Chapter 4:
Non-Parametric Techniques

- Introduction
- Density Estimation
- Parzen Windows
- Kn-Nearest Neighbor Density Estimation
- K-Nearest Neighbor (KNN) Decision Rule

Supervised Learning

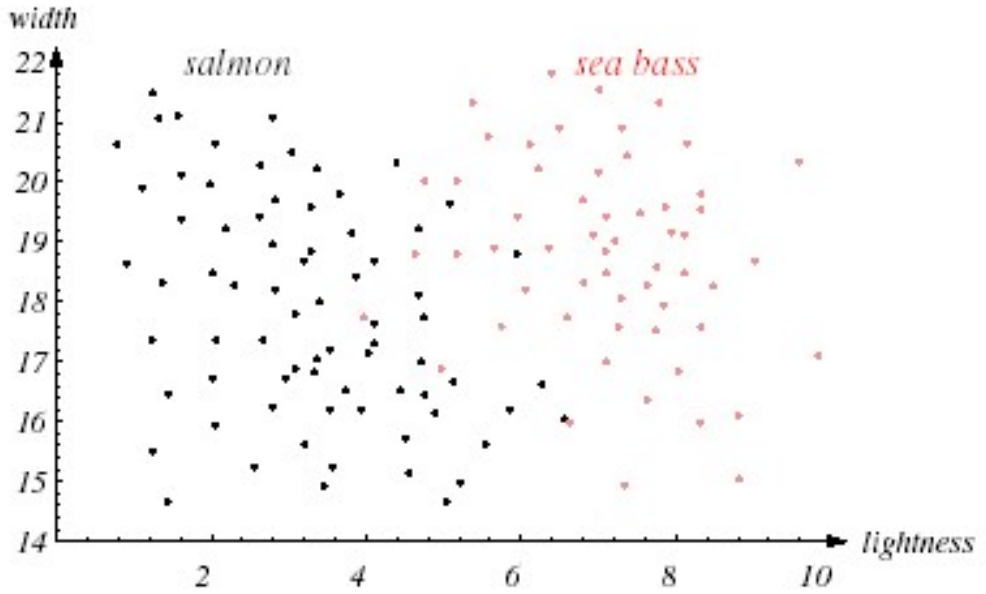
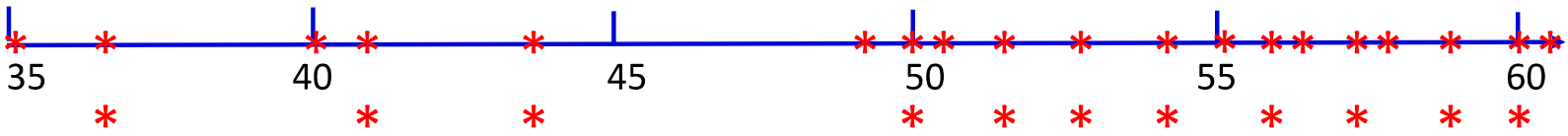
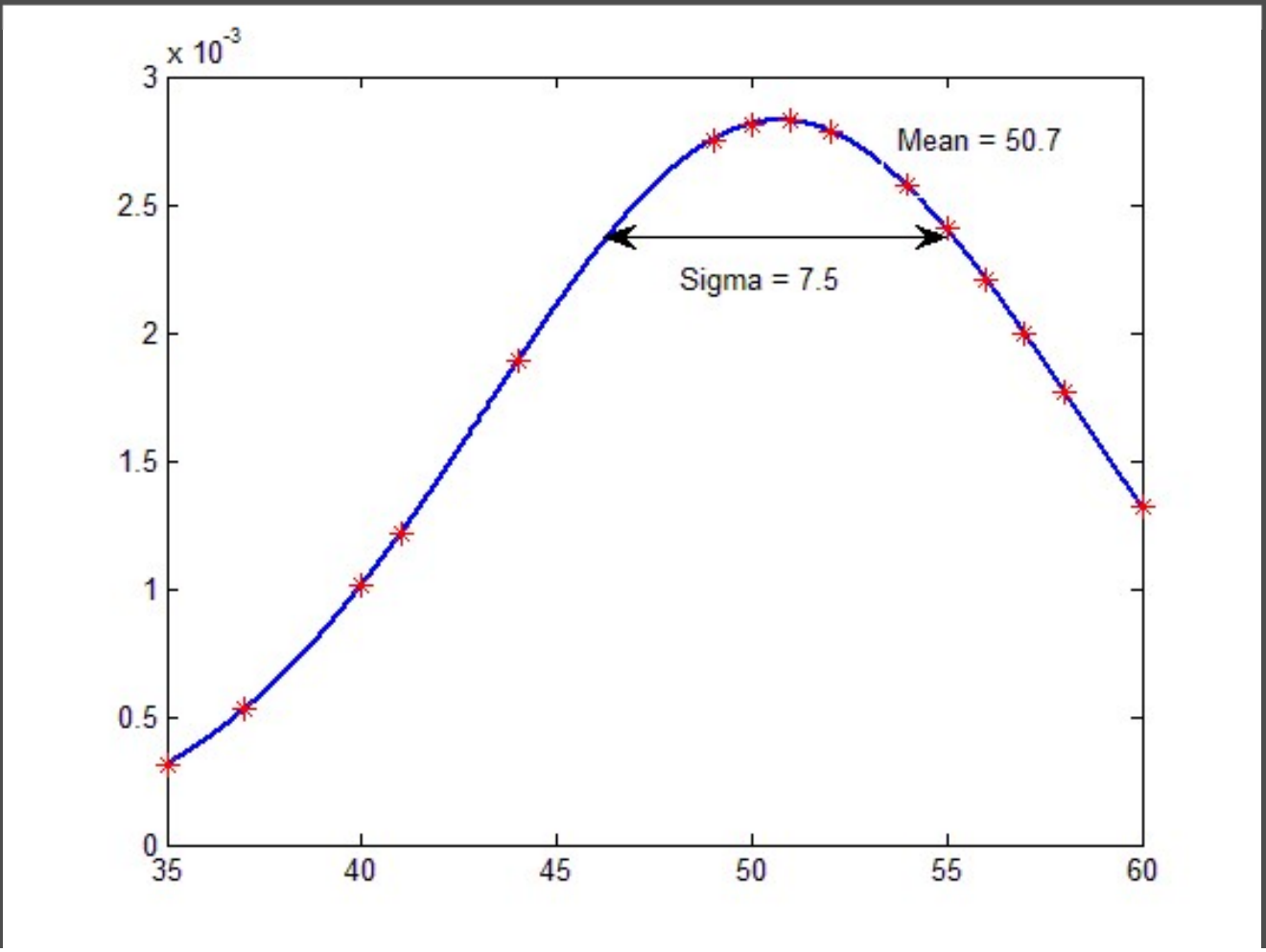


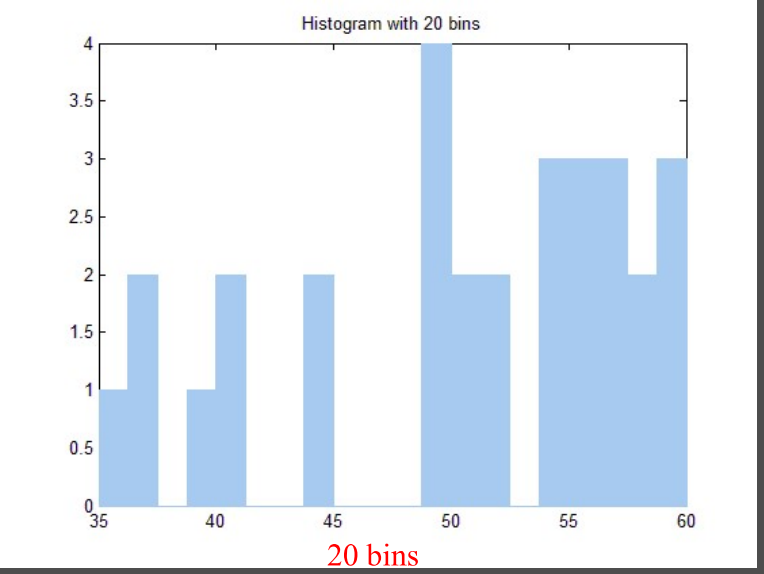
FIGURE 1.4. The two features of lightness and width for sea bass and salmon. The dark line could serve as a decision boundary of our classifier. Overall classification error on the data shown is lower than if we use only one feature as in Fig. 1.3, but there will still be some errors. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

How to fit a density function to 1-D data?

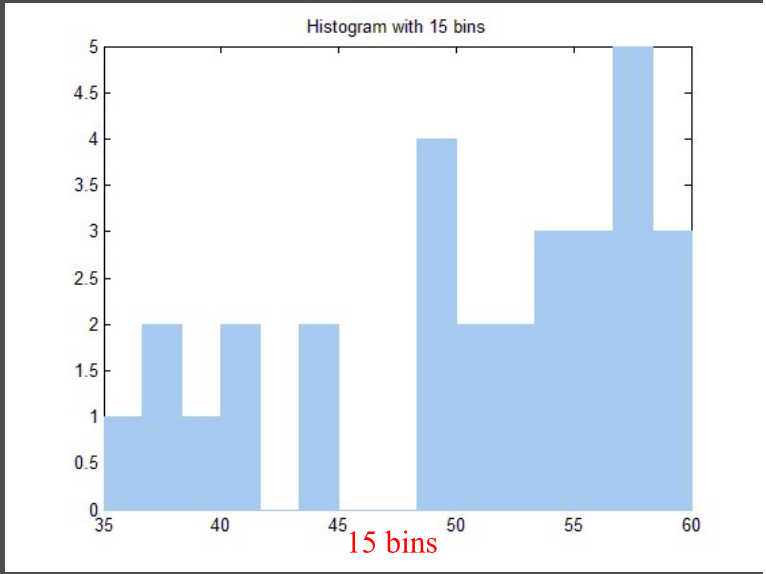


Fitting a Normal density using MLE

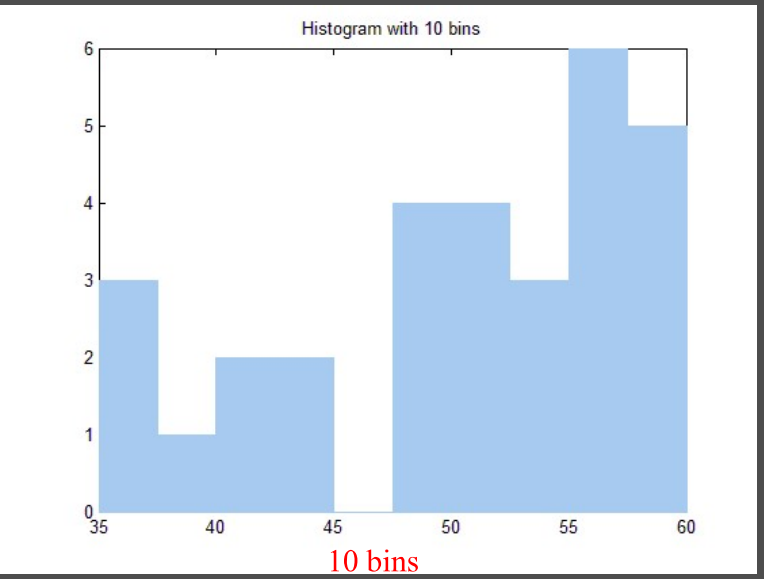




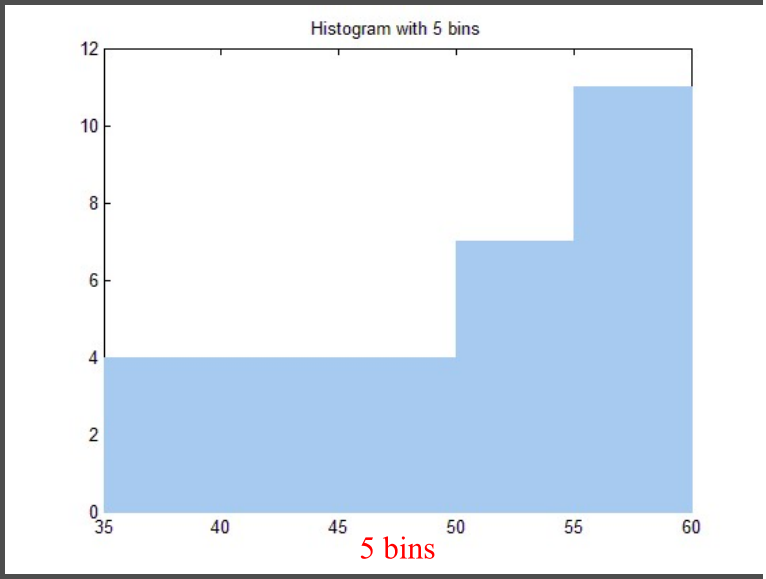
20 bins



15 bins



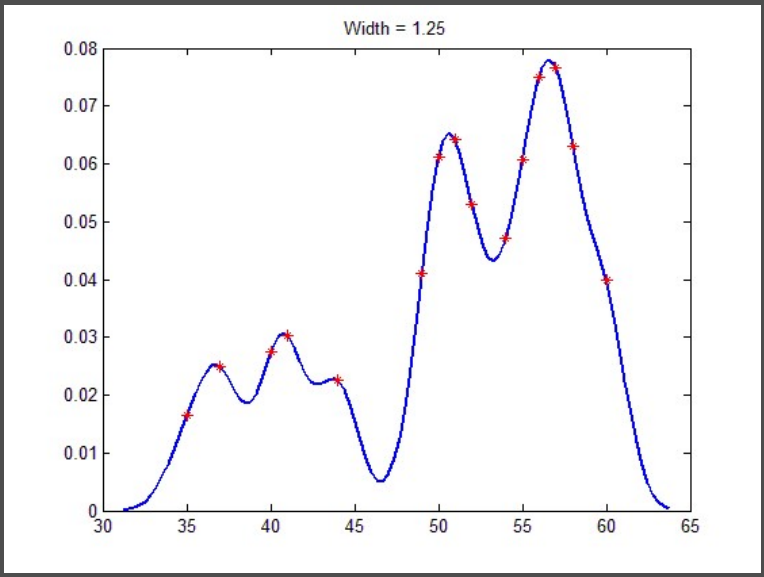
10 bins



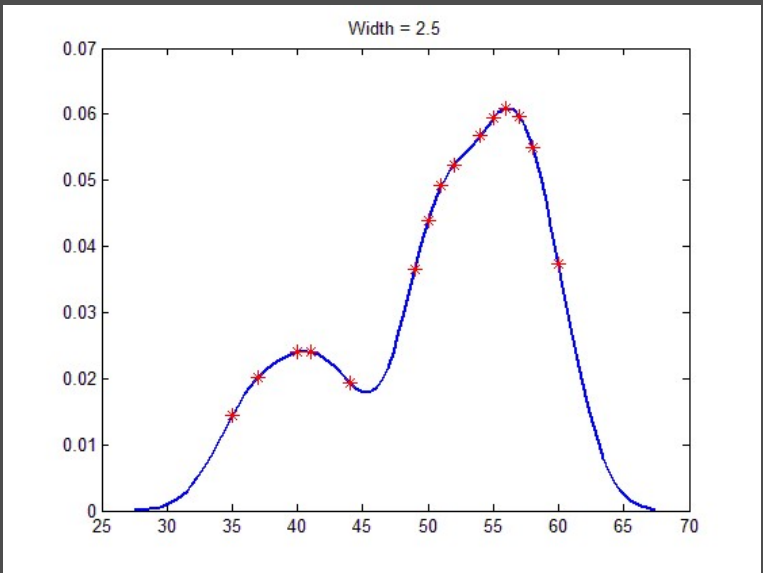
5 bins

How many bins?

Parzen Windows

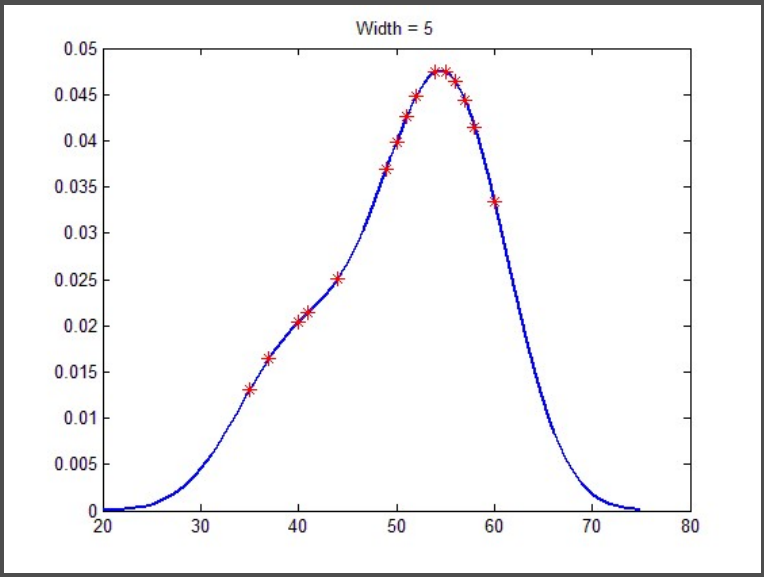


Width = 1.25

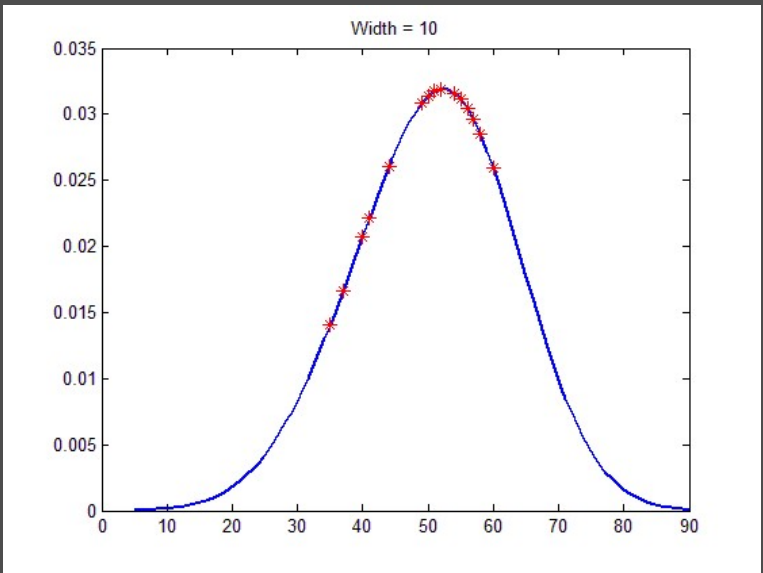


Width = 2.5

Parzen Windows



Width = 5

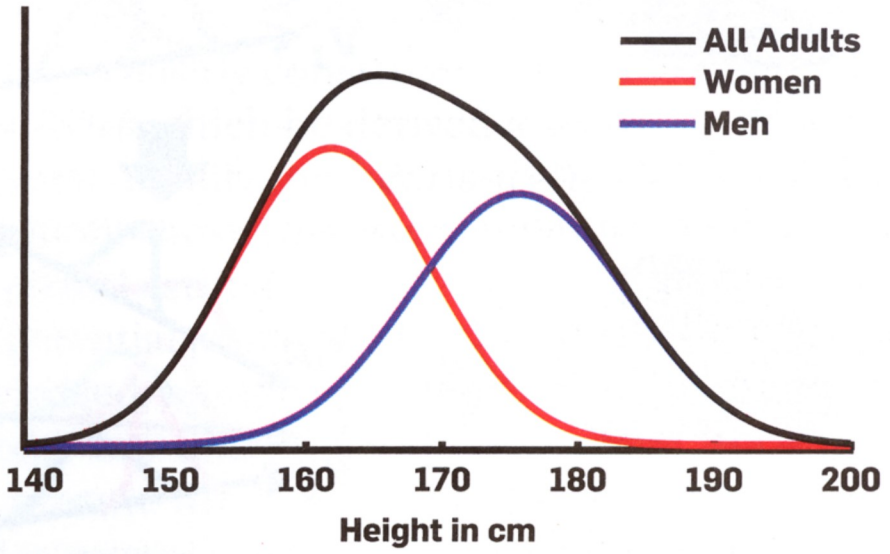


Width = 10

Gaussian Mixture Model

- A weighted combination of a small but unknown no. of Gaussians
- Can one recover the parameters of Gaussians from unlabeled data?

Figure 1. The Gaussian approximations of the heights of adult women (red) and men (blue). Can one recover estimates of these Gaussian components given only the aggregate data without gender labels (black)? [Raw data from NHANES].



Introduction

- All parametric densities are unimodal (have a single local maximum), whereas many practical problems involve multi-modal densities (subgroups)
- Nonparametric procedures can be used with arbitrary distributions and without the assumption that the form of the underlying densities is known
- Two types of nonparametric methods:
  - Estimating class conditional densities, P$(x \mid \omega_j)$
  - Bypass class-conditional density estimation and directly estimate the a-posteriori probability

Density Estimation

- Basic idea: 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). How do we estimate P?
- If we have a sample of size n (i.i.d from p(x)), the probability that k of the n points fall in R is (binomial law):

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

and the expected value for k is:

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

- ML estimation of unknown $P = \theta$

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

- Ratio k/n is a good estimate for the probability P and can be used to estimate density fn. p (x)

- Assume $p(x)$ is continuous and region R is so small that p does not vary significantly within it. Then

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

where x is a point in R and V the volume enclosed by R

Combining equations (1) , (3) and (4) yields:

$$p(x) \cong \frac{k/n}{V}$$

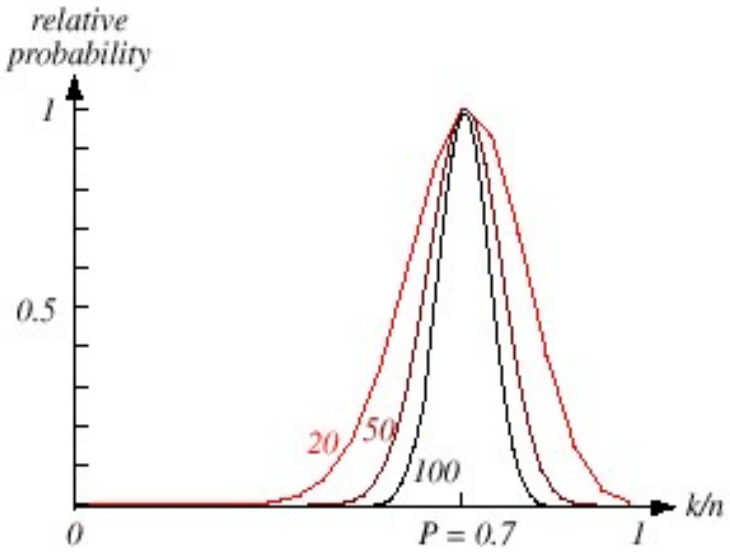


FIGURE 4.1. The relative probability an estimate given by Eq. 4 will yield a particular value for the probability density, here where the true probability was chosen to be 0.7. Each curve is labeled by the total number of patterns n sampled, and is scaled to give the same maximum (at the true probability). The form of each curve is binomial, as given by Eq. 2. For large n, such binomials peak strongly at the true probability. In the limit $n \to \infty$, the curve approaches a delta function, and we are guaranteed that our estimate will give the true probability. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

- Conditions for convergence

The fraction *k/(nV)* is a "space averaged" value of *p(x).* Convergence to true *p(x)* is obtained only if V approaches zero.

$$\lim_{V \to 0,\, k=0} p(x) = 0 \quad (\text{if } n = \text{fixed})$$

This is the case where no samples are included in R

$$\lim_{V \to 0,\, k \neq 0} p(x) = \infty$$

In this case, the estimate diverges

- The volume V needs to approach 0 if we want to obtain p(x) rather than just an averaged version of it.
  - In practice, V cannot be allowed to become small since the number of samples , n, is always limited
  - We have to accept a certain amount of averaging in p(x)
  - Theoretically, if an unlimited number of samples is available, we can circumvent this difficulty as follows:

    To estimate the density of x, we form a sequence of regions $R_1, R_2, \ldots$ containing x: the first region contains one sample, the second two samples and so on.

    Let V_n be the volume of R_n, k_n be the number of samples falling in R_n and $p_n(x)$ be the n^{th} estimate for $p(x)$:

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

Necessary conditions for $p_n(x)$ to converge to $p(x)$:

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

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

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

Two ways of defining sequences of regions that satisfy these conditions:

(a) Shrink an initial region where $V_n = 1/\sqrt{n}$; it can be shown that

$$p_n(x) \underset{n\to\infty}{\longrightarrow} p(x)$$

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

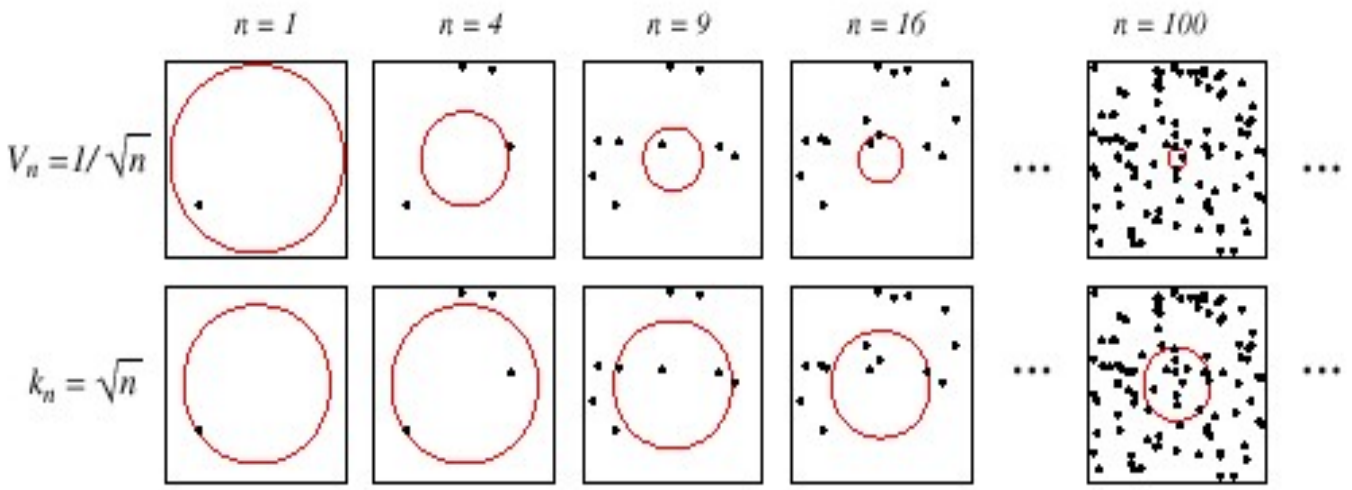


FIGURE 4.2. There are two leading methods for estimating the density at a point, here at the center of each square. The one shown in the top row is to start with a large volume centered on the test point and shrink it according to a function such as $V_n = 1/\sqrt{n}$. The other method, shown in the 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

- For simplicity, let us assume that the region R_n is a d-dimensional hypercube

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

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

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

- $\varphi((x-x_i)/h_n)$ 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:

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

$P_n(x)$ estimates $p(x)$ as an average of functions of x and the samples x_i, $i = 1, \ldots, n$. These functions φ can be of any form, e.g., Gaussian.

- The behavior of the Parzen-window method

  - Case where $p(x) \rightarrow N(0,1)$

    Let $\varphi(u) = (1/\sqrt{(2\pi)}) \exp(-u^2/2)$ and $h_n = h_1/\sqrt{n}$ $(n>1)$

    (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.

Parzen Window Density Estimate

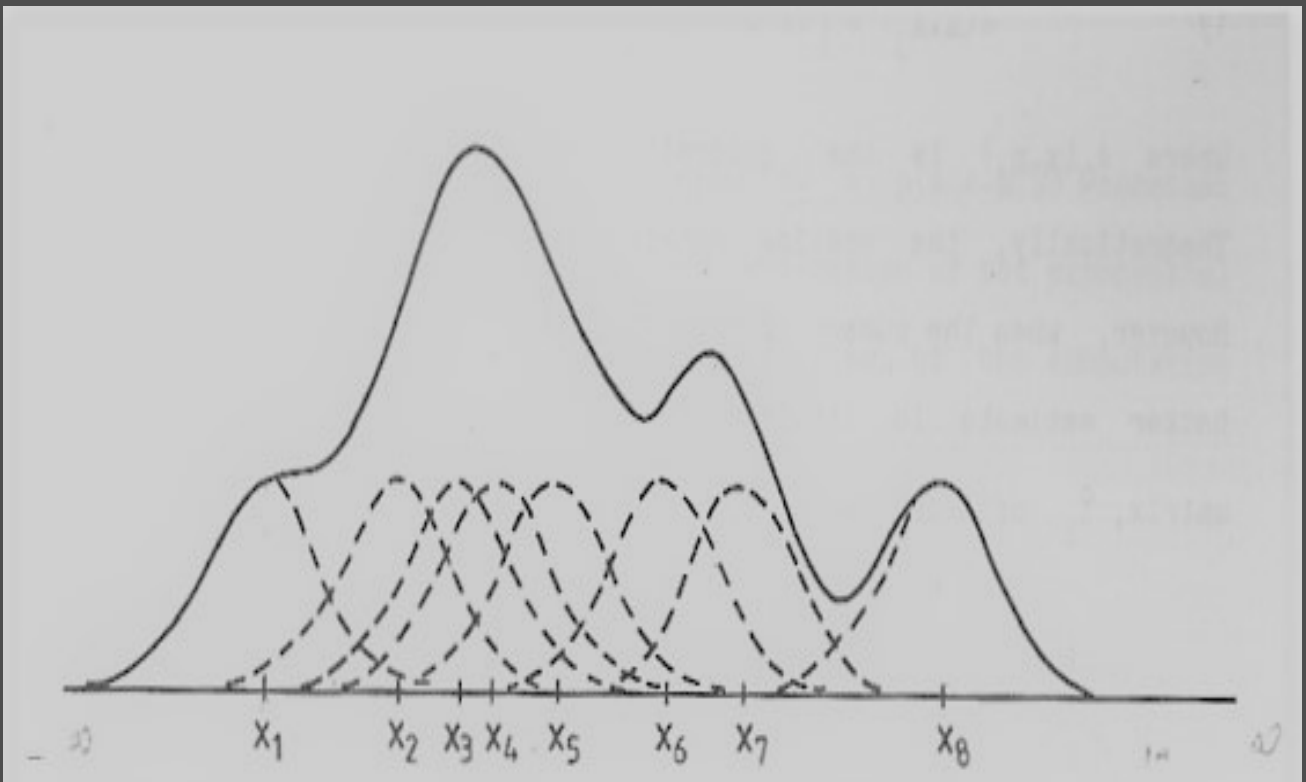


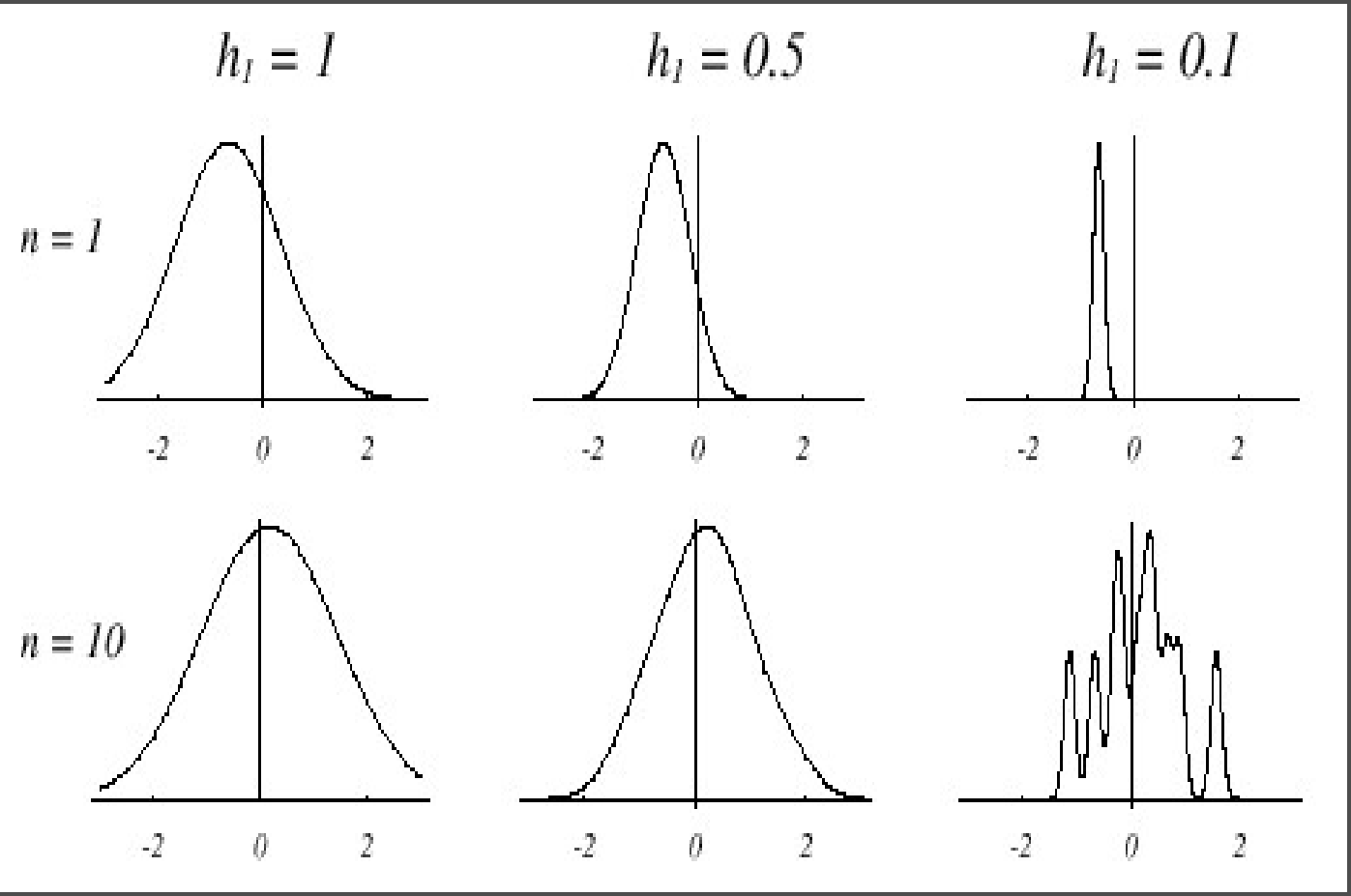
Figure A.2 Summing the contributions of the kernels to obtain a Parzen p.d.f. estimate.

- Numerical results:

For $n = 1$ and $h_1 = 1$

$$p_1(x) = \varphi(x - x_1) = \frac{1}{\sqrt{2\pi}} e^{-1/2(x - x_1)^2} \rightarrow N(x_1, 1)$$

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


$h_1 = 1$
$h_1 = 0.5$
$h_1 = 0.1$
$n = 1$
$n = 10$
-2
0
2

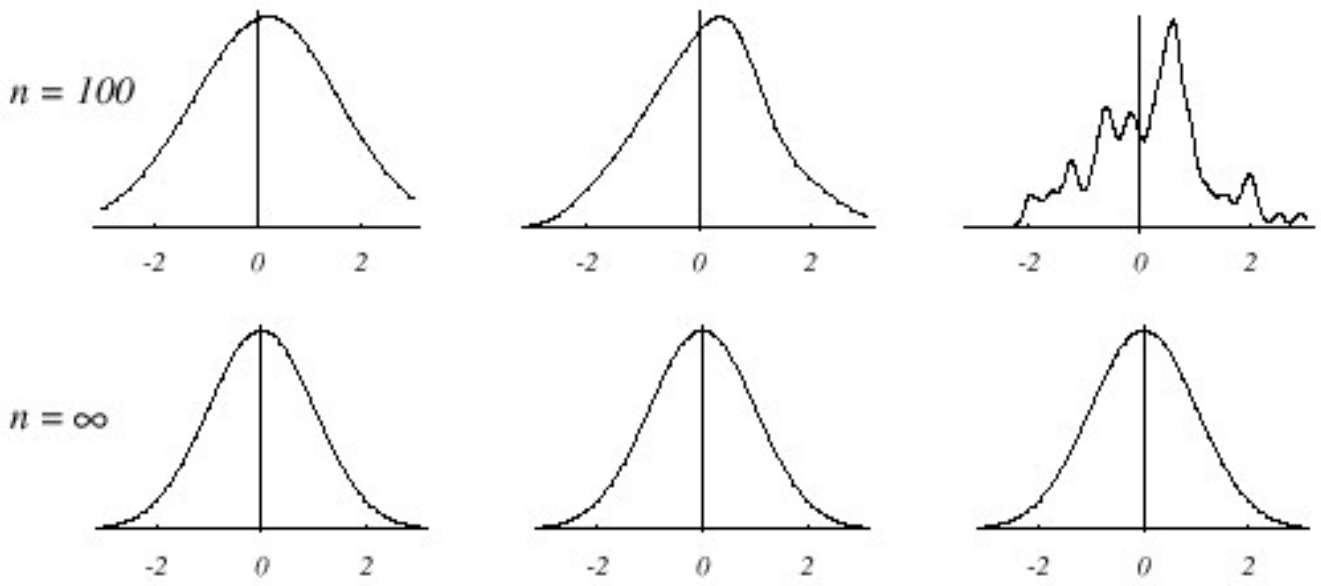
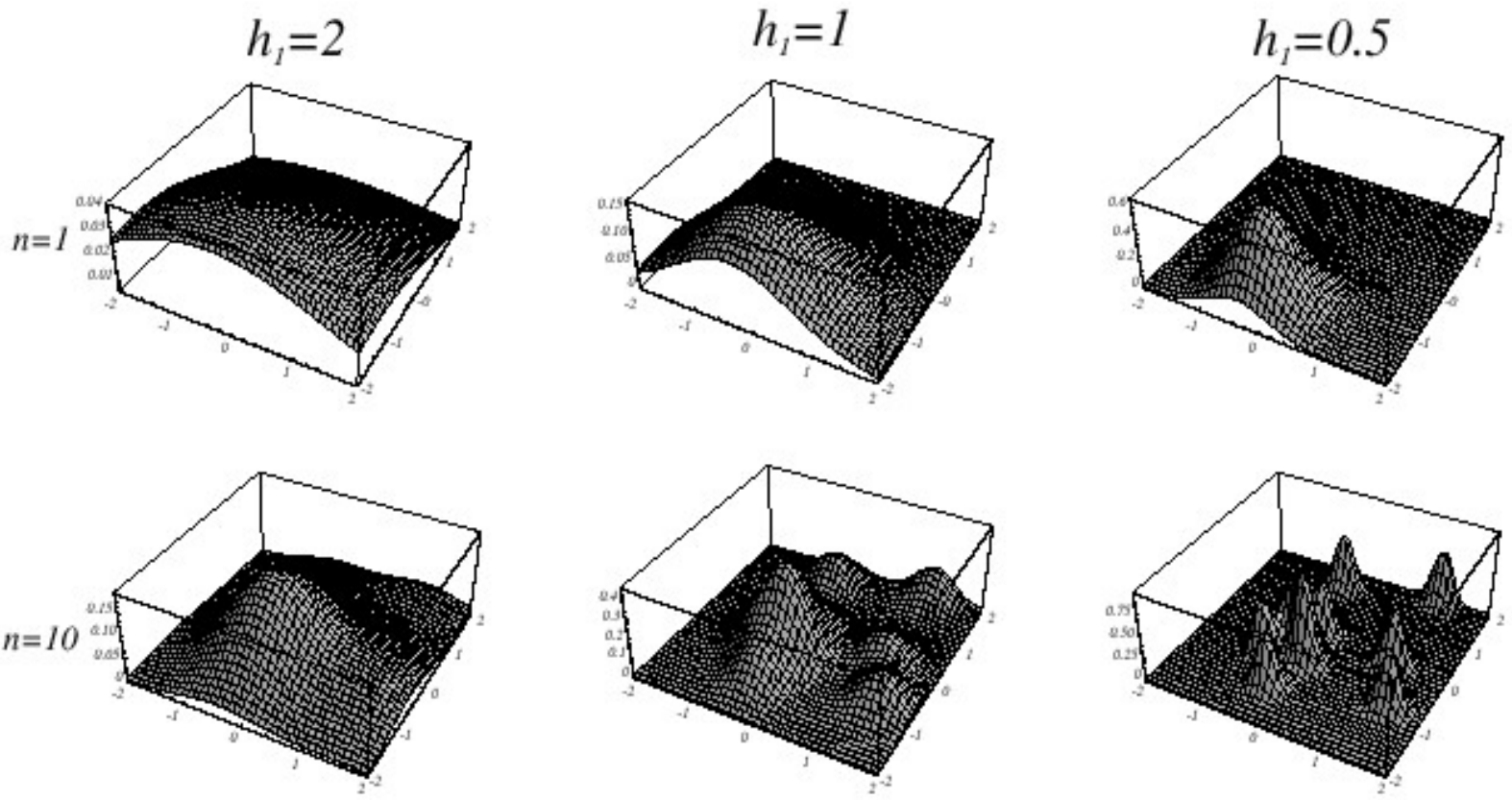


FIGURE 4.5. Parzen-window estimates of a univariate normal density using different window widths and numbers of samples. The vertical axes have been scaled to best show the structure in each graph. Note particularly that the $n = \infty$ estimates are the same (and match the true density function), regardless 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:



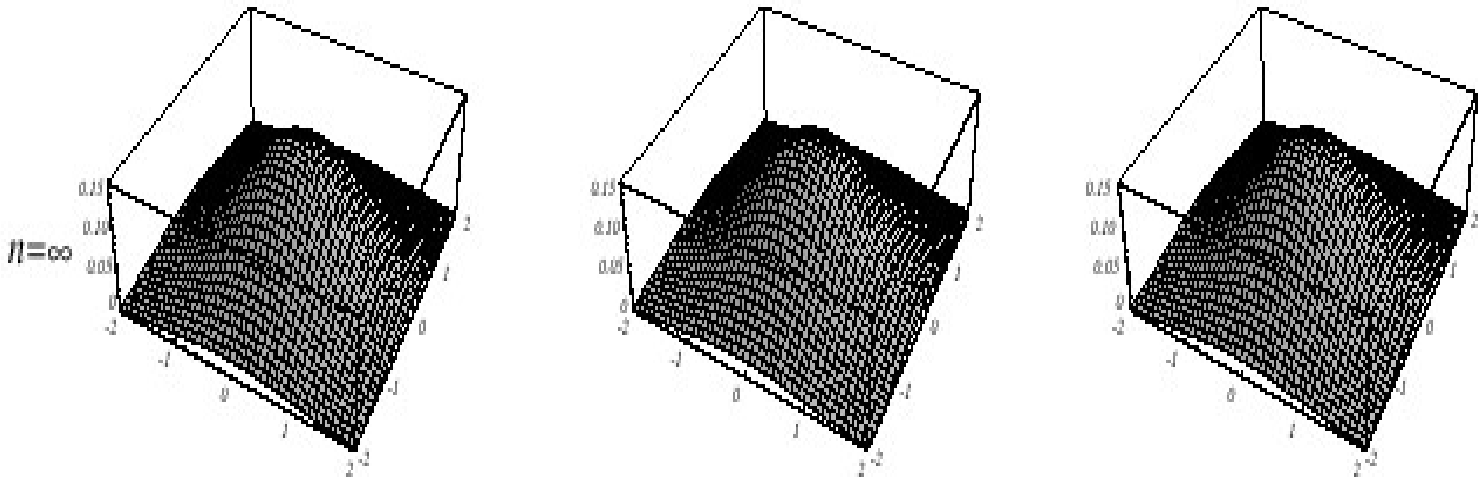
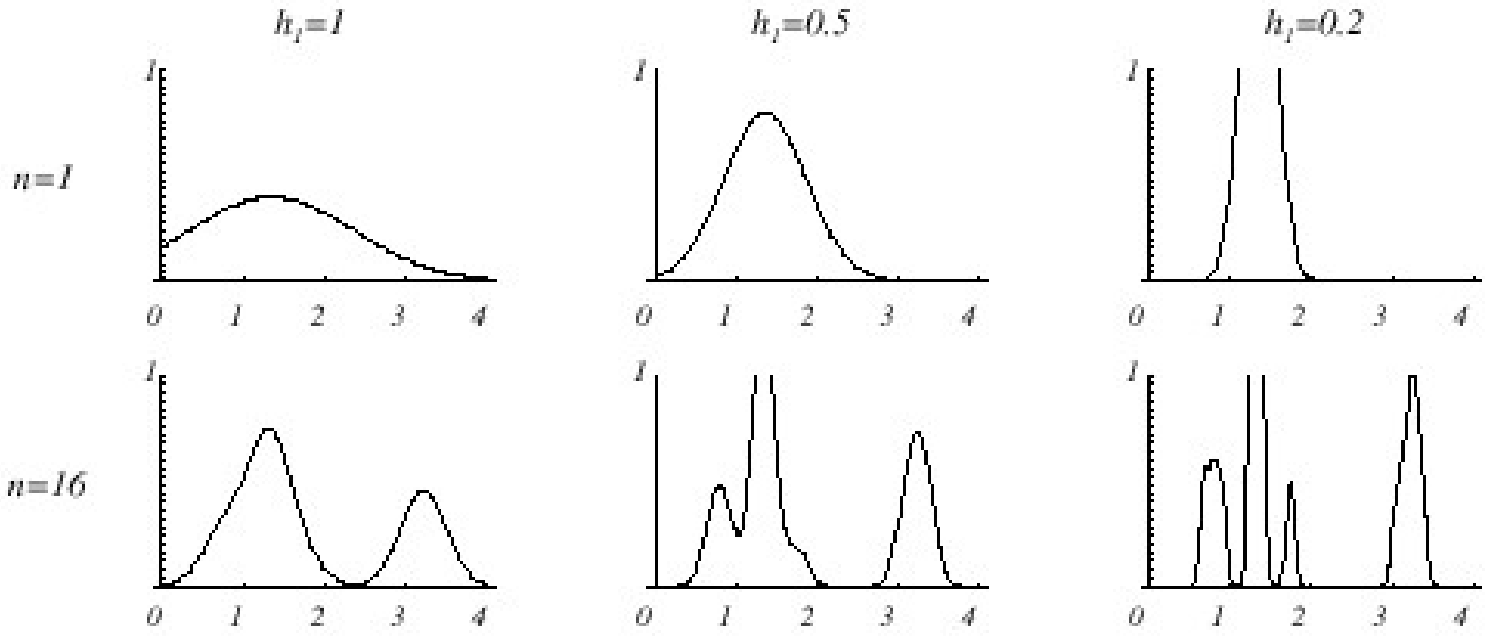


FIGURE 4.6. Parzen-window estimates of a bivariate normal density using different window widths and numbers of samples. The vertical axes have been scaled to best show the structure in each graph. Note particularly that the $n = \infty$ estimates are the same (and match the true distribution), regardless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

Unknown density $p(x) = \lambda_1.U(a,b) + \lambda_2.T(c,d)$ *is a* mixture of a uniform and a triangle density



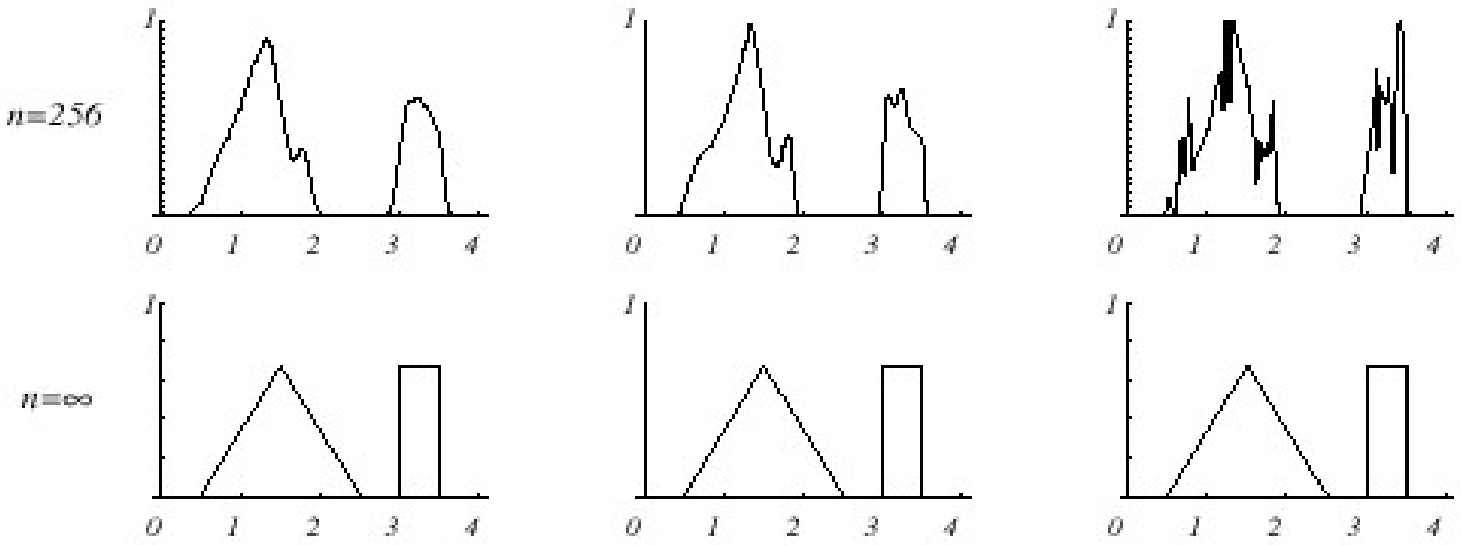


FIGURE 4.7. Parzen-window estimates of a bimodal distribution using different window widths and numbers of samples. Note particularly that the $n = \infty$ estimates are the same (and match the true distribution), regardless of window width. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern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

Parzen-window density estimates

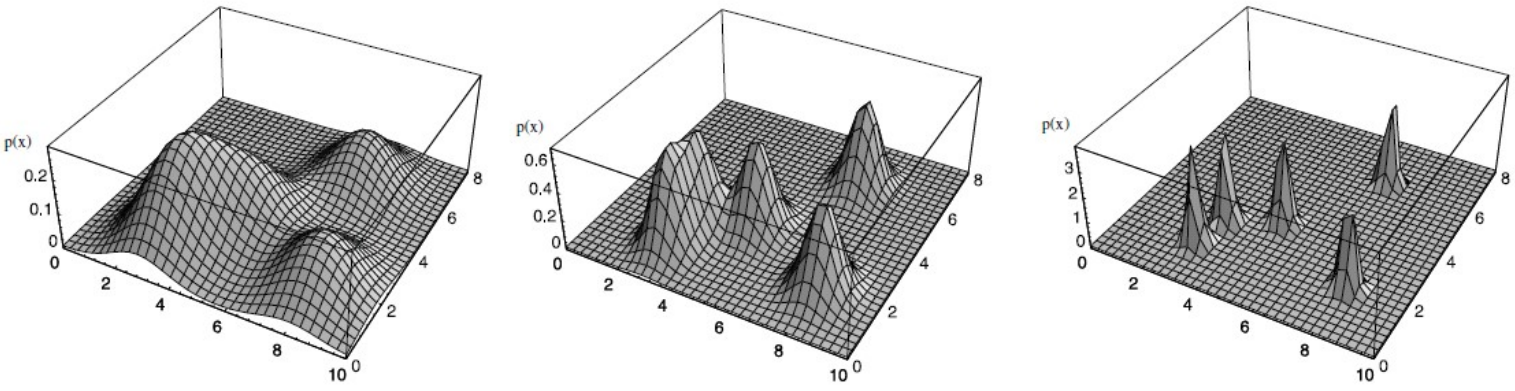


Figure 4.4: Three Parzen-window density estimates based on the same set of five samples, using the window functions in Fig. 4.3. As before, the vertical axes have been scaled to show the structure of each function.

- Classification example

  How do we design classifiers based on Parzen-window density estimation?

  - Estimate the class-conditional densities for each class; classify a test sample by the label corresponding to the maximum a posteriori density

  - Decision region for a Parzen-window classifier depends upon the choice of window function and the window width; in practice window function chosen is Gaussian

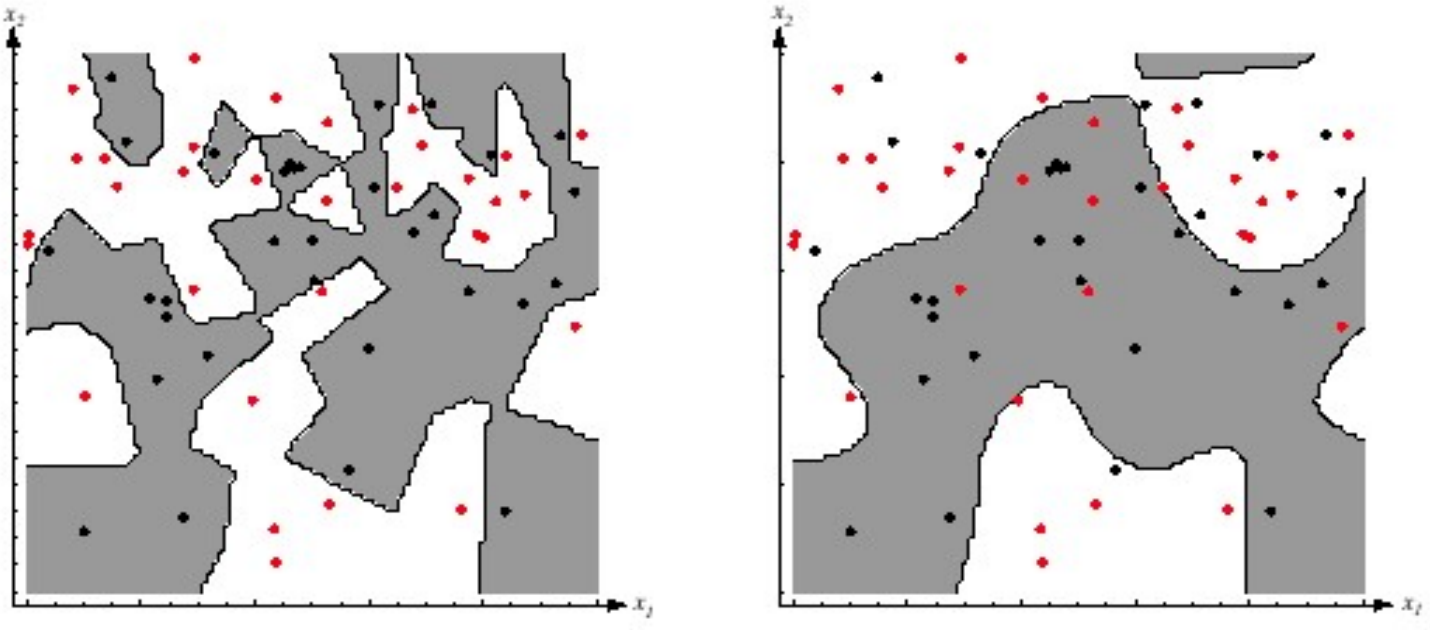


FIGURE 4.8. The decision boundaries in a two-dimensional Parzen-window dichotomizer depend on the window width h. At the left a small h leads to boundaries that are more complicated than for large h on same data set, shown at the right. Apparently, for these data a small h would be appropriate for the 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*.

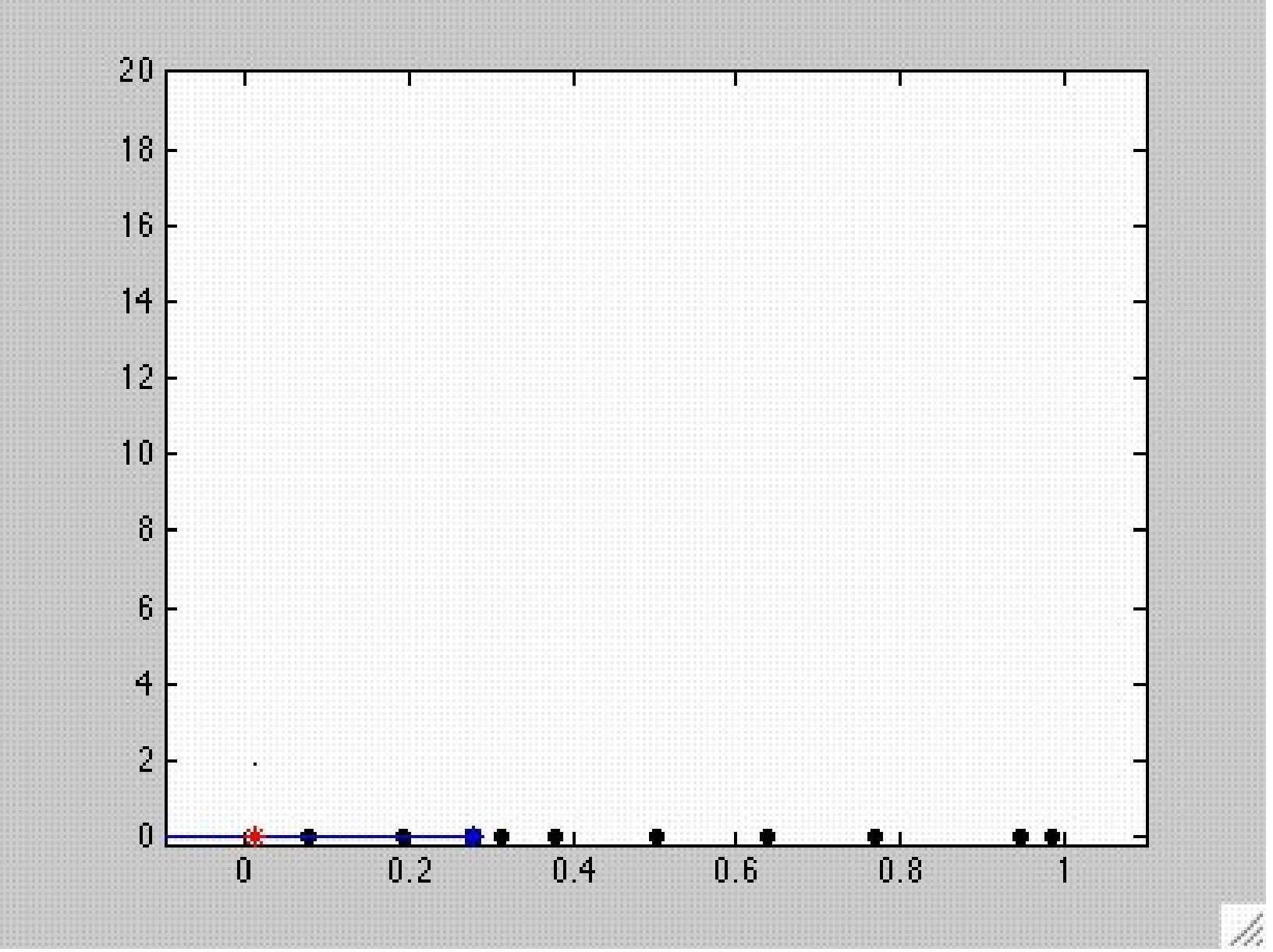
K_n - Nearest Neighbor Density Estimation

- Let the region volume be a function of the training data
- Center a region about x and let it grows until it captures k_n samples $(k_n = f(n))$
- k_n *denotes* the k_n nearest-neighbors of x

Two possibilities can occur:

- If the true density is high near x, the region will be small which provides a good resolution
- If the true density at x is low, the region will grow large to capture k_n samples

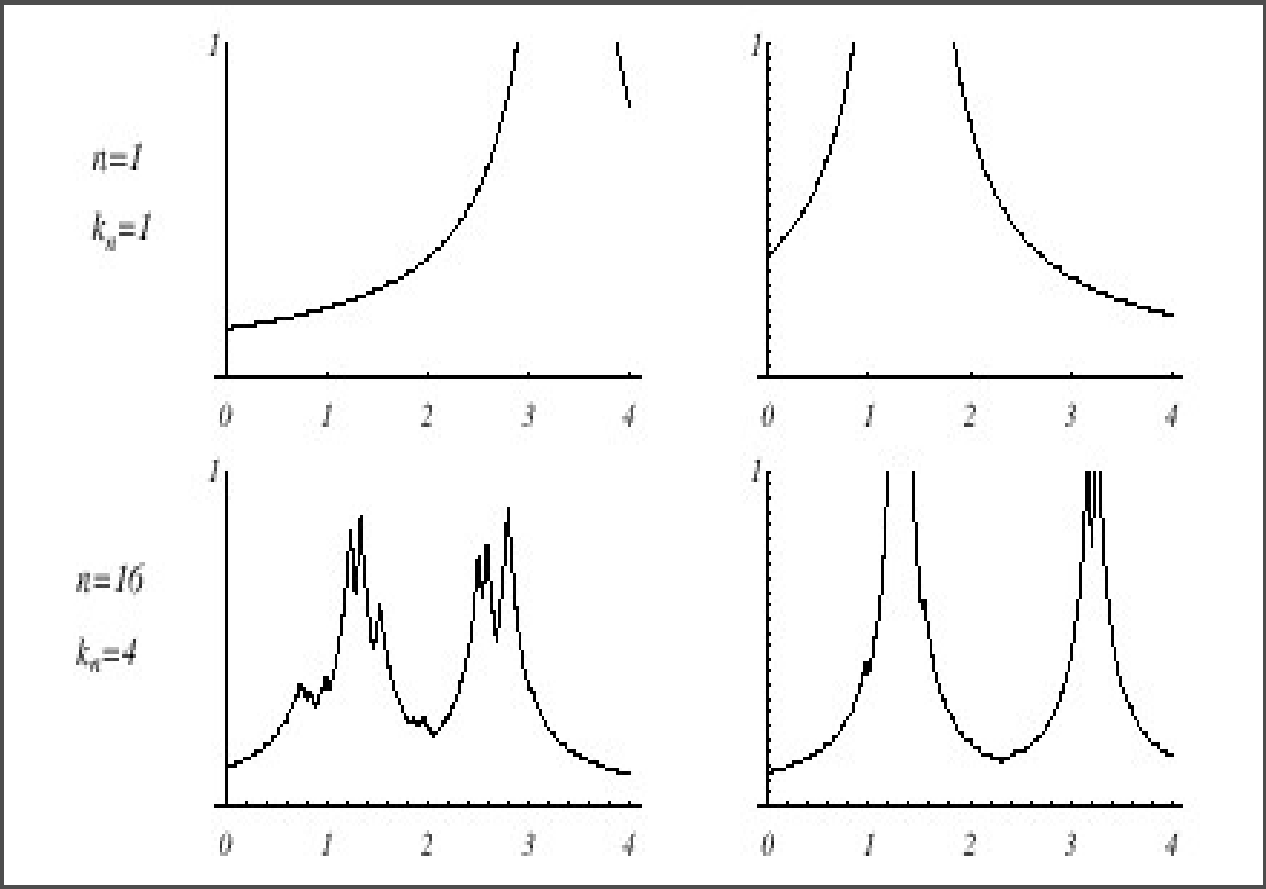
We can obtain a family of estimates by setting $k_n=k_1/\sqrt{n}$ and choosing different values for k_1


20
18
16
14
12
10
8
6
4
2
0
0
0.2
0.4
0.6
0.8
1

Illustration

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

$$P_n(x) = k_n / n.V_n = 1 / V_1 = 1 / 2|x-x_1|$$


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

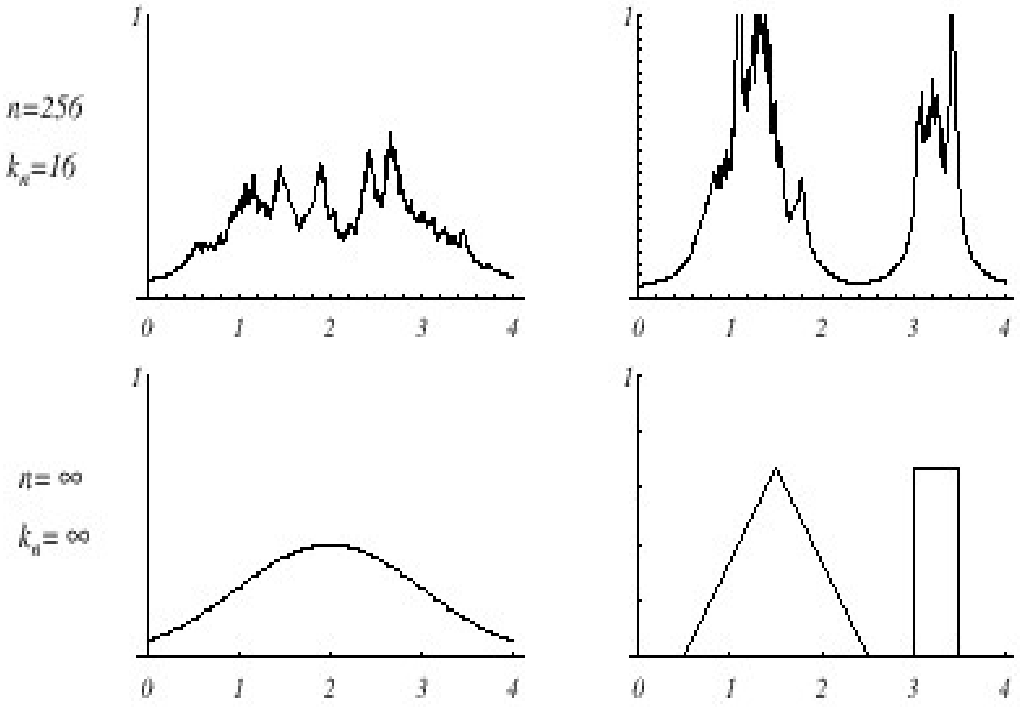


FIGURE 4.12. Several *k*-nearest-neighbor estimates of two unidimensional densities: a Gaussian and a bimodal distribution. Notice how the finite *n* estimates can be quite "spiky." From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons,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

- Estimation of a-posteriori 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
    - Suppose k_i samples amongst k are labeled ω_i then:

$$p_n(x, \omega_i) = k_i / n.V$$

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_j

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

- If k is large and the cell is sufficiently small, the performance will approach the optimal Bayes rule

Nearest –neighbor Decision 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, namely the Bayes error 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 proved!)

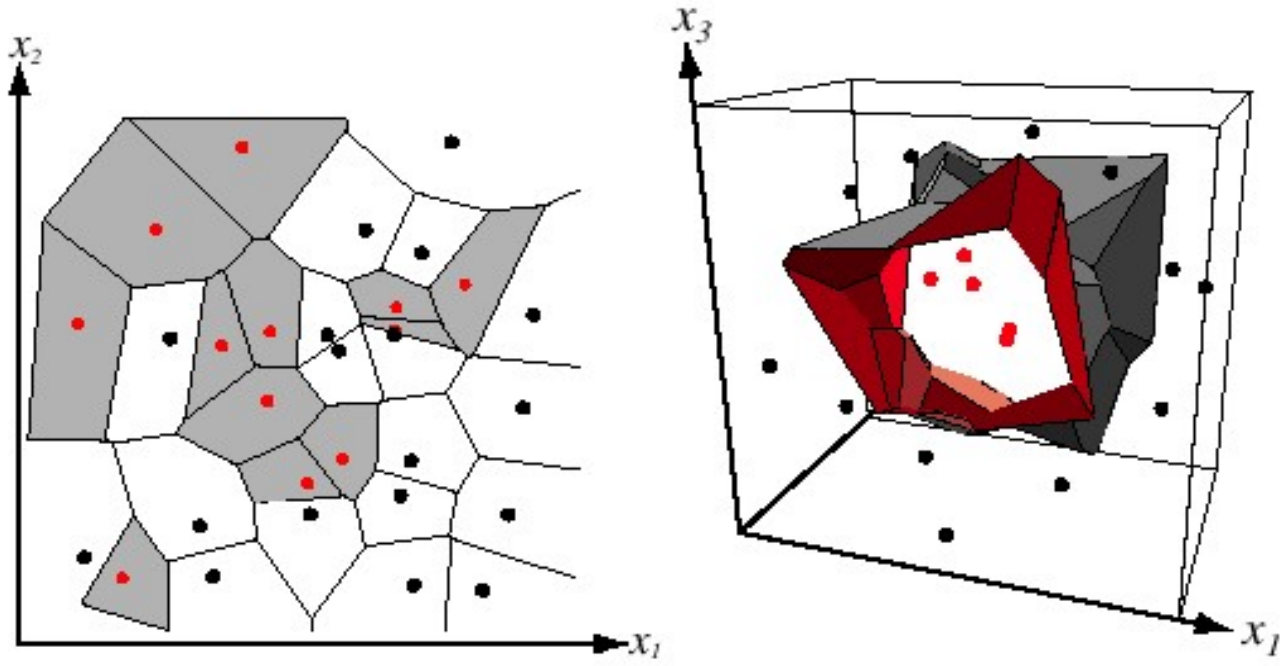
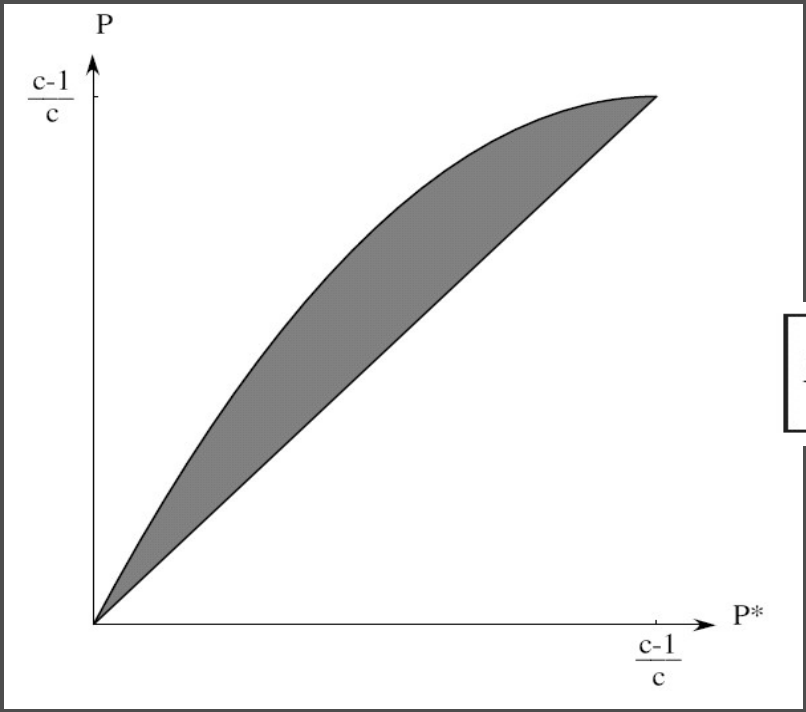


FIGURE 4.13. In two dimensions, the nearest-neighbor algorithm leads to a partitioning of the input space into Voronoi cells, each labeled by the category of the training point it contains. In three dimensions, the cells are three-dimensional, and the decision boundary resembles the surface of a crystal.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

Bound on the Nearest Neighbor Error Rate



$$P^* \leq P \leq P^*\left(2 - \frac{c}{c-1}P^*\right).$$

Figure 4.14: Bounds on the nearest-neighbor error rate P in a c-category problem given infinite training data, where P^* is the Bayes error (Eq. 53). At low error rates, the nearest-neighbor error rate is bounded above by twice the Bayes rate.

k – Nearest Neighbor (k-NN) Decision Rule

- Classify the test sample x by assigning it the label most frequently represented among the k nearest samples of x

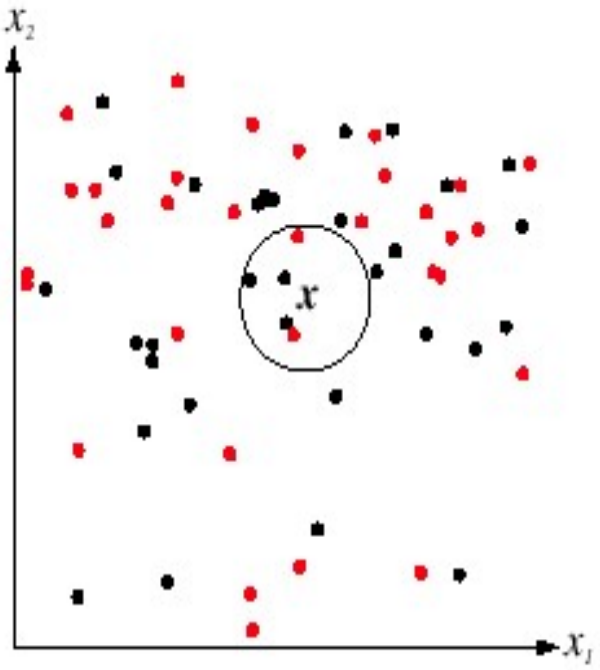
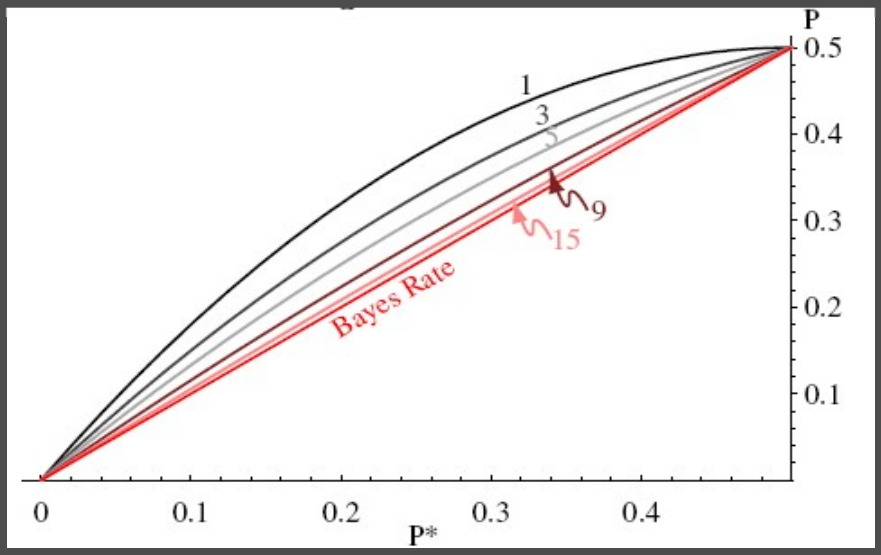


FIGURE 4.15. The k-nearest-neighbor query starts at the test point **x** and grows a spherical region until it encloses k training samples, and it labels the test point by a majority vote of these samples. In this $k = 5$ case, the test point **x** would be labeled the category of the black points. From: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

The *k*-Nearest Neighbor Rule



The bounds on *k*-NN error rate for different values of *k.*

As *k* increases, the upper bounds get progressively closer to the lower bound – the Bayes Error rate.

When *k* goes to infinity, the two bounds meet and *k*-NN rule becomes optimal.

Metric

- Nearest neighbor rule relies on a metric or a "distance" function between the patterns
- A metric D(.,.) is a function that takes two vectors **a** and **b** as arguments, and returns a scalar function.
- For a function D(.,.) to be called a metric, it needs to satisfy the following properties for any given vectors **a, b** and **c**.

$$\textbf{non-negativity: } D(\mathbf{a}, \mathbf{b}) \geq 0$$

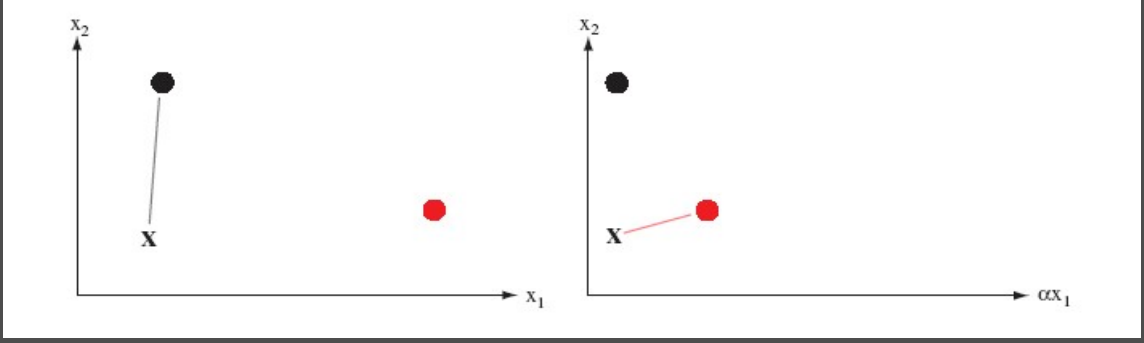
$$\textbf{reflexivity: } D(\mathbf{a}, \mathbf{b}) = 0 \text{ if and only if } \mathbf{a} = \mathbf{b}$$

$$\textbf{symmetry: } D(\mathbf{a}, \mathbf{b}) = D(\mathbf{b}, \mathbf{a})$$

$$\textbf{triangle inequality: } D(\mathbf{a}, \mathbf{b}) + D(\mathbf{b}, \mathbf{c}) \geq D(\mathbf{a}, \mathbf{c}).$$

Metrics and Nearest-neighbors

- A metric remains a metric even when each dimension is scaled (in fact, under any positive definite linear transformation).



- Left plot shows a scenario where the black point is closer to *x*. Scaling the x_1 axis by 1/3, however, brings the red point closer to *x* than the black point.

Example Metrics

- Minkowski metric

$$L_k(\mathbf{a}, \mathbf{b}) = \left(\sum_{i=1}^{d} |a_i - b_i|^k \right)^{1/k}$$

  - K = 1 gives the Manhattan distance
  - K = 2 gives the Euclidean distance
  - K = ∞ gives the max-distance (maximum value of the absolute differences between the elements of the two vectors)
- Tanimoto metric: used for finding distance between sets

$$D_{Tanimoto}(\mathcal{S}_1, \mathcal{S}_2) = \frac{n_1 + n_2 - 2n_{12}}{n_1 + n_2 - n_{12}},$$

Computational Complexity of K-NN Rule

- Complexity both in terms of time (search) and space (storage of prototypes) has received a lot of attention
- Suppose we are given n labeled samples in d dimensions and we seek the sample closest to a test point x. Naïve approach would require calculating the distance of x to each stored point and then find the closest.
- Three general algorithmic techniques for reducing the computational burden
  - computing partial distances
  - Pre-structuring
  - editing the stored prototypes

Nearest-Neighbor Editing

- For each given test point, k-NN has to make N distance computations, followed by selection of k-closest neighbors.
  - Imagine a billion web-pages, or a million images.
- Different ways to reduce search complexity
  - Pre-structuring the data. (Eg. Divide 2D data uniformly distributed in $[-1\ 1]^2$ into quadrants and compare the given example to points only in that quadrant)
  - Build search trees
  - Nearest neighbor editing
- Nearest-neighbor editing eliminates useless prototypes
  - Eliminate prototypes that are surrounded by training examples of the same category label
  - Leaves the decision boundaries intact

Nearest neighbor editing: Algorithm

Algorithm 3 (Nearest-neighbor editing)

1 **begin initialize** $j = 0, \mathcal{D}$ = data set, n = #prototypes
2 construct the full Voronoi diagram of $\mathcal{D}$
3 **do** $j \leftarrow j + 1$; for each prototype $\mathbf{x}'_j$
4 Find the Voronoi neighbors of $\mathbf{x}'_j$
5 **if** any neighbor is not from the same class as $\mathbf{x}'_j$ **then** mark $\mathbf{x}'_j$
6 **until** $j = n$
7 Discard all points that are not marked
8 Construct the Voronoi diagram of the remaining (marked) prototypes
9 **end**

The complexity of this algorithm is $O(d^3 n^{[d/2]} \ln n)$, where [d/2] is the floor of d/2.

“A Branch and Bound Algorithm for Computing k-Nearest Neighbors”

K. Fukunga & P. Narendra, IEEE Trans. Computers, July 1975

KNN

- K-NN generally requires large number of computations for large *n* (number of training samples)
- Several solutions to reduce complexity
  - Hart's method: condense the training dataset size while retaining most of the samples that provide sufficient discriminatory information between classes
  - Fischer and Patrick: re-order the examples such that many distance computations can be eliminated
  - Fukunaga and Narendra: hierarchically decompose the design samples into disjoint subsets and apply a branch-and-bound algorithm for searching through the hierarchy.

Stage 1: Hierarchical Decomposition

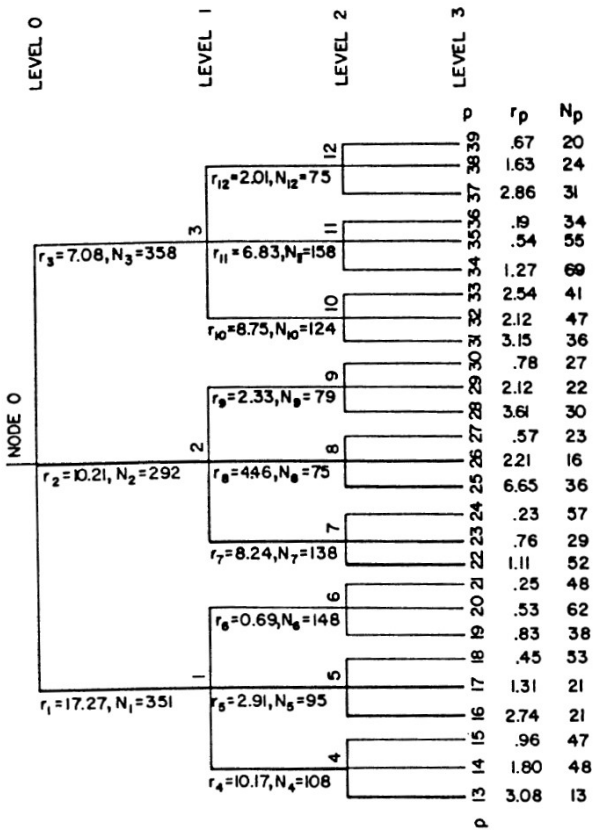


Fig. 1. Results of the hierarchical decomposition (bold lines indicate the nodes expanded by the algorithm for the typical test sample of Experiment 1).

Given a dataset of n samples, divide it into l groups, and recursively divide each of the l-groups further into l-groups each.

Partitioning must be done such that similar samples are placed in the same group.

K-means algorithm is used to divide the dataset, since it scales well to large datasets.

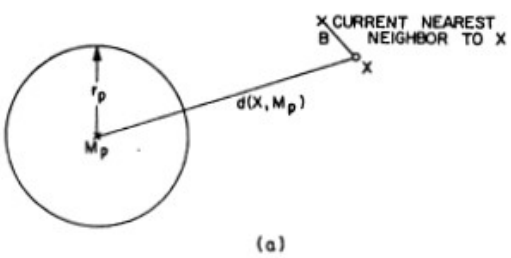
Searching the Tree

- For each node in the tree, evaluate the following quantities

S_p	Set of samples associated with node p
N_p	Number of samples at node p
M_p	Sample mean of samples at node p
r_p	Farthest distance from mean M_p

- Use two rules for eliminating examples for nearest neighbor computation.

Rule 1



- Rule 1: No sample x_i in S_p can be nearest neighbor to test point x if

$$B + r_p < d\,(X, M_p).$$

where B is the distance to the current nearest neighbor.

Proof:

For any point x_i in S_p if test point x is in the same node, by triangle inequality, we have

$$d(X, X_i) + d(X_i, M_p) \geq d(X, M_p)$$

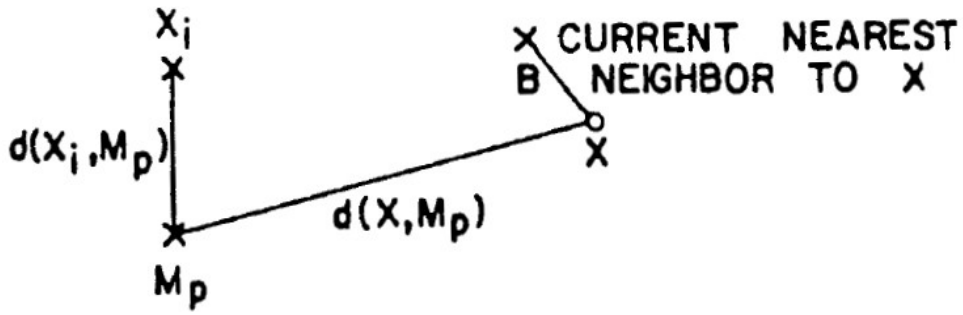
Since $d(x_i, M_p) < r_p$ by definition,

$$d(X, X_i) \geq d(X, M_p) - r_p.$$

Therefore x_i cannot be nearest neighbor to x if

$$d(X, X_i) \geq d(X, M_p) - r_p > B.$$

Rule 2



- Rule 2: A sample x_i in S_p cannot be a nearest neighbor to x if

$$B + d(X_i, M_p) < d(X, M_p),$$

Proof: On similar lines as the previous proof using triangle inequality.

This rule is applied only on the leaf nodes. The distances $d(x_i, M_p)$ are computed for all the points in the leaf nodes; only requires additional storage of n real numbers.

Branch and Bound Search Algorithm

1. Set B = ∞. Current level L = 1, and Node = 0.
2. *Expansion:* Place all nodes that are immediate successors of current node in the *active set of nodes.*
3. For each node, test Rule 1. Remove nodes that fail.
4. Pick the closest node to the test point *x* and expand using step 2. If you are at the final level, go to step 5.
5. Apply Rule 2 to all the points in the last node.
    1. For all the points that fail, do not compute the distance to *x.*
    2. For points that succeed, compute the distances.
6. Pick the closest point using the computed distances.

Extension to k-Nearest Neighbors

- The same algorithm may be applied with *B* representing the distance to the *k-th* closest point in the dataset.
- When a distance is actually calculated in Step 5, it is compared against distances to its current k-nearest neighbors, and the current k-nearest neighbor table is updated as necessary.