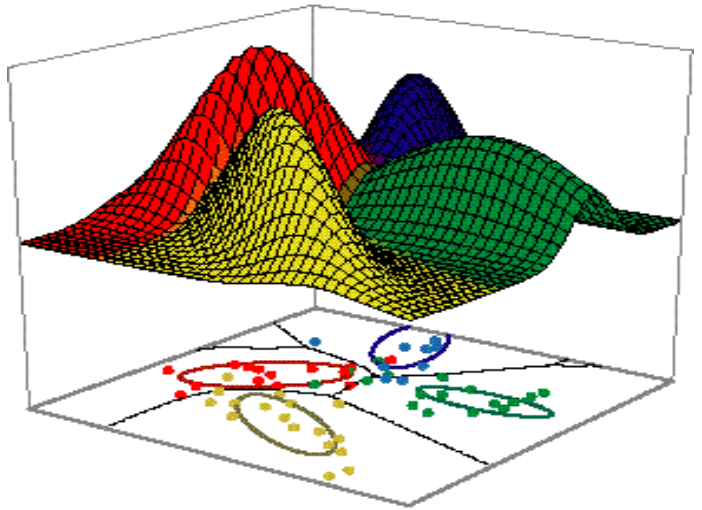
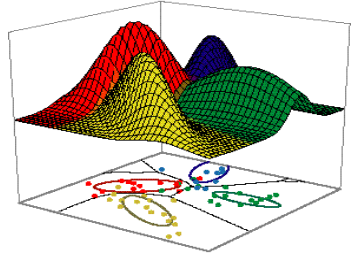
Part 7:
Nonparametric Techniques



Intro to Nonparametric Density Estimation
Parzen Windows Method
Probabilistic Neural Networks
K-NN Density Estimation
(K) Nearest Neighbour Decision Rule

Some materials in these slides were taken from Pattern Classification (2nd ed) by R. O. Duda, P. E. Hart and D. G. Stork, John Wiley & Sons, 2000, **Chapter 4.1-4.5.**

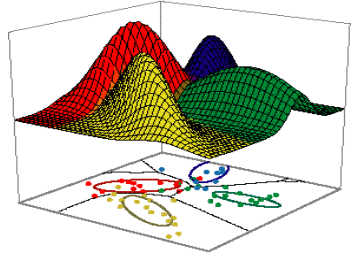
Nonparametric Techniques



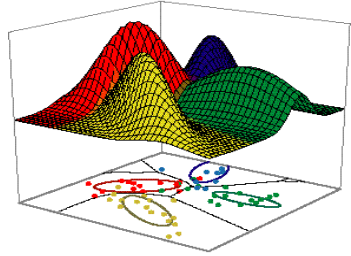
(Chapter 4.1-4.5)

- Limitations of parameter estimation:
  - Assumed we knew the form of the distribution – not always the case.
  - All (classic) parametric densities are unimodal (have a single local maximum), whereas many practical problems involve multi-modal densities
- Nonparametric procedures can be used with arbitrary distributions and without the assumption that the forms of the underlying densities are known

Nonparametric Techniques



- There are two types of nonparametric methods:
- 1) Nonparametric density estimation
  - Estimate P*(x* | ω_j *)*
    - *e.g., Parzen windows, K-NN estimation*
  - Then apply Bayes' rule for classification
- 2) Nonparametric decision rules
  - Bypass probability and go directly to posterior probability estimation
    - e.g., Nearest neighbour / K-NN decision rules



Nonparametric Density Estimation

- Basic idea:
  - Probability that a vector x will fall in region R is:

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

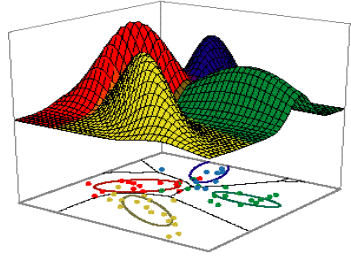
  - P is a smoothed (or averaged) version of the density function p(x)
  - If we draw an IID sample of size n, the probability that k of n points fall within region R is then:

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

and the expected value for k is:

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

Nonparametric Density Estimation



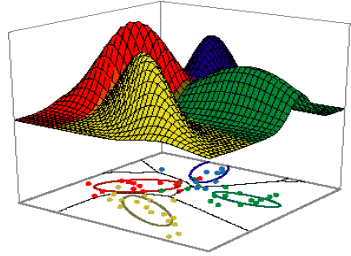
- ML estimation of $P = \theta$

  $\underset{\theta}{Max}(P_k \mid \theta)$ 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).
- *Assume that p(x)* is continuous and that the region $\mathcal{R}$ is so small that p does not vary significantly within it (i.e., since *p(x)* = constant, it is not a part of the sum), we can write:

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

where x' is a point within $\mathcal{R}$ and V the volume enclosed by $\mathcal{R}$.



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

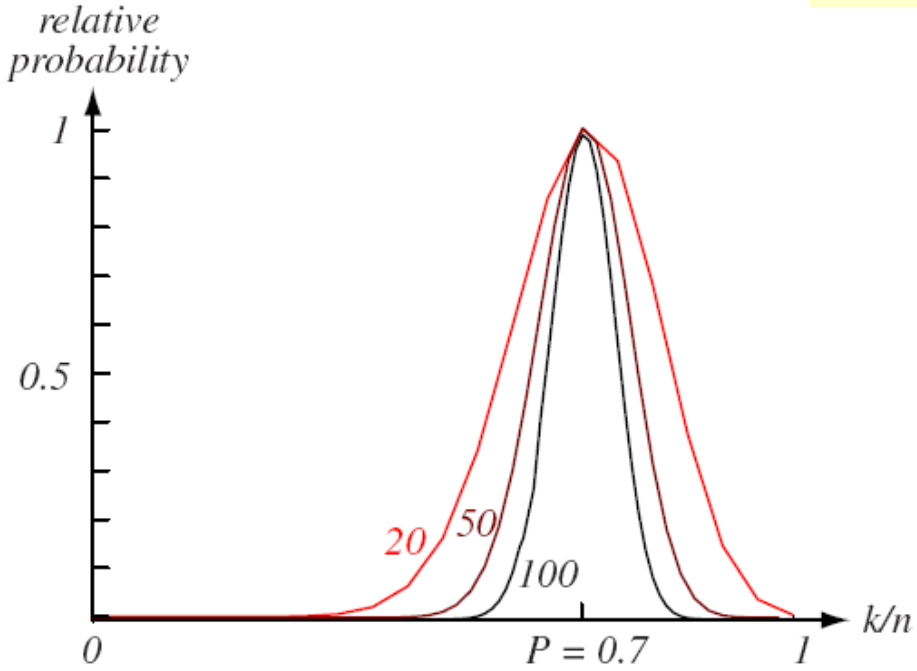
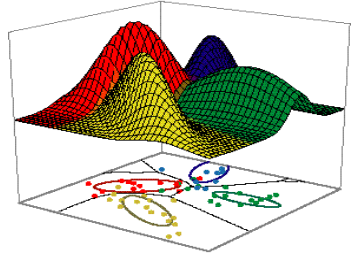


FIGURE 4.1. The relative probability an estimate given by Eq. 4 will yield a particular value for the probability density, here where the true probability was chosen to be 0.7. Each curve is labeled by the total number of patterns n sampled, and is scaled to give the same maximum (at the true probability). 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.

Nonparametric Density Estimation



- Condition for convergence
  - The fraction $\frac{k}{nV}$ is a space-averaged value of *p(x).*
    - *p(x)* is obtained only if V approaches zero.
  - Problem 1: for fixed n, as V→0, so does k:

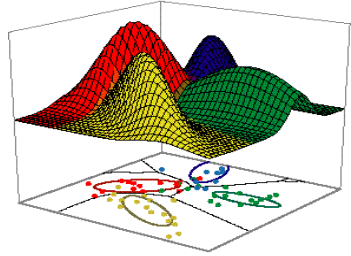
$$\lim_{V \to 0,\, \mathrm{k}=0} p(x) = 0 \quad (\text{if n} = \text{fixed})$$

    - This is the case where no samples are included in *R*.
      - It is an uninteresting case!
  - Problem 2: if *R* does contain a single point, then:

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

    - In this case, the estimate diverges.
      - Also an uninteresting case!

Nonparametric Density Estimation



- The volume V needs to approach 0 if we want to use this estimation
  - 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
  - One will have to accept a certain amount of space-averaging of p(x)

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

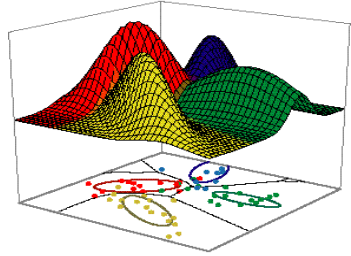
    To estimate the density of x, we form a sequence of regions

    $R_1, R_2, \ldots$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 $p(x)$:

$$p_n(x) = \frac{k_n / n}{V_n} = \frac{k_n}{nV_n} \quad (7)$$

Nonparametric Density Estimation



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

$$1)\ \lim_{n\to\infty} V_n = 0 \qquad \text{P/V}\rightarrow\text{p}$$

$$2)\ \lim_{n\to\infty} k_n = \infty \qquad k_n/n\rightarrow p$$

$$3)\ \lim_{n\to\infty} k_n / n = 0 \qquad \text{convergence of (7)}$$

- 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

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

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"

Nonparametric Density Estimation

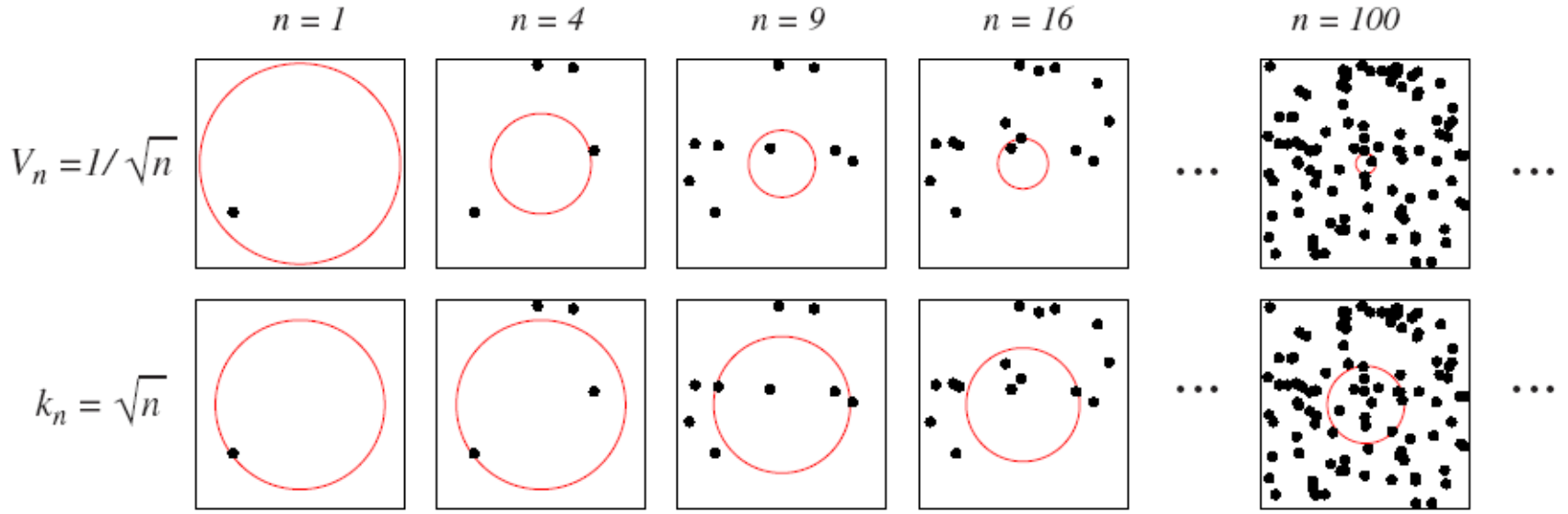
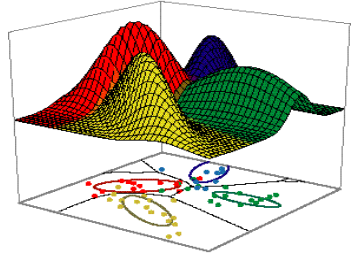


FIGURE 4.2. There are two leading methods for estimating the density at a point, here at the center of each square. The one shown in the top row is to start with a large volume centered on the test point and shrink it according to a function such as $V_n = 1/\sqrt{n}$. The other method, shown in the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



- First assume that the region R_n is a d-dimensional hypercube:

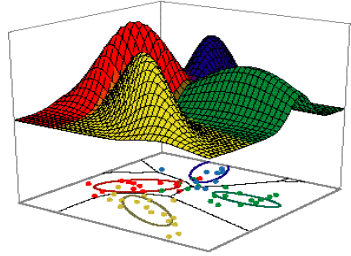
$$V_n = h_n^d \quad (\mathrm{h_n}: \text{length of the edge of } \Re_n)$$

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

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

- The function $\varphi\left(\frac{x - x_i}{h_n}\right)$ is equal to 1 if x_i falls within the hypercube of volume V_n centered at x and equal to 0 otherwise.

Parzen Windows



- The number of samples that fall inside this hypercube is:

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

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

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

- $P_n(x)$ estimates $p(x)$ as an average of n functions of x and the samples (x_i) $(i = 1, \ldots, n)$.
  - Window function is being used for interpolation
    - Each sample contributes to the estimate depending on its dist from x
  - These functions φ can be general…
  - Require that φ be a density function such that p_n will be a density
    - φ ≥ 0 and integrates to 1

Parzen Windows

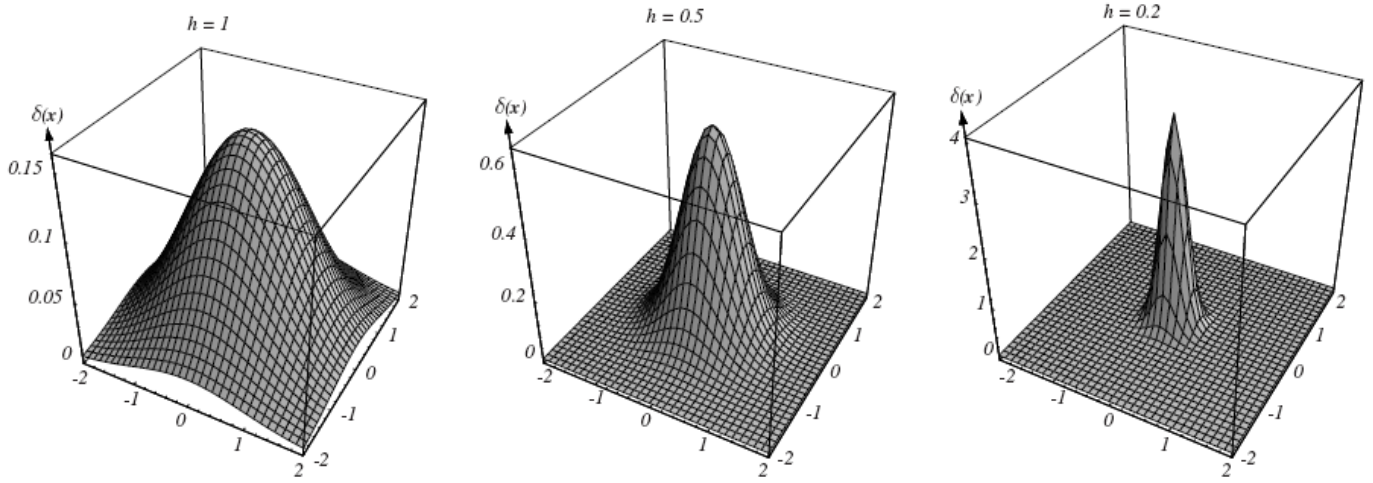


FIGURE 4.3. Examples of two-dimensional circularly symmetric normal Parzen windows for three different values of h. Note that because the $\delta(\mathbf{x})$ are normalized, different vertical scales must be used to show their structure. From: Richard O. Duda, Peter E.

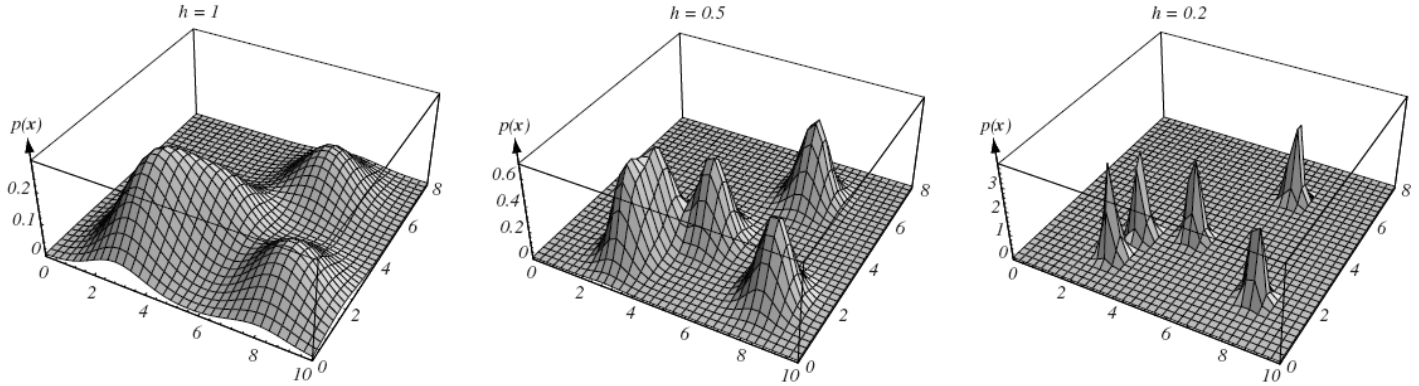
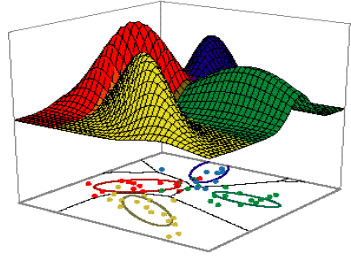


FIGURE 4.4. Three Parzen-window density estimates based on the same set of five samples, using the window functions in Fig. 4.3. As before, the vertical axes have been scaled to show the structure of each distribution. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Parzen Windows – An illustration



- The behavior of the Parzen-window method

  - Case where *p(x)* →*N(0,1)*

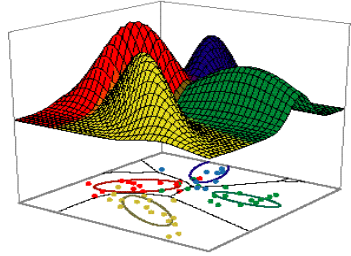
Let $$\varphi(u)=\frac{1}{\sqrt{2\pi}}e^{\left(\frac{-1}{2}u^2\right)} \quad \text{and} \quad h_n=\frac{h_1}{\sqrt{n}}\ ,(n>1)$$

(h_1: known parameter)

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

is an average of normal densities centered at the samples x_i.

Parzen Windows – An illustration

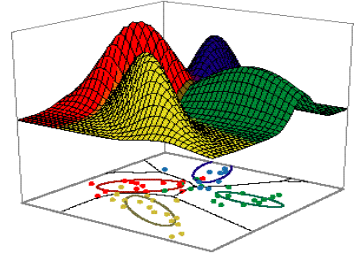


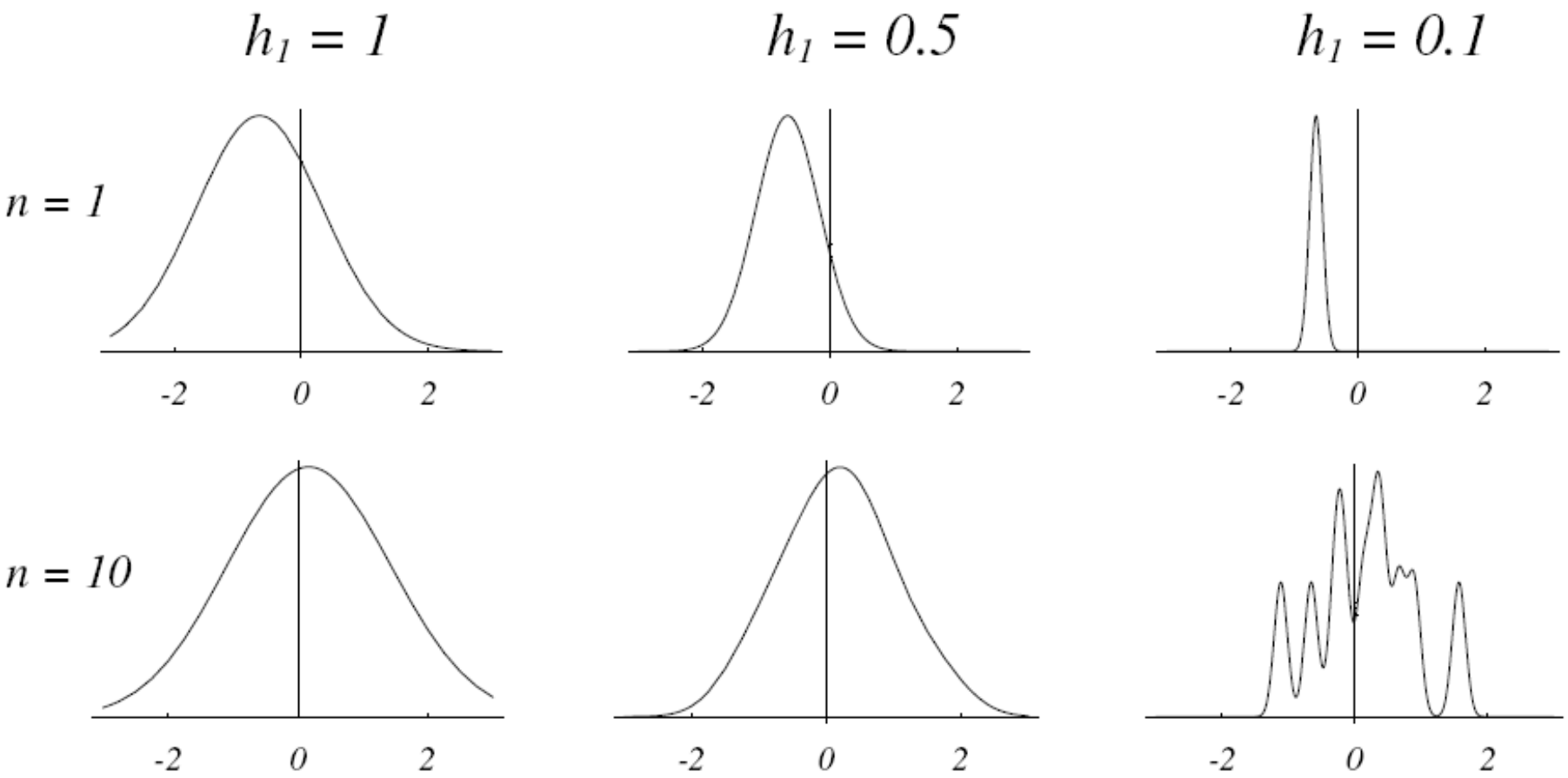
- Effect of h_n
  - h_n too large → blurred estimate
  - h_n too small → peaky estimate
- Numerical results:
  - For $n = 1$ and $h_1 = 1$

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

For $n = 10$ and $h_1 = 0.1$, the contributions of the individual samples are clearly observable !

Parzen Windows – An illustration





Parzen Windows – An illustration

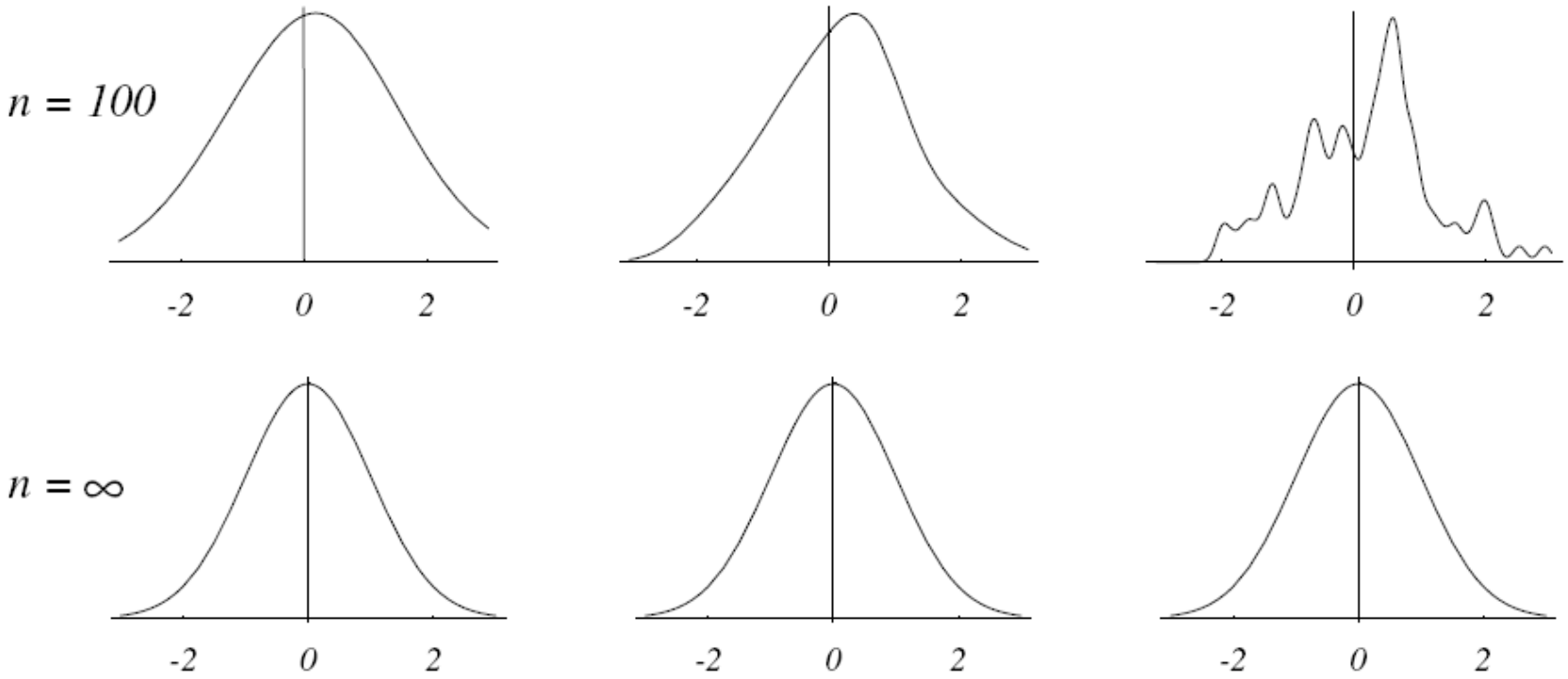
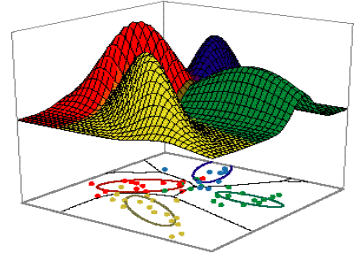
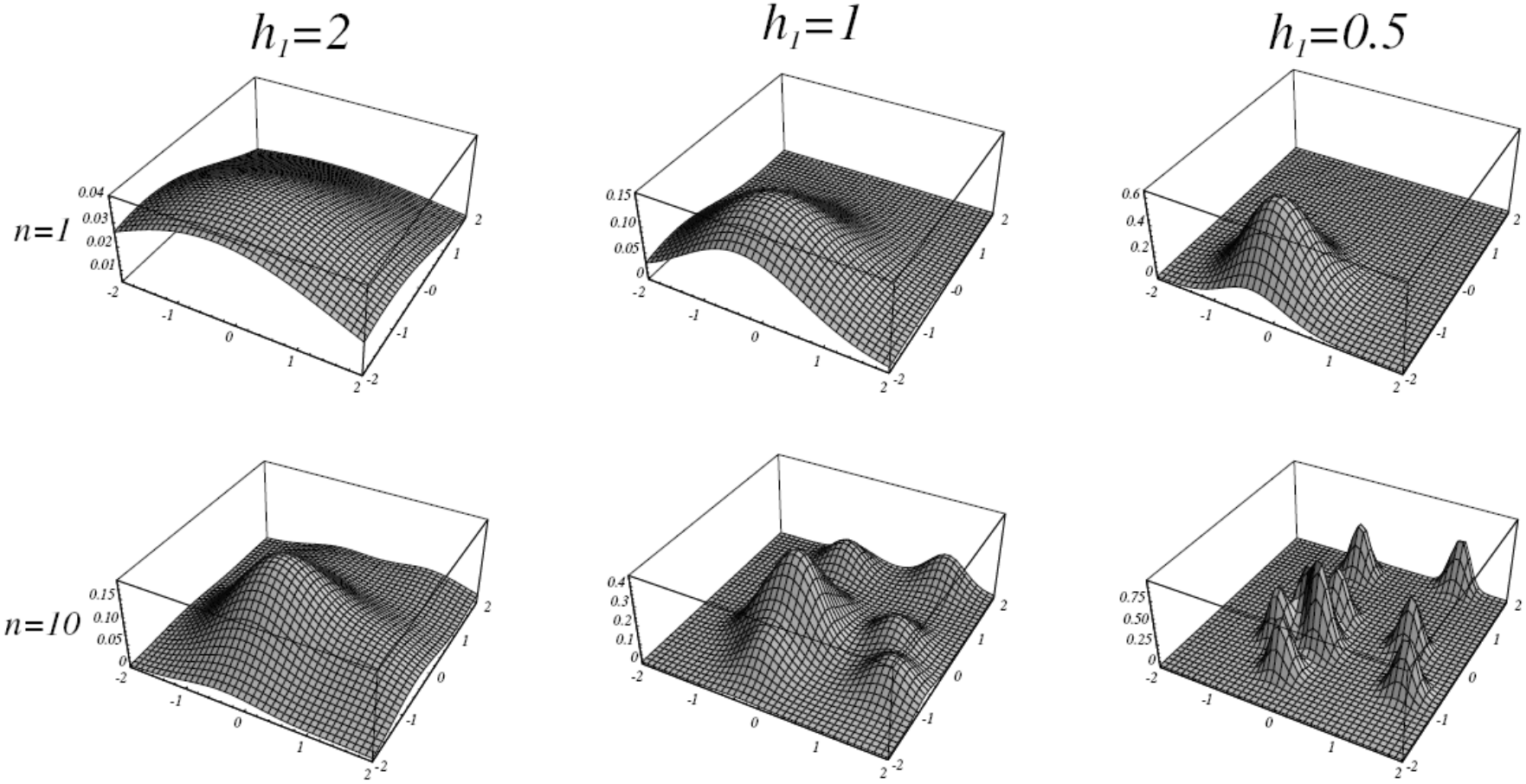


FIGURE 4.5. Parzen-window estimates of a univariate normal density using different window widths and numbers of samples. The vertical axes have been scaled to best show the structure in each graph. Note particularly that the $n = \infty$ estimates are the same (and match the true density function), regardless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

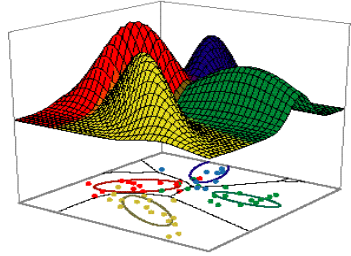
Parzen Windows – An illustration



Analogous results are also obtained in two dimensions as illustrated:



Parzen Windows – An illustration



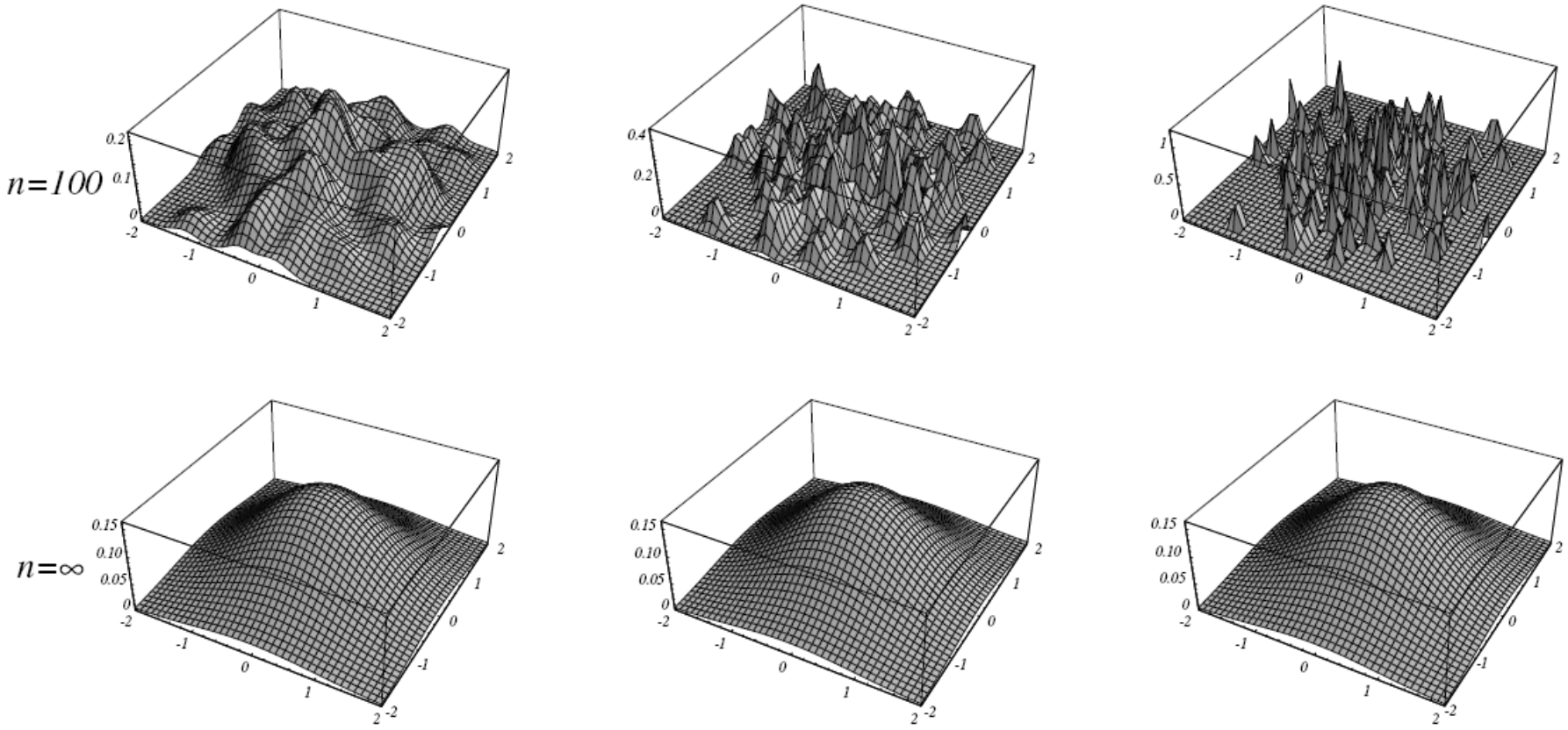
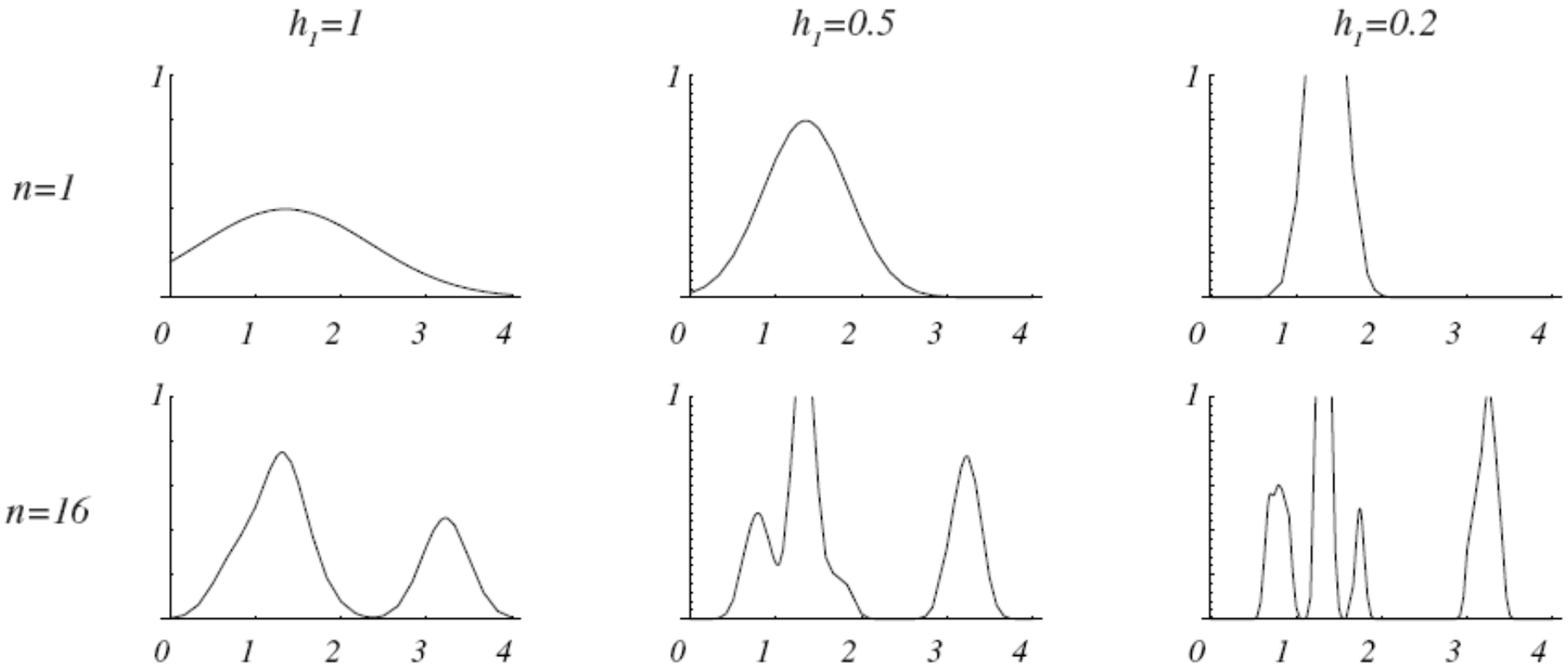


FIGURE 4.6. Parzen-window estimates of a bivariate normal density using different window widths and numbers of samples. The vertical axes have been scaled to best show the structure in each graph. Note particularly that the $n = \infty$ estimates are the same (and match the true distribution), regardless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

How to choose h? Can look at consistency between LOO density estimates…

Parzen Windows – Example 2

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



Parzen Windows – Example 2

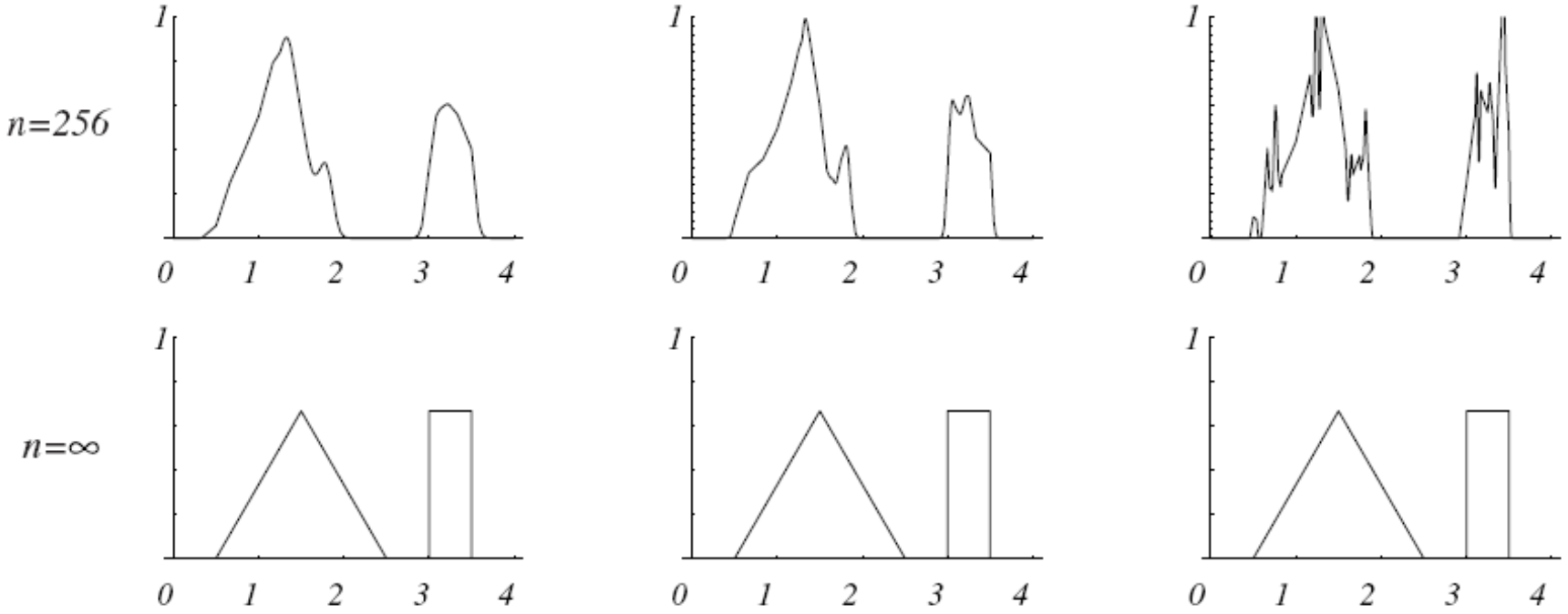
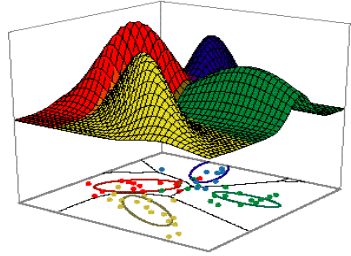


FIGURE 4.7. Parzen-window estimates of a bimodal distribution using different window widths and numbers of samples. Note particularly that the $n = \infty$ estimates are the same (and match the true distribution), regardless of window width. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern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

Parzen Windows – 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.

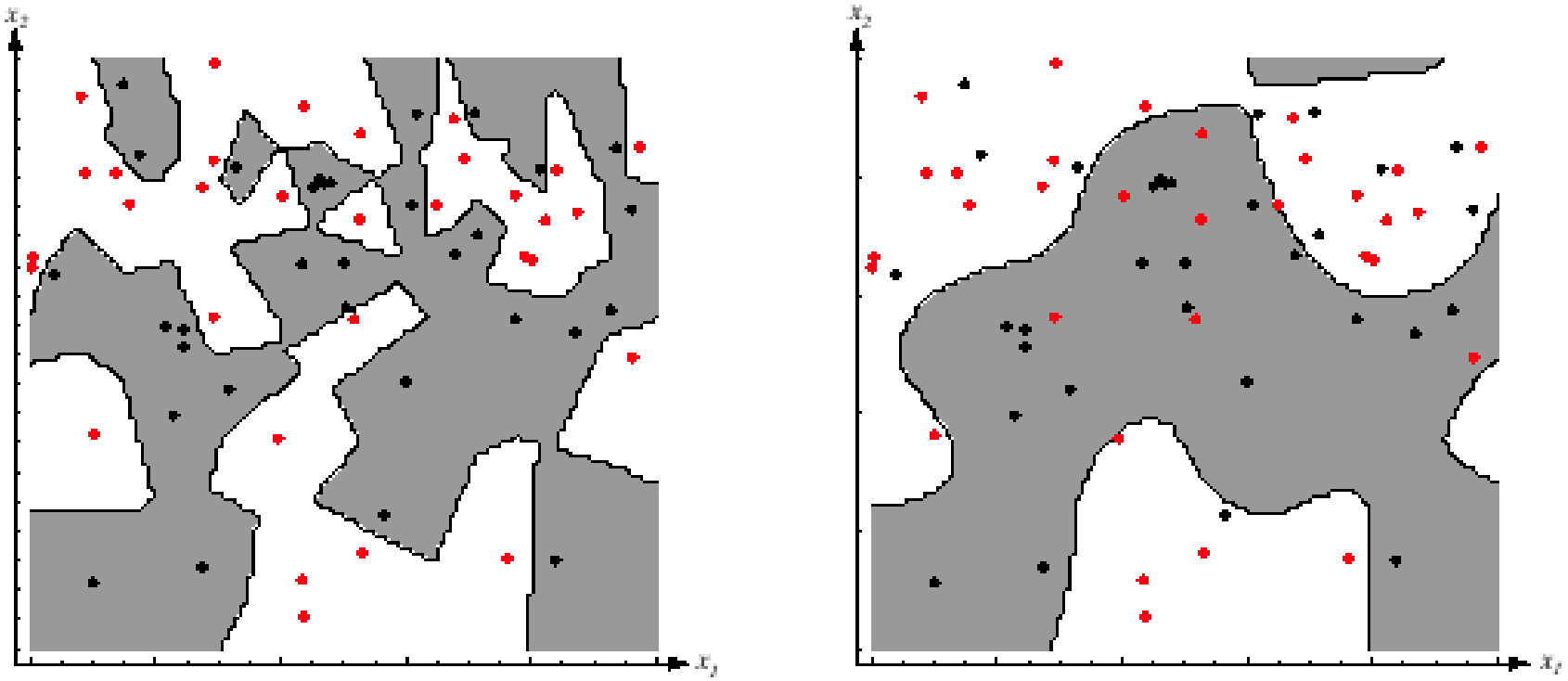
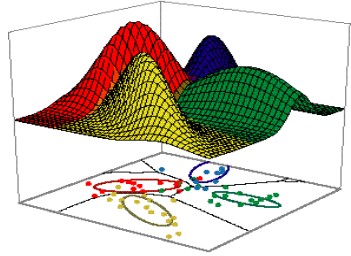
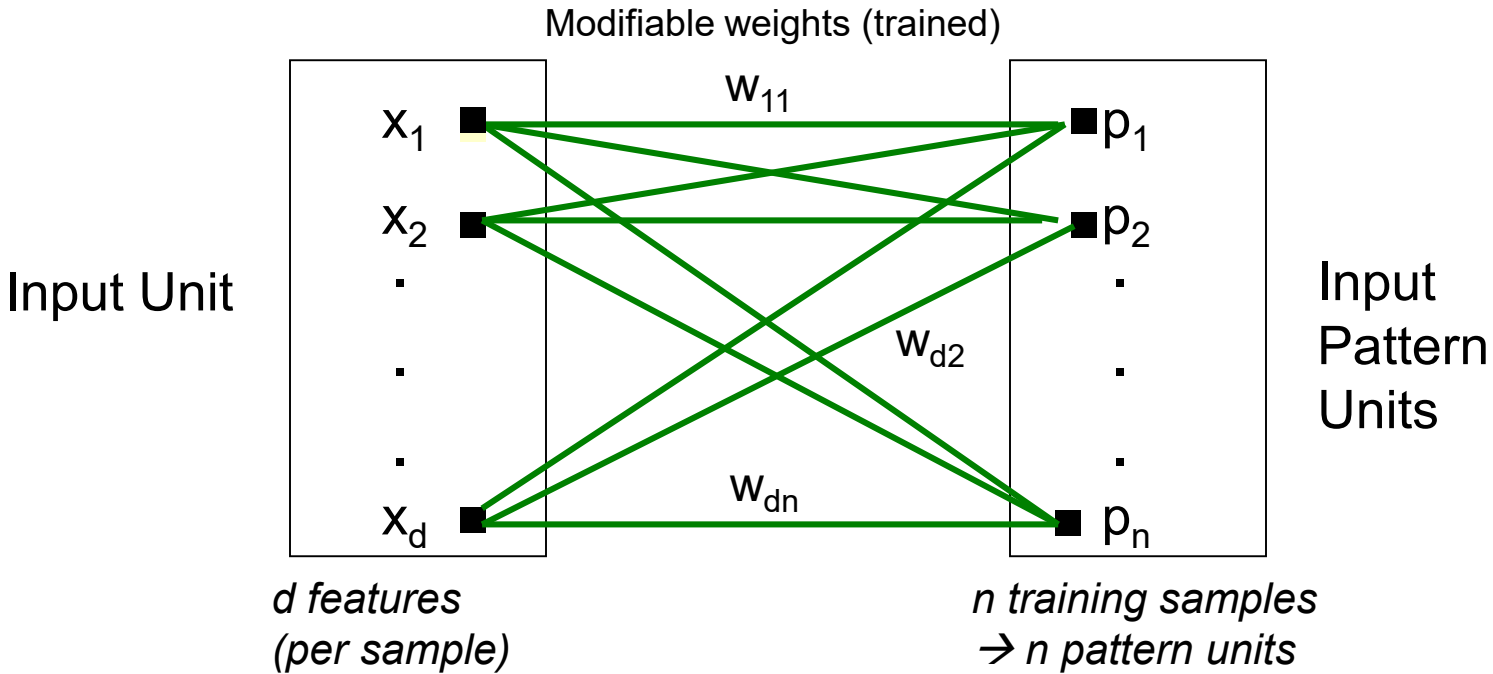


FIGURE 4.8. The decision boundaries in a two-dimensional Parzen-window dichotomizer depend on the window width h. At the left a small h leads to boundaries that are more complicated than for large h on same data set, shown at the right. Apparently, for these data a small h would be appropriate for the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

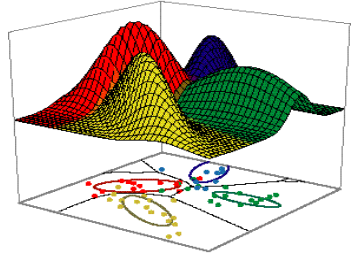
Probabilistic Neural Networks

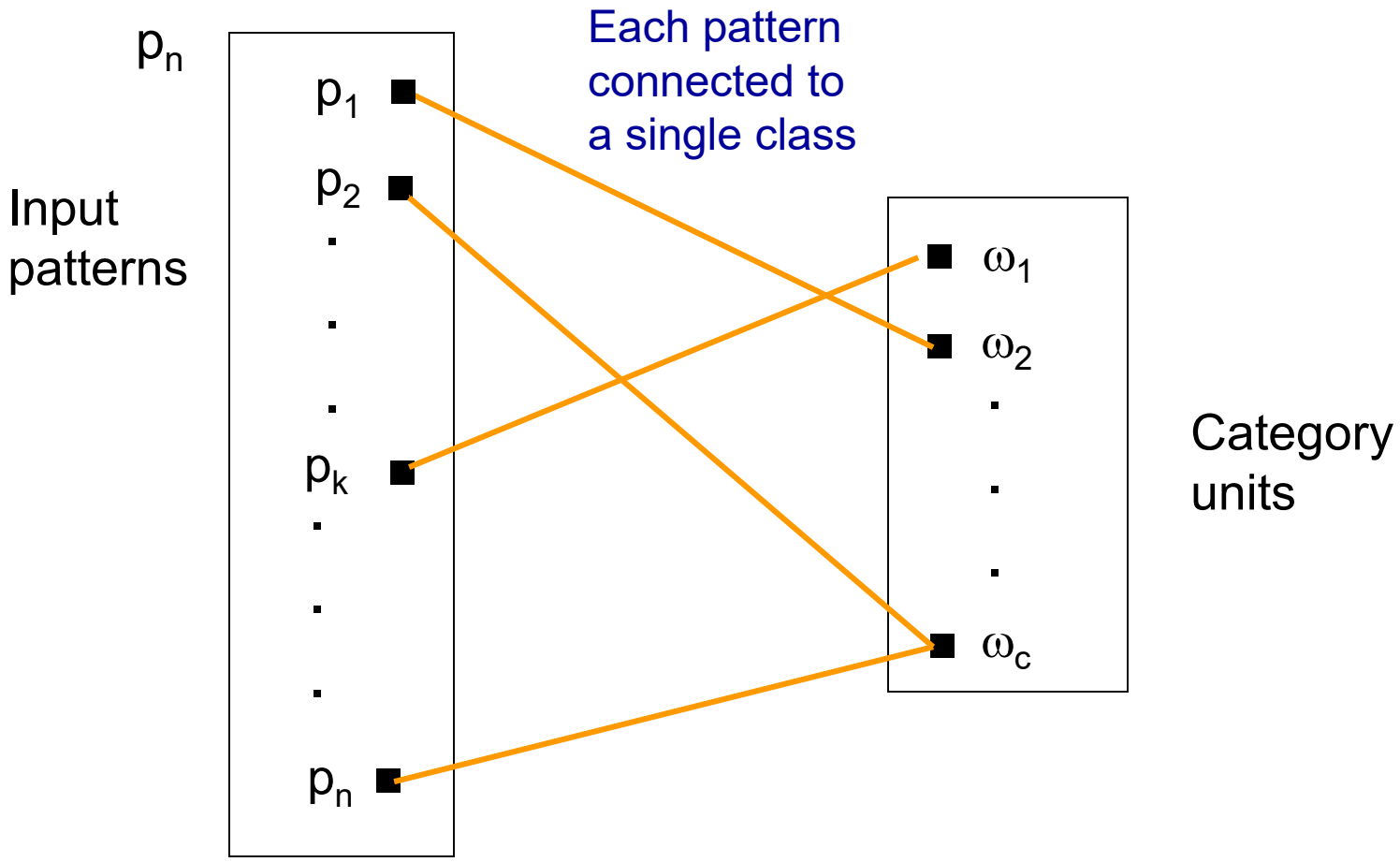


- PNN is a parallel implementation of Parzen windows.
  - Compute a Parzen estimate based on *n* patterns
  - n patterns with d features sampled from *c* classes
  - All input units are connected to all *n* patterns

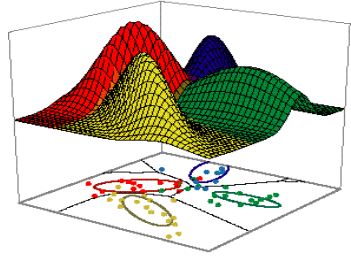


Probabilistic Neural Networks





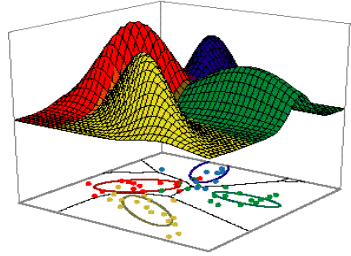
Probabilistic Neural Networks

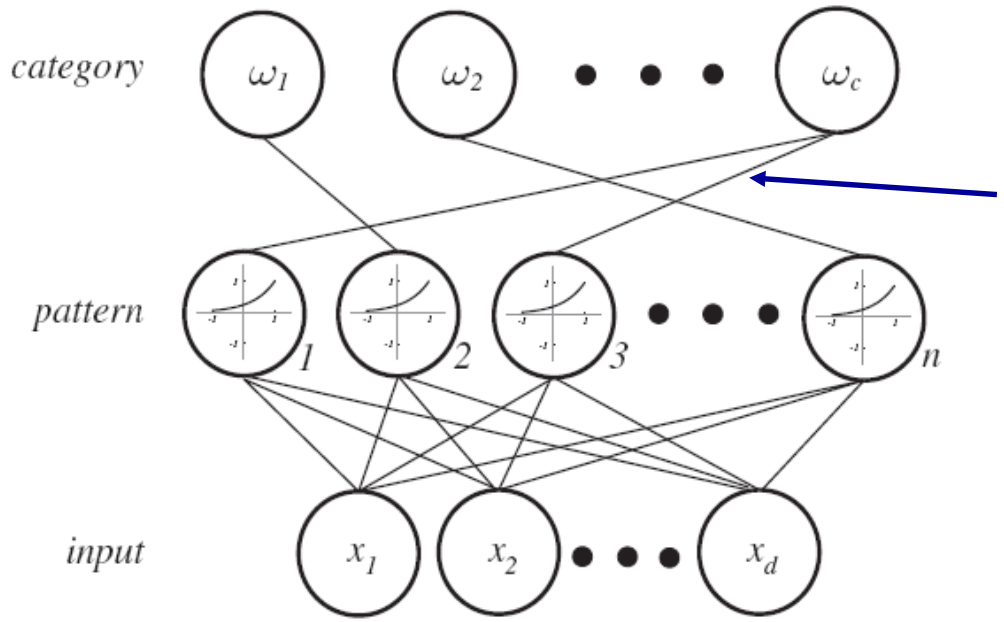


- Training the network
  1. Normalize each pattern x of the training set to 1
  2. Place the first training pattern on the input units
  3. Set the weights linking the input units and the first pattern units such that: $w_{i1} = x_{i1}$, $i=1..d$
  4. Make a single connection from the first pattern unit to the category unit corresponding to the known class of that pattern
  5. Repeat the process for all remaining training patterns by setting the weights such that $w_{ik} = x_{ik}$ $(i=1..d;\ k = 1..n)$

  We finally obtain the following network…

Probabilistic Neural Networks

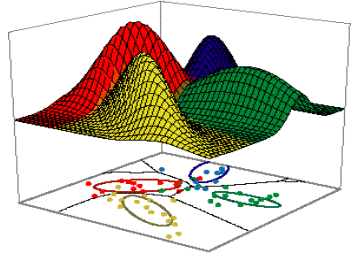




Value represents probability that test point came from a Gaussian centered on training pattern 3
(which was from class c)

FIGURE 4.9. A probabilistic neural network (PNN) consists of d input units, n pattern units, and c category units. Each pattern unit forms the inner product of its weight vector and the normalized pattern vector $\mathbf{x}$ to form $z = \mathbf{w}^t\mathbf{x}$, and then it emits $\exp[(z-1)/\sigma^2]$. Each category unit sums such contributions from the pattern unit connected to it. This ensures that the activity in each of the category units represents the Parzen-window density estimate using a circularly symmetric Gaussian window of covariance $\sigma^2\mathbf{I}$, where $\mathbf{I}$ is the $d \times d$ identity matrix. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Probabilistic Neural Networks



- Testing the network
  1. Normalize the test pattern x and place it at the input units
  2. Each pattern unit computes the inner product of the weight vector and the input x in order to yield the net activation

$$net_k = w_k^t x$$

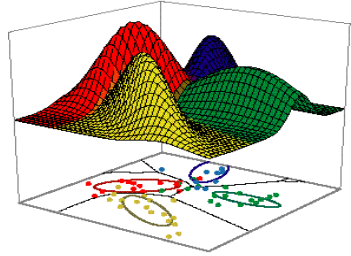
and emit a nonlinear function $f(net_k) = exp\left[\frac{net_k - 1}{\sigma^2}\right]$

  3. Each output unit sums the contributions from all pattern units connected to it

$$P_n(x \mid \omega_j) = \sum_{i=1}^{n} \varphi_i \propto P(\omega_j \mid x)$$

  4. Classify by selecting the maximum value of $P_n(x \mid \omega_j)$ $(j = 1, \ldots, c)$

k_n-Nearest-Neighbour Estimation



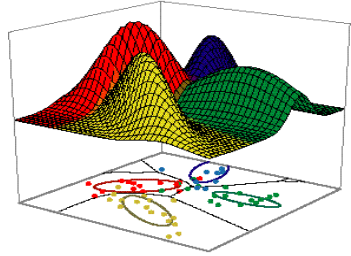
- Goal: a solution for the problem of the unknown "best" window function
  - Let the cell volume be a function of the training data
  - Center a cell about x and let it grow until it captures k_n samples $(k_n = f(n))$
  - k_n are called the k_n nearest-neighbors of x

2 possibilities can occur:

- Density is high near x; therefore the cell will be small which provides a good resolution
- Density is low; therefore the cell will grow large and stop until higher density regions are reached

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

k_n-Nearest-Neighbour Estimation



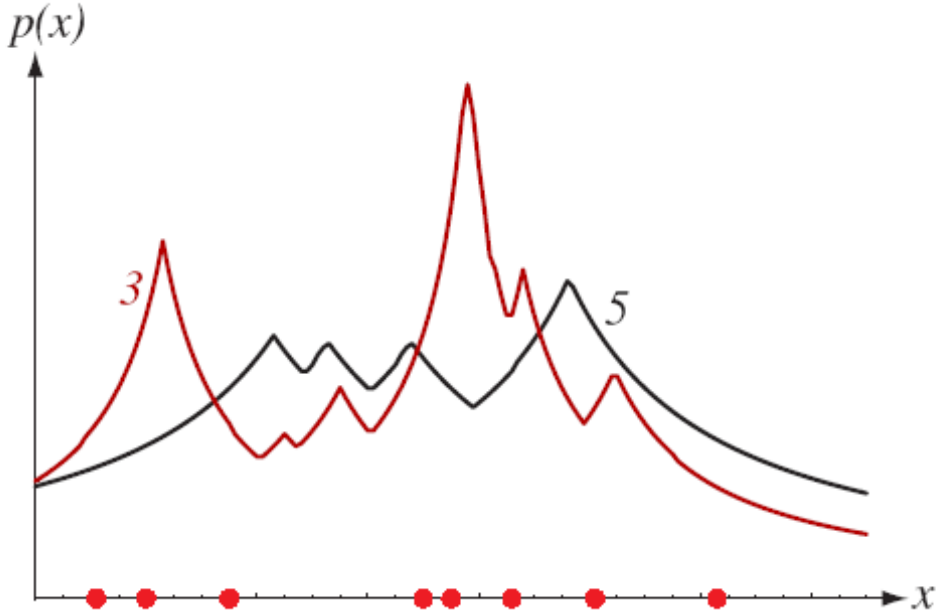
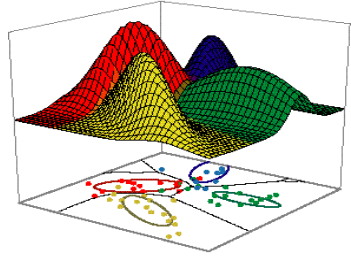


FIGURE 4.10. Eight points in one dimension and the *k*-nearest-neighbor density estimates, for $k = 3$ and 5. Note especially that the discontinuities in the slopes in the estimates generally lie *away* from the positions of the prototype points. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

k-Nearest-Neighbour Estimation

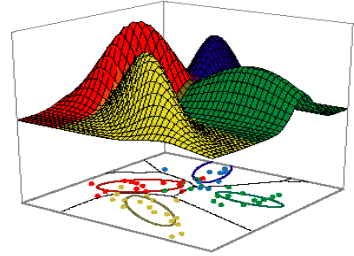


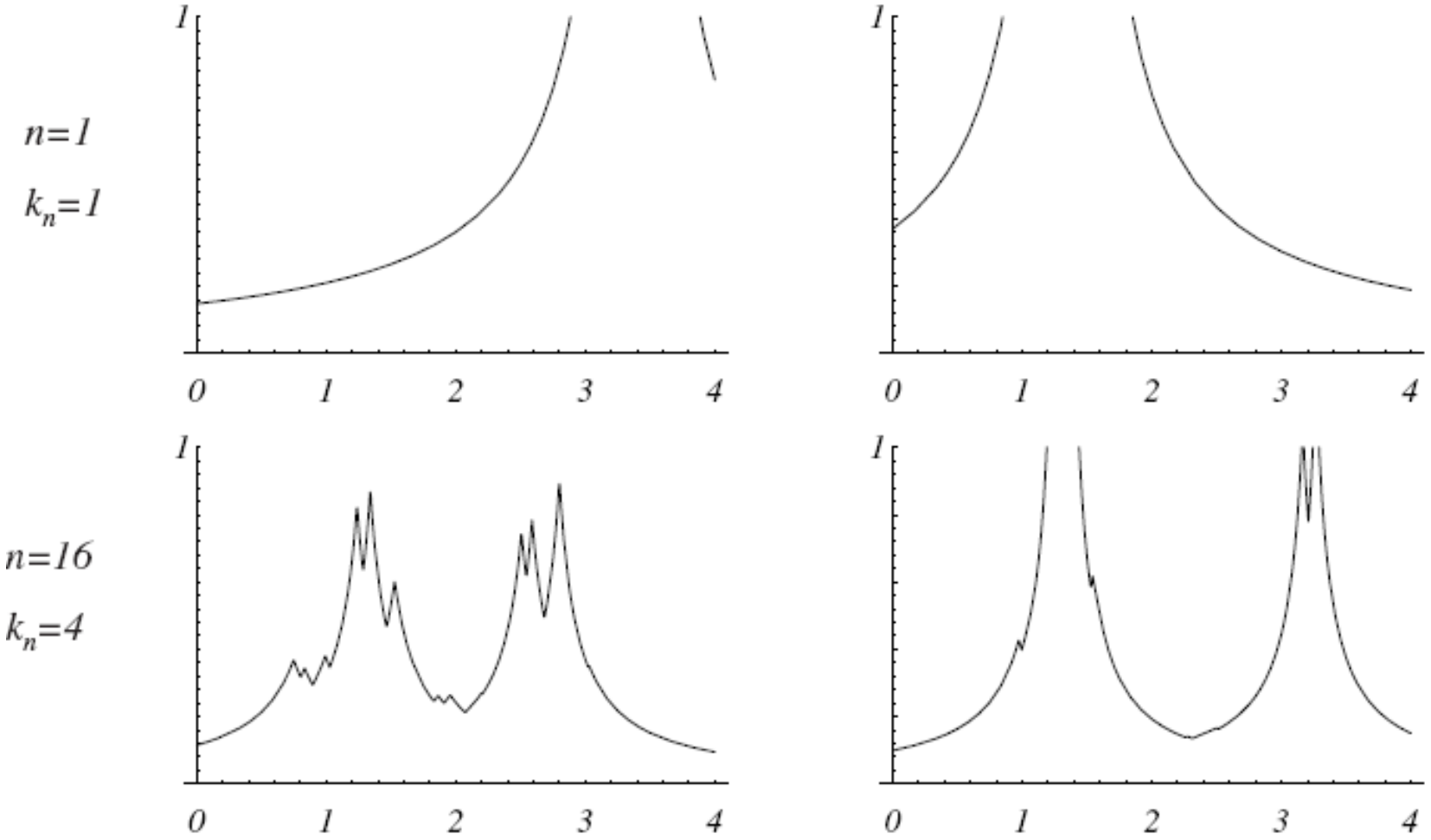
Illustration

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

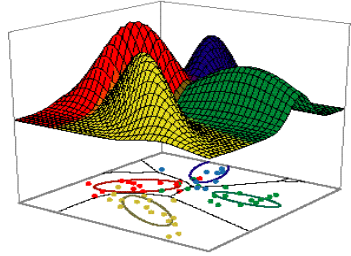
$$p_n(x) = \frac{k_n / n}{V_n} = \frac{1}{V_1} = \frac{1}{2|x - x_1|}$$

k-Nearest-Neighbour Estimation





k-Nearest-Neighbour Estimation



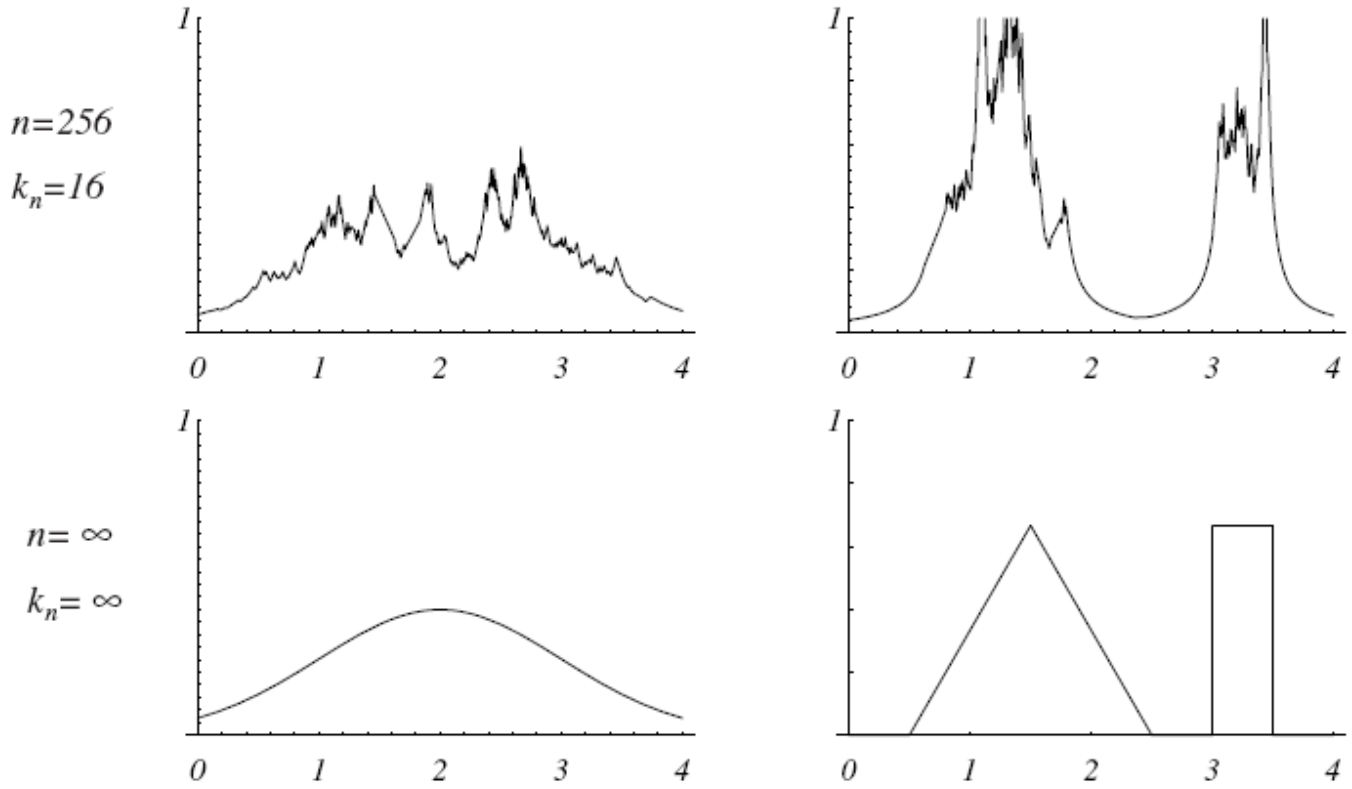
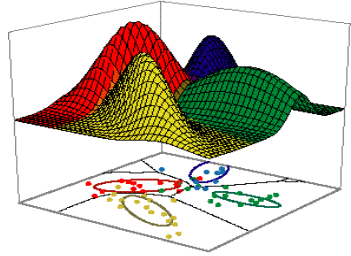


FIGURE 4.12. Several *k*-nearest-neighbor estimates of two unidimensional densities: a Gaussian and a bimodal distribution. Notice how the finite *n* estimates can be quite "spiky." From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons,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

k-Nearest-Neighbour Estimation



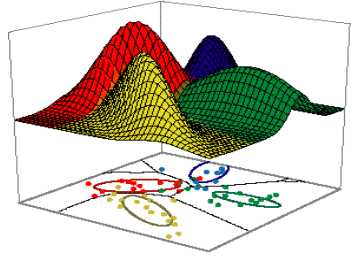
- Estimation of a-posteriori probabilities
  - Goal: estimate $P(\omega_i \mid x)$ from a set of n labeled samples
    - Let's place a cell of volume V around x and capture k samples
    - k_i samples amongst k turned out to be labeled ω_i then:

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

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

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

k-Nearest-Neighbour Estimation

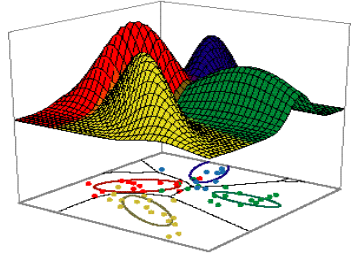


- $p_n(\omega_i \mid x) = \frac{k_i}{k}$ is the fraction of the samples within the cell that are labeled ω_i

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

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

The Nearest-Neighbour 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 demonstrated!)
- If $n \rightarrow \infty$, it is always possible to find x' sufficiently close so that:

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

The Nearest-Neighbour Rule

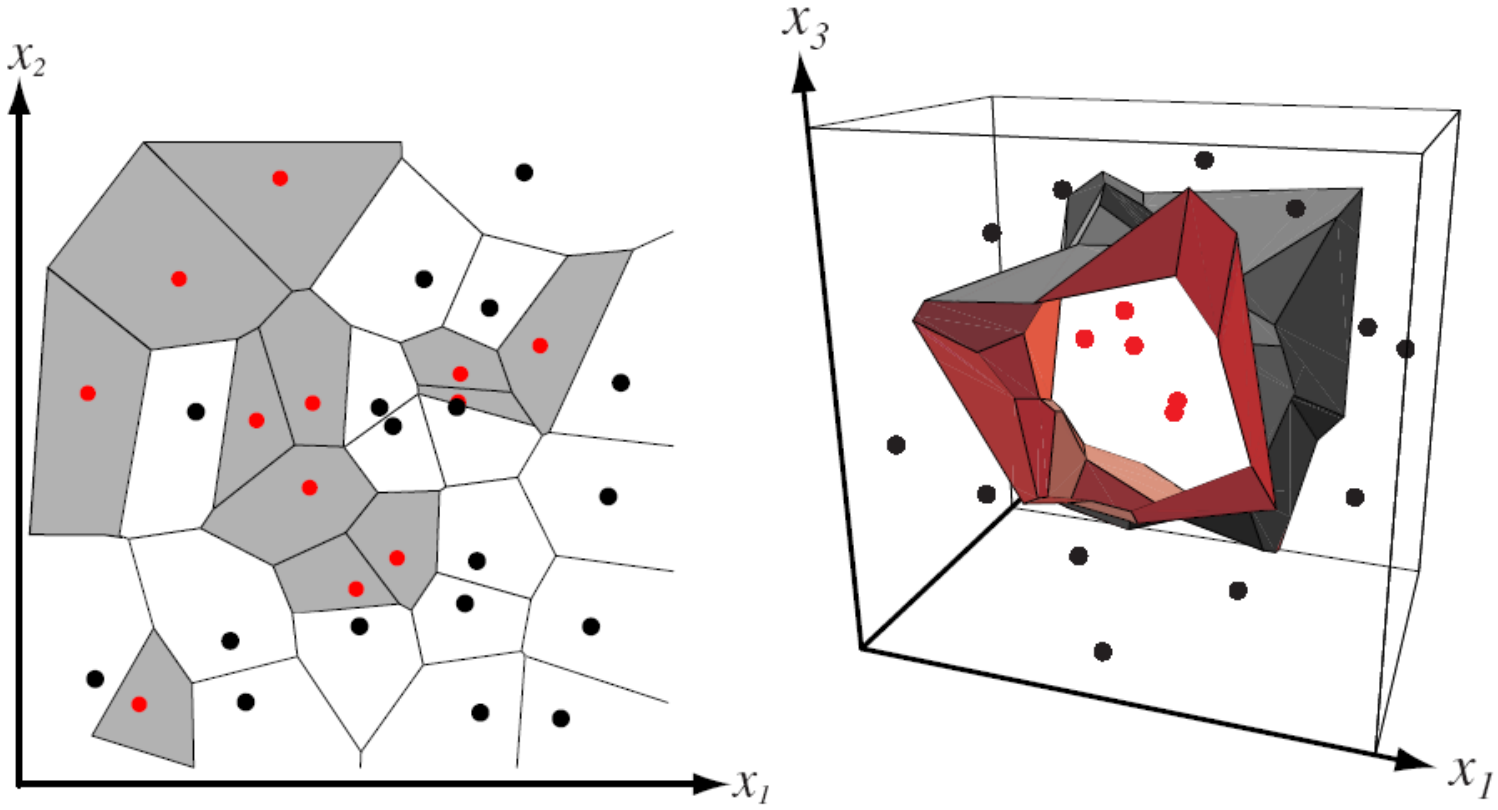
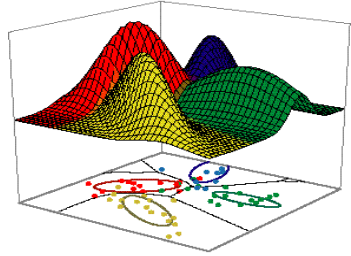


FIGURE 4.13. In two dimensions, the nearest-neighbor algorithm leads to a partitioning of the input space into Voronoi cells, each labeled by the category of the training point it contains. In three dimensions, the cells are three-dimensional, and the decision boundary resembles the surface of a crystal.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

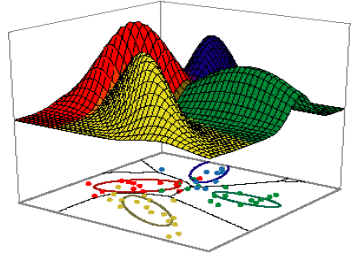
The k-Nearest-Neighbour Rule



- Goal: Classify x by assigning it the label most frequently represented among the k nearest samples and use a voting scheme
- Follows directly from k-NN estimation of posterior probability

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

The k-Nearest-Neighbour Rule



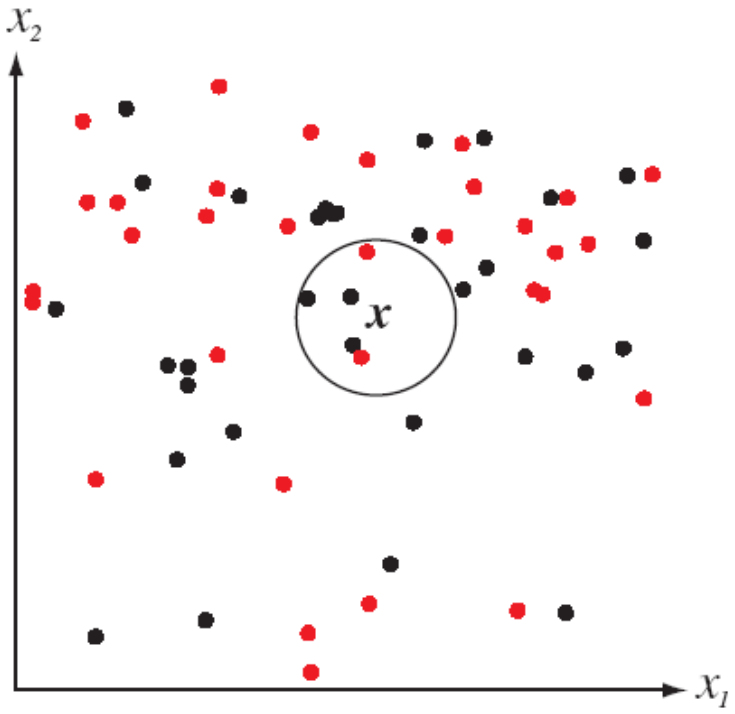
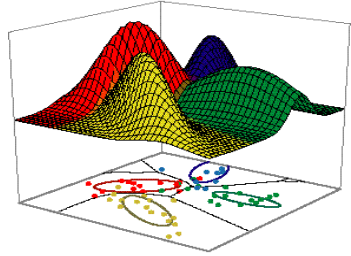


FIGURE 4.15. The *k*-nearest-neighbor query starts at the test point **x** and grows a spherical region until it encloses *k* training samples, and it labels the test point by a majority vote of these samples. In this $k = 5$ case, the test point **x** would be labeled the category of the black points.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

The k-Nearest-Neighbour Rule



Example:

$k = 3$ (odd value) and $x = (0.10, 0.25)^t$

Prototypes	Labels
(0.15, 0.35)	ω_1
(0.10, 0.28)	ω_2
(0.09, 0.30)	ω_1
(0.12, 0.20)	ω_2

Closest vectors to x with their labels are:

$\{(0.10, 0.28, \omega_2); (0.12, 0.20, \omega_2); (0.15, 0.35, \omega_1)\}$

One voting scheme assigns the label ω_2 to x since ω_2 is the most frequently represented