Pattern Classification - Chapter 4
Nonparametric techniques

R.O. Duda, P.E. Hart, D.G. Stork

Alexandru Ionescu
22.01.2019

Introduction

- In most pattern recognition applications, the common parametric forms of the underlying density function rarely fit the densities actually encountered in practice.
- Classical parametric densities are unimodal (have a single local maximum), whereas many practical problems involve multimodal densities.
- Types of nonparametric methods in pattern recognition:
  - Procedures for estimating the density functions $p(\mathrm{x}|\omega_j)$ from sample patterns; if these estimates are satisfactory, they can be substituted for the true densities when designing the classifier.
  - Procedures for directly estimating the a posteriori probabilities $P(\omega_j|\mathrm{x})$; closely related to nonparametric design procedures such as the nearest-neighbor rule, which bypass probability estimation and go directly to decision functions.
  - Procedures for transforming the feature space in the hope that it may be possible to employ parametric methods in the transformed space.

Density estimation

- Most fundamental techniques rely on the fact that the probability P that a vector x will fall in a region $\mathcal{R}$ is given by

$$P = \int_{\mathcal{R}} p(\mathrm{x}')d\mathrm{x}'.$$

- We can estimate a smoothed value of the density function $p(\mathrm{x})$ by estimating the probability P.
- For n samples $\mathrm{x}_1, \ldots, \mathrm{x}_n$ from a i.i.d. distribution with the probability law $p(\mathrm{x})$, the probability that k of them fall in $\mathcal{R}$ is given by

$$P_k = C_n^k P^k (1-P)^{n-k},$$

  with the expected value for k, $\varepsilon[k] = nP$.
- The ration k/n is a very good estimate for the probability P, and hence for the smoothed density function.
- Under the assumptions that $p(\mathrm{x})$ is continuous and that the region $\mathcal{R}$ is so small that p does not vary appreciably within it,

$$\int_{\mathcal{R}} p(\mathrm{x}')d\mathrm{x}' \approx p(\mathrm{x})V,$$

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

$$p(\mathrm{x}) \approx \frac{k/n}{V}.$$

- The procedure for estimating the density at x implies forming a sequence of regions $\mathcal{R}_1, \mathcal{R}_2, \ldots,$ containing x – the first region to be used with one sample, the second with two, and so on.
- V_n – the volume of $\mathcal{R}_n$, k_n – the number of samples falling in $\mathcal{R}_n$, $p_n(\mathrm{x})$ – the n-th estimate for $p(\mathrm{x})$:

$$p_n(\mathrm{x}) \approx \frac{k_n/n}{V_n}.$$

- The three conditions for $p_n(\mathrm{x})$ to converge to $p(\mathrm{x})$:
  - $\lim_{n\to\infty} V_n = 0$ (the space averaged P/V converges to $p(\mathrm{x})$)
  - $\lim_{n\to\infty} k_n = \infty$ (the frequency ratio converges in probability to the probability P)
  - $\lim_{n\to\infty} k_n/n = 0$ (necessary for $p_n(\mathrm{x})$ to converge at all; although a huge number of samples will eventually fall within the small region $\mathcal{R}_n$, they will form a negligibly small fraction of the total number of samples)

- The first common way of obtaining adequate sequences of regions is to shrink an initial region by specifying the volume V_n as some function of n, such as $V_n = 1/\sqrt{n}$, and to prove that $p_n(\mathrm{x}) \to p(\mathrm{x})$ (the *Parzen-window method*).
- The second method is to specify k_n as some function of n, such as $k_n = \sqrt{n}$. Here, the volume V_n is grown until it encloses k_n neighbors of x (the *k_n-nearest neighbor estimation* method).

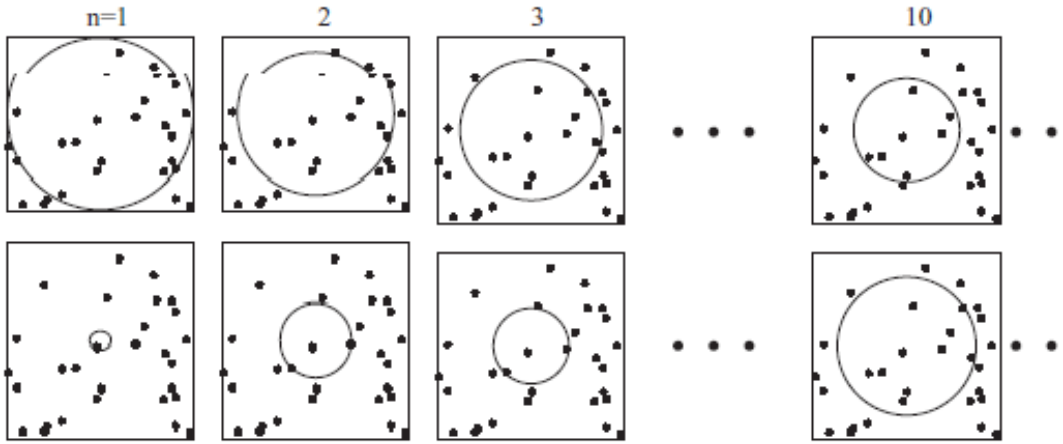


Figure: Two methods for estimating the density at the center of each square.

Parzen windows

- Assume that the region $\mathcal{R}_n$ is a d-dimensional hypercube, with h_n – the length of an edge; its volume:

$$V_n = h_n^d.$$

- Define the *window function* (describing a unit hypercube centered at the origin):

$$\varphi(\mathrm{u}) = \begin{cases} 1, & |u_j| \leq 1/2,\ j = 1, \ldots, d \\ 0, & \textit{otherwise}. \end{cases}$$

- Thus, $\varphi((\mathrm{x} - \mathrm{x}_i)/h_n)$ is equal to unity if x_i falls within the hypercube of volume V_n centered at x, and is zero otherwise; the number of samples in this hypercube is

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

- Hence, the estimate is

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

(the window function is used for interpolation).

- For the estimate $p_n(\mathrm{x})$ to be a legitimate density function, the window function must be a density function itself, i.e.,

$$\varphi(\mathrm{x}) \geq 0$$

$$\int \varphi(\mathrm{u}) d\mathrm{u} = 1$$

- To study the effect of the *window width* h_n over $p_n(\mathrm{x})$, take

$$\delta_n(\mathrm{x}) = \frac{1}{V_n}\varphi\left(\frac{\mathrm{x}}{h_n}\right),$$

such that

$$p_n(\mathrm{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{V_n}\delta(\mathrm{x} - \mathrm{x}_i).$$

- If h_n is very large, the amplitude of δ_n is small, and x must be far from x_i before $\delta_n(\mathrm{x} - \mathrm{x}_i)$ changes much from $\delta_n(0)$; in this case, $p_n(\mathrm{x})$ is the superposition of n broad, slowly changing functions and is a very smooth "out-of-focus" estimate of $p(x)$.
- If h_n is very small, the peak value of $\delta_n(\mathrm{x} - \mathrm{x}_i)$ is large and occurs near $\mathrm{x} = \mathrm{x}_i$; in this case, $p(x)$ is the superposition of n sharp pulses centered at the samples - an erratic, "noisy" estimate.
- For any value of h_n, the distribution is normalized, such that as h_n approaches zero, $\delta_n(\mathrm{x} - \mathrm{x}_i)$ approaches a Dirac delta function centered at x_i, and $p_n(\mathrm{x})$ approaches a superposition of delta functions centered at the samples.

- If V_n is too large, the estimate will suffer from too little resolution; if V_n is too small, the estimate will suffer from too much statistical variability.

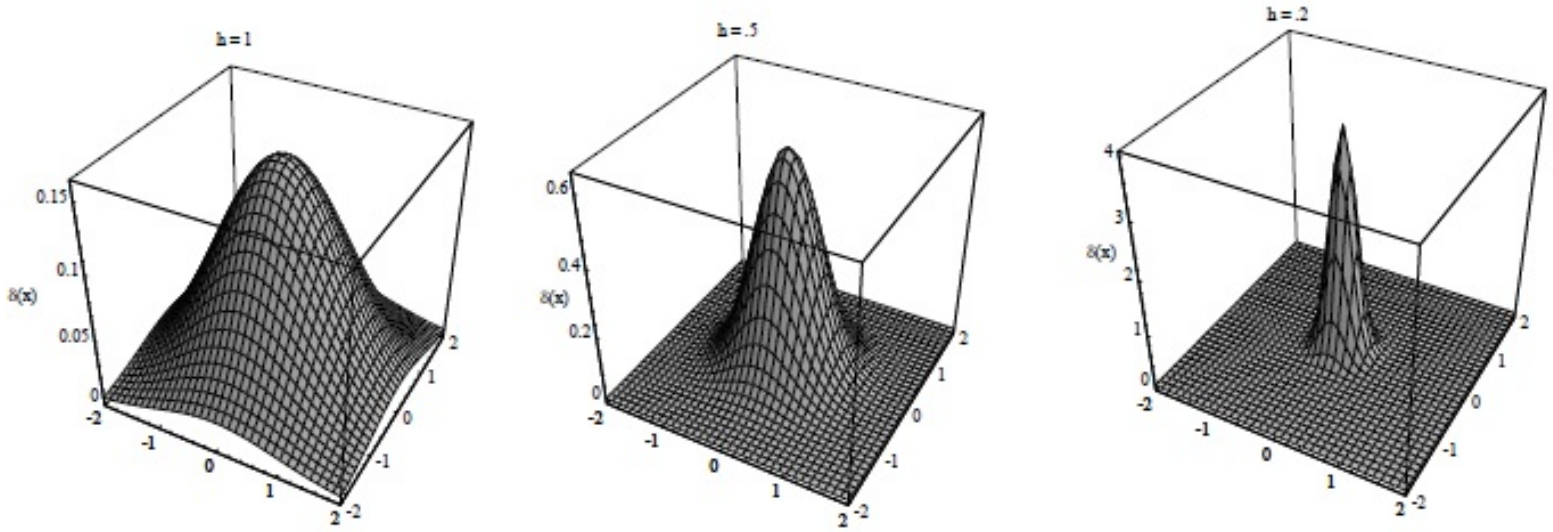


Figure: Two-dimensional symmetric normal Parzen windows $\varphi(\mathrm{x}/h)$ for three different values of h.

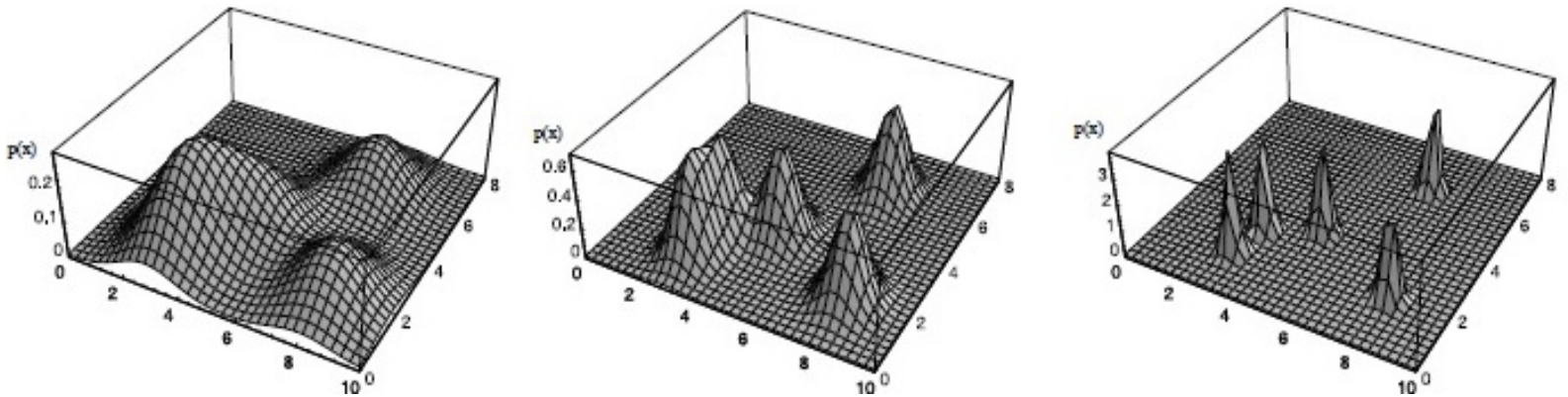


Figure: Parzen-window density estimates based on the same set of five samples, using the window functions in the previous figure.

Example 1

- $p(x)$ = a zero-mean, unit-variance, univariate normal density;
- the window function $\varphi(u) = \frac{1}{\sqrt{2\pi}} e^{-u^2/2}$;
- $h_n = h_1/\sqrt{n}$, where h_1 – parameter;
- $p_n(x)$ is an average of normal densities centered at the samples:

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

- For $n = 1$, $p_n(x)$ is merely a single Gaussian centered about the first sample.
- For $n = 10$ and $h_1 = 0.1$, the contributions of the individual samples are clearly discernible; this is not the case for $h_1 = 1$ and $h_1 = 5$.
- As n gets larger, the ability of $p_n(x)$ to resolve variations in $p(x)$ increases; also, $p_n(x)$ appears to be more sensitive to local sampling irregularities when n is large.
- $p_n(x)$ will converge to the smooth normal curve as n goes to infinity.

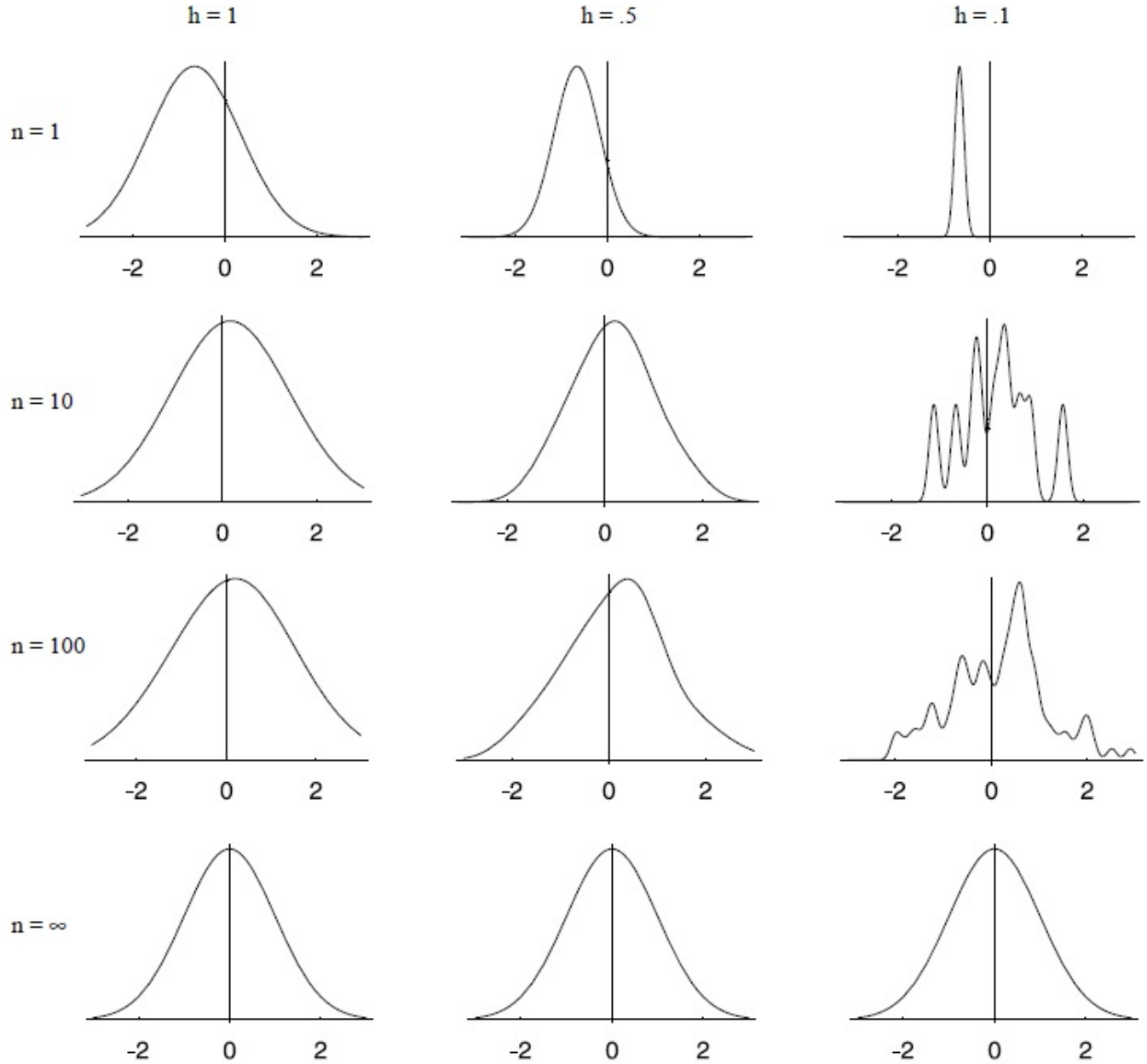


Figure: Parzen-window estimates of a univariate normal density using different window widths and numbers of samples.

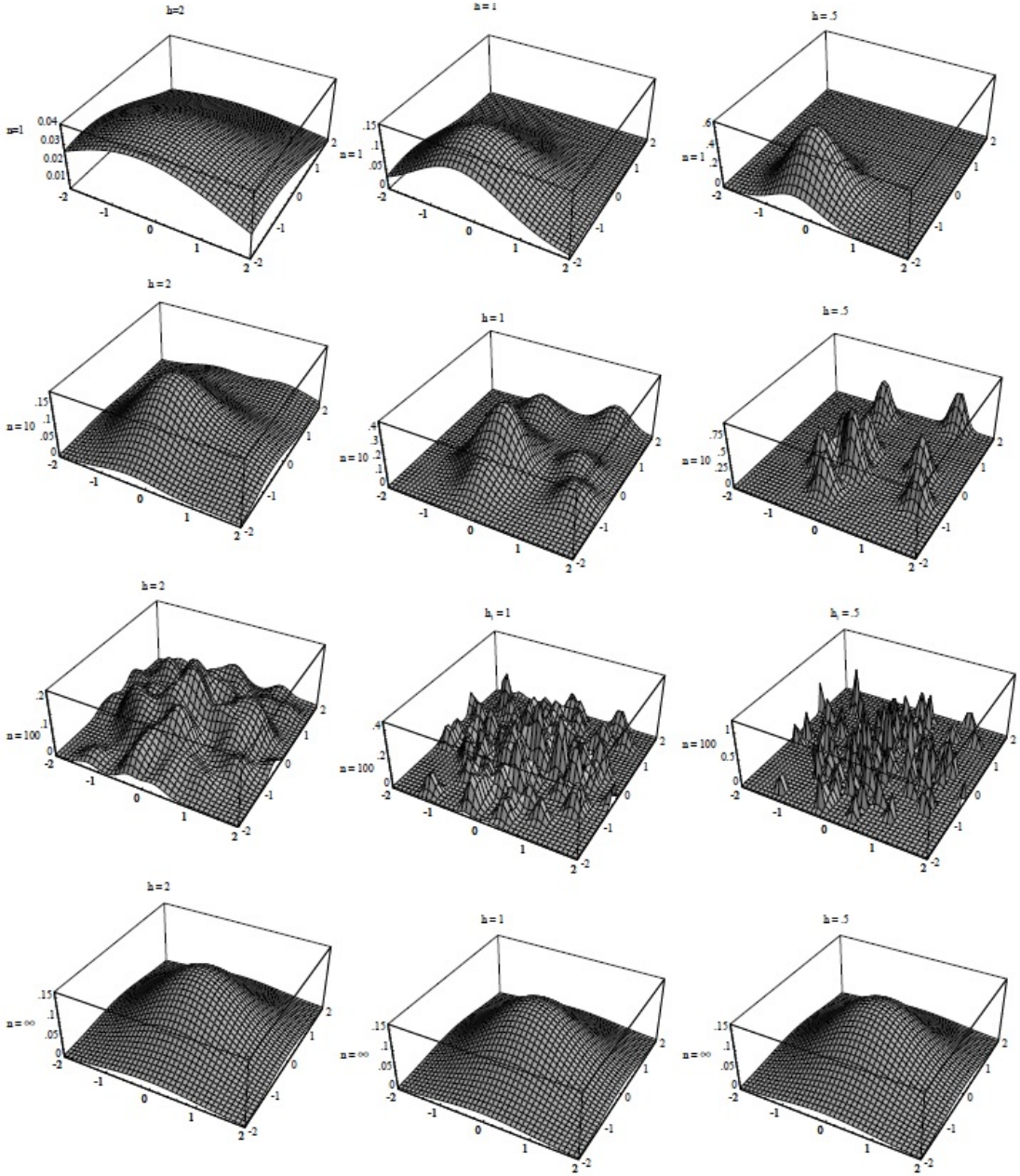
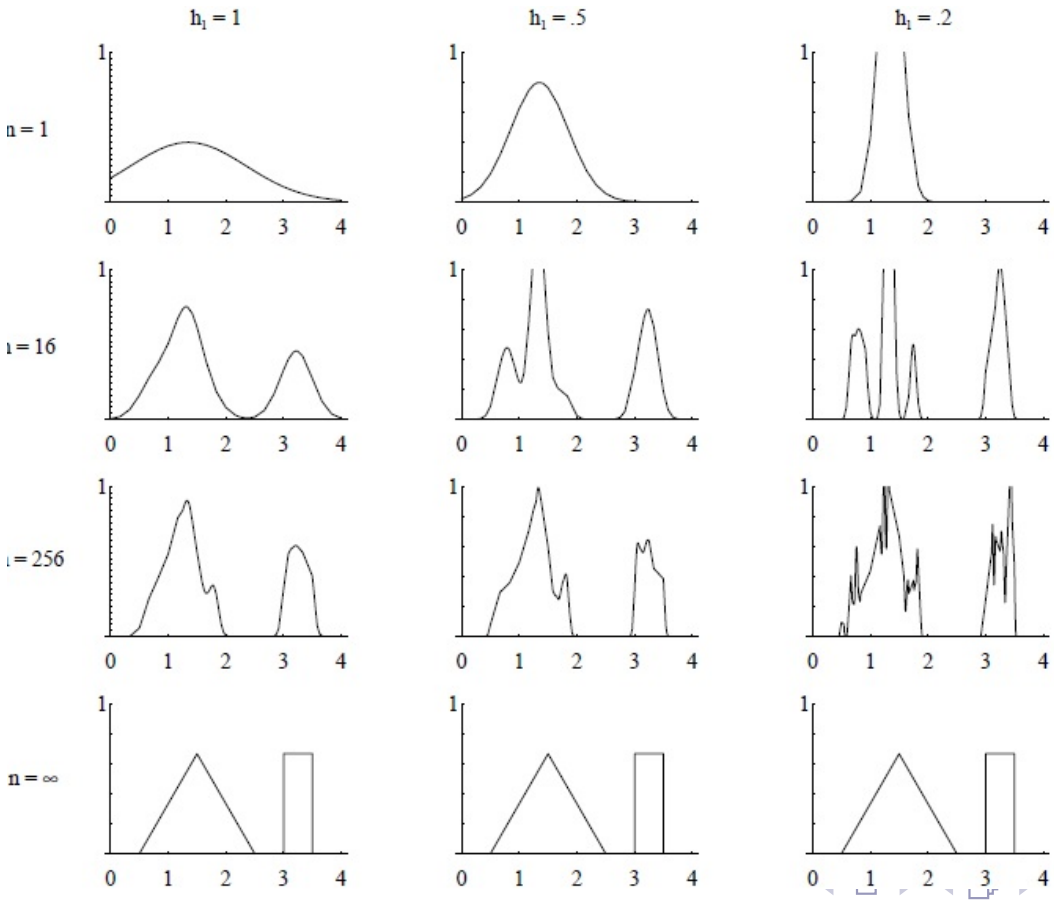


Figure: Parzen-window estimates of a bivariate normal density using different window widths and numbers of samples.

Example 2

- Take $\varphi(x)$ and h_n to be the same as in Example 1.
- $\varphi(x)$ is a mixture of two uniform densities:

$$p(x) = \begin{cases} 1, & -2.5 < x < -2 \\ 1/4, & 0 < x < 2 \\ 0, & \textit{otherwise} \end{cases}$$



Classification example

- Estimation is carried out on the densities for each category and a test point is classified by the label corresponding to the maximum posterior.
- In general, the training error – the empirical error on the training points themselves – can be made arbitrarily low by making the window width sufficiently small; this does *not* guarantee a small test error.
- In the absence of some information about the underlying distributions, there is little theoretical justification of one window width over another.
- The power of nonparametric methods resides in their generality. With enough samples, we are essentially assured of convergence to an arbitrarily complicated target density.
- On the other hand, the number of samples needed may be very large indeed much greater than would be required if we knew the form of the unknown density. These methods have severe requirements for computation time and storage. Moreover, the demand for a large number of samples grows exponentially with the dimensionality of the feature space.

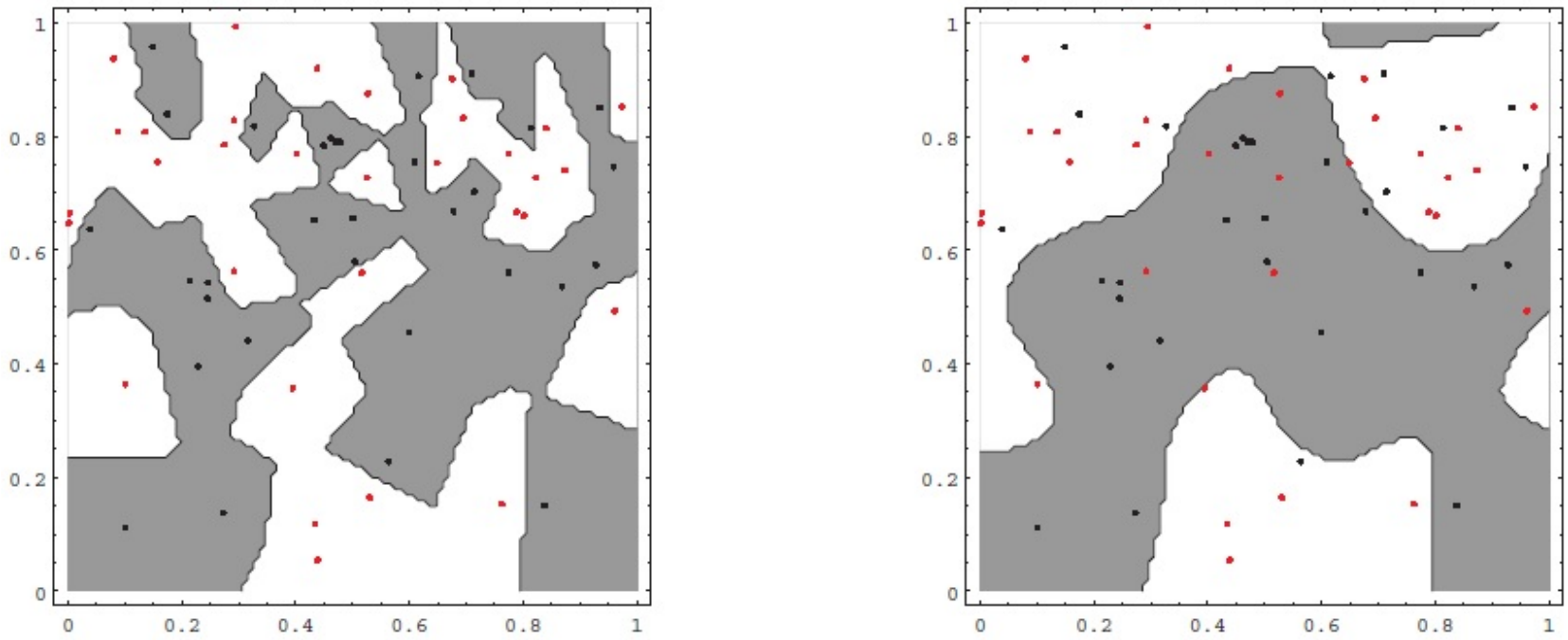
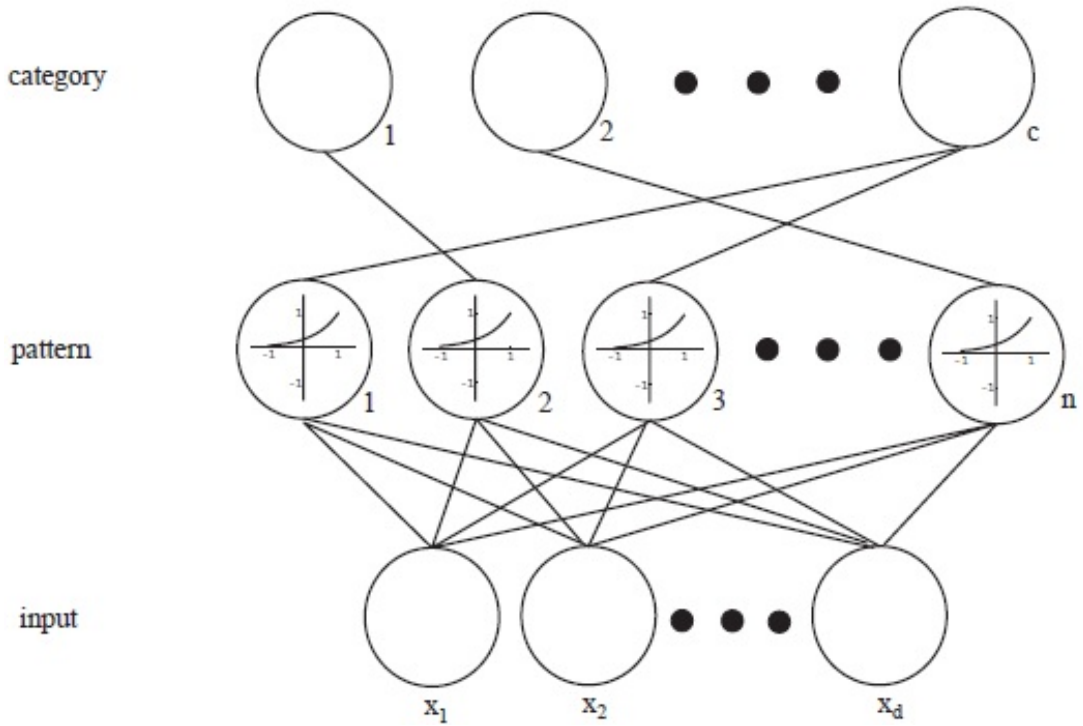


Figure: The decision boundaries in a two-dimensional Parzen-window dichotomizer depend on the window width h. At the left a small h leads to boundaries that are more complicated than for large h on same data set, shown at the right. Apparently, for this data a small h would be appropriate for the upper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

Probabilistic Neural Networks (PNNs)

- We wish to form a Parzen estimate based on n patterns, each of which is d-dimensional, randomly sampled from c classes.
- The connections from the d input units to the n pattern units represent modifiable weights, which will be trained. Each link from a pattern unit to its associated category unit is of a single constant magnitude.



- PNN training
  1. Each pattern $\mathbf{x}$ of the training set is normalized such that $\sum_{i=1}^{d} x_i^2 = 1$.
  2. The first normalized training pattern is placed on the input units; the modifiable weights linking the input units and the first pattern unit are set such that $w_1 = x_1$.
  3. A single connection from the first pattern unit is made to the category unit corresponding to the known class of that pattern.
  4. The process is repeated with each of the remaining training patterns, setting the weights to the successive pattern units such that $w_k = x_k$ for $k = 1, 2, \ldots, n$.

```
1 begin initialize j = 0, n = #patterns
2         do j ← j + 1
3             normalize : x_jk ← x_jk / ( Σ_i x_ji^2 )^(1/2)
4             train : w_jk ← x_jk
5             if x ∈ ω_i then a_ic ← 1
6         until j = n
7 end
```

- PNN classification
  1. A normalized test pattern x is placed at the input units; each pattern unit computes the inner product
  $$z_k = \mathrm{w}_k^t \mathrm{x}$$
  and emits a nonlinear function of z_k.
  2. Each output unit sums the contributions from all pattern units connected to it. The nonlinear function is $e^{(z_k - 1)/\sigma^2}$, where σ is a parameter set by the user.
  3. Each pattern unit contributes to its associated category unit a signal equal to the probability the test point was generated by a Gaussian centered on the associated training point; the sum of these local estimates (computed at the corresponding category unit) gives the discriminant function $g_i(\mathrm{x})$ - the Parzen window estimate of the underlying distribution.
  4. The desired category for the test point is $\max_i g_i(\mathrm{x})$.

```
1 begin initialize k = 0, x = test pattern
2         do k ← k + 1
3             z_k ← w_k^t x
4             if a_kc = 1 then g_c ← g_c + exp[(z_k − 1)/σ^2]
5         until k = n
6   return class ← arg max_i g_i(x)
7 end
```

- PNNs advantages:
  1. speed of learning (the learning rule $w_k = x_k$ is simple and requires only a single pass through the training data)
  2. new training patterns can be incorporated into a previously trained classifier quite easily
- PNNs disadvantage: space complexity ($O((n+1)d)$).
- **Choosing the window function** is the main problem in the Parzen-window/PNN approach.

k_n-Nearest-Neighbour Estimation

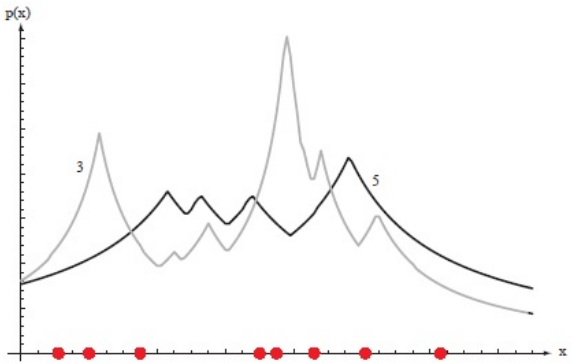
- Let the cell volume be a function of the training data, rather than some arbitrary function of the overall number of samples.
- To estimate $p(\mathrm{x})$ from n *prototypes*, center a cell about x and let it grow until it captures k_n samples, where k_n is some specified function of n.
- If the density is high near x, the cell will be relatively small; if the density is low, the cell will grow large, but it will stop soon after it enters regions of higher density.
- Take

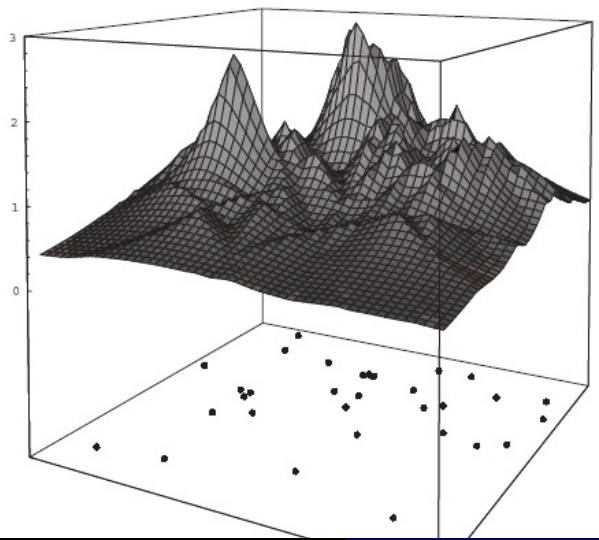
$$p_n(\mathrm{x}) = \frac{k_n/n}{V_n}$$

  and ask that $k_n \to \infty$, such that k_n/n is a good estimate of the probability that a point will fall in the cell of volume V_n.
- The ratio k_n/n must go to zero, since k_n should grow sufficiently slowly that the size of the cell needed to capture k_n training samples will shrink to zero.

Examples

- For $k_n = \sqrt{n}$, assuming that $p_n(\mathrm{x})$ is a reasonably good approximation to $p(\mathrm{x})$, then $V_n \approx 1/(\sqrt{n}p(\mathrm{x}))$; hence, the initial volume V_1 is determined by the nature of the data rather than by some arbitrary choice.
- There are nearly always discontinuities in the slopes of these estimates, and these lie away from the prototypes themselves:





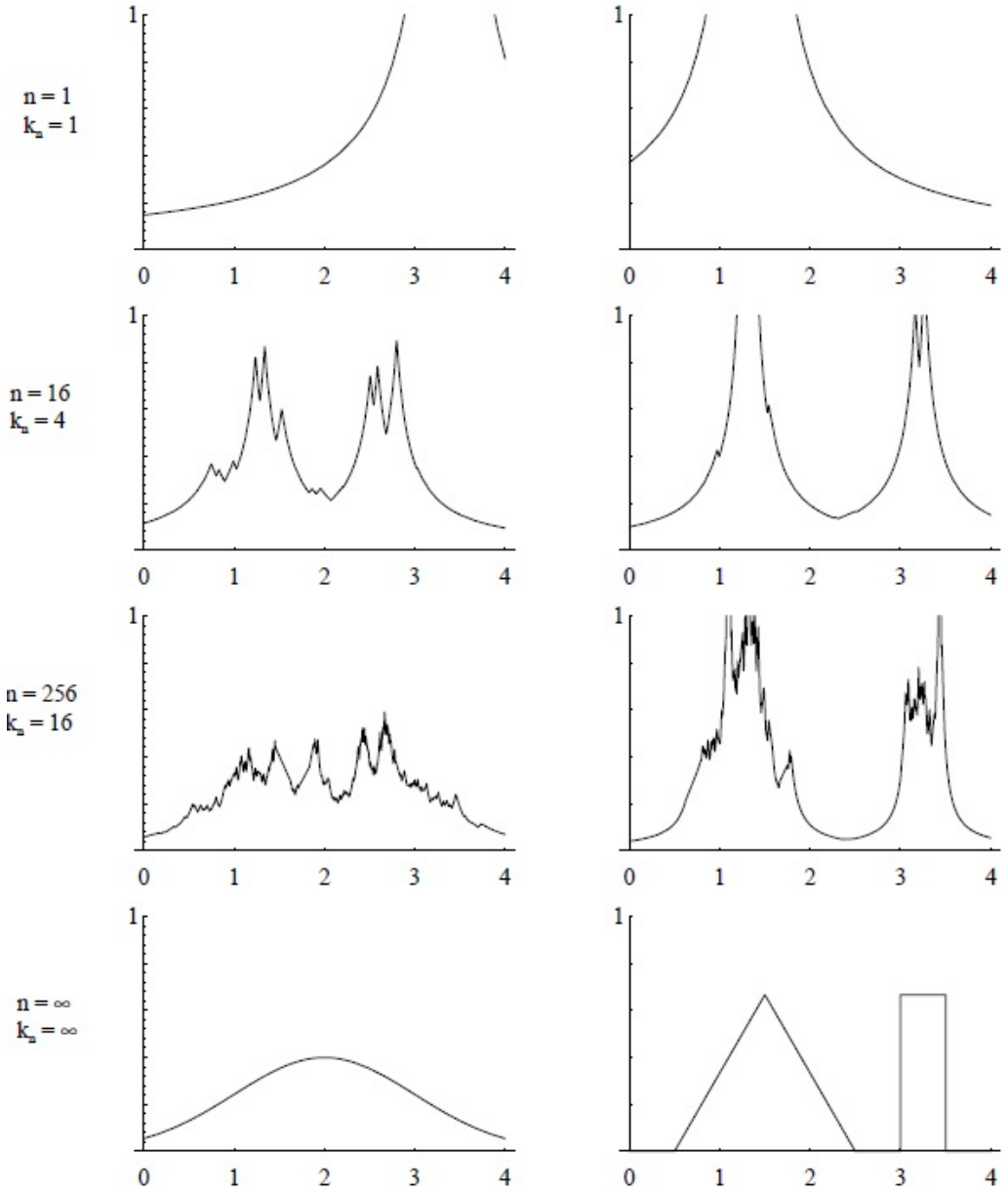


Figure: Several k_n-nearest-neighbor estimates of two unidimensional densities: a Gaussian and a bimodal distribution.

Estimation of a posteriori probabilities

- Place a cell of volume V around x and capture k samples, k_i of which turn out to be labeled ω_i.
- The estimation for the joint probability $p(\mathrm{x}, \omega_i)$ is

$$p_n(\mathrm{x}, \omega_i) = \frac{k_i/n}{V},$$

hence a reasonable estimate for $P(\omega_i|\mathrm{x})$ is

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

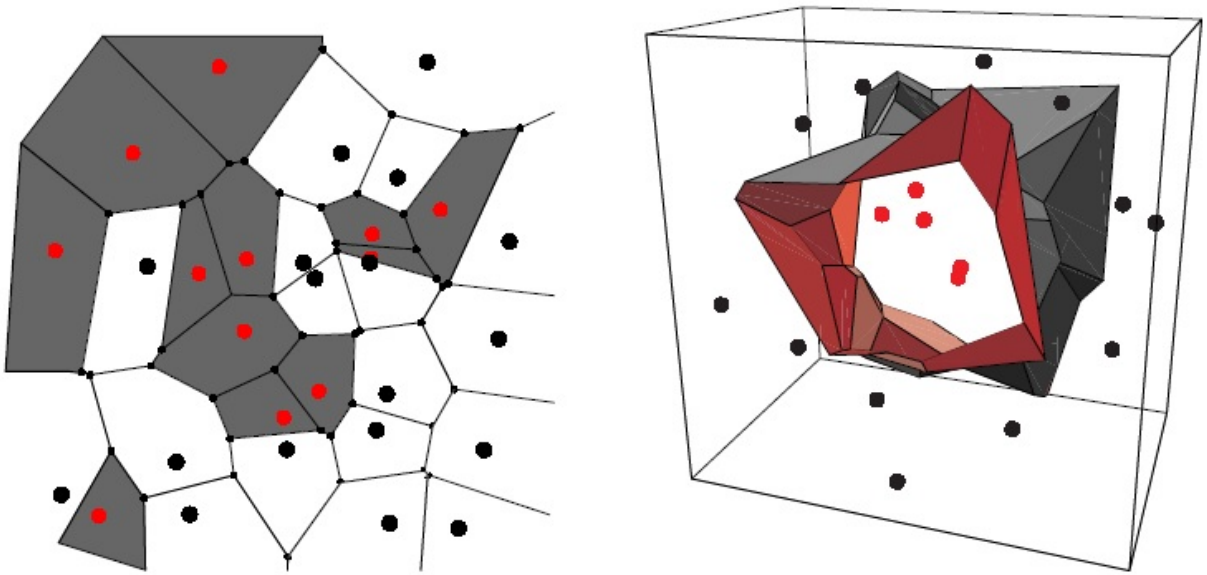
(the estimate of the a posteriori probability that ω_i is the state of nature is merely the fraction of the samples within the cell that are labelled ω_i).

- For minimum error rate, the category most frequently represented within the cell must be selected.
- The size of the cell can be chosen using either the Parzen-window approach (where V_n is some specified function of n), or the k_n-nearest-neighbor approach (where V_n should be expanded until some k samples are captured).
- *The cell volume could become arbitrarily small for $n \to \infty$ and yet contain an arbitrarily large number of samples!*

The Nearest-Neighbor Rule

- $\mathcal{D}^n = \{\mathrm{x}_1, \ldots, \mathrm{x}_n\}$ – a set of n labeled prototypes
- $\mathrm{x}' \in \mathcal{D}^n$ – the prototype nearest to a test point x
- The *nearest-neighbor rule* (assign x the label of x') usually leads to an error rate greater than the minimum possible, the Bayes rate; however, with an unlimited number of prototypes, the error rate is never worse than twice the Bayes rate.
- The label θ' associated with the nearest neighbor is a random variable, and the probability that $\theta' = \omega_i$ is merely the a posteriori probability $P(\omega_i|\mathrm{x}')$. For a very large number of samples, it is reasonable to assume that x' is sufficiently close to x that $P(\omega|\mathrm{x}') = P(\omega_i|\mathrm{x})$ (the nearest-neighbor rule is effectively matching probabilities with nature).

- Define $\omega_m(\mathrm{x})$ by
$$P(\omega_m|\mathrm{x}) = \max_i P(\omega_i|\mathrm{x}),$$
such that Bayes decision rule always selects ω_m.
- We partition the feature space into cells consisting of all points closer to a given training point x' than to any other training points; all points in such a cell are thus labelled by the category of the training point (the *Voronoi tesselation* of the space):



- It can be shown that **at least half of the classification information in an infinite data set resides in the nearest neighbor** (the nearest-neighbor error rate is bounded above by twice the Bayes rate).

The k-Nearest-Neighbor Rule

- We classify x by assigning it the label most frequently represented among the k nearest samples:

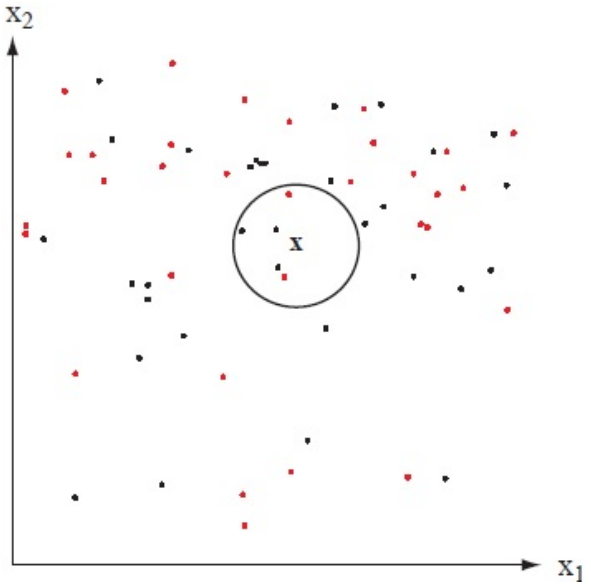


Figure: The case $k = 5$, when test point x would be labeled the category of the black points (in the two-class case, k should be odd).

- The labels on each of the k-nearest-neighbors are random variables, which independently assume the values ω_i with probabilities $P(\omega_i|\mathrm{x})$.
- We want to use a large value of k to obtain a reliable estimate. On the other hand, we want all of the k nearest neighbors x' to be very near x to be sure that $P(\omega_i|\mathrm{x}')$ is approximately the same as $P(\omega_i|\mathrm{x})$; we must choose a compromise k that is a small fraction of the number of samples.

Computational Complexity of the k-Nearest-Neighbor Rule

- For n labeled training samples in d dimensions, seeking to find the closest to a test point $\mathbf{x}$ ($k = 1$) means inspecting each stored point in turn, calculating its Euclidean distance to $\mathbf{x}$, retaining the identity only of the current closest one; the complexity: $O(dn^2)$.
- Algorithmic techniques for reducing the computational burden in nearest-neighbor searches:
  1. *Computing partial distances*: calculate the distance using some partial subset r of the full d dimensions, and if this partial distance is too great we do not compute further.
  2. *Prestructuring*: create some form of search tree in which prototypes are selectively linked; during classification, we compute the distance of the test point to one tree or a few stored "entry" or "root" prototypes and then consider only the prototypes linked to it. Of these, we find the one that is closest to the test point, and recursively consider only subsequent linked prototypes.
  3. *Editing*: eliminate "useless" prototypes during training (prototypes that are surrounded by training points of the same category label); example algorithm of complexity $O(d^3 n^{\lfloor d/2 \rfloor} \ln n)$:

```
1 begin initialize j = 0, D = data set, n = #prototypes
2         construct the full Voronoi diagram of D
3         do j ← j + 1;  for each prototype x'_j
4             Find the Voronoi neighbors of x'_j
5             if any neighbor is not from the same class as x'_j then mark 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
```