p(x_k \mid \mu)\, p(\mu)$$

- α depends on D, not μ
- (1) shows how training samples affect our idea about the true value of μ

$$p(\mu \mid \mathrm{D}) = \frac{p(\mathrm{D} \mid \mu)\, p(\mu)}{\int p(\mathrm{D} \mid \mu)\, p(\mu)\, d\mu} \qquad (1)$$

$$= \alpha \prod_{k=1}^{k=n} p(x_k \mid \mu)\, p(\mu)$$

Reproducing density (remains Gaussian)

$$p(\mu \mid \mathrm{D}) \sim N(\mu_n, \sigma_n^2) \qquad (2)$$

(1) and (2) yield:

$$\mu_n = \left(\frac{n\sigma_0^2}{n\sigma_0^2 + \sigma^2} \right) \hat{\mu}_n + \frac{\sigma^2}{n\sigma_0^2 + \sigma^2}\, \mu_0$$

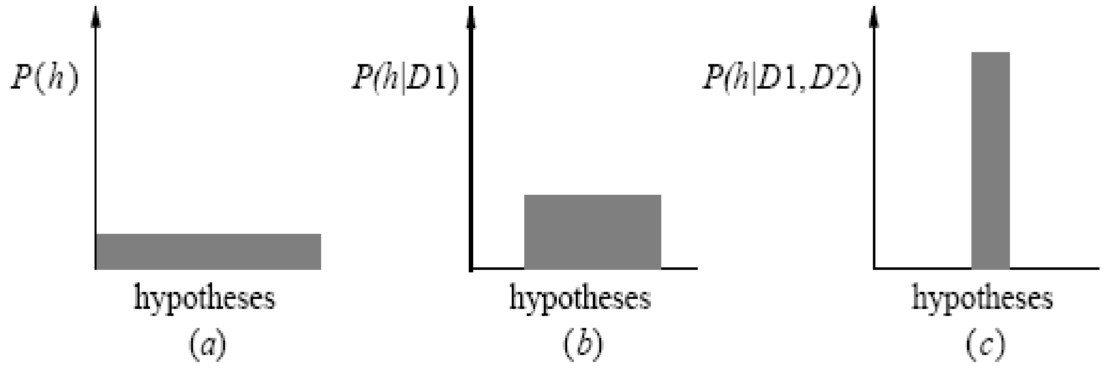
$$and \;\; \sigma_n^2 = \frac{\sigma_0^2 \sigma^2}{n\sigma_0^2 + \sigma^2}$$

Empirical (sample) mean

$$\mu_n = \left(\frac{n\sigma_0^2}{n\sigma_0^2 + \sigma^2} \right) \hat{\mu}_n + \frac{\sigma^2}{n\sigma_0^2 + \sigma^2} \mu_0$$

$$and \;\; \sigma_n^2 = \frac{\sigma_0^2 \sigma^2}{n\sigma_0^2 + \sigma^2}$$

- μ is linear combination of empirical and prior information
- Each additional observation decreases uncertainty about μ



– The univariate case p(x | D)

- p(μ | D) computed
- p(x | D) remains to be computed*

$$p(x \mid \mathrm{D}) = \int p(x \mid \mu)\, p(\mu \mid \mathrm{D})\, d\mu \ \text{ is Gaussian}$$

It provides: $p(x \mid \mathrm{D}) \sim N(\mu_n, \sigma^2 + \sigma_n^2)$

* Desired class-conditional density p(x | D_j, ω_j)

Using Bayes formula, we obtain the Bayesian classification rule:

$$\underset{\omega_j}{Max}\left[p(\omega_j \mid x, \mathrm{D})\right] \equiv \underset{\omega_j}{Max}\left[p(x \mid \omega_j, \mathrm{D}_j)\, p(\omega_j)\right]$$

$$p(x \mid \mathrm{D}) \sim N(\mu_n, \sigma^2 + \sigma_n^2)$$

- We have:
  - Replaced mean with conditional mean
  - Increased variance to account for additional uncertainty in x due to inexact knowledge of mean

Bayesian Parameter Estimation: General Theory

– $p(x \mid D)$ computation can be applied to any situation in which the unknown density can be parameterized: the basic assumptions are:

- The form of $p(x \mid \theta)$ is assumed known, but the value of θ is not known exactly
- Our knowledge about θ is assumed to be contained in a known prior density $p(\theta)$
- The rest of our knowledge θ is contained in a set D of n random variables $x_1, x_2, \ldots, x_n$ that follows $p(x)$

The basic problem is:

"Compute the posterior density p(θ | D)"

then "Derive p(x | D)"

Using Bayes formula, we have:

$$p(\theta \mid \mathrm{D}) = \frac{p(\mathrm{D} \mid \theta)\, p(\theta)}{\int p(\mathrm{D} \mid \theta)\, p(\theta) d\theta}$$

And by the independence assumption:

$$p(\mathrm{D} \mid \theta) = \prod_{k=1}^{k=n} p(x_k \mid \theta)$$

Recursive Bayes Learning

- Assume that training samples become available one by one

$$p(\mathrm{D}^{\mathrm{n}} \mid \theta) = p(x_n \mid \theta)\, p(\mathrm{D}^{\mathrm{n-1}} \mid \theta)$$

- Due to independence, result is independent of order:

$$p(\mathrm{D} \mid \theta) = \prod_{k=1}^{k=n} p(x_k \mid \theta)$$

Estimation of p(x|D)

- The basic problem is: Compute p(x | D)

$$p(\mathbf{x} \mid D) = \int \underbrace{p(\mathbf{x} \mid \theta)}_{\text{known}} \underbrace{p(\theta \mid D)}_{\text{unknown}} d\theta$$

- Compute the posterior density p(θ | D)

$$p(\theta \mid \mathrm{D}) = \frac{p(\mathrm{D} \mid \theta)\, p(\theta)}{\int p(\mathrm{D} \mid \theta)\, p(\theta)\, d\theta}$$

- Then derive p(x | D)
- Repeat for all classes to obtain $p(x \mid \omega_i)$
- Combine with $p(\omega_i)$ to get posteriors

Conjugate Priors

- Prior is conjugate to likelihood if it leads to itself as posterior
- Closed form representation of posterior
- If the prior on θ, with hyperparameters α, has some $p(\theta|\alpha)$, the posterior given data D is of the same form but with updated hyperparameters

$$p(\theta|D,\alpha) = p(\theta|\alpha')$$

Bayesian Inference of Mean and Variance

- Uni-variate Gaussian

$$p(\mathcal{X}|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{N/2}}\exp\left(-\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x^n-\mu)^2\right)$$

- Posterior of parameters

$$p(\mu,\sigma^2|\mathcal{X}) \propto p(\mathcal{X}|\mu,\sigma^2)p(\mu,\sigma^2) = p(\mathcal{X}|\mu,\sigma^2)p(\mu|\sigma^2)p(\sigma^2)$$

- Prior of mean (Gaussian)

$$p(\mu|\mu_0,\sigma_0^2) = \frac{1}{\sqrt{2\pi\sigma_0^2}}\exp\left(-\frac{1}{2\sigma_0^2}(\mu_0-\mu)^2\right)$$

Bayesian Inference of Mean and Variance

- Posterior

$$p(\mu, \sigma^2 | \mathcal{X}) \propto \frac{1}{\sigma_0} \frac{1}{\sigma^N} \exp\left(-\frac{1}{2\sigma_0^2} (\mu_0 - \mu)^2 - \frac{1}{2\sigma^2} \sum_n (x^n - \mu)^2 \right) p(\sigma^2)$$

$$p(\mu, \sigma^2 | \mathcal{X}) \propto \underbrace{\sqrt{a} \exp\left(-\frac{1}{2} a \left(\mu - \frac{b}{a} \right)^2 \right)}_{p(\mu|\mathcal{X},\sigma^2)} \underbrace{\frac{1}{\sqrt{a}} \exp\left(-\frac{1}{2} \left(c - \frac{b^2}{a} \right) \right) \frac{1}{\sigma_0} \frac{1}{\sigma^N} p(\sigma^2)}_{p(\sigma^2|\mathcal{X})}$$

after some manipulation …

Bayesian Inference of Mean and Variance

- Use inverse Gamma distribution for p(σ^2)

$$p(\mu,\sigma^2) = \mathcal{N}\left(\mu \middle| \mu_0, \gamma\sigma^2\right) InvGam\left(\sigma^2 | \alpha, \beta\right)$$

- Then, posterior is also Gauss-Inverse-Gamma

$$p(\mu,\sigma^2|\mathcal{X}) = \mathcal{N}\left(\mu \middle| \frac{\tilde{b}}{\tilde{a}}, \frac{\sigma^2}{\tilde{a}}\right) InvGam\left(\sigma^2 | \alpha + \frac{N}{2}, \beta + \frac{1}{2}\left(\tilde{c} - \frac{\tilde{b}^2}{\tilde{a}}\right)\right)$$

ML vs. Bayesian Parameter Estimation: Summary

BE vs. MLE

- BE: p(x|D) can be thought of as the weighted average of the proposed model for all possible values of θ

$$p(x \mid D) = \int \underbrace{p(x \mid \theta)}_{\substack{\textit{proposed model} \\ \textit{with certain } \theta}} \overbrace{p(\theta \mid D)}^{\substack{\textit{support } \theta \textit{ receives} \\ \textit{from the data}}} d\theta$$

- Contrast this with the MLE solution which always gives us a single model:

$$p(x|\hat{\theta})$$

- When we have many possible solutions, taking their sum averaged by their probabilities seems better than pick just one solution

Bayesian Estimation vs. MLE

- In practice, it may be hard to do integration analytically and we may have to resort to numerical methods
- The MLE solution requires differentiation, instead of integration, to get

$$p(x|\hat{\theta})$$

  - Differentiation is easy and can always be done analytically

When do Maximum-Likelihood and Bayes Methods Differ?

- Equivalent asymptotically (for infinite training data)
  - For reasonable prior distributions
  - When prior p(θ) is uninformative and p(θ|D) is peaked
- MLE computationally cheaper, simpler solutions
- BE uses more information (more general model)

Naïve Bayes Classifier (not BE)

- Simple classifier that applies Bayes' rule with strong (naive) independence assumptions
- A.k.a. the "independent feature model”

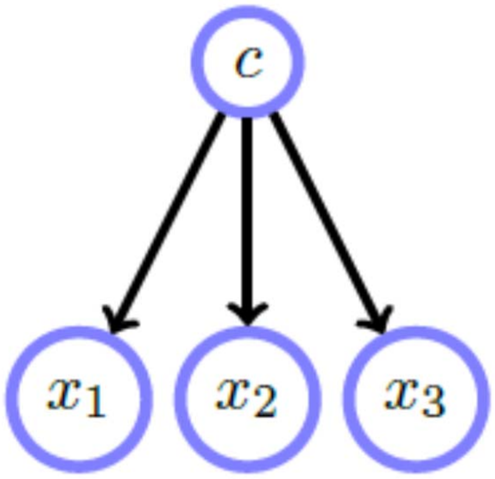
- $p(\omega_i|x_1, x_2,\ldots)= \alpha\, p(x_1|\omega_i)\, p(x_2|\omega_i)\ldots\, p(\omega_i)$

- Often performs reasonably well despite simplicity

Naïve Bayes Classifier

- NB is known to produce posteriors closer to extremes (0 or 1) than true posteriors
  - Why?

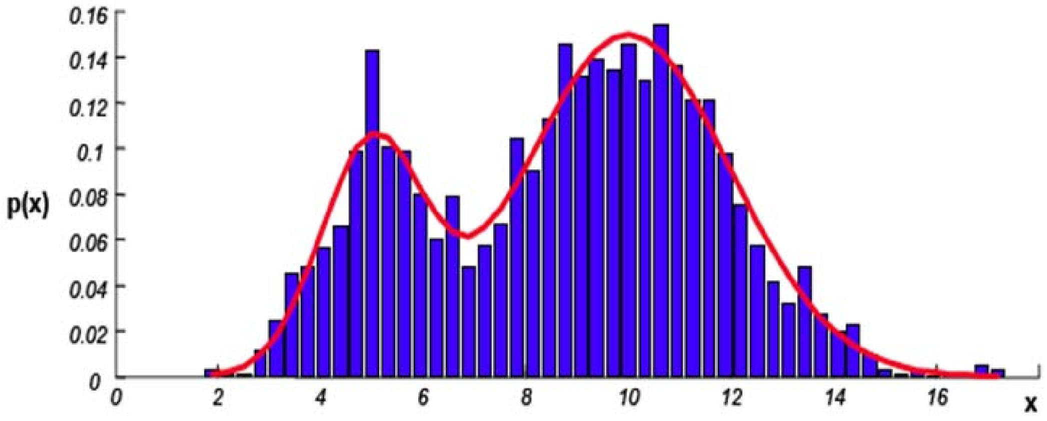
- NB performs well when only small amounts of training data are available
  - Why?



Non-parametric Classification

The Histogram

- The simplest form of non-parametric density estimation is the histogram
  - Divide sample space in number of bins
  - Approximate the density at the center of each bin by the fraction of points that fall into the bin
  - Two parameters: bin width and starting position of first bin (or other equivalent pairs)
- Drawbacks:
  - Depends on position of bin centers
    - Often compute two histograms, offset by ½ bin width
  - Discontinuities as an artifact of bin boundaries
  - Curse of dimensionality



Introduction

- All parametric densities are unimodal (have a single local maximum), whereas many practical problems involve multi-modal densities

- Non-parametric procedures can be used with arbitrary distributions and without the assumption that the forms of the underlying densities are known

- There are two types of non-parametric methods:
  - Estimate $P(x \mid \omega_j)$
  - Bypass density function and go directly to posterior probability estimation

Density Estimation

– Probability that a vector x will fall in region R is:

$$P = \int_{\Re} p(x')dx' \qquad (1)$$

– P is a smoothed (or averaged) version of the density function p(x) if we have a sample of size **n**; therefore, the probability that k points fall in R is:

$$P_k = \binom{n}{k} P^k (1 - P)^{n-k} \qquad (2)$$

and the expected value for ***k*** is:

$$E(k) = nP \qquad (3)$$

ML Estimate

ML estimation of P = θ

$\underset{\theta}{Max}(P_k \mid \theta)$ is reached for $\hat{\theta} = \frac{k}{n} \cong P$

Therefore, the ratio **k/n** is a good estimate for the probability P and hence for the density function p(x) (for large n)

Assumptions

p(x) is continuous and the region *R* is so small that p does not vary significantly within it, we can write:

$$\int_{\Re} p(x')dx' \cong p(x)V \qquad (4)$$

where **x** is a point within *R* and V the volume enclosed by *R*.

Combining equation (1) , (3) and (4) yields: $p(x) \cong \frac{k/n}{V}$

- The volume V needs to approach 0, if we want to use this estimate

  - Practically, V cannot be allowed to become small since the number of samples is always limited

  - One will have to accept a certain amount of variance in the ratio k/n

  - Theoretically, if an unlimited number of samples is available, we can circumvent this difficulty

    To estimate the density of x, we form a sequence of regions

    $\mathcal{R}_1, \mathcal{R}_2, \ldots$ containing x: the first region contains one sample, the second two samples and so on.

    Let V_n be the volume of $\mathcal{R}_n$, k_n the number of samples falling in $\mathcal{R}_n$ and $p_n(x)$ be the nth estimate for $p(x)$:

$$p_n(x) = (k_n/n)/V_n \qquad (7)$$

Three necessary conditions should apply if we want *$p_n(x)$* to converge to *$p(x)$*:

$$1)\ \lim_{n\to\infty} V_n = 0$$

$$2)\ \lim_{n\to\infty} k_n = \infty$$

$$3)\ \lim_{n\to\infty} k_n / n = 0$$

There are two different ways of obtaining sequences of regions that satisfy these conditions:

(a) Shrink an initial region where $V_n = 1/\sqrt{n}$ and show that

$$p_n(x) \xrightarrow[n\to\infty]{} p(x)$$

This is called "the Parzen-window estimation method"

(b) Specify k_n as some function of n, such as $k_n = \sqrt{n}$; the volume V_n is grown until it encloses k_n neighbors of x. This is called "the k_n-nearest neighbor estimation method"

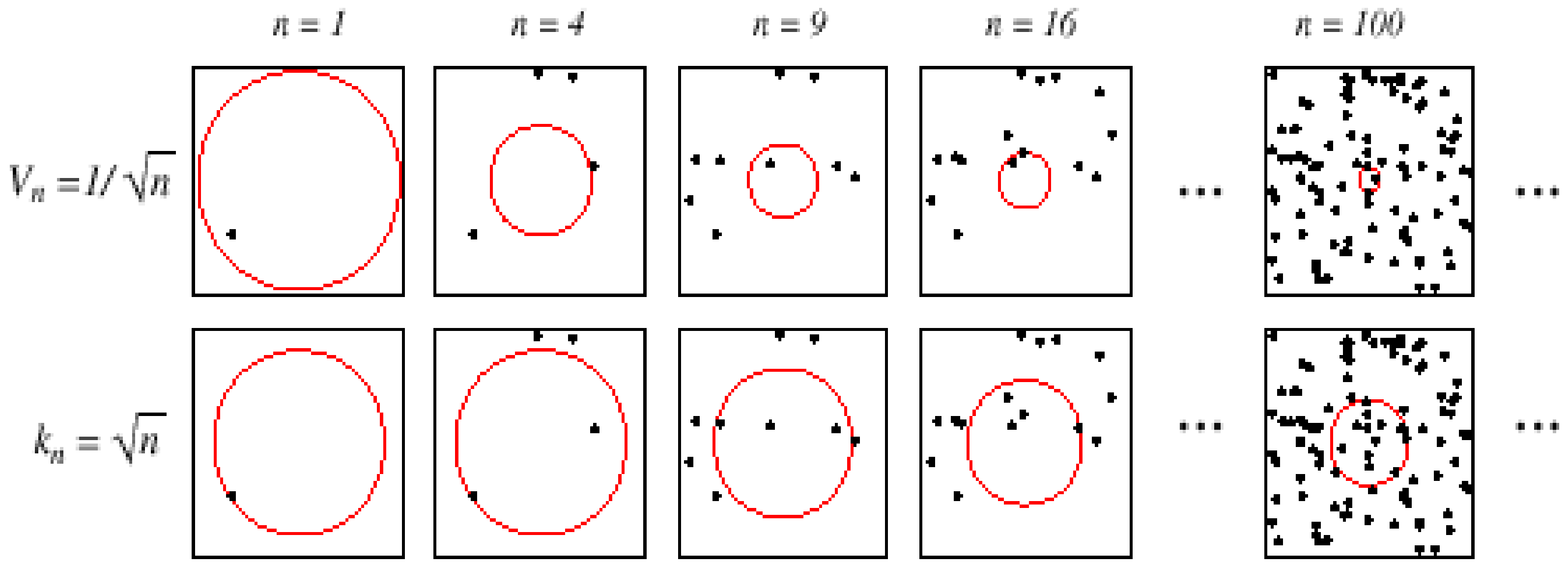


FIGURE 4.2. There are two leading methods for estimating the density at a point, here at the center of each square. The one shown in the top row is to start with a large volume centered on the test point and shrink it according to a function such as $V_n = 1/\sqrt{n}$. The other method, shown in the bottom row, is to decrease the volume in a data-dependent way, for instance letting the volume enclose some number $k_n = \sqrt{n}$ of sample points. The sequences in both cases represent random variables that generally converge and allow the true density at the test point to be calculated. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Parzen Windows

– The Parzen-window approach to estimate densities assumes that the region R_n is a d-dimensional hypercube

$$V_n = h_n^d \ (h_n : \text{length of the edge of } \Re_n)$$

Let $\varphi(u)$ be the following window function :

$$\varphi(u) = \begin{cases} 1 & |u_j| \le \frac{1}{2} \quad j = 1,\dots,d \\ 0 & \text{otherwise} \end{cases}$$

– $\varphi((x-x_i)/h_n)$ is equal to unity if x_i falls within the hypercube of volume V_n centered at x and equal to zero otherwise

– The number of samples in this hypercube is:

$$k_n = \sum_{i=1}^{i=n} \varphi\left(\frac{x - x_i}{h_n}\right)$$

By substituting k_n in equation (7), we obtain the following estimate:

$$\mathbf{p_n(x) = \frac{1}{n}\sum_{i=1}^{i=n} \frac{1}{V_n}\,\varphi\left(\frac{x - x_i}{h_n}\right)}$$

$P_n(x)$ estimates $p(x)$ as an average of functions of x and the samples $\{x_i\}$ (i = 1,… ,n). These functions φ can be general

Window Functions

- Conditions for estimating legitimate density function
  - Non-negative $\varphi(\mathrm{x}) \geq 0$
  - Integrate to 1

$$\int \varphi(x)dx = 1$$

- In other words, the window function should be a probability density function

Illustration

- The behavior of the Parzen-window method
  - Case where p(x) →N(0,1)
  - Let

$$\varphi(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}}$$

and $h_n = \frac{h_1}{\sqrt{n}}$ (h$_1$: known parameter)

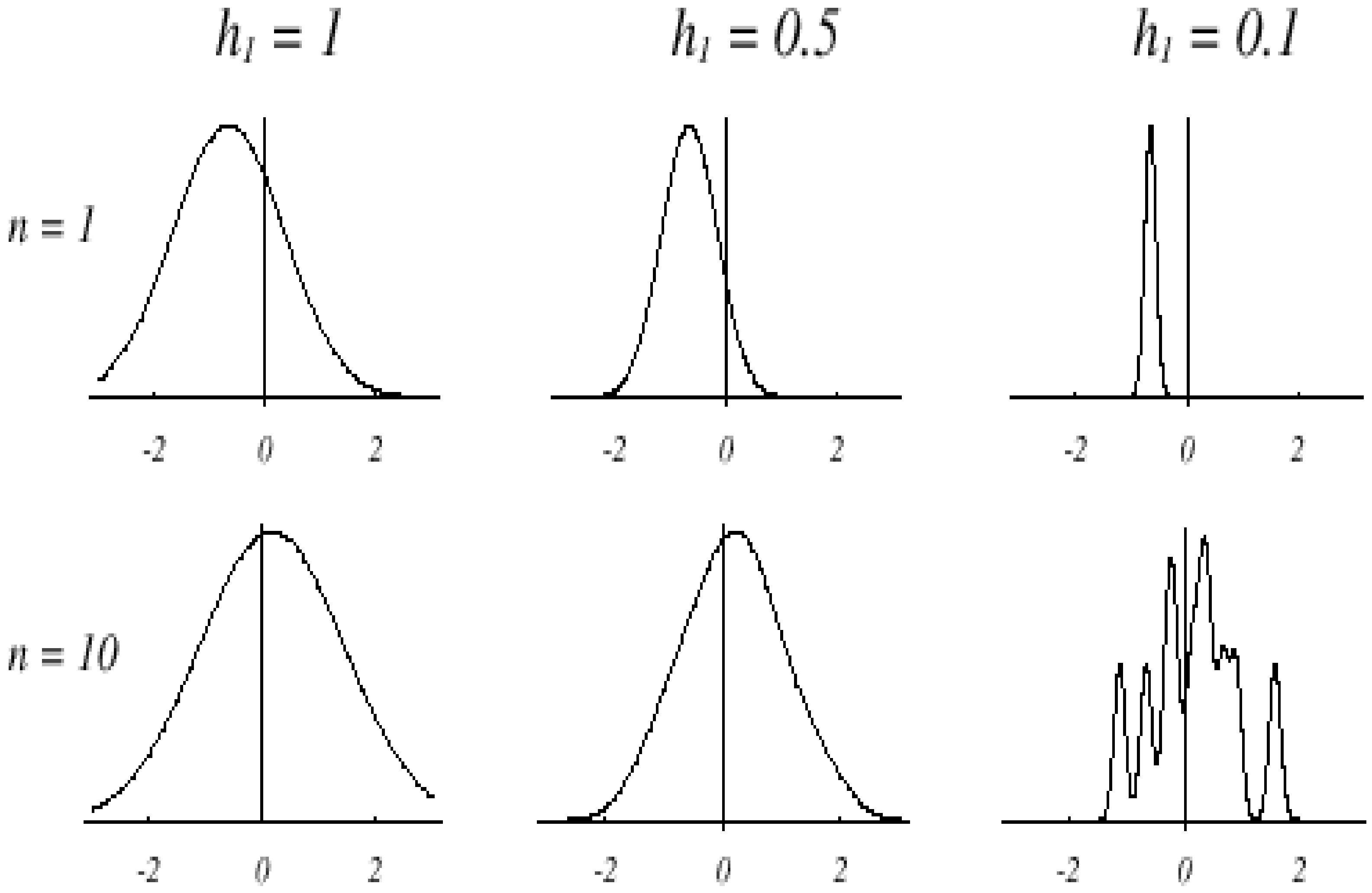
Thus: $$p_n(x) = \frac{1}{n}\sum_{i=1}^{i=n} \frac{1}{h_n}\varphi\left(\frac{x - x_i}{h_n}\right)$$

is an average of normal densities centered at the samples x_i

Numerical Results

For *n = 1* and *h_1=1*

$$\mathrm{p}_1(x) = \varphi(x - x_1) = \frac{1}{\sqrt{2\pi}} e^{-1/2}(x - x_1)^2 \rightarrow N(x_1, 1)$$



For n = 10 and h = 0.1, the contributions of the individual samples are clearly observable

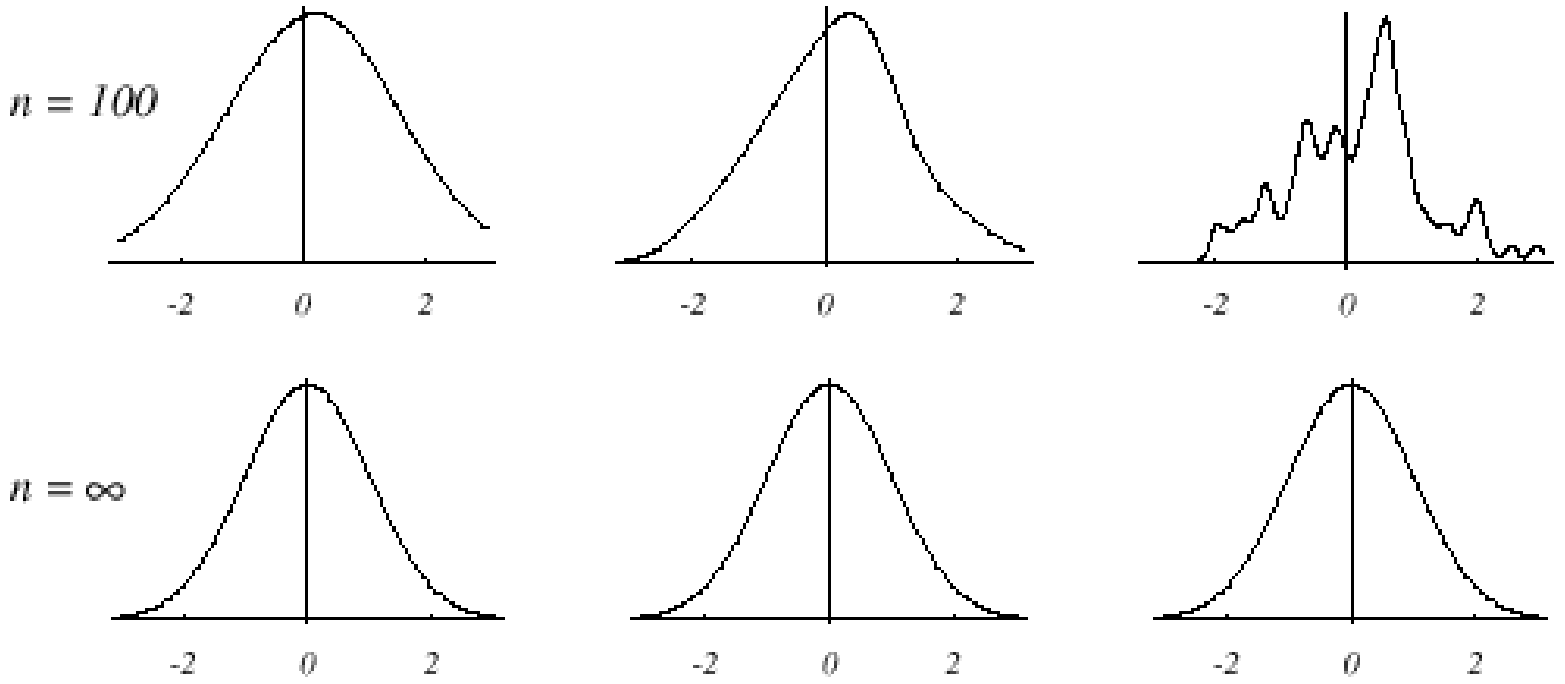
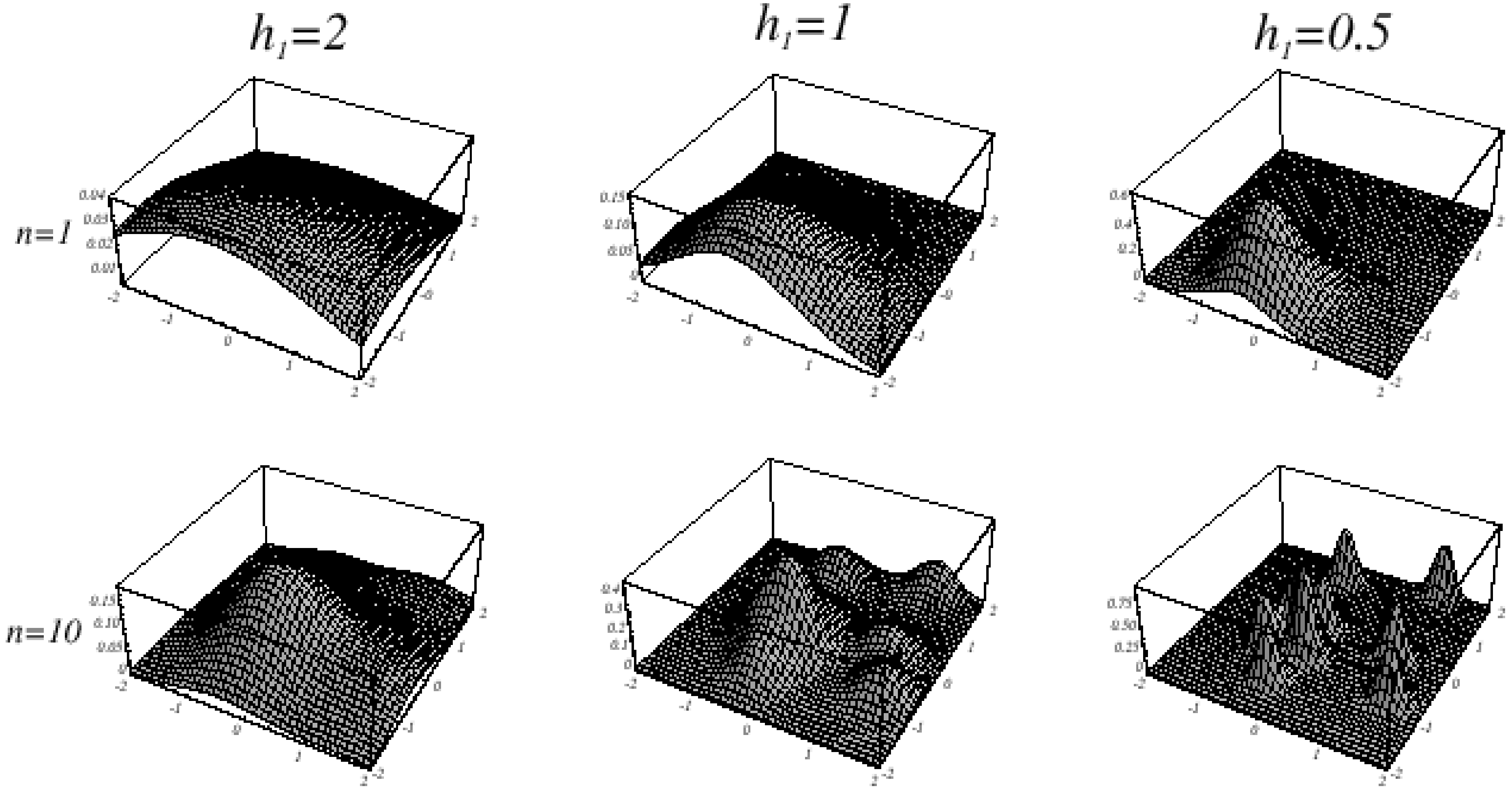


FIGURE 4.5. Parzen-window estimates of a univariate normal density using different window widths and numbers of samples. The vertical axes have been scaled to best show the structure in each graph. Note particularly that the $n = \infty$ estimates are the same (and match the true density function), regardless of window width. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Analogous results are also obtained in two dimensions as illustrated:



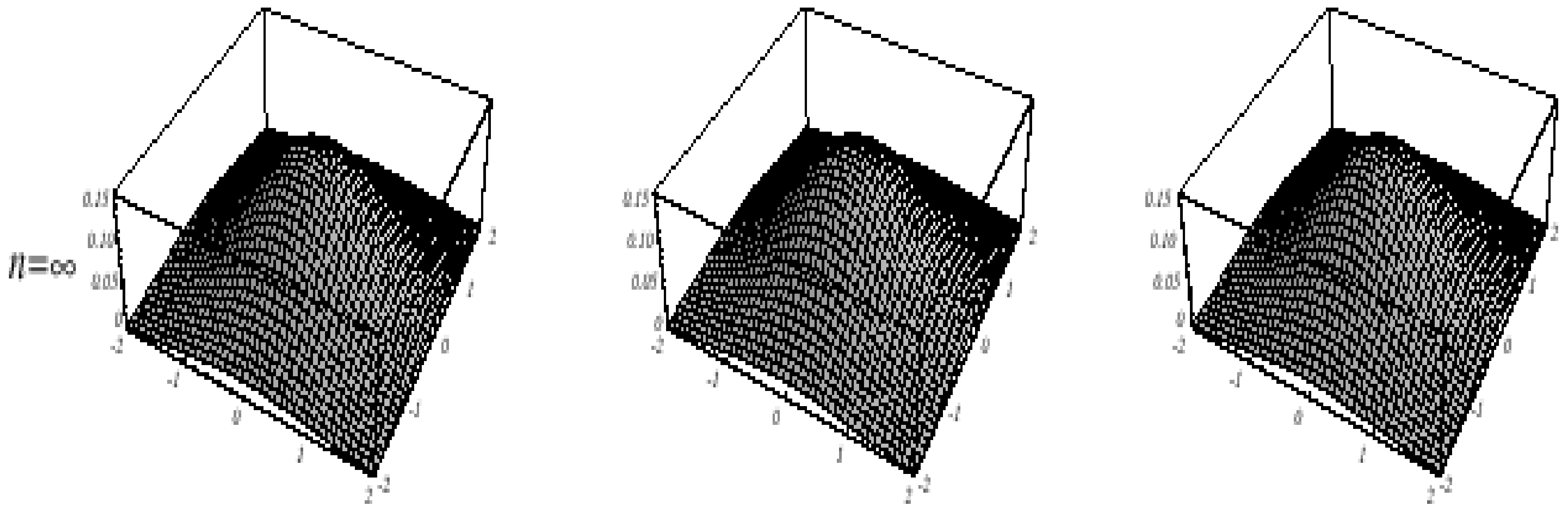
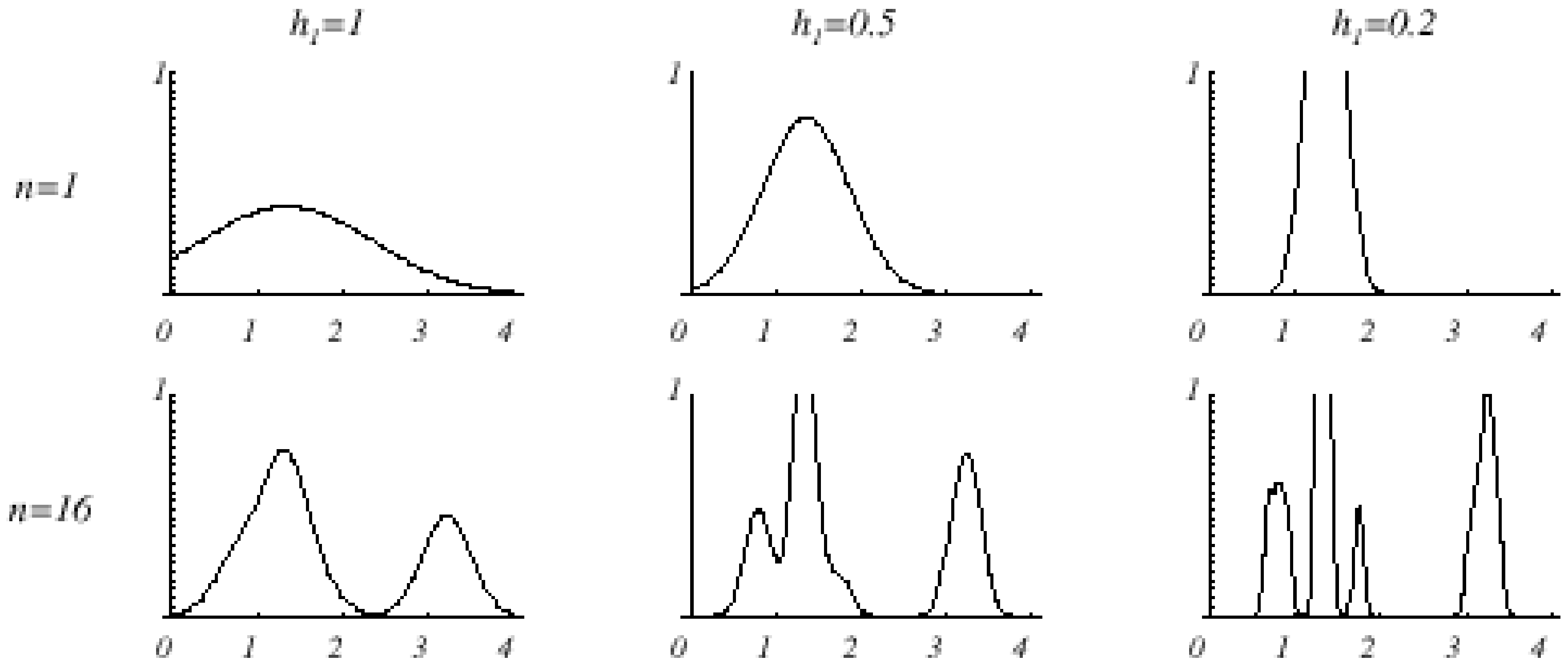


FIGURE 4.6. Parzen-window estimates of a bivariate normal density using different window widths and numbers of samples. The vertical axes have been scaled to best show the structure in each graph. Note particularly that the $n = \infty$ estimates are the same (and match the true distribution), regardless of window width. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

- Case where **$p(x) = \lambda_1 U(a,b) + \lambda_2 T(c,d)$**
  - unknown density, mixture of a uniform and a triangle density



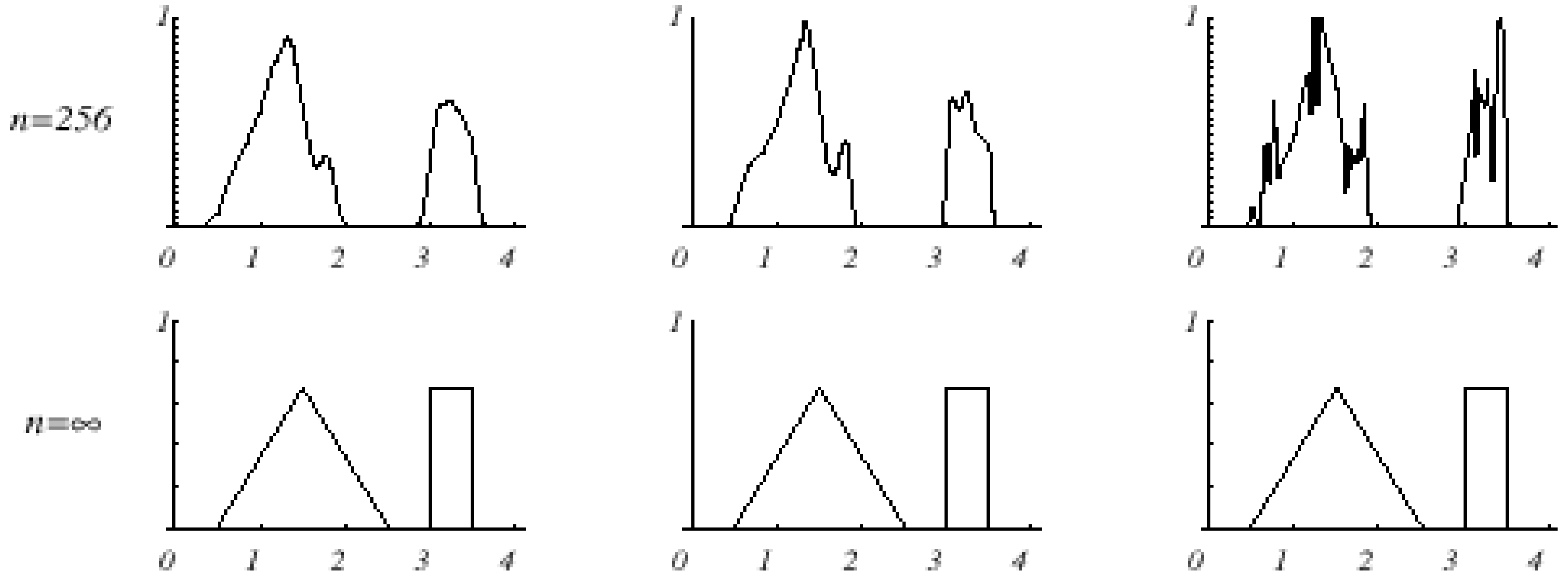


FIGURE 4.7. Parzen-window estimates of a bimodal distribution using different window widths and numbers of samples. Note particularly that the $n = \infty$ estimates are the same (and match the true distribution), regardless of window width. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Classification

- In classifiers based on Parzen-window estimation:
  - We estimate the densities for each category and classify a test point by the label corresponding to the maximum posterior
  - The decision region for a Parzen-window classifier depends upon the choice of window function as illustrated in the following figure

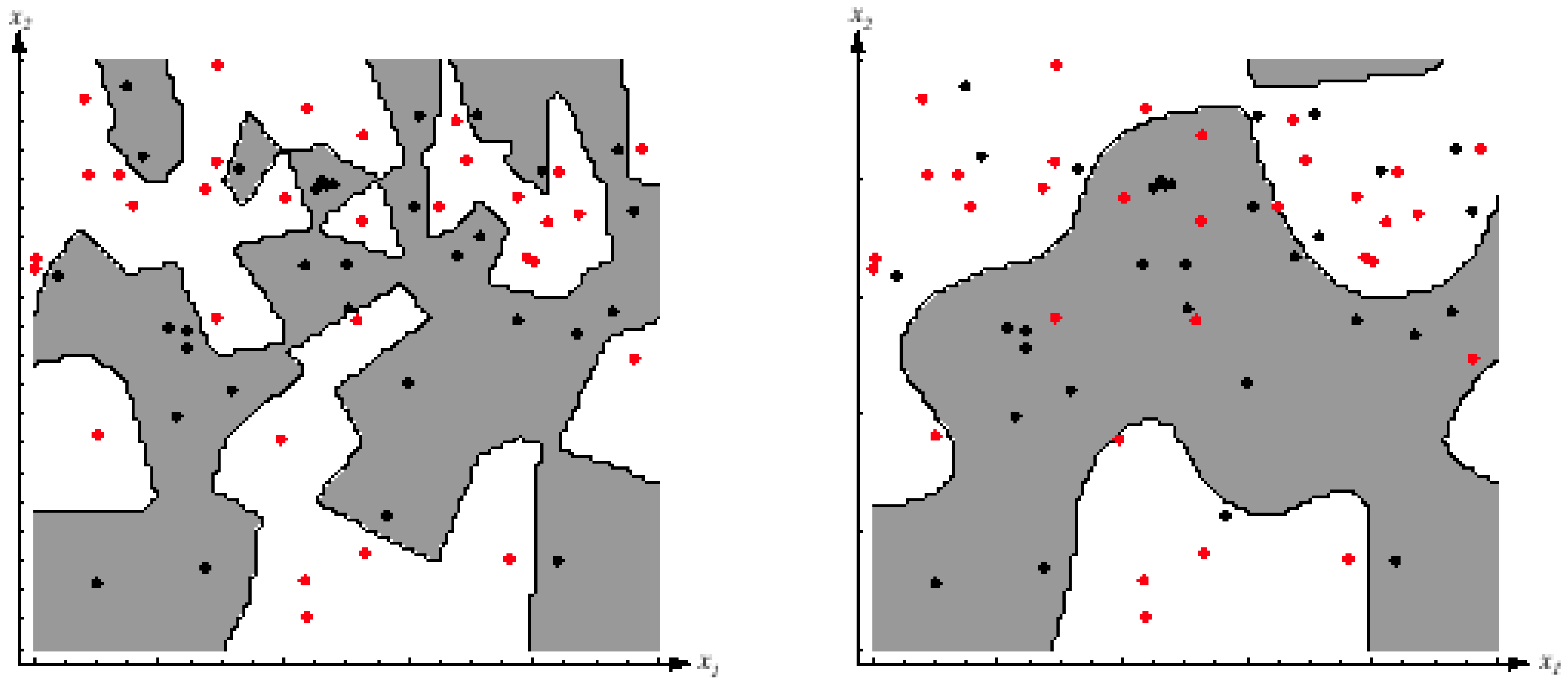


FIGURE 4.8. The decision boundaries in a two-dimensional Parzen-window dichotomizer depend on the window width *h*. At the left a small *h* leads to boundaries that are more complicated than for large *h* on same data set, shown at the right. Apparently, for these data a small *h* would be appropriate for the upper region, while a large *h* would be appropriate for the lower region; no single window width is ideal overall. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Remember discussion on overfitting

K - Nearest Neighbor Estimation

- Goal: a solution for the problem of the unknown “best” window function
  - Let the cell volume be a function of the training data
  - Center a cell about x and let it grow until it captures k_n samples ($k_n = f(n)$)
  - k_n are called the k_n nearest-neighbors of x

- Benefits
  - If density is high near x, the cell will be small which provides a good resolution
  - If density is low, the cell will grow large and stop when higher density regions are reached

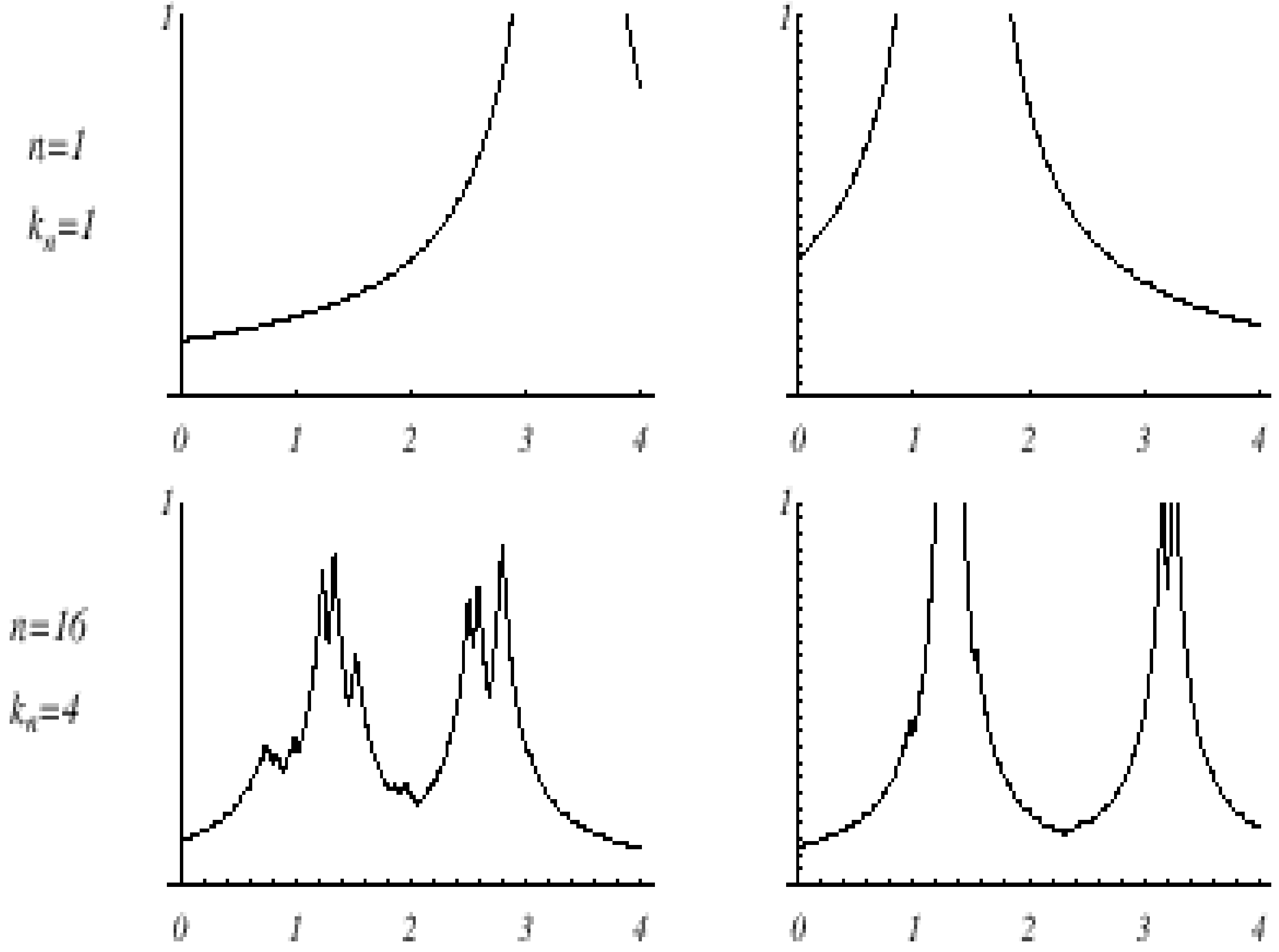
We can obtain a family of estimates by setting $k_n = k_1/\sqrt{n}$ and choosing different values for k_1

Illustration

For $k_n = \sqrt{n} = 1$; the estimate becomes:

$P_n(x) = k_n / nV_n = 1 / V_1 = 1 / 2|x-x_1|$

(goes to infinity at x_1)



$n=1$
$k_n=1$
$n=16$
$k_n=4$
1
0
1
2
3
4

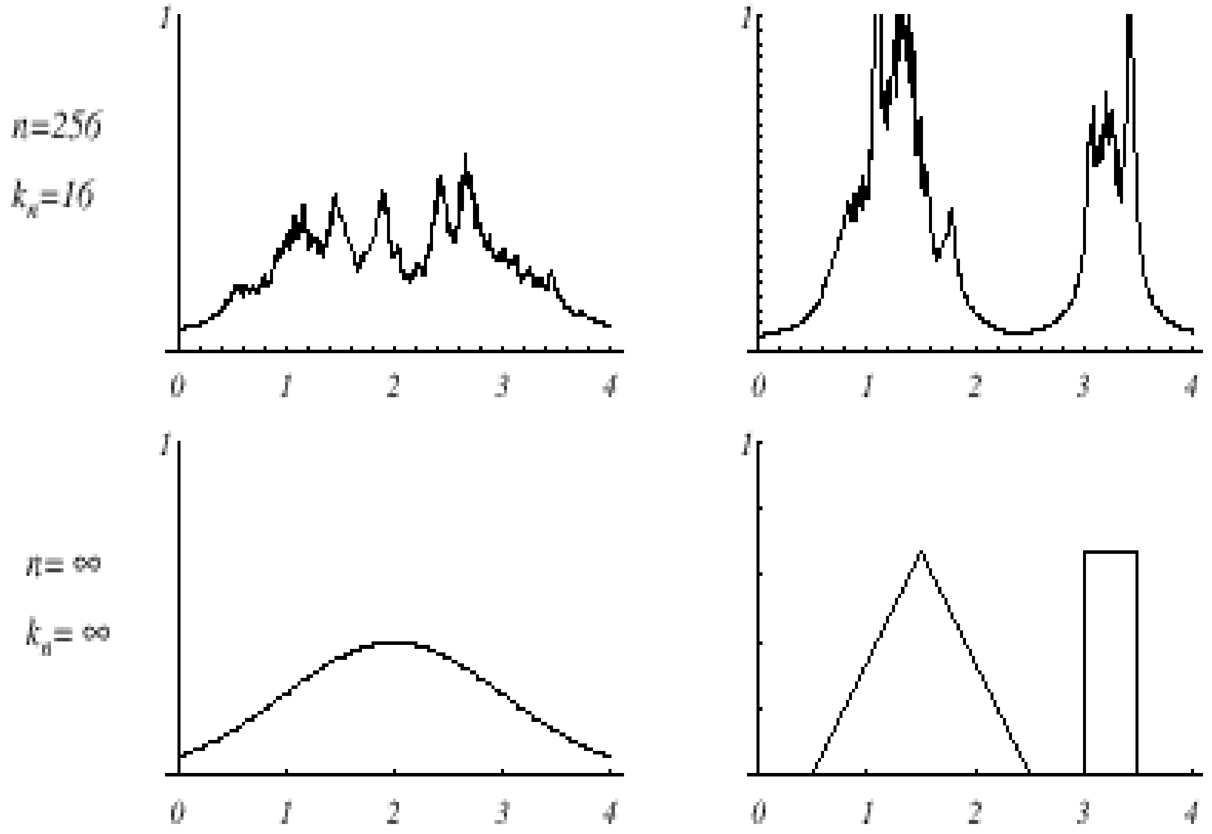


FIGURE 4.12. Several *k*-nearest-neighbor estimates of two unidimensional densities: a Gaussian and a bimodal distribution. Notice how the finite *n* estimates can be quite "spiky." From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Estimation of Posterior Probabilities

- Goal: estimate $P(\omega_i \mid x)$ from a set of n labeled samples
- Place a cell of volume V around x and capture k samples
- k_i samples amongst k turned out to be labeled ω_i then:

$$p_n(x, \omega_i) = k_i / nV$$

An estimate for $p_n(\omega_i \mid x)$ is:

$$p_n(\omega_i \mid x) = \frac{p_n(x, \omega_i)}{\sum_{j=1}^{j=c} p_n(x, \omega_j)} = \frac{k_i}{k}$$

– k_i/k is the fraction of the samples within the cell that are labeled ω_i

– For minimum error rate, the most frequently represented category within the cell is selected

=> This is equivalent to posterior estimation

– If k is large and the cell sufficiently small, the performance will approach the best possible

The Nearest-Neighbor Rule

- Let $D_n = \{x_1, x_2, \ldots, x_n\}$ be a set of n labeled prototypes
- Let $x' \in D_n$ be the closest prototype to a test point x then the nearest-neighbor rule for classifying x is to assign it the label associated with x'
- The nearest-neighbor rule leads to an error rate greater than the minimum possible: the Bayes rate
- If the number of prototypes is large (unlimited), the error rate of the nearest-neighbor classifier is never worse than twice the Bayes rate (it can be proven!)
- If $n \to \infty$, it is always possible to find x' sufficiently close so that:

$$P(\omega_i \mid x') \approx P(\omega_i \mid x)$$

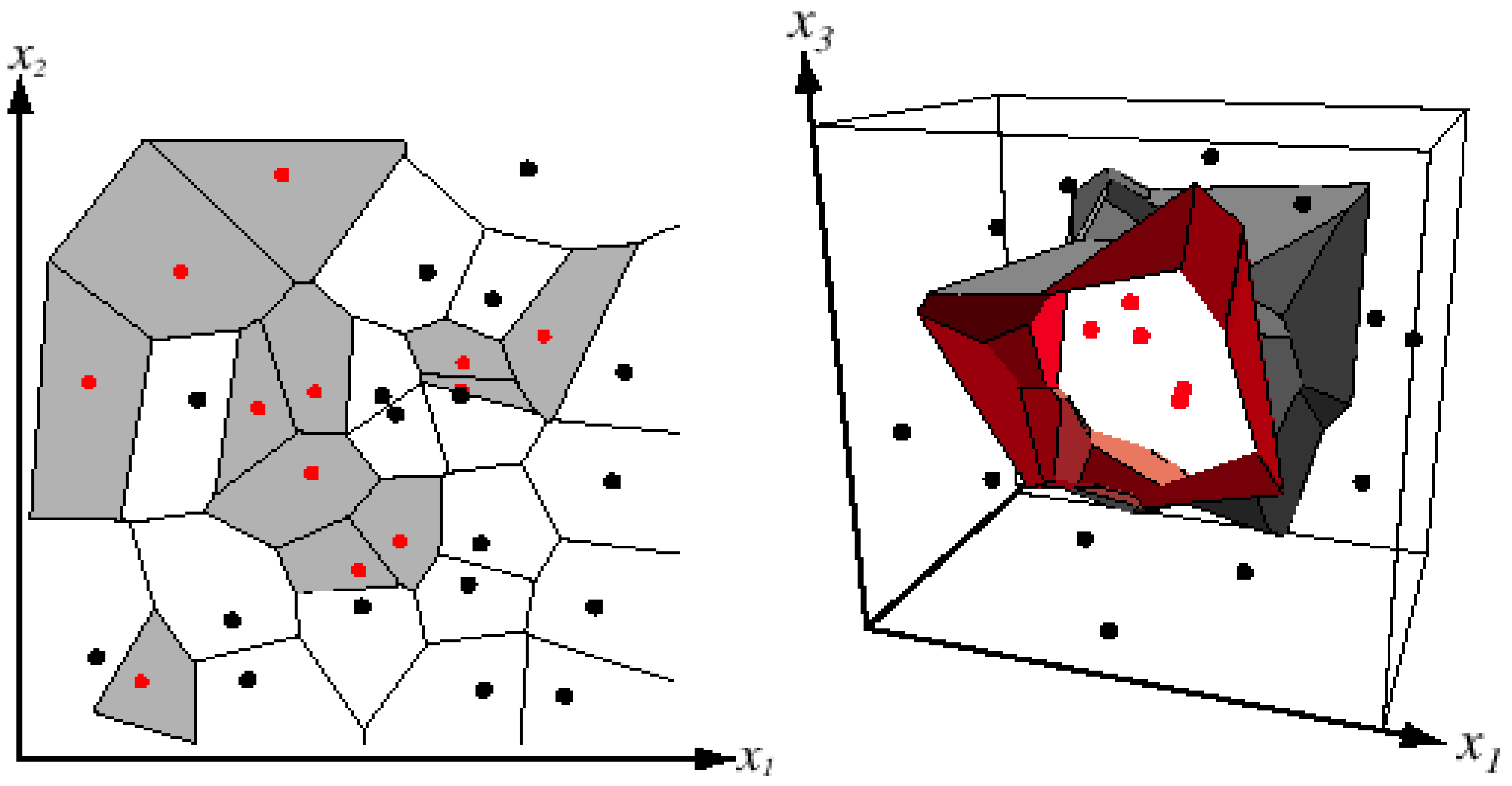


FIGURE 4.13. In two dimensions, the nearest-neighbor algorithm leads to a partitioning of the input space into Voronoi cells, each labeled by the category of the training point it contains. In three dimensions, the cells are three-dimensional, and the decision boundary resembles the surface of a crystal. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

The k-Nearest-Neighbor Rule

- Goal: Classify x by assigning it the label most frequently represented among the k nearest samples
- Use a voting scheme

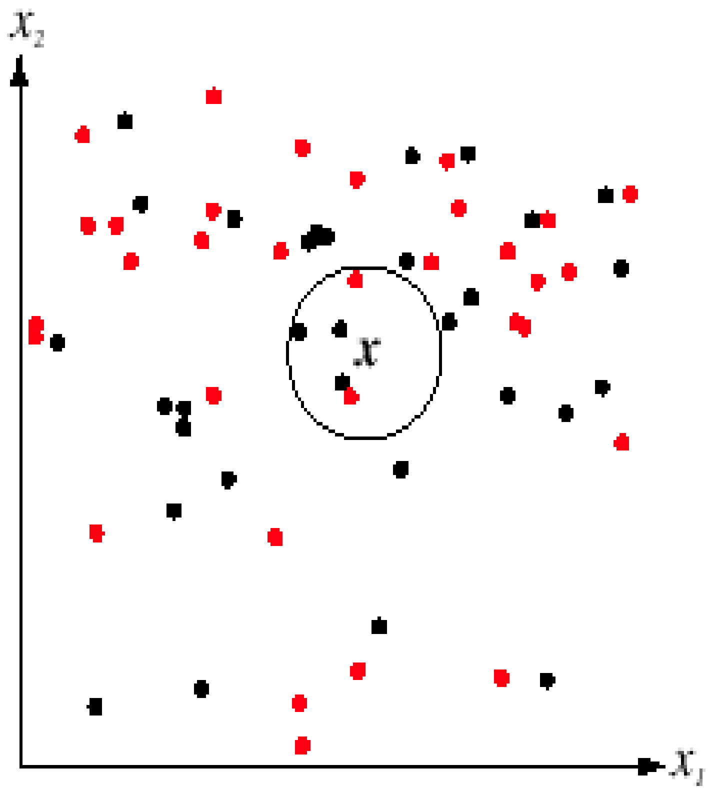


FIGURE 4.15. The k-nearest-neighbor query starts at the test point **x** and grows a spherical region until it encloses k training samples, and it labels the test point by a majority vote of these samples. In this $k = 5$ case, the test point **x** would be labeled the category of the black points. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Matlab Example

```
data = dlmread('pima-indians-diabetes.data');

data = reshape(data,[],9);

% use randperm to re-order data. ignore if not using Matlab
rp = randperm(length(data));
data=data(rp,:);

%split = length(data)/2;
split = 300;

train_data = data(1:split,:);
test_data = data(split+1:end,:);
```

```
% pick features
active_feat = [1:3];

% training
% NOT NEEDED

% testing
correct=0;
wrong=0;
```

```
for i=1:length(test_data)

    sample=test_data(i,active_feat);

    dist = train_data(:,active_feat)-repmat(sample,length(train_data),1);
    dist = dist*dist';

    % we are only interested in the diagonal elements
    % DON'T USE QUADRATIC DISTANCE COMPUTATION IN PRACTICE
    fin_dist = diag(dist);
    [min_d index] = min(fin_dist);

    if(test_data(i,9) == train_data(index,9))
        correct = correct+1;
    else
        wrong = wrong+1;
    end
end
```