6 Statistical Estimation & Machine Learning

Outline

- Nonparametric approach to estimate the probability distribution density
  - Parzen-window approach
  - k–nearest-neighbor kNN rule
- Parametric approach to estimate the probability distribution density
  - Maximum likelihood (ML) estimation
  - Multivariate Gaussian probability distribution density
- Unsupervised learning
  - K-means clustering algorithm
  - Gaussian Mixture model and expectation-maximization (EM) algorithm

6 Statistical Estimation & Machine Learning

- We understand that the best decision or optimal classification is:

$$\text{Decide } \omega_k = \arg\min_{\omega_i}[p(e_i \mid \mathbf{x})] = \arg\min_{\omega_i}[1 - p(\omega_i \mid \mathbf{x})]$$
$$= \arg\max_{\omega_i}[p(\omega_i \mid \mathbf{x})] = \arg\max_{\omega_i}[p(\omega_i)\, p(\mathbf{x} \mid \omega_i)]$$

- To design an automatic pattern recognition system, we need know the probability distribution/density function (PDF) for all values of $\mathbf{x}$ so that the system can make decision for any value of received data $\mathbf{x}$. In practice, the PDF is often unknown and can only be estimated by collected examples/samples $\{\mathbf{x}_i\}=[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\mathrm{n}]$ for the design of the pattern recognition system.
- Determining some rules or some deterministic values on a random variable $\mathbf{x}$ based on a set of training samples $D=\{\mathbf{x}_i\}$ is the task of statistical estimation or machine learning.

6 Statistical Estimation & Machine Learning

- The probability distribution/density function (PDF) is the complete information about a random variable.
- It is difficult to estimate the posterior probability $p(\omega_k|\mathbf{x})$ function directly. So we learn the prior probability $p(\omega_k)$ and the class-conditional probability function $p(\mathbf{x}|\omega_k)$.
- The c prior probabilities can be easily estimated by

$$\hat{p}(\omega_k) = n_k / n$$

  where n_k and n are the number of training samples of class ω_k and the total number of training samples.
- As the estimation method is the same for all classes, we simplifying the notation of $p(\mathbf{x}|\omega_k)$ to $p(\mathbf{x})$ and suppose we have n samples $\{\mathbf{x}_i\}$=[$\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_n$] drawn independently and identically distributed (i.i.d.) according to the probability law $p(\mathbf{x})$.

6 Statistical Estimation & Machine Learning

- According to the definition of probability density function, the probability P that $\mathbf{x}$ falls in a region R is:

$$P(\mathbf{x}) = \int_{R_\mathbf{x}} p(\mathbf{t})d\mathbf{t} \approx p(\mathbf{x})V$$

- where V is the volume of the region $R_\mathbf{x}$.
- If out of all n training samples of this class, there are k samples fall in the region R, the estimate of P is then.

$$\hat{P}(\mathbf{x}) = \frac{k}{n}$$

- Therefore, the estimate of the PDF $p(\mathbf{x})$ is

$$\hat{p}(\mathbf{x}) = \frac{k}{nV}$$

- This is called nonparametric approach to the estimation of the probability density function because no assumption of the model is applied.

6 Statistical Estimation & Machine Learning

- The procedure to estimate the PDF based on n training samples is: Given a value of $\mathbf{x}$, select a region/cell of size (volume) V centered at $\mathbf{x}$, count the number of samples k fall in the region/cell. The probability density $p(\mathbf{x})$ at $\mathbf{x}$ is then estimated as :

$$\hat{p}(\mathbf{x}) = \frac{k}{nV}$$

- Obviously, different shape and size of the region/cell lead to different estimate of PDF.
- In general, $\mathbf{x}$ is multiple dimensional $\mathbf{x}=[x_1, x_2, \ldots, x_d]$. If we choose a d-dimensional hypercube of the side length h as the region, a sample $\mathbf{x}_i=[x_{i1}, x_{i2}, \ldots, x_{id}]$ will fall into the hypercube if

$$\frac{|x_j - x_{ij}|}{h} < \frac{1}{2} \quad \text{for } \forall j = 1,\ 2, \ldots,\ d$$

6 Statistical Estimation & Machine Learning

- Therefore, we can express the number of samples falling into the cell k mathematically as

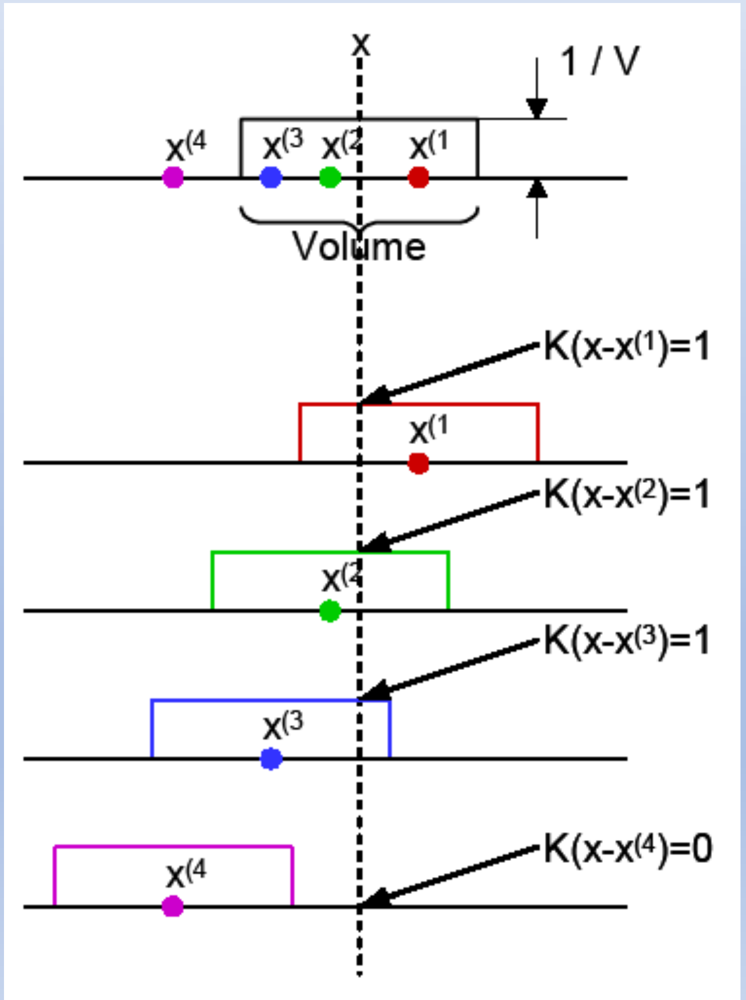
$$k = \sum_{i=1}^{n} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)$$

- Where the kernel function called Parzen-window is defined as

$$K(\mathbf{u}) = \begin{cases} 1 & \text{if } |u_j| < 1/2 \quad \text{for } \forall j = 1,\ 2, \ldots,\ d \\ 0 & \text{otherwise} \end{cases}$$

- The probability density $p(\mathbf{x})$ at $\mathbf{x}$ is then estimated as :

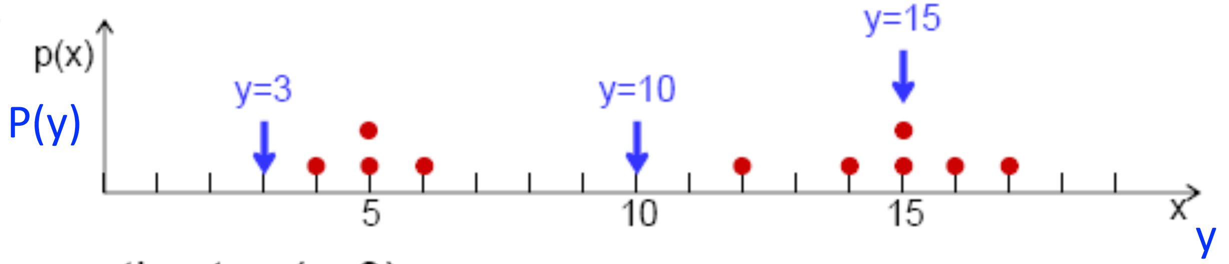
$$\hat{p}(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)$$



6 Statistical Estimation & Machine Learning

Example:

- **Given the dataset below, use Parzen windows to estimate the density p(x) at y=3,10,15. Use a bandwidth of h=4**
  - $X = \{x^{(1}, x^{(2}, \ldots x^{(N}\} = \{4, 5, 5, 6, 12, 14, 15, 15, 16, 17\}$
- **Solution**
  - Let's first draw the dataset to get an idea of what numerical results we should expect



  - Let's now estimate p(y=3):

$$p_{KDE}(y=3) = \frac{1}{Nh^D}\sum_{n=1}^{N} K\left(\frac{y-x^{(n}}{h}\right) = \frac{1}{10\times 4^1}\left[\underbrace{K\left(\frac{3-4}{4}\right)}_{-1/4} + \underbrace{K\left(\frac{3-5}{4}\right)}_{-1/2} + \underbrace{K\left(\frac{3-5}{4}\right)}_{-1/2} + \underbrace{K\left(\frac{3-6}{4}\right)}_{-1} + \ldots + \underbrace{K\left(\frac{3-17}{4}\right)}_{-13/4}\right] =$$

$$= \frac{1}{10\times 4^1}[1+0+0+0+0+0+0+0+0+0] = \frac{1}{10\times 4} = 0.025$$

  - Similarly

$$p_{KDE}(y=10) = \frac{1}{10\times 4^1}[0+0+0+0+0+0+0+0+0+0] = \frac{0}{10\times 4} = 0$$

$$p_{KDE}(y=15) = \frac{1}{10\times 4^1}[0+0+0+0+0+1+1+1+1+0] = \frac{4}{10\times 4} = 0.1$$

Note here:

$$K(\mathbf{u}) = \begin{cases} 1 \text{ if } |u_j| < 1/2 \\ 0 \text{ otherwise} \end{cases}$$

6 Statistical Estimation & Machine Learning

- The rectangular kernel function produces unsmoothed PDF due to the unsmooth rectangular kernel function. In fact, we can choose any function as the kernel so long as:

$$K(\mathbf{u}) \geq 0$$

$$\int_{-\infty}^{\infty} K(\mathbf{u}) d\mathbf{u} = 1 \quad \text{i.e.} \quad \int_{-\infty}^{\infty} \frac{1}{h^d} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right) d\mathbf{x} = 1$$

$$\because \int_{-\infty}^{\infty} K(\mathbf{u}) d\mathbf{u} = \int_{-\infty}^{\infty} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right) d\frac{\mathbf{x}-\mathbf{x}_i}{h} = \int_{-\infty}^{\infty} \frac{1}{h^d} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right) d\mathbf{x}$$

- Therefore, any DPF function can be served as the kernel function. This approach is called Parzen-window approach, it in fact interpolates the discrete points $\{\mathbf{x}_i\}=[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\mathrm{n}]$ into a continuous function PDF $p(\mathbf{x})$.

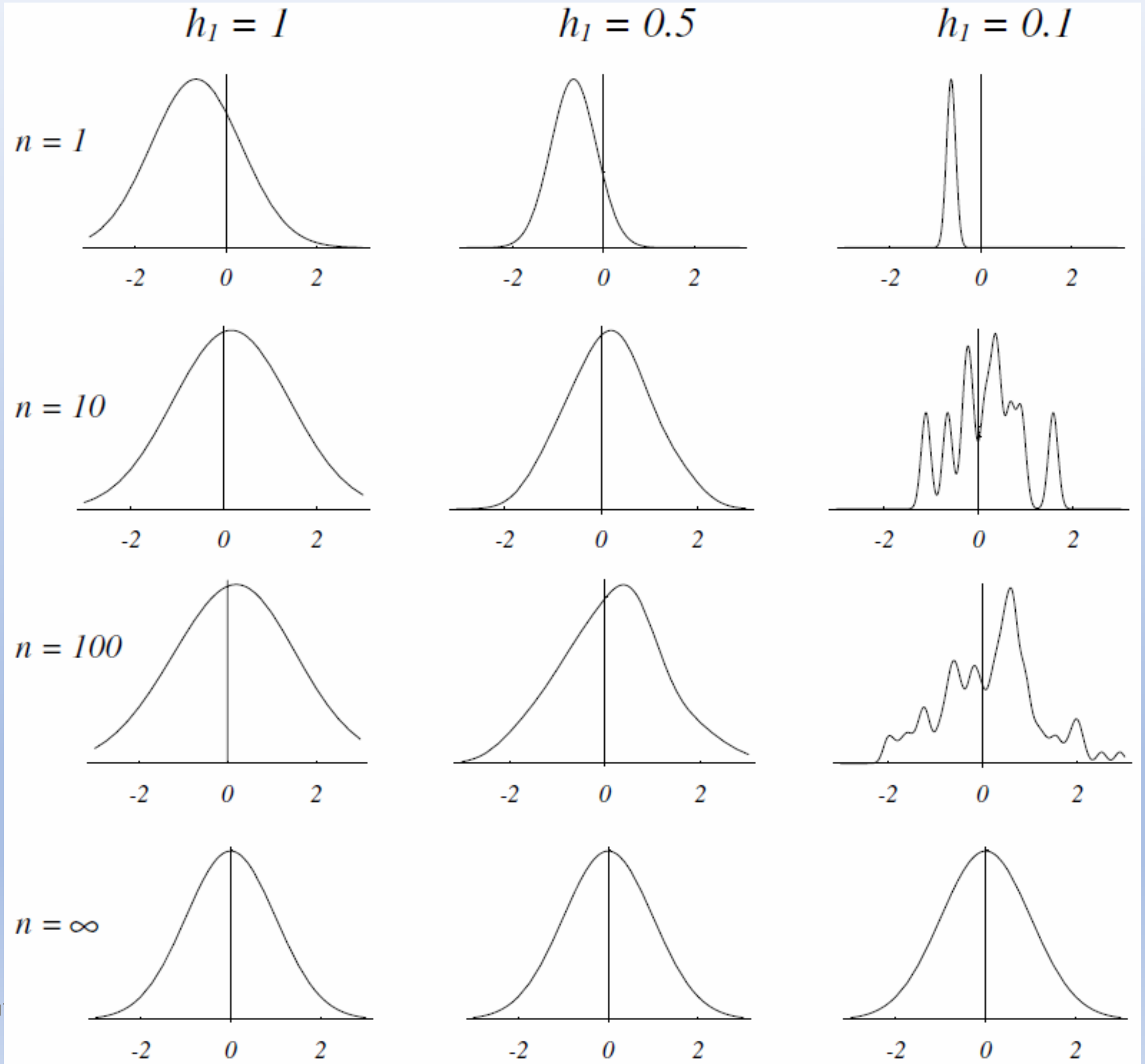
6 Statistical Estimation & Machine Learning

A simple 1-D example.

$$K(\mathbf{u}) = N(0, \mathbf{I})$$

$$= \frac{1}{(2\pi)^{d/2}} \exp\left[-\frac{1}{2}\mathbf{u}^T\mathbf{u}\right]$$

$$\hat{p}(\mathbf{x}) = \frac{1}{nh^d}\sum_{i=1}^{n} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)$$



6 Statistical Estimation & Machine Learning

another

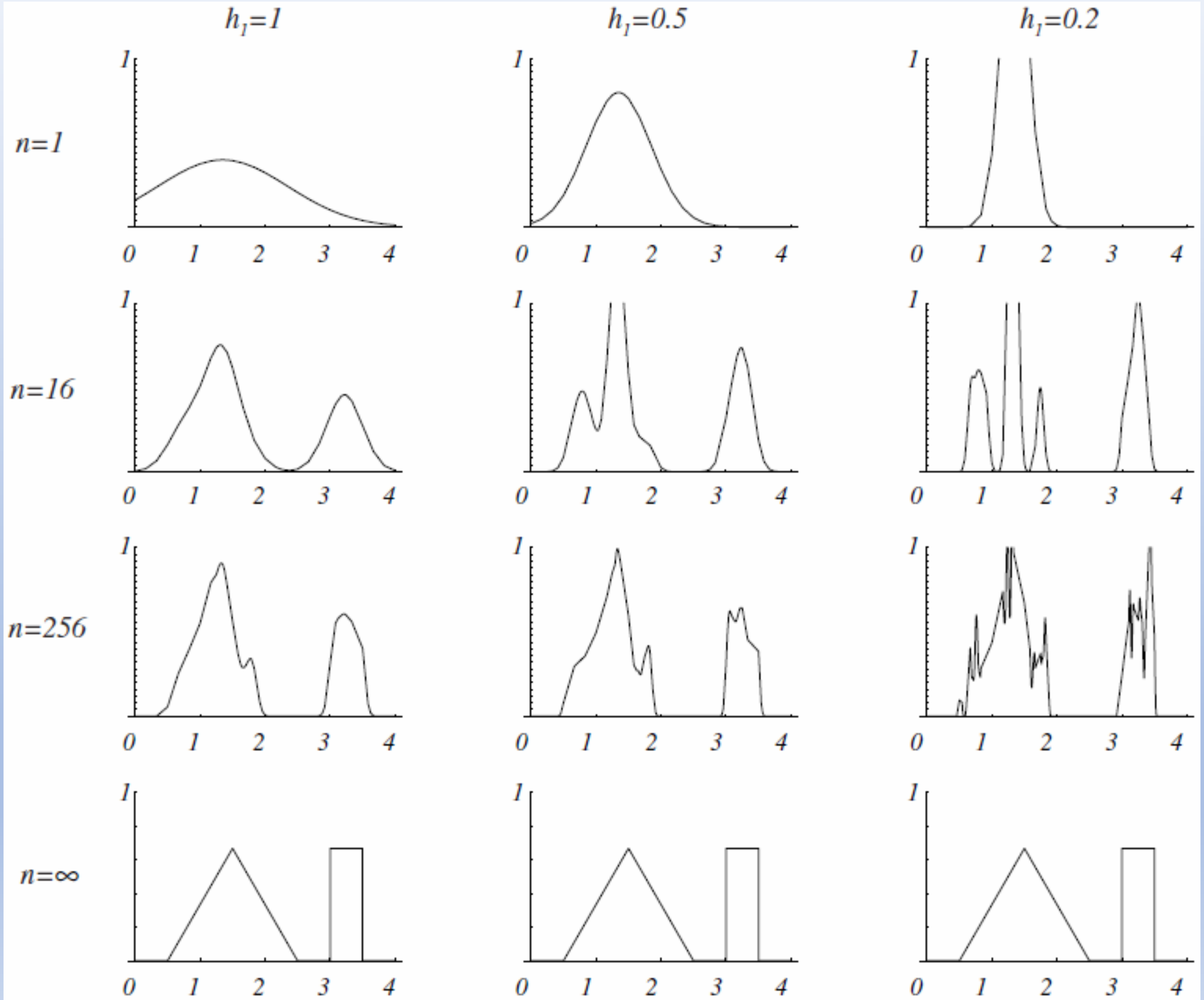
1-D example.

Bimodal distribution

$$K(\mathbf{u}) = N(0, \mathbf{I})$$

$$= \frac{1}{(2\pi)^{d/2}} \exp\left[-\frac{1}{2}\mathbf{u}^T\mathbf{u}\right]$$

$$\hat{p}(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right)$$



6 Statistical Estimation & Machine Learning

- A single 2-D training sample with Gaussian kernel function of different width h.

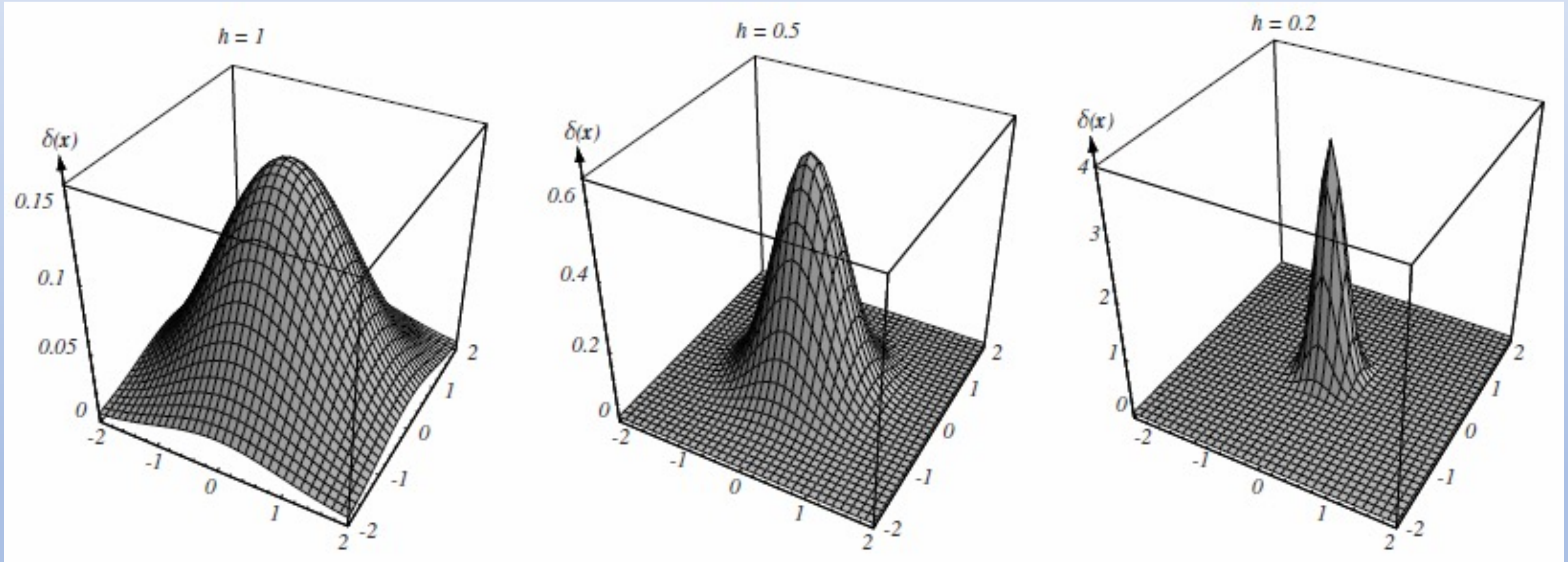


FIGURE 4.3. Examples of two-dimensional circularly symmetric normal Parzen windows for three different values of *h*. Note that because the $\delta(\mathbf{x})$ are normalized, different

6 Statistical Estimation & Machine Learning

- 5 2-D training samples with Gaussian kernel function of different width h.

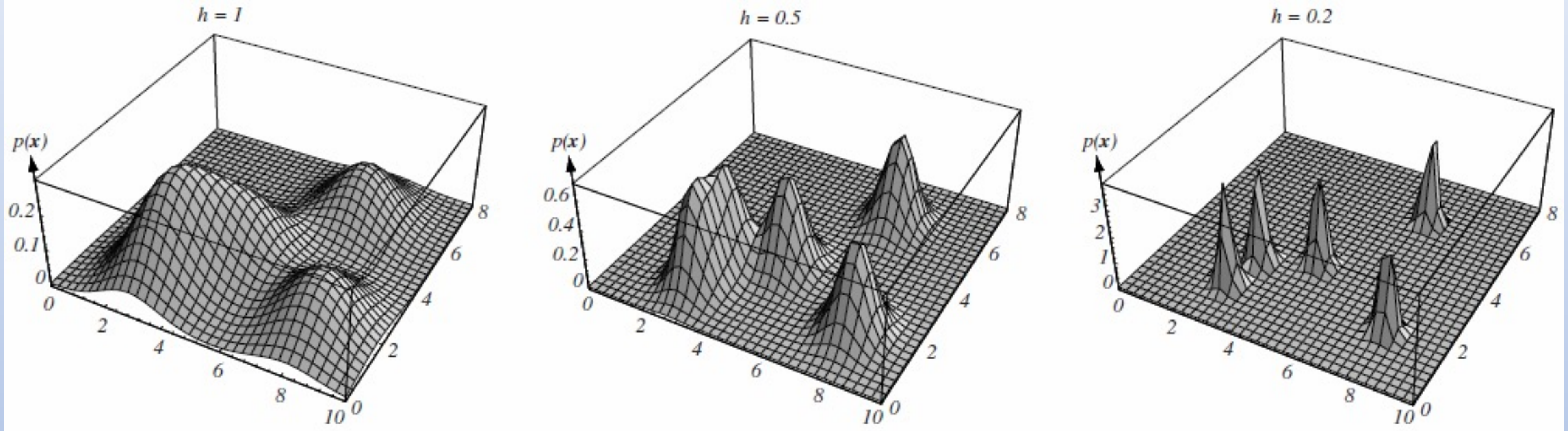


FIGURE 4.4. Three Parzen-window density estimates based on the same set of five samples, using the window functions in Fig. 4.3. As before, the vertical axes have been scaled to show the structure of each distribution.

- The probabilistic neural networks and the RBF neural networks are almost equivalent to the Parzen-window approach.

$$\hat{p}(\mathbf{x}) = \frac{1}{nh^d}\sum_{i=1}^{n} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)$$

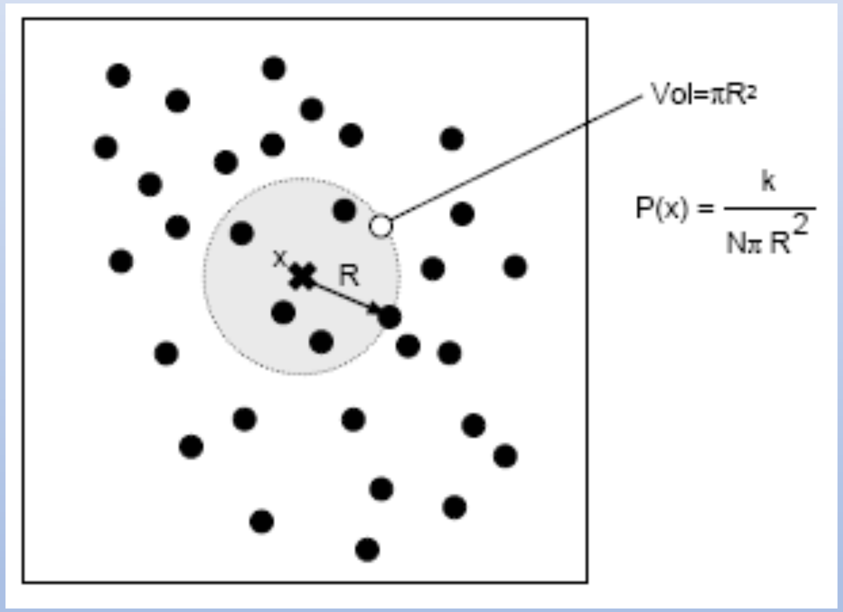
6 Statistical Estimation & Machine Learning

- To estimate the PDF by $\hat{p}(\mathbf{x}) = \dfrac{k}{nV}$

  the Parzen-window approach selects a region/cell of fixed size (volume) V centered at $\mathbf{x}$ and counts the number of samples k fall in the region/cell for the estimation of PDF.

- We can also select the fixed number of samples k and compute the size/volume that just encloses k samples to estimate the PDF by
- This approach is called

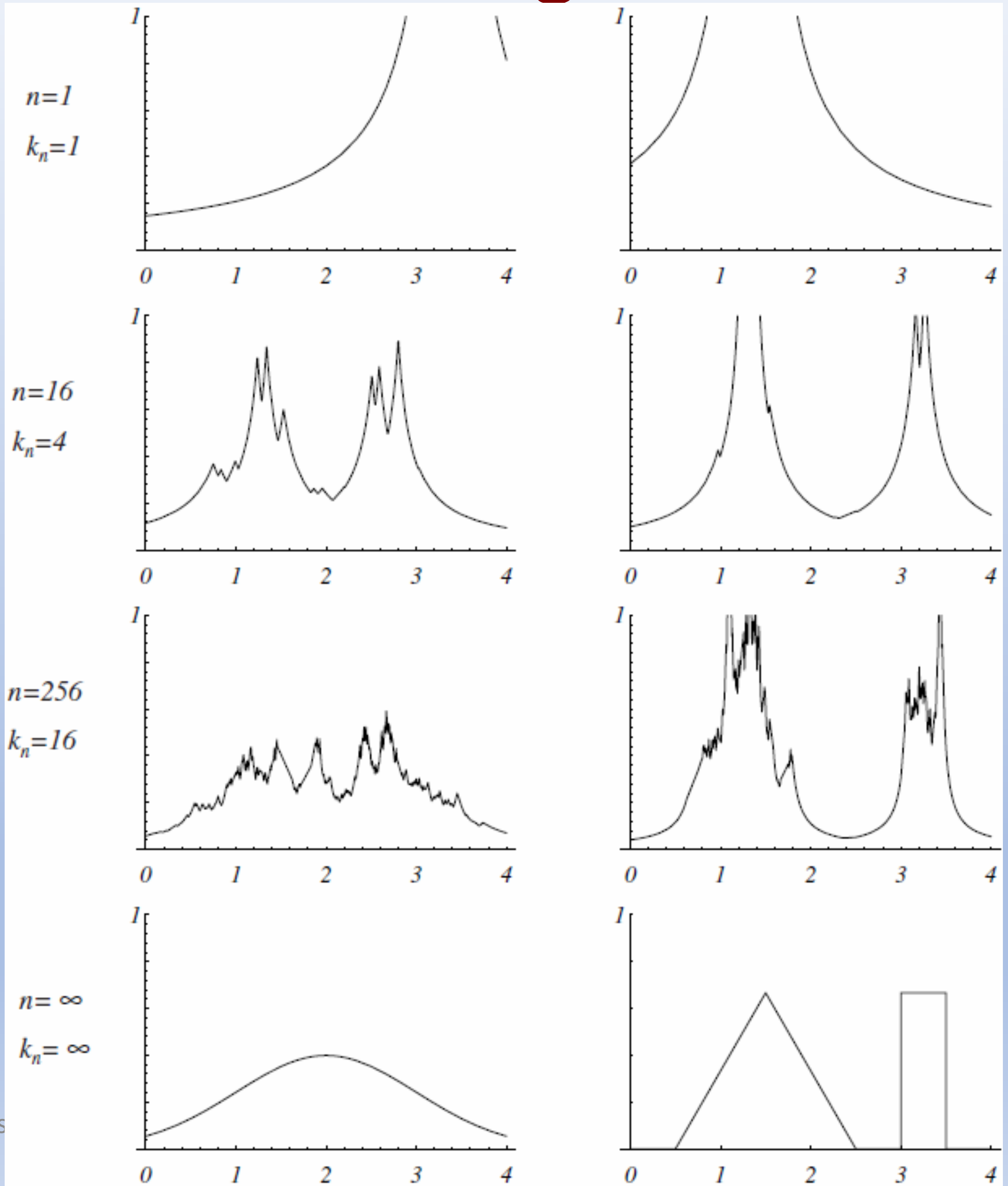
  k–nearest-neighbor kNN estimation.



6 Statistical Estimation & Machine Learning

- Several k-nearest-neighbor estimates of two one-dimensional densities:

$$\hat{p}(\mathbf{x}) = \frac{k}{nV(k)}$$



6 Statistical Estimation & Machine Learning

$$\hat{p}(\mathbf{x}) = \frac{k}{nV(k)}$$

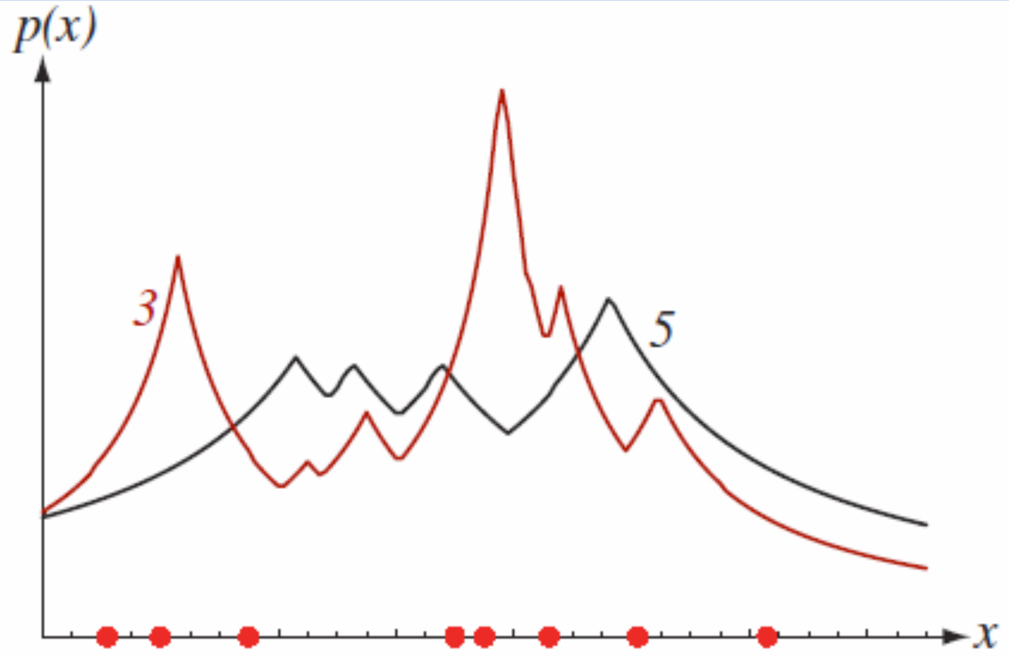


FIGURE 4.10. Eight points in one dimension and the *k*-nearest-neighbor density estimates, for $k = 3$ and 5. Note especially that the discontinuities in the slopes in the estimates generally lie *away* from the positions of the prototype points. From: Richard

6 Statistical Estimation & Machine Learning

$$\hat{p}(\mathbf{x}) = \frac{k}{nV(k)}$$

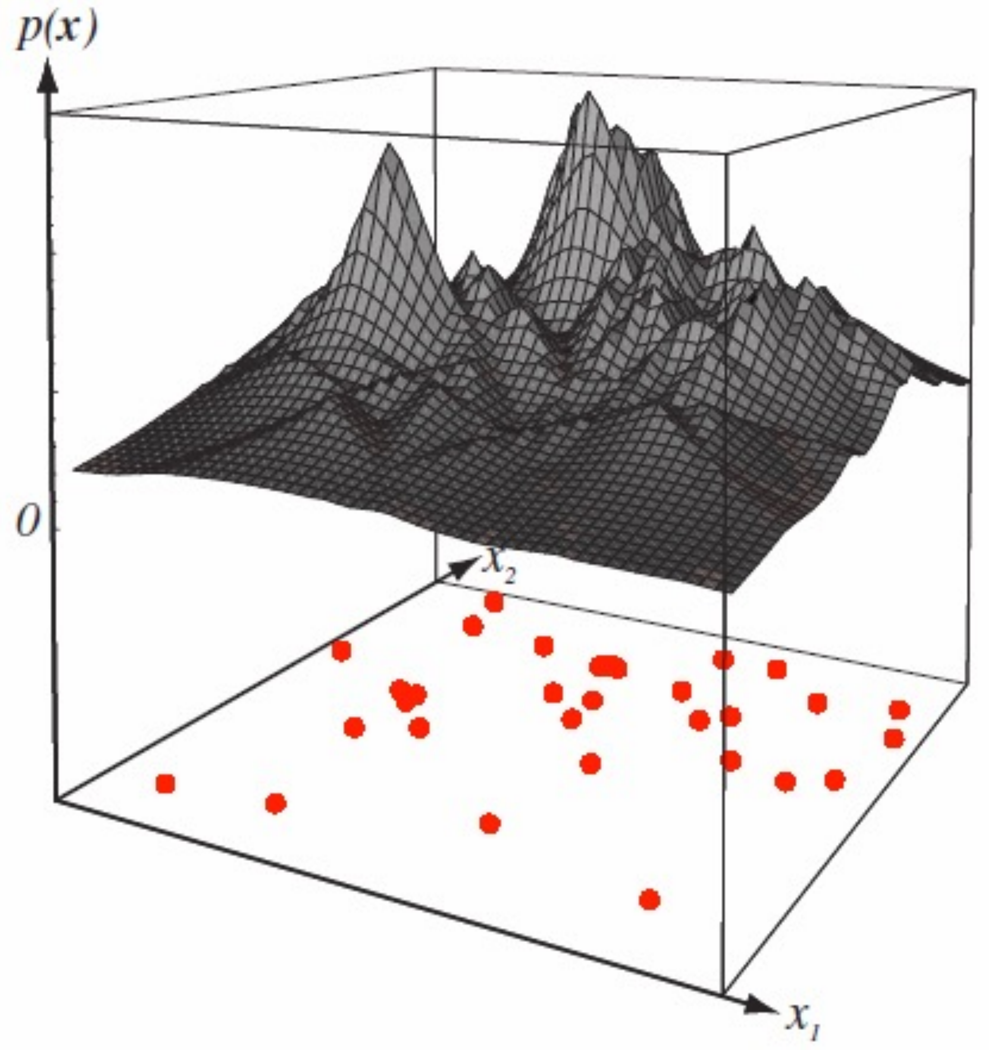


FIGURE 4.11. The k-nearest-neighbor estimate of a two-dimensional density for $k = 5$.

6 Statistical Estimation & Machine Learning

- The KNN technique can also be used for estimation of a-posteriori probabilities $P(\omega_i|\mathrm{x})$ from a set of n labeled samples. (Compare to PNNs and Parzen window estimates.)
- Suppose that we place a cell of volume V around x and capture k samples, k_i of which turn out to be labeled ω_i.
- An estimate for the joint probability is $p_n(\mathrm{x},\omega_i) = \frac{k_i}{nV}$.
- Hence, we can estimate $P(\omega_i|\mathrm{x})$ by

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

$$\hat{p}(\mathbf{x}) = \frac{k}{nV(k)}$$

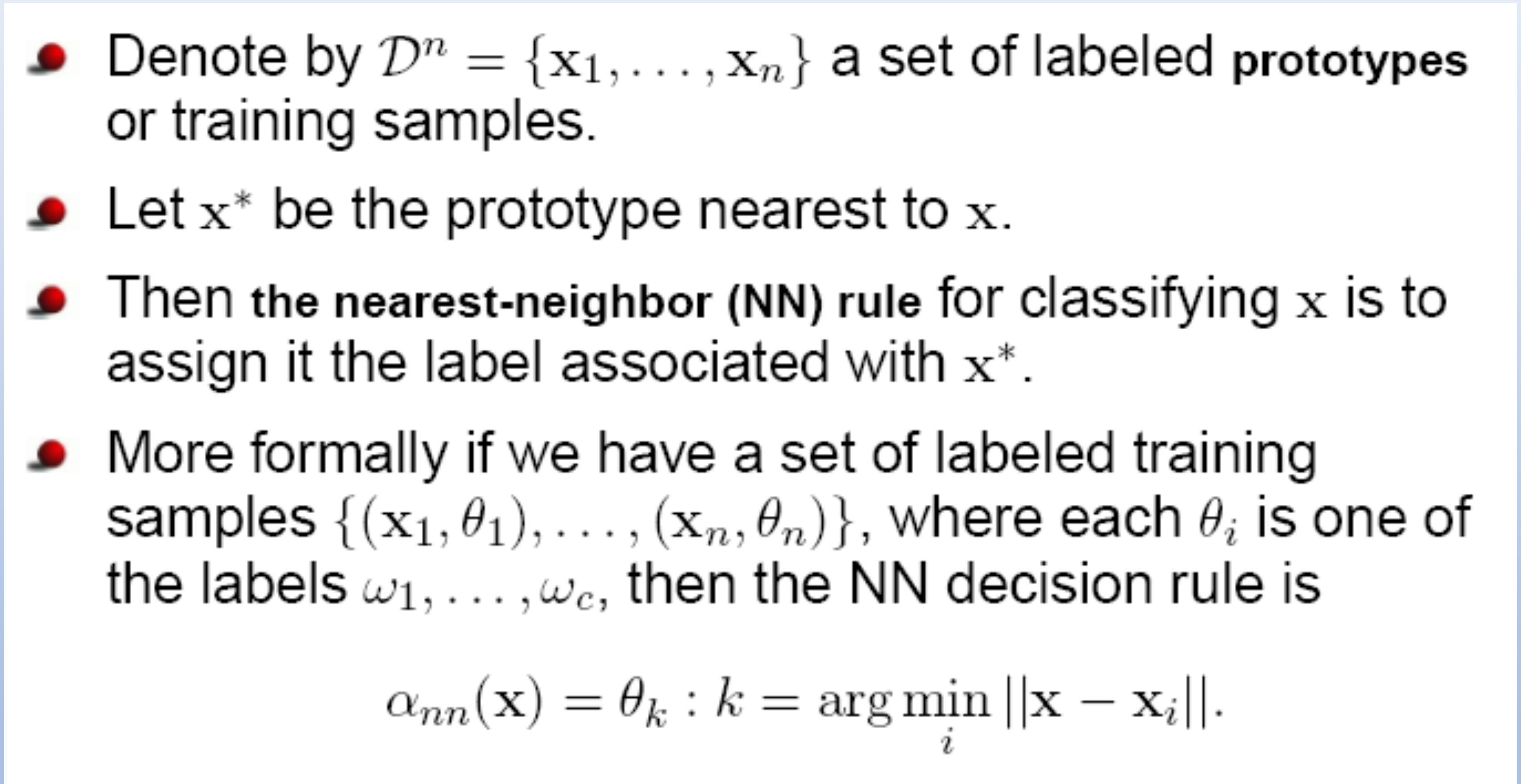
$$\hat{P}(\omega_i) = \frac{n_i}{n}$$

$$\hat{p}(\mathbf{x}\,|\,\omega_i) = \frac{k_i}{n_i V(k)}$$

$$\hat{p}(\omega_i\,|\,\mathbf{x}) = \hat{P}(\omega_i)\hat{p}(\mathbf{x}\,|\,\omega_i) \,/\, \hat{p}(\mathbf{x})$$
$$= \frac{n_i}{n}\frac{k_i}{n_i V(k)}\frac{nV(k)}{k} = \frac{k_i}{k}$$

k–nearest-neighbor classifier, *k*NN classifier

6 Statistical Estimation & Machine Learning



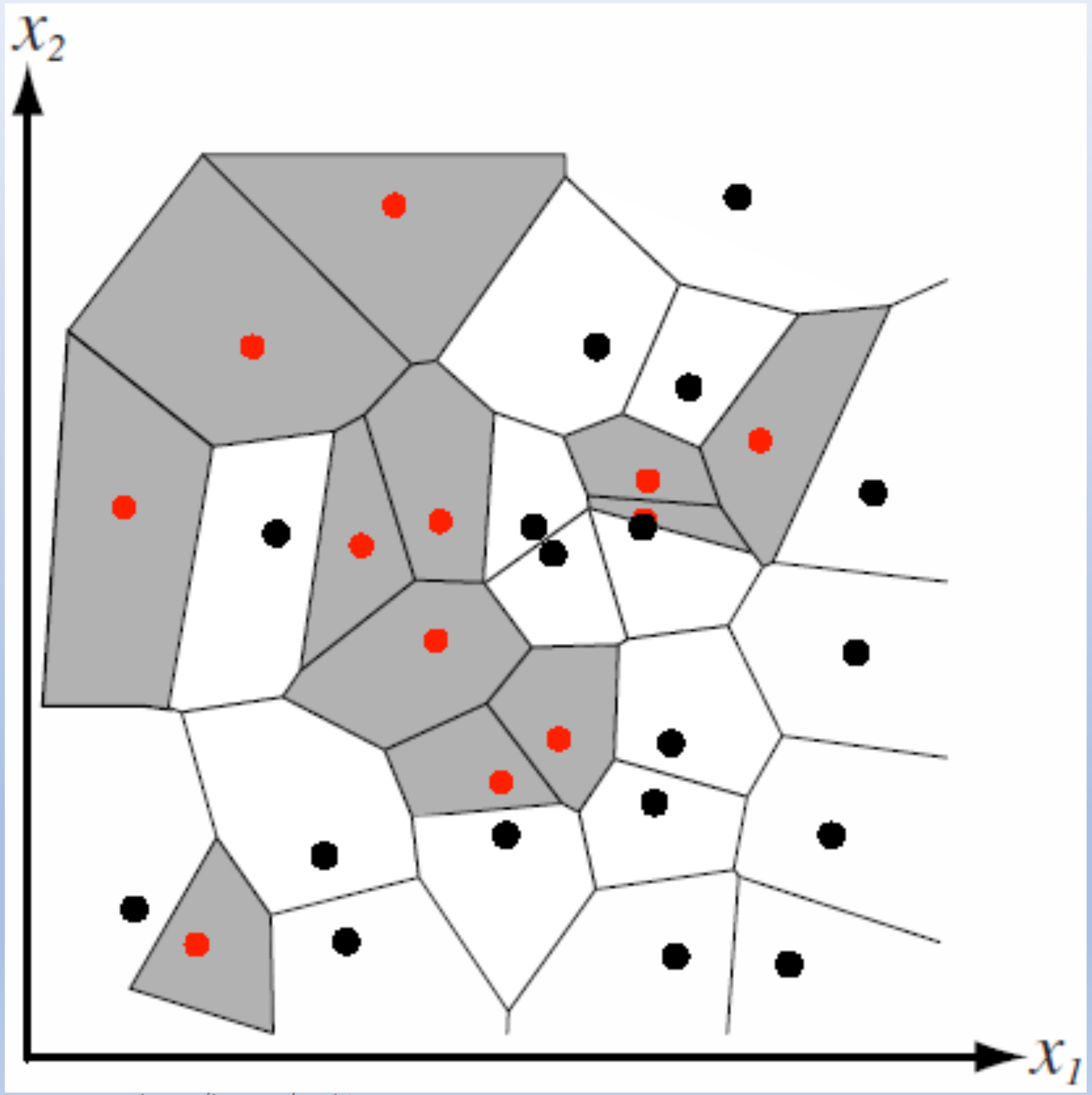
- Denote by $\mathcal{D}^n = \{\mathrm{x}_1, \ldots, \mathrm{x}_n\}$ a set of labeled **prototypes** or training samples.
- Let x^* be the prototype nearest to x.
- Then **the nearest-neighbor (NN) rule** for classifying x is to assign it the label associated with x^*.
- More formally if we have a set of labeled training samples $\{(\mathrm{x}_1, \theta_1), \ldots, (\mathrm{x}_n, \theta_n)\}$, where each θ_i is one of the labels $\omega_1, \ldots, \omega_c$, then the NN decision rule is

$$\alpha_{nn}(\mathrm{x}) = \theta_k : k = \arg\min_i ||\mathrm{x} - \mathrm{x}_i||.$$

1st–nearest-neighbor classifier

6 Statistical Estimation & Machine Learning

- Classification boundary of 1st-NN classifier, also simply called NN classifier.



6 Statistical Estimation & Machine Learning

- The error rate of NN classifier P for very large number of training samples are bounded as $P^* \le P \le P^*\left(2 - \frac{c}{c-1}P^*\right)$

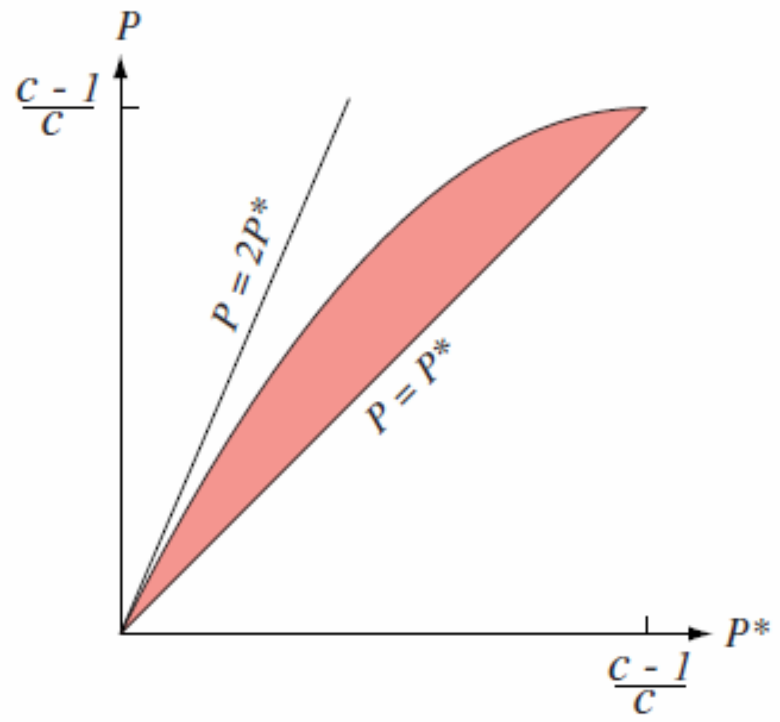
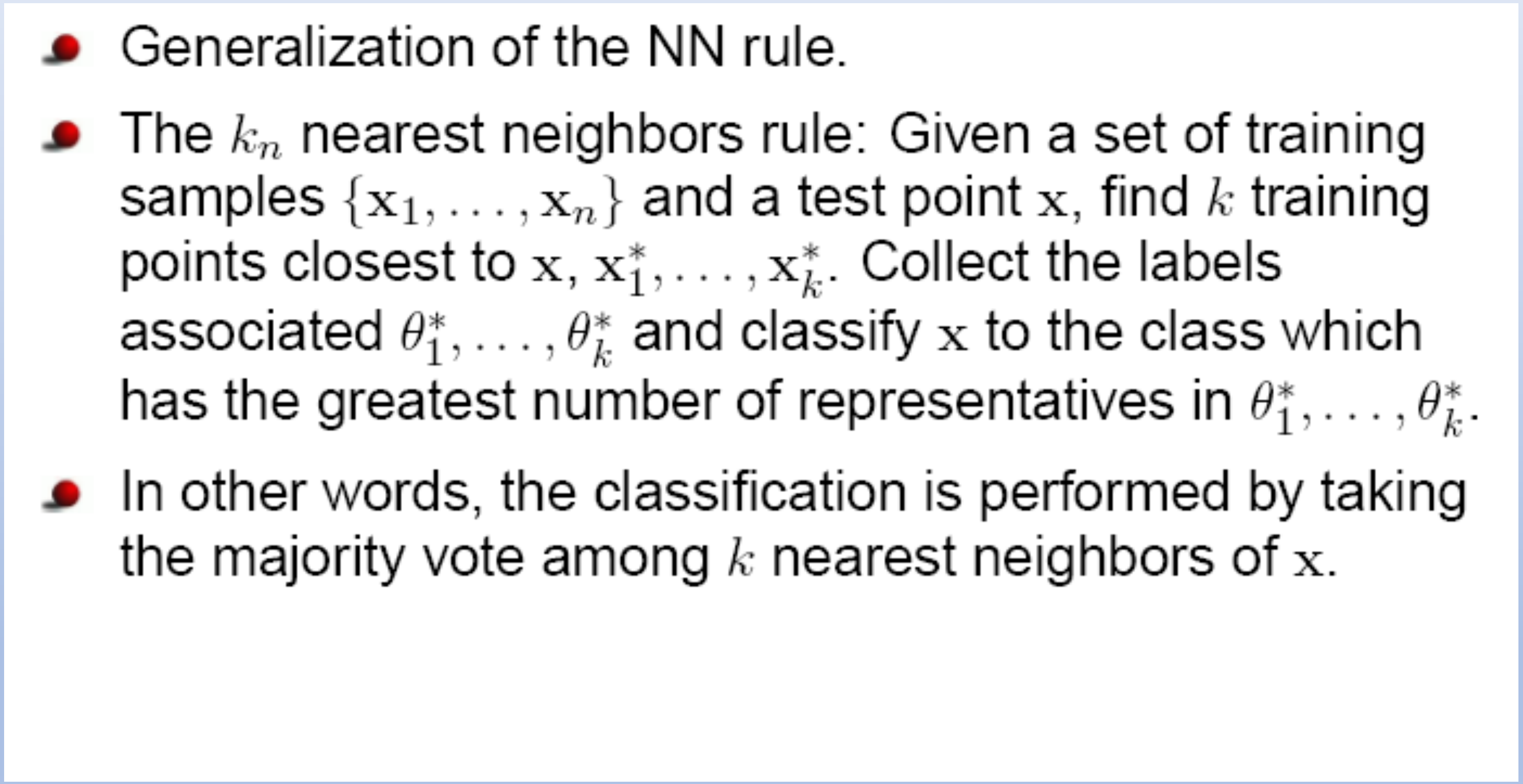


FIGURE 4.14. Bounds on the nearest-neighbor error rate *P* in a *c*-category problem given infinite training data, where *P** is the Bayes error (Eq. 52). At low error rates, the nearest-neighbor error rate is bounded above by twice the Bayes rate. From: Richard O.

6 Statistical Estimation & Machine Learning



- Generalization of the NN rule.
- The k_n nearest neighbors rule: Given a set of training samples $\{\mathrm{x}_1, \ldots, \mathrm{x}_n\}$ and a test point x, find k training points closest to x, $\mathrm{x}_1^*, \ldots, \mathrm{x}_k^*$. Collect the labels associated $\theta_1^*, \ldots, \theta_k^*$ and classify x to the class which has the greatest number of representatives in $\theta_1^*, \ldots, \theta_k^*$.
- In other words, the classification is performed by taking the majority vote among k nearest neighbors of x.

6 Statistical Estimation & Machine Learning

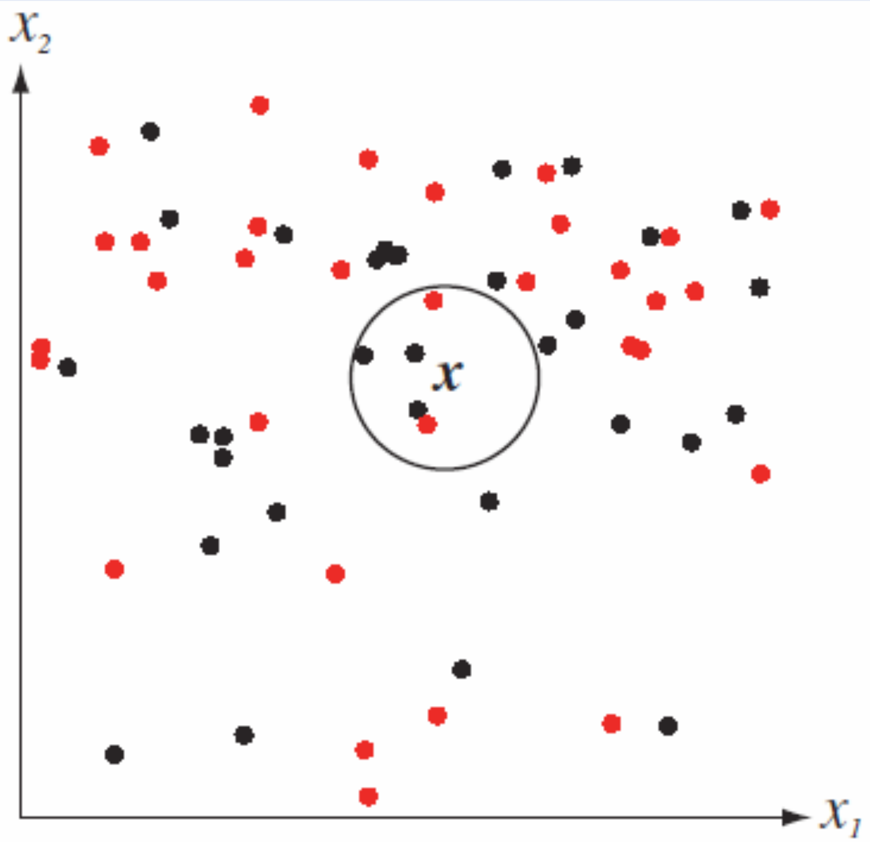


FIGURE 4.15. The k-nearest-neighbor query starts at the test point **x** and grows a spherical region until it encloses k training samples, and it labels the test point by a majority vote of these samples. In this $k = 5$ case, the test point **x** would be labeled the category of the black points. From: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

6 Statistical Estimation & Machine Learning

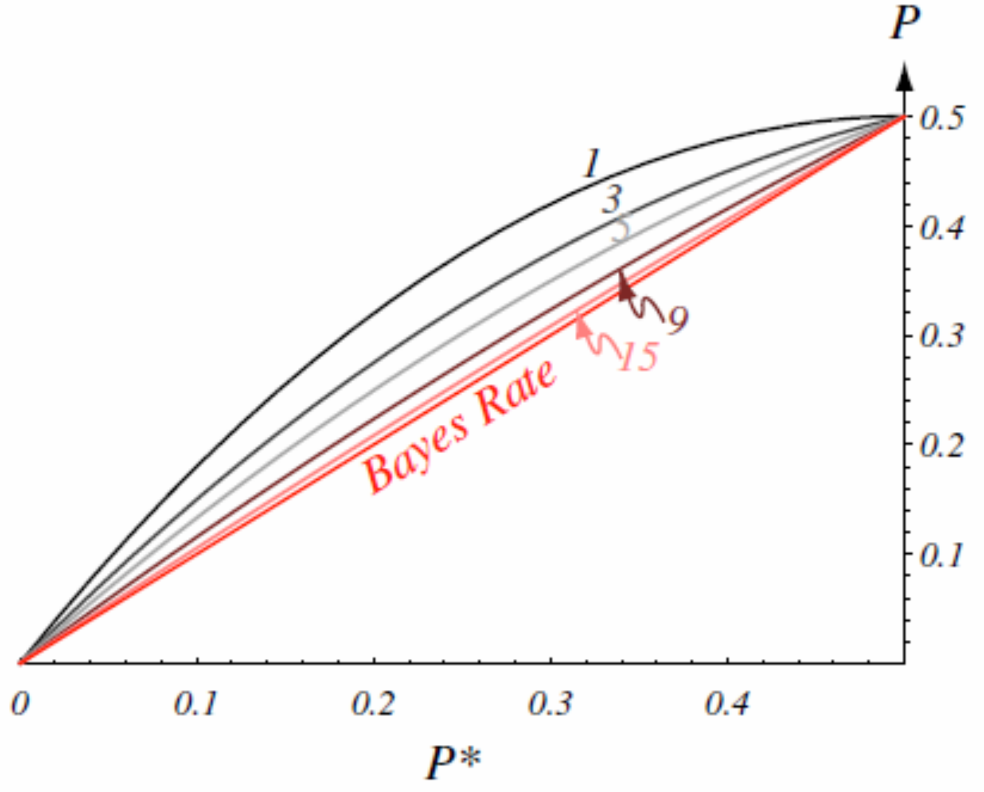


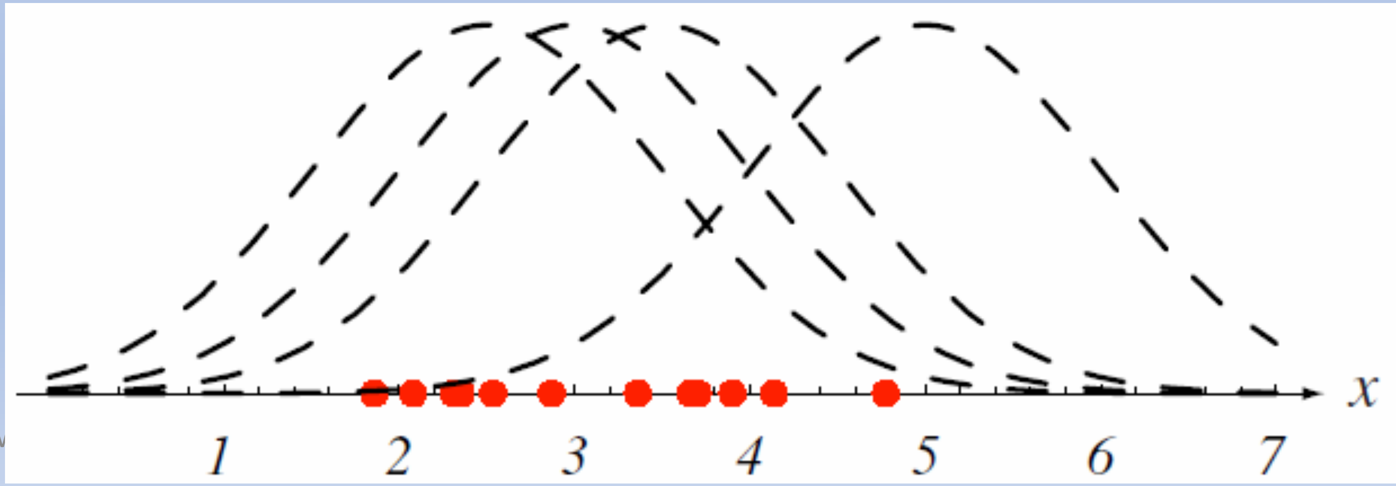
FIGURE 4.16. The error rate for the k-nearest-neighbor rule for a two-category problem is bounded by $C_k(P^*)$ in Eq. 54. Each curve is labeled by k; when $k = \infty$, the estimated probabilities match the true probabilities and thus the error rate is equal to the Bayes rate, that is, $P = P^*$. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern*

6 Statistical Estimation & Machine Learning

- We see that the learned conditional PDF from training samples could greatly deviate from the true PDF of the population, especially in case of small number of training samples.
- If our general knowledge about the problem permits us to model the conditional PDF, i.e. using a mathematical analytical function to represent the PDF with unknown parameter. The severity of these problems can be reduced significantly. Here we parameterize the conditional PDF, which is called parametric method.
- Neural networks and deep learning are also parametric method.
- But we start from the simplest. Suppose that $p(\mathbf{x}|\omega_k)$ is a Gaussian density with mean $\boldsymbol{\mu}_k$ and covariance matrix Σ_k. Although we do not know their values, this knowledge simplifies the problem from estimating an unknown function $p(\mathbf{x}|\omega_k)$ to estimating the unknown parameters $\boldsymbol{\mu}_k$ and Σ_k only.

6 Statistical Estimation & Machine Learning

- Now we will use a set of training samples $D=\{\mathbf{x}_i\}=[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$ drawn independently from the probability density $p(\mathbf{x}|\theta)$ to estimate the unknown parameter vector θ.
- Lets see an example to generate idea how to estimate the parameter of a given probability density $p(\mathbf{x}|\theta)$ reasonably based on the training data D.
- The graph shows several training points in one dimension, known or assumed to be drawn from a Gaussian of a particular variance, but unknown mean. Four PDF with 4 different means are shown in dashed lines.
- Which PDF you should choose?



6 Statistical Estimation & Machine Learning

- Now we formulate our idea mathematically.
- Obviously, the probability that a sample $\mathbf{x}_k$ occurs is $p(\mathbf{x}_k|\boldsymbol{\theta})$. As all samples in the training set are independently collected (occur), the probability that all samples occur is

$$p(D\,|\,\boldsymbol{\theta}) = \prod_{k=1}^{n} p(\mathbf{x}_k\,|\,\boldsymbol{\theta})$$

- Intuitively, we should select the parameter so that the probability density $p(\mathbf{x}|\boldsymbol{\theta})$ best supports the actually observed training samples, i.e. to make the probability of all training data occur $p(D|\boldsymbol{\theta})$ maximal. Note that $p(D|\boldsymbol{\theta})$ is called the likelihood of $\boldsymbol{\theta}$ with respect to the set of samples D. Thus, this method is called the maximum likelihood (ML) estimation.

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} p(D\,|\,\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \prod_{k=1}^{n} p(\mathbf{x}_k\,|\,\boldsymbol{\theta})$$

6 Statistical Estimation & Machine Learning

- It is often not easy to get an analytical solution of

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} p(D \mid \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \prod_{k=1}^{n} p(\mathbf{x}_k \mid \boldsymbol{\theta})$$

  due to the multiplication of the functions of $\boldsymbol{\theta}$ and $p(\mathbf{x}_k|\boldsymbol{\theta})$ is often nonlinear function of $\boldsymbol{\theta}$.

- Since the logarithm is monotonically increasing, maximizing the logarithm of a function also maximizes the function itself. The logarithm has nice property that converts the multiplication into summation and simplifies the exponential function.

- Thus, we maximize the log-likelihood instead of maximizing the likelihood $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \ln p(D \mid \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \sum_{k=1}^{n} \ln p(\mathbf{x}_k \mid \boldsymbol{\theta})$

6 Statistical Estimation & Machine Learning

- The solution can be found by the standard methods of differential calculus: Solving the equation that the gradient is zero.

$$\nabla_{\boldsymbol{\theta}} \ln p(D \mid \boldsymbol{\theta}) = 0$$

$$\sum_{k=1}^{n} \nabla_{\boldsymbol{\theta}} \ln p(\mathbf{x}_k \mid \boldsymbol{\theta}) = 0$$

- If the number of parameters to be estimated is q, then $\boldsymbol{\theta}$ is a q-component vector $\boldsymbol{\theta}=(\theta_1, \theta_2, \ldots, \theta_q)^T$. The gradient is a vector that contains partial differentiation against all components of $\boldsymbol{\theta}$.

$$\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) \triangleq \begin{pmatrix} \dfrac{\partial f(\boldsymbol{\theta})}{\partial \theta_1} \\ \vdots \\ \dfrac{\partial f(\boldsymbol{\theta})}{\partial \theta_q} \end{pmatrix}$$

6 Statistical Estimation & Machine Learning

- To see how maximum likelihood methods results apply to a specific case, suppose that the samples are drawn from a multivariate Gaussian population with unknown mean μ and covariance matrix Σ.

$$p(\mathbf{x}\,|\,\boldsymbol{\theta}) = \frac{1}{(2\pi)^{d/2}\,|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$

- The log-likelihood of a single sample is

$$\ln p(\mathbf{x}_k\,|\,\boldsymbol{\theta}) = -\frac{1}{2}\ln[(2\pi)^d\,|\boldsymbol{\Sigma}|] - \frac{1}{2}(\mathbf{x}_k-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}_k-\boldsymbol{\mu})$$

- Consider first the univariate case with $\boldsymbol{\theta}=(\theta_1, \theta_2)^T=(\mu, \sigma^2)^T$.

$$p(x\,|\,\boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$$

6 Statistical Estimation & Machine Learning

- Here the log-likelihood of a single sample is simplified as

$$\ln p(x_k \mid \boldsymbol{\theta}) = -\frac{1}{2}\ln[2\pi\sigma^2] - \frac{1}{2\sigma^2}(x_k - \mu)^2$$

- Its derivative is

$$\nabla_{\boldsymbol{\theta}} \ln p(x_k \mid \boldsymbol{\theta}) = \begin{pmatrix} \dfrac{1}{\sigma^2}(x_k - \mu) \\ -\dfrac{1}{2\sigma^2} + \dfrac{(x_k - \mu)^2}{2\sigma^4} \end{pmatrix}$$

6 Statistical Estimation & Machine Learning

- Applying ML

$$\nabla_{\boldsymbol{\theta}} \ln p(D \mid \boldsymbol{\theta}) = \sum_{k=1}^{n} \nabla_{\boldsymbol{\theta}} \ln p(\mathbf{x}_k \mid \boldsymbol{\theta}) = 0$$

- We have

$$\sum_{k=1}^{n} \frac{1}{\sigma^2}(x_k - \mu) = 0$$

$$-\sum_{k=1}^{n} \frac{1}{2\sigma^2} + \sum_{k=1}^{n} \frac{(x_k - \mu)^2}{2\sigma^4} = 0$$

- Solve these two equations we have the following maximum likelihood estimates

$$\hat{\mu} = \frac{1}{n} \sum_{k=1}^{n} x_k$$

$$\hat{\sigma}^2 = \frac{1}{n} \sum_{k=1}^{n} (x_k - \hat{\mu})^2$$

6 Statistical Estimation & Machine Learning

- While the analysis of the multivariate case is basically very similar, considerably more manipulations are involved. The result is that the maximum likelihood estimates for mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ of multivariate Gaussian PDF

$$p(\mathbf{x} \mid \boldsymbol{\theta}) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right]$$

are given by

$$\hat{\boldsymbol{\mu}} = \frac{1}{n} \sum_{k=1}^{n} \mathbf{x}_k$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{n} \sum_{k=1}^{n} (\mathbf{x}_k - \hat{\boldsymbol{\mu}})(\mathbf{x}_k - \hat{\boldsymbol{\mu}})^T$$

Unsupervised Learning and Clustering

- Previously, all our training samples were labeled: these samples were said "supervised".
- We now investigate a number of "unsupervised" procedures which use unlabeled samples because collecting and labeling a large set of sample patterns can be costly or in some cases not possible based on human knowledge.

- We can use unsupervised methods to identify features that will then be useful for categorization.
- We gain some insight into the nature (or structure) of the data.

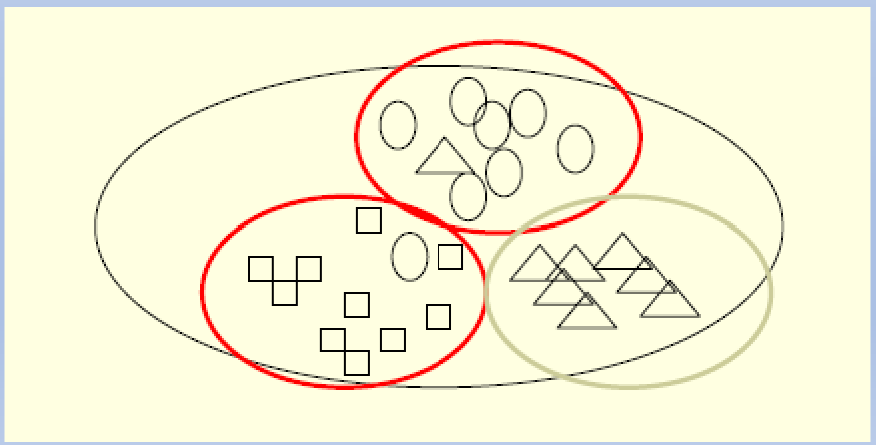
- Clustering that partitions samples into a number of groups without knowing the class membership of the samples belong to unsupervised learning.

Unsupervised Learning and Clustering

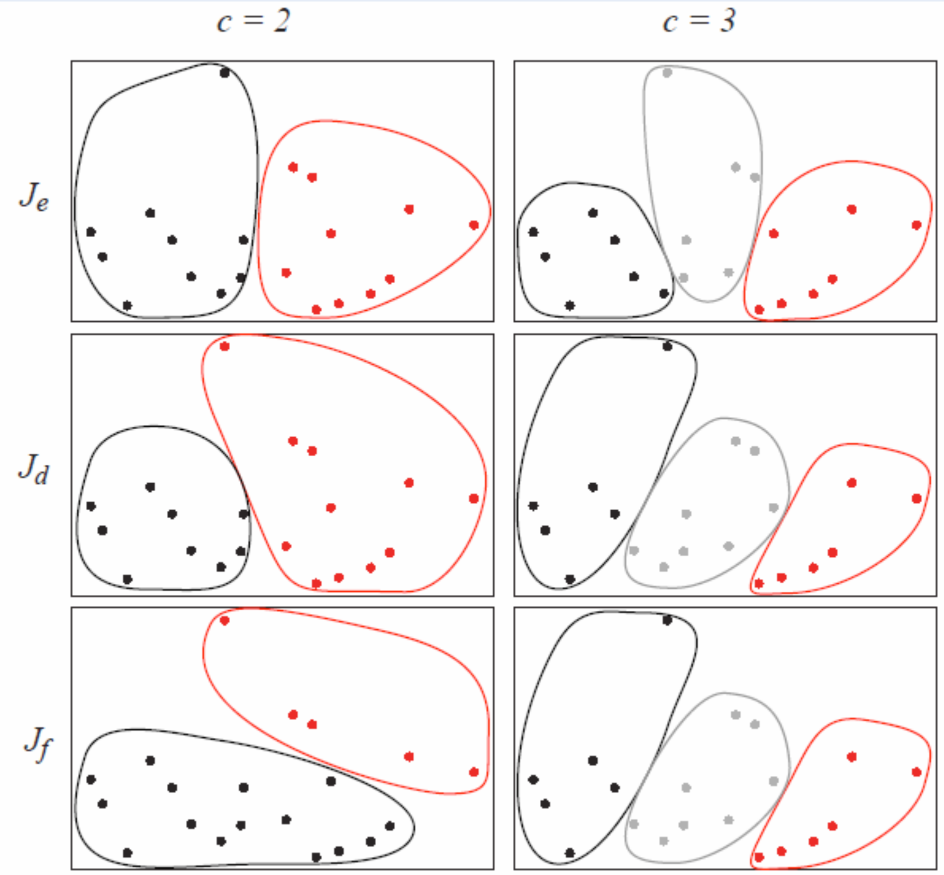
- Clustering is a process of partitioning a set of data (or objects) in a set of meaningful sub-classes, called clusters. – Helps users understand the natural grouping or structure in a data set.
- Cluster: a collection of data objects that are "similar" to one another and thus can be treated collectively as one group.
- Given a database $D=\{t_1,t_2,\ldots,t_n\}$ of *n* samples and an integer value *k*, the *Clustering Problem* is to define a mapping $f: D \rightarrow \{1,..,k\}$ where each t_i is assigned to one cluster K_j, $1<j<k$.
- A *Cluster*, K_j, contains those samples mapped to it. Unlike classification problem, clusters are not known a priori.
  - To group data into un-predefined classes
  - To make data within the same cluster have high similarity, while data points in different clusters have low similarity.

Unsupervised Learning and Clustering

- A good method will produce high quality clusters in which:
  - the intra-cluster similarity is high.
  - the inter-cluster similarity is low.
- The quality of a clustering result also depends on both the similarity measure used by the method and its implementation.
- The quality of a clustering method is also measured by its ability to discover the hidden patterns.



Unsupervised Learning and Clustering



The clusters found by minimizing a criterion depends upon the criterion function as well as the assumed number of clusters. The sum-of-squared-error criterion J_e (Eq. 49), the determinant criterion J_d (Eq. 63) and the more subtle trace criterion J_f (Eq. 65) were applied to the 20 points in the table with the assumption of $c = 2$ and $c = 3$ clusters. (Each point in the table is shown, with bounding boxes defined

Unsupervised Learning and Clustering

Major Categories of Algorithms:

- Partitioning : Construct a partition of a database *D* of *n* objects into a set of *k* clusters that optimizes the chosen partitioning criterion.
  - K-means algorithm
- Model-based : A model is hypothesized for each of the clusters and the idea is to find the best fit of that model to each other.
  - Gaussian Mixture model

Unsupervised Learning and Clustering

Partitioning method: Construct a partition of a database

$\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ of n objects into a set of k clusters

$\mathcal{D}_j, j = 1, 2, \cdots, k$ that optimizes the chosen partitioning criterion.

k-means algorithm--principle:

- Each cluster is represented by the center (mean) of the cluster.

$$\mathcal{M} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \cdots, \boldsymbol{\mu}_k\}$$

- Estimate the unknown k cluster centers (means)

$$\mu_j = \frac{1}{|\mathcal{D}_j|} \sum_{\mathbf{x}_i \in \mathcal{D}_j} \mathbf{x}_i$$

to minimize the sum of error squares.

$$e(\mathcal{M}) = \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in \mathcal{D}_j} \| \mathbf{x}_i - \boldsymbol{\mu}_j \|^2$$

Unsupervised Learning and Clustering

k-means algorithm--implementation:

1. Initialize k clusters, e.g. randomly pick samples.

$$\mathcal{M} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \cdots, \boldsymbol{\mu}_k\} = \{\mathbf{x}_{r1}, \mathbf{x}_{r2}, \cdots, \mathbf{x}_{rk}\}$$

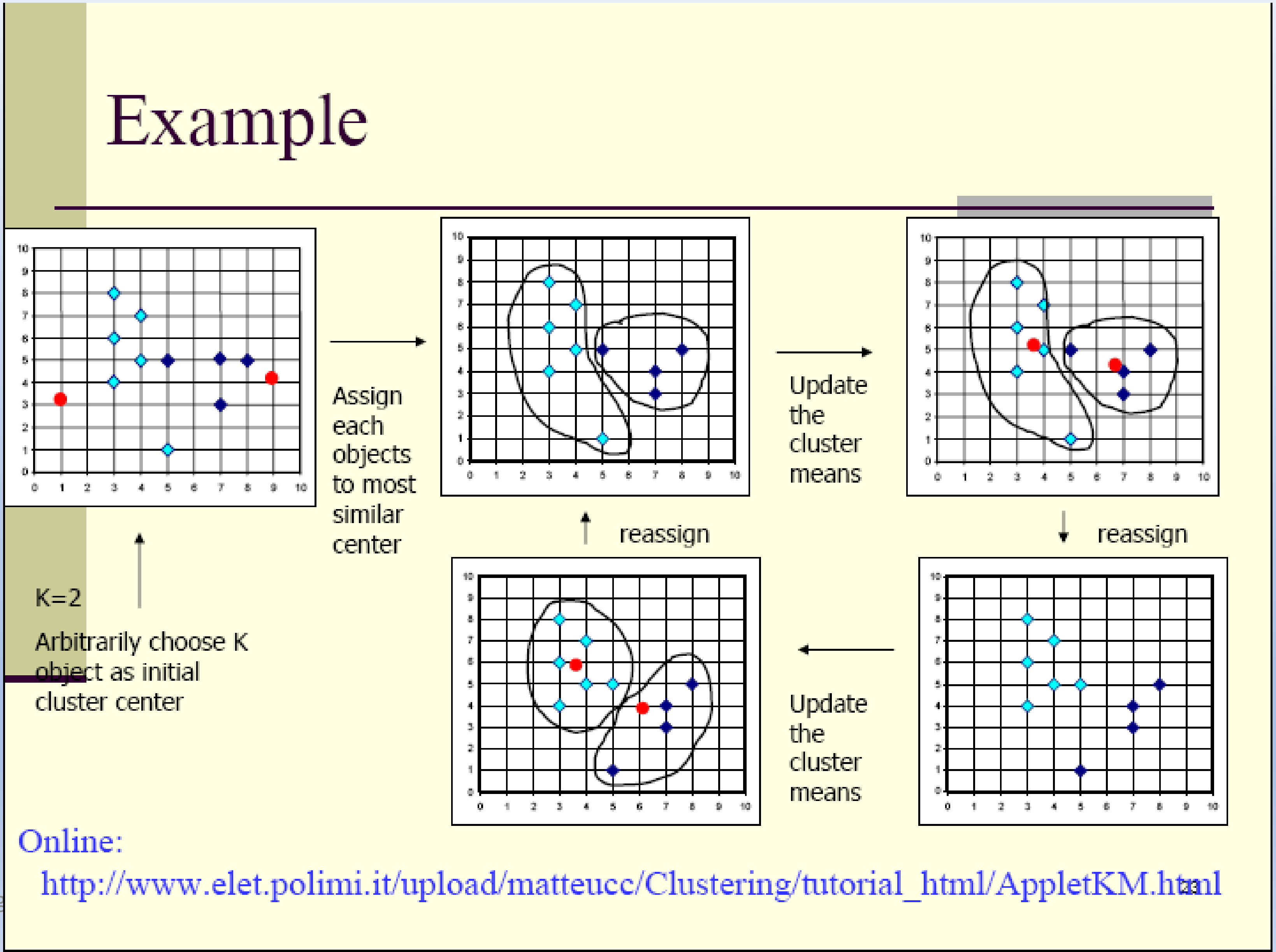
2. Classify n samples into k clusters according to nearest to $\boldsymbol{\mu}_j$.
3. Re-compute all cluster centers (means).

$$\mathcal{M} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \cdots, \boldsymbol{\mu}_k\} \qquad \boldsymbol{\mu}_j = \frac{1}{|\mathcal{D}_j|} \sum_{\mathbf{x}_i \in \mathcal{D}_j} \mathbf{x}_i, \quad j = 1, 2, \cdots, k$$

4. Go back step 2 until no change in .

- The computational complexity of the algorithm is $O(ndkT)$ where d the number of features and T the number of iterations. In practice, the number of iterations is generally much less than the number of samples.

Unsupervised Learning and Clustering



Example

K=2
Arbitrarily choose K object as initial cluster center
Assign each objects to most similar center
Update the cluster means
reassign
reassign
Update the cluster means

Online:
http://www.elet.polimi.it/upload/matteucc/Clustering/tutorial_html/AppletKM.html

Unsupervised Learning and Clustering

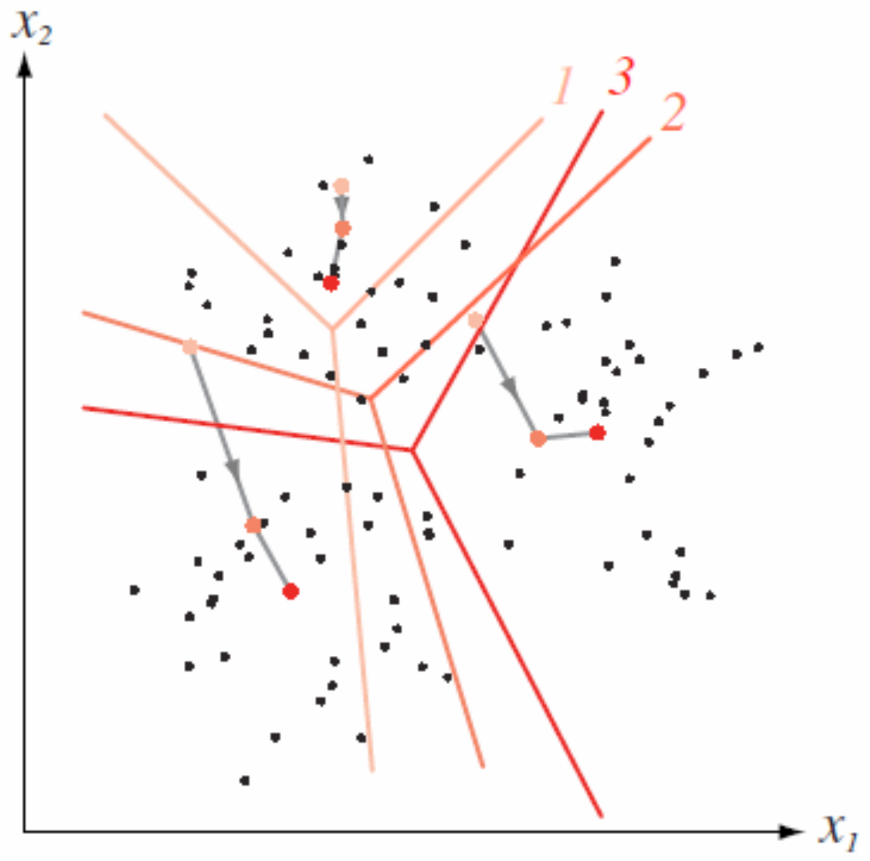


FIGURE 10.3. Trajectories for the means of the *k*-means clustering procedure applied to two-dimensional data. The final Voronoi tesselation (for classification) is also shown— the means correspond to the "centers" of the Voronoi cells. In this case, convergence is obtained in three iterations. From: Richard O. Duda, Peter E. Hart, and David G. Stork,

Unsupervised Learning and Clustering

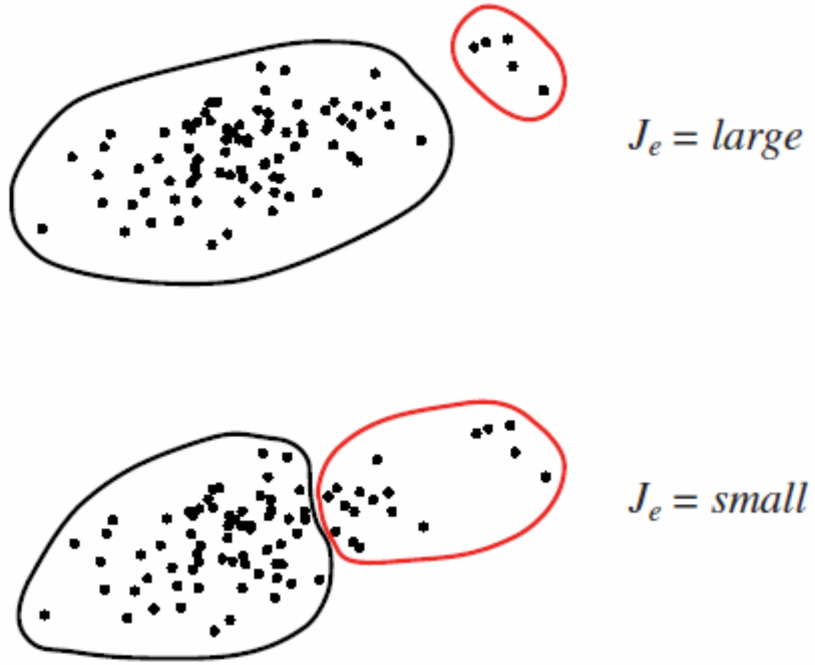


FIGURE 10.10. When two natural groupings have very different numbers of points, the clusters minimizing a sum-squared-error criterion J_e of Eq. 54 may not reveal the true underlying structure. Here the criterion is smaller for the two clusters at the bottom than for the more natural clustering at the top. From: Richard O. Duda, Peter E. Hart, and

M. Liu, X.D. Jiang and A. Kot, "A Multi-Prototype Clustering Algorithm," *Pattern Recognition*, vol. 42, no. 5, pp. 689-698, May 2009.

Unsupervised Learning and Clustering

Model-based : A model is hypothesized for each of the clusters and the idea is to find the best fit of that model to each other.

- Assumption -- the functional forms for the underlying probability densities are known and that the only thing that must be learned is the value of an unknown parameter vector.
- Further assumptions:
  - The samples come from a known number *of* classes c
  - The prior probabilities $P(\omega_j)$ for each class are known, $(j = 1, \ldots, c)$
  - $P(x|\omega_j, \theta_j)$ $(j = 1, \ldots, c)$ are known.
  - The values of the c parameter vectors $\theta_1, \theta_2, \ldots, \theta_c$ are unknown

Unsupervised Learning and Clustering

Model-based :

- The category labels are unknown

$$p(\mathbf{x}\,|\,\boldsymbol{\theta}) = \sum_{j=1}^{c} \overbrace{p(\mathbf{x}\,|\,\omega_j, \boldsymbol{\theta}_j)}^{\text{component densities}} \bullet \underbrace{P(\omega_j)}_{\text{mixing parameters}} \quad , \quad \text{where } \boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_c)^T$$

- This density function is called a mixture density.
- Our goal will be to use samples drawn from this mixture density to estimate the unknown parameter vector $\boldsymbol{\theta}$.
- Once $\boldsymbol{\theta}$ is known, we can decompose the mixture into its components and use a MAP classifier on the derived densities.
- $p(\mathbf{x}|\omega_j,\boldsymbol{\theta}_j) \sim N(\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j) \rightarrow$ Gaussian Mixture Model (GMM).
- A popular technique to solve this problem is expectation-maximization (EM) algorithm.