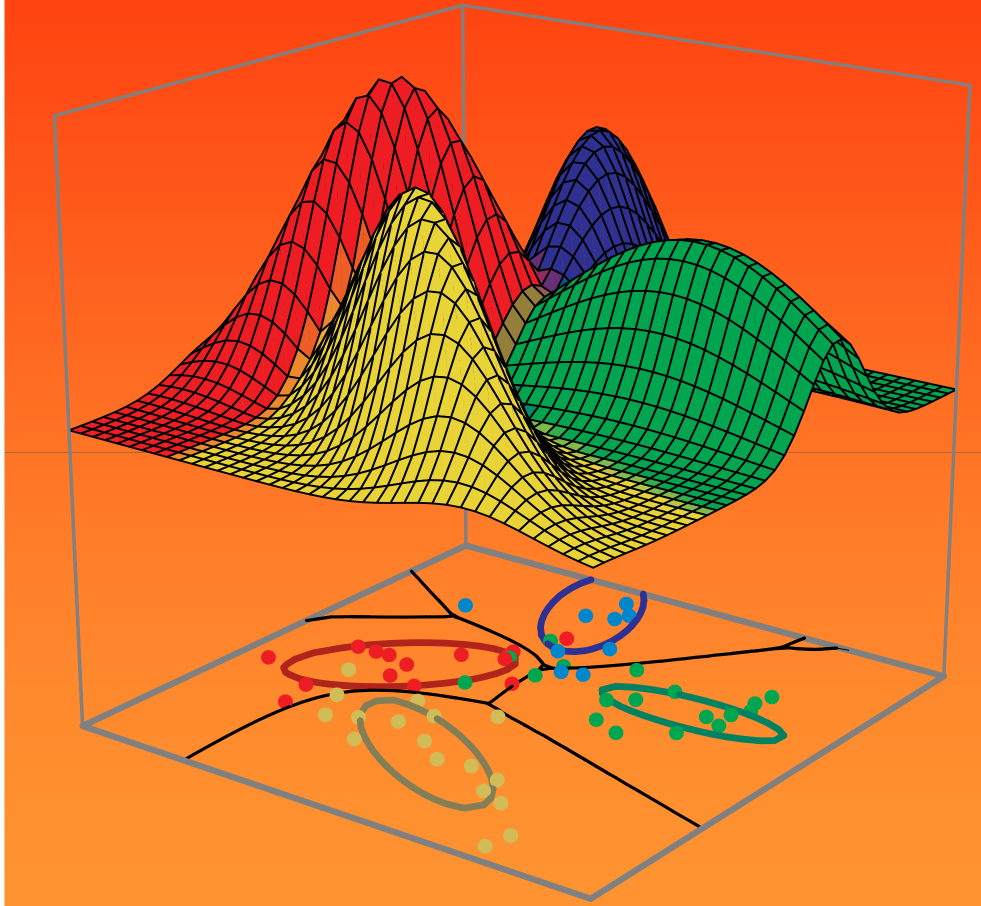


Pattern Classification

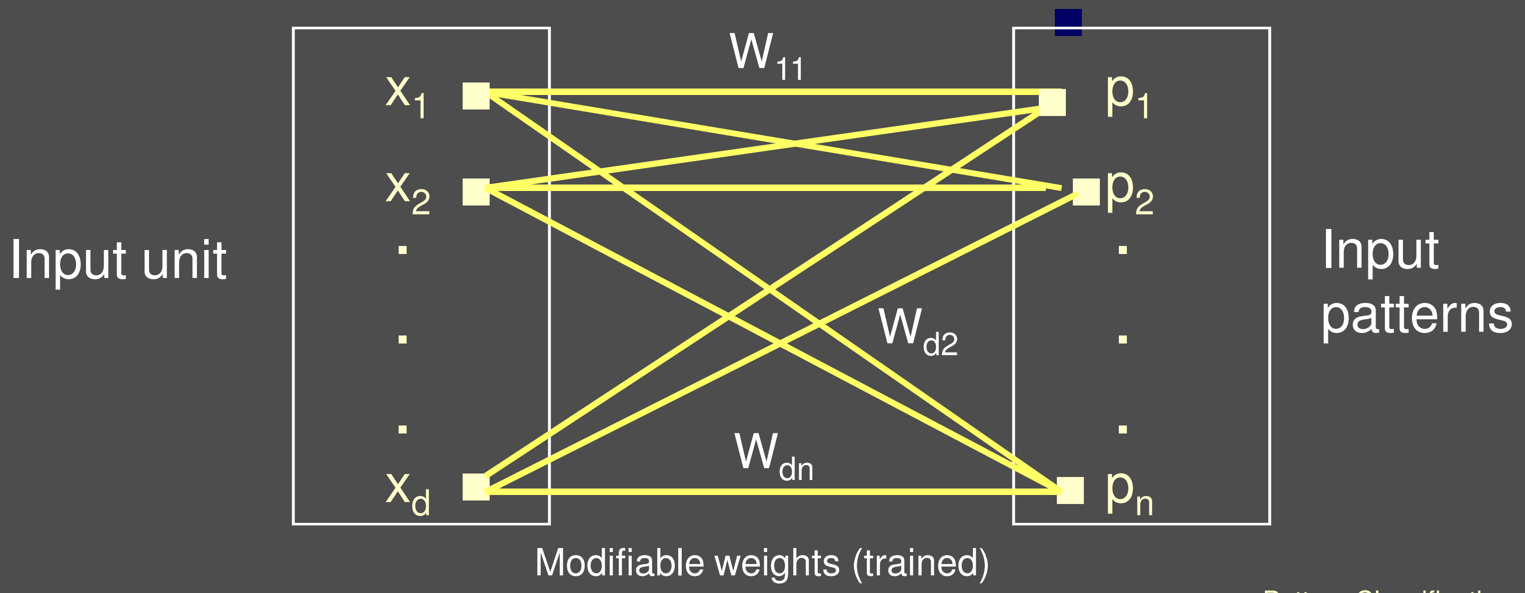
All materials in these slides were taken from *Pattern Classification (2nd ed) by R. O. Duda, P. E. Hart and D. G. Stork, John Wiley & Sons, 2000*
with the permission of the authors and the publisher

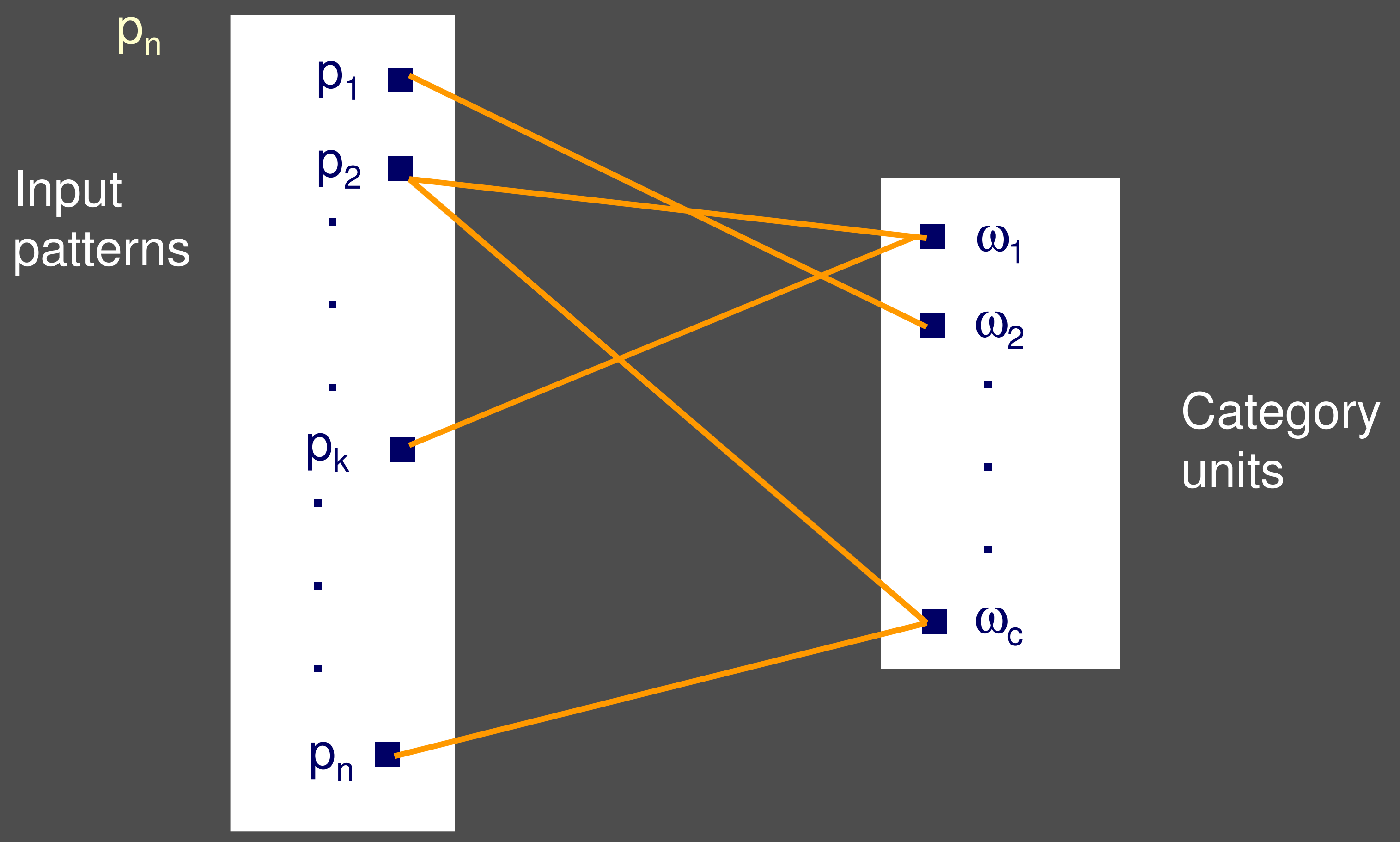
Chapter 4 (part 2): Non-Parametric Classification (Sections 4.3-4.5)

- Parzen Window (cont.)
- Kn –Nearest Neighbor Estimation
- The Nearest-Neighbor Rule

Parzen Windows (cont.)

- Parzen Windows – Probabilistic Neural Networks
  - Compute a Parzen estimate based on n patterns
    - Patterns with d features sampled from c classes
    - The input unit is connected to n patterns





Activations
(Emission of nonlinear functions)

- Training the network

  - Algorithm

    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: $w_1 = x_1$
    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 $w_k = x_k$ $(k = 1, 2, \ldots, n)$

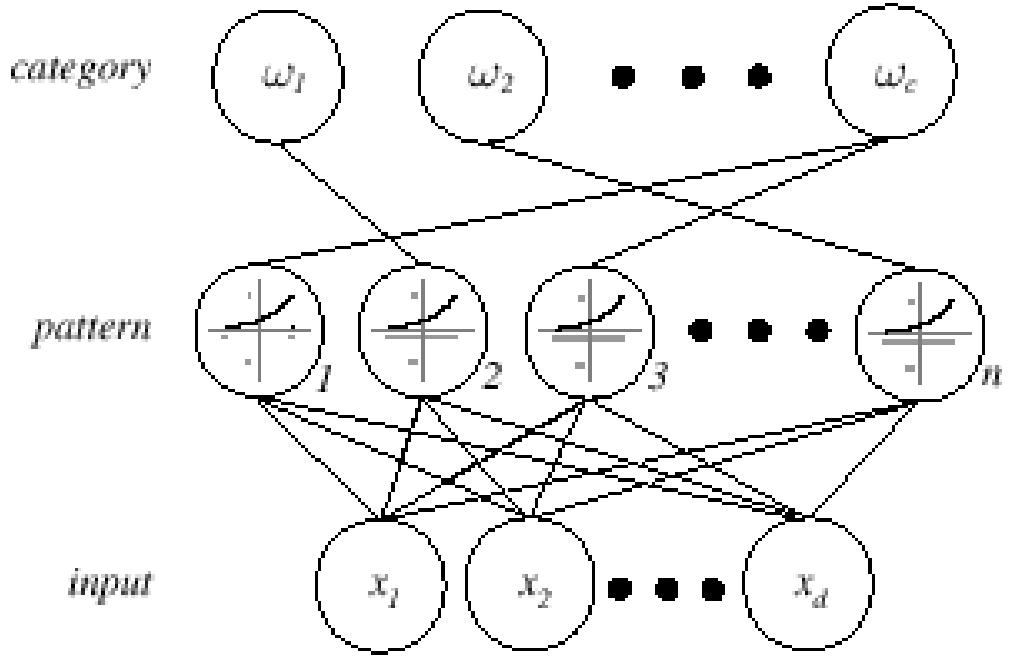


FIGURE 4.9. A probabilistic neural network (PNN) consists of d input units, n pattern units, and c category units. Each pattern unit forms the inner product of its weight vector and the normalized pattern vector $\mathbf{x}$ to form $z = \mathbf{w}^t\mathbf{x}$, and then it emits $\exp[(z-1)/\sigma^2]$. 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.

- Testing the network
  - Algorithm

1. Normalize the test pattern x and place it at the input units
2. Each pattern unit computes the inner product 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 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 grows 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 $k_n=k_1/\sqrt{n}$ and choosing different values for k_1

K-nearest neighbor estimates

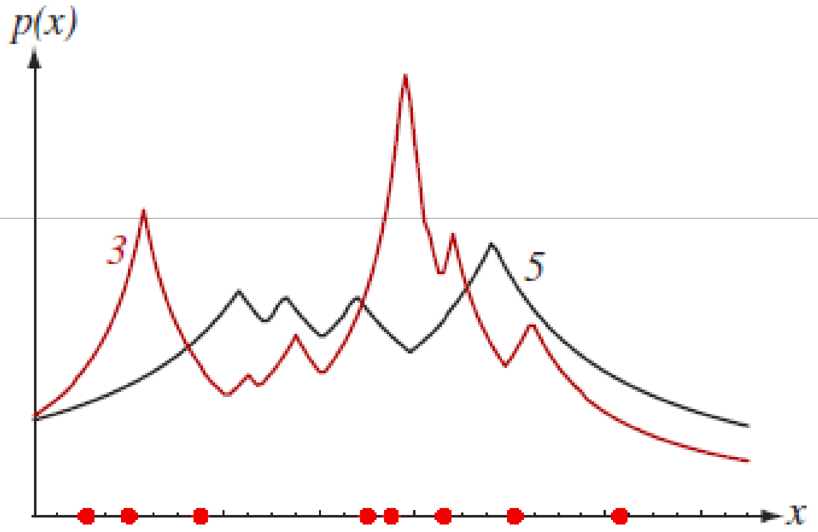


FIGURE 4.10. Eight points in one dimension and the *k*-nearest-neighbor density estimates, for $k = 3$ and 5. Note especially that the discontinuities in the slopes in the estimates generally lie *away* from the positions of the prototype points. From: Richard O. Duda, Peter E. Hart, and David G. Stork,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

K-nearest neighbor estimates

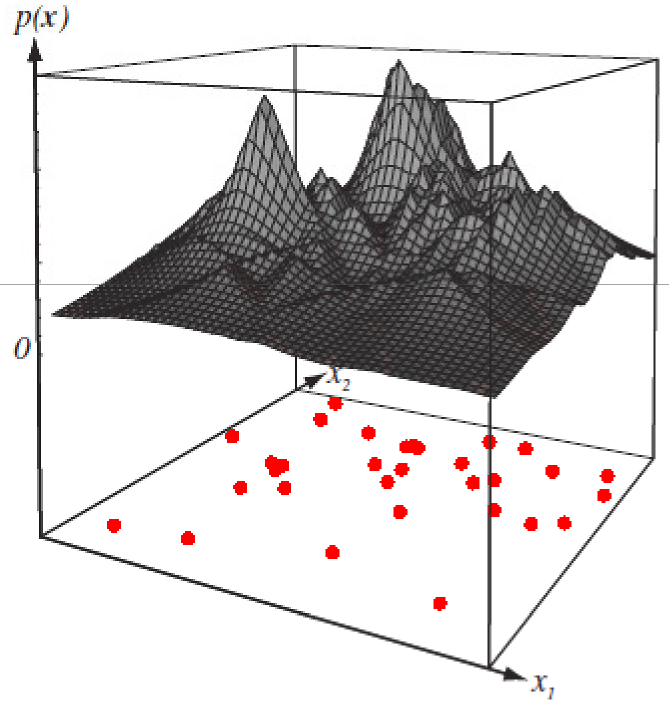
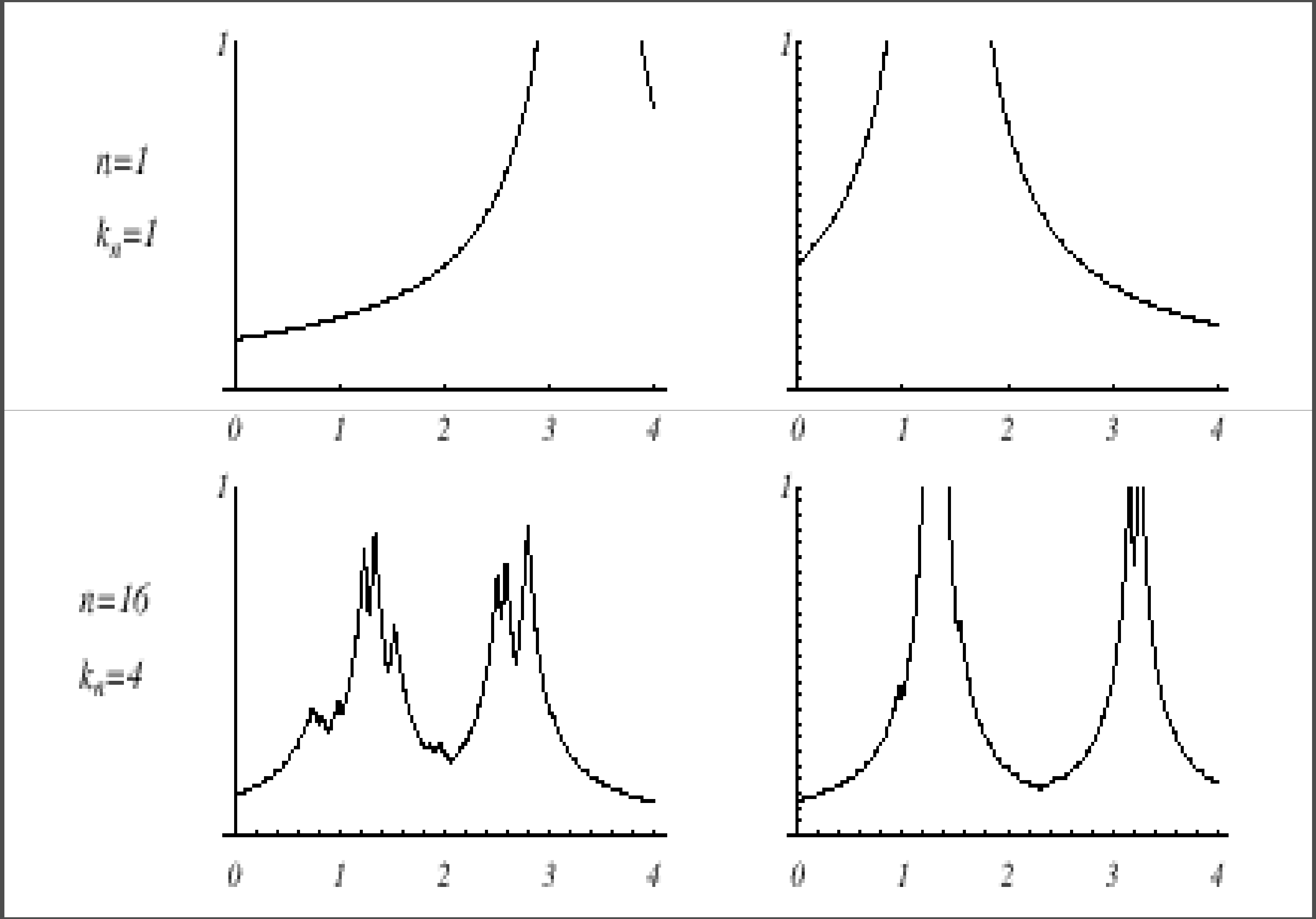


FIGURE 4.11. The k-nearest-neighbor estimate of a two-dimensional density for $k = 5$. Notice how such a finite n estimate can be quite "jagged," and notice that discontinuities in the slopes generally occur along lines away from the positions of the points themselves. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

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

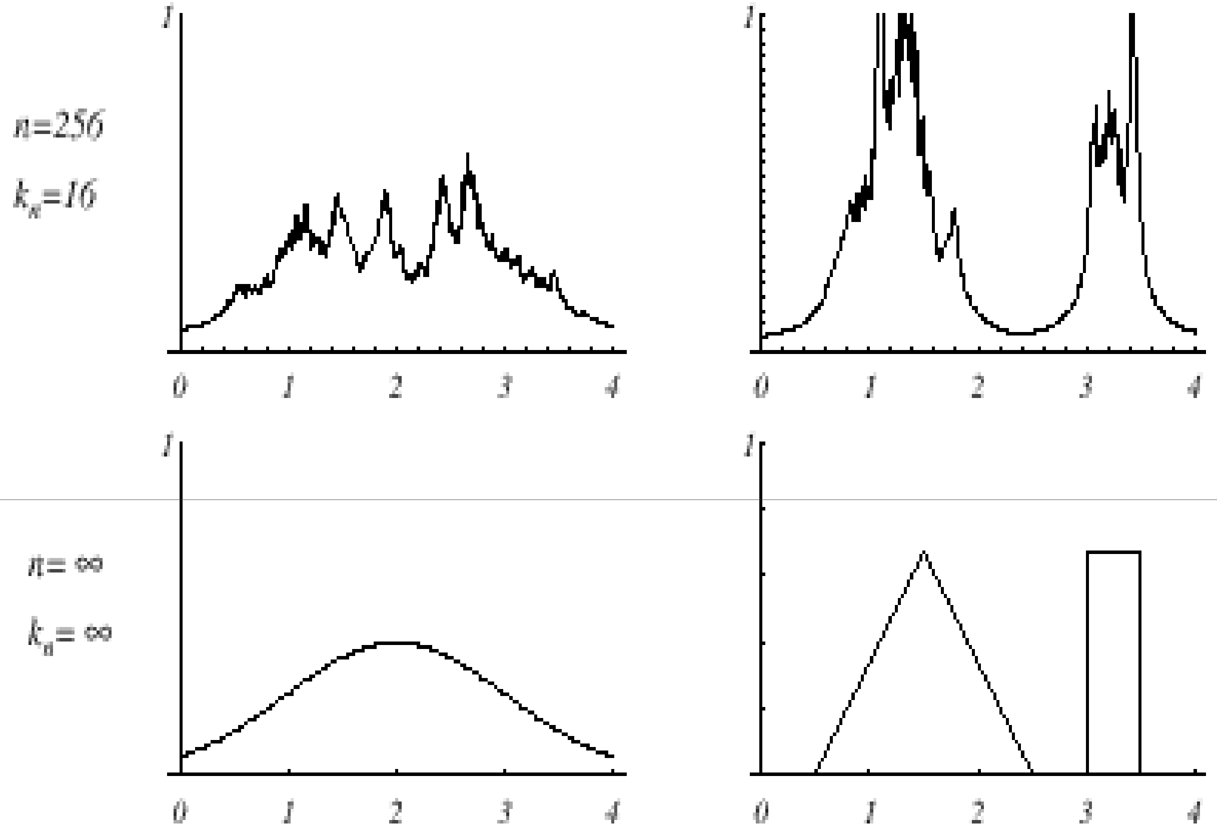


FIGURE 4.12. Several *k*-nearest-neighbor estimates of two unidimensional densities: a Gaussian and a bimodal distribution. Notice how the finite *n* estimates can be quite "spiky." From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons,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

Nearest Neighbor

- Feature space (Voronoi) tesselation → Assign x to the respective cell

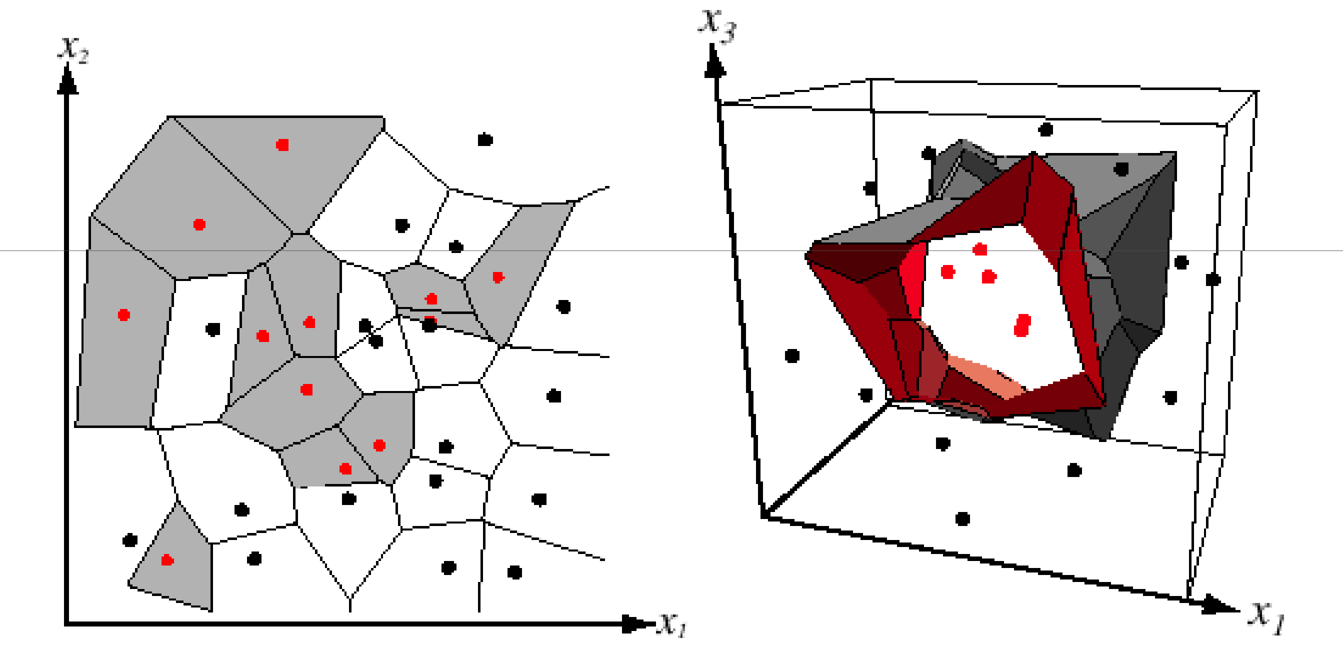


FIGURE 4.13. In two dimensions, the nearest-neighbor algorithm leads to a partitioning of the input space into Voronoi cells, each labeled by the category of the training point it contains. In three dimensions, the cells are three-dimensional, and the decision boundary resembles the surface of a crystal.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

- Estimation of a-posteriori probabilities
  - Goal: estimate $P(\omega_i \,/\, x)$ 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:

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

      An estimate for $p_n(\omega_i / x)$ 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

- 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 prototype 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:

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

- If $P(\omega_m \mid x) \cong 1$, then the nearest neighbor selection is almost always the same as the Bayes selection (Min. Prob. error is small → nearest neighbor prob. error is small too!)
- If $P(\omega_m \mid x) \cong 1/c$ – *The decisions of the are rarely the same! However prob.error = 1-1/c*

Example:

$x = (0.68, 0.60)^t$

Prototypes	Labels	A-posteriori probabilities estimated
$(0.50, 0.30)$	ω_2	0.25
	ω_3	$0.75 = P(\omega_m \mid x)$
$(0.70, 0.65)$	ω_5	0.70
	ω_6	0.30

Decision: ω_3 is the label assigned to x

- RECALL Minimizing the probability of error

- Bayes Decision (Minimize the probability of error)

Decide ω_1 *if* $P(\omega_1 \mid x) > P(\omega_2 \mid x)$; otherwise decide ω_2

Therefore:

$$P(error \mid x) = \min [P(\omega_1 \mid x), P(\omega_2 \mid x)]$$

- The k – nearest-neighbor rule
  - Goal: Classify x by assigning it the label most frequently represented among the k nearest samples and use a voting scheme

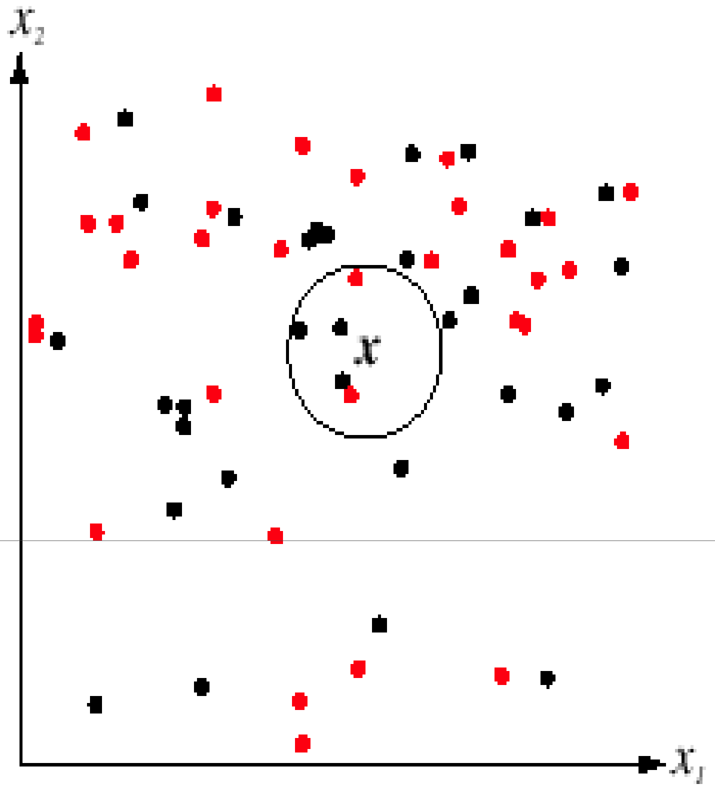


FIGURE 4.15. The *k*-nearest-neighbor query starts at the test point **x** and grows a spherical region until it encloses *k* training samples, and it labels the test point by a majority vote of these samples. In this $k = 5$ case, the test point **x** would be labeled the category of the black points.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

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_5
(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