Chapter 4

Nonparametric Techniques

Introduction

- Supervised learning under the assumption that the forms of the underlying density functions were known – Chap. 3
  - In most PR applications this assumption is suspect:
    - the common parametric forms rarely fit the densities actually encountered in practice.
  - Many practical problems involve multimodal densities, instead of unimodal
  - High-dimensional density might be accurately represented as the product of one-dimensional functions?

Introduction

- Nonparametric procedures that can be used with arbitrary distributions and without the assumption are introduced
  - Estimating the density functions $p(\mathbf{x}|\omega_j)$ from sample patterns
  - Procedures for directly estimating the a posteriori probabilities $P(\omega_j|\mathbf{x})$.

Density Estimation

- Estimating an unknown probability density function
  - The probability P that a vector $\mathbf{x}$ will fall in a region R is given by

$$P = \int_R p(\mathbf{x}')d\mathbf{x}'$$

    - P is an averaged version of the density function $p(\mathbf{x})$.
  - Smoothed value of p can be estimated by estimating the P.
    - Supposed that $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are drawn i.i.d. according to the probability law $p(\mathbf{x})$. The probability that k of these n fall in R is given by the binomial law.

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

The expected value for k is

$$E[k] = nP$$

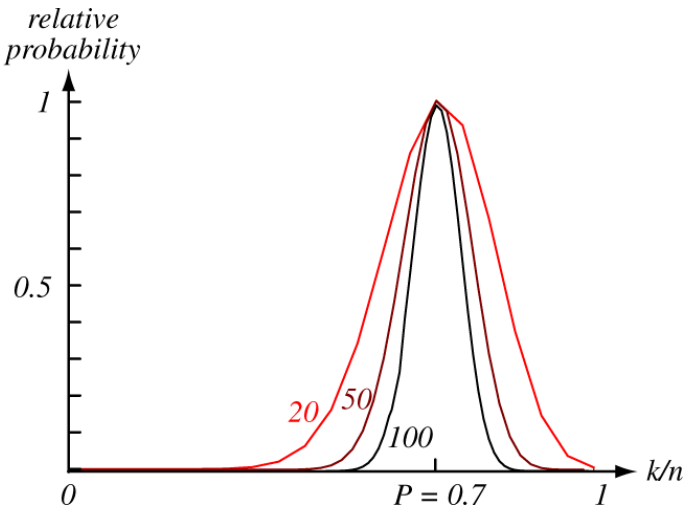
Density Estimation

- This binomial distribution peaks very sharply about the mean.
  - The ratio k/n will be a very good estimate for the P.
  - This estimate is accurate when n is very large.
- If $p(\mathbf{x})$ is continuous and R is so small that p does not vary appreciably within it

$$\int_R p(\mathbf{x}')d\mathbf{x}' \cong p(\mathbf{x})V$$

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

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



Density Estimation

- Problems remaining:
  - If we fix V and take more and more training samples, k/n will converge, but then we have only obtained an estimate of the space-averaged value of $p(\mathbf{x})$

$$\frac{P}{V} = \frac{\int_R p(\mathbf{x}')d\mathbf{x}'}{\int_R d\mathbf{x}'}$$

  - From a practical standpoint:
    - we note that *the number of samples is always limited*. Thus V cannot be allowed to become arbitrarily small. If this kind of estimate is to be used, one will have to accept a certain amount of variance in the k/n and a certain amount of averaging of the density $p(\mathbf{x})$.

Density Estimation

- From a theoretical standpoint:
  - It is interesting to ask how these limitations can be circumvented *if an unlimited number of samples is available*. Let V_n be the volume of R_n, k_n be the number of samples falling in R_n, and $p_n(\mathbf{x})$ be the n-th estimate for $p(\mathbf{x})$:

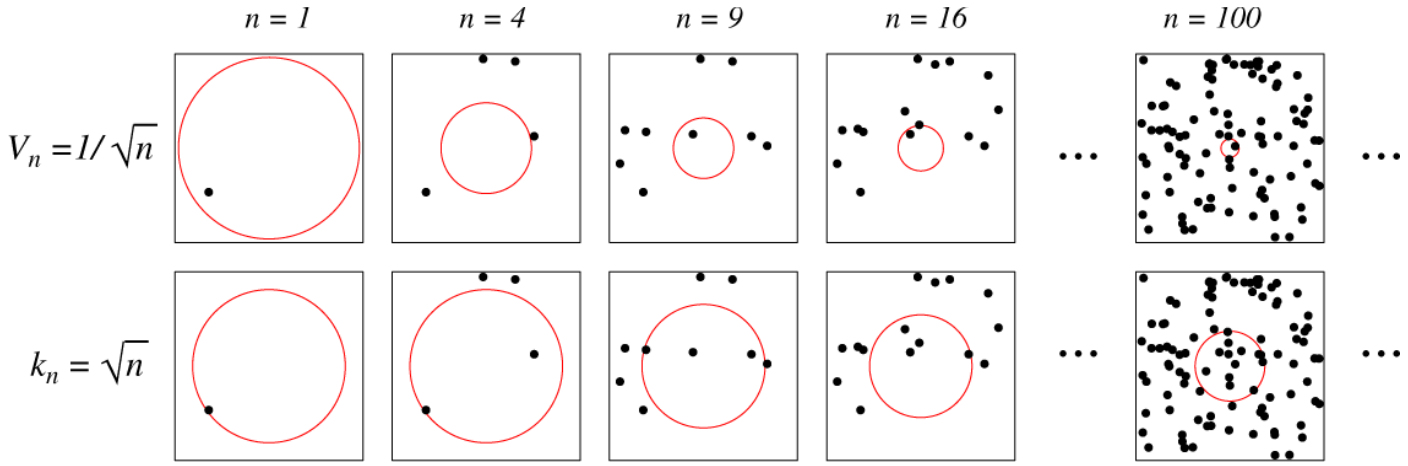
$$p_n(\mathbf{x}) = \frac{k_n/n}{V_n}$$

  - If $p_n(\mathbf{x})$ is to be converge to $p(\mathbf{x})$, following conditions appear to be required

$\lim_{n\to\infty} V_n = 0$	the space averaged P/V will converge to $p(\mathbf{x})$, provided that the regions shrink uniformly and that $p(\cdot)$ is continuous at $\mathbf{x}$.
$\lim_{n\to\infty} k_n = \infty$	It only makes sense if $p(\mathbf{x}) \neq 0$, assures us that the frequency ratio will converge to P.
$\lim_{n\to\infty} k_n/n = 0$	It is clearly necessary if $p_n(\mathbf{x})$ is to converge at all. It also says that although a huge number of samples will eventually fall within the small region R_n, they will form a negligibly small fraction of the total number of samples.

Density Estimation

- Two common ways of obtaining sequences of regions that satisfy these conditions.



  - The sequences in both cases represent random variables that generally converge and allow the true density at the test point to be calculated.

Parzen Windows

- The Parzen-window approach to estimating densities
  - Temporarily assume that the R_n is a d-dimensional hypercube. If h_n is the length of an edge of that hypercube, the its volume is

$$V_n = h_n^d$$

- Window function
  - An analytic expression for k_n, the number of samples falling in the hypercube, can be expressed by defining the window function:

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

  - Thus, the function defines a unit hypercube centered at the origin.
  - $\varphi(\frac{\mathbf{x}-\mathbf{x}_i}{h_n})$ is equal to unity if $\mathbf{x}_i$ falls within the hypercube of V_n centered at $\mathbf{x}$ and zero otherwise.

Parzen Windows

- The number of samples in the hypercube is

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

and the estimate becomes $p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{V_n}\varphi\left(\frac{\mathbf{x} - \mathbf{x}_i}{h_n}\right)$

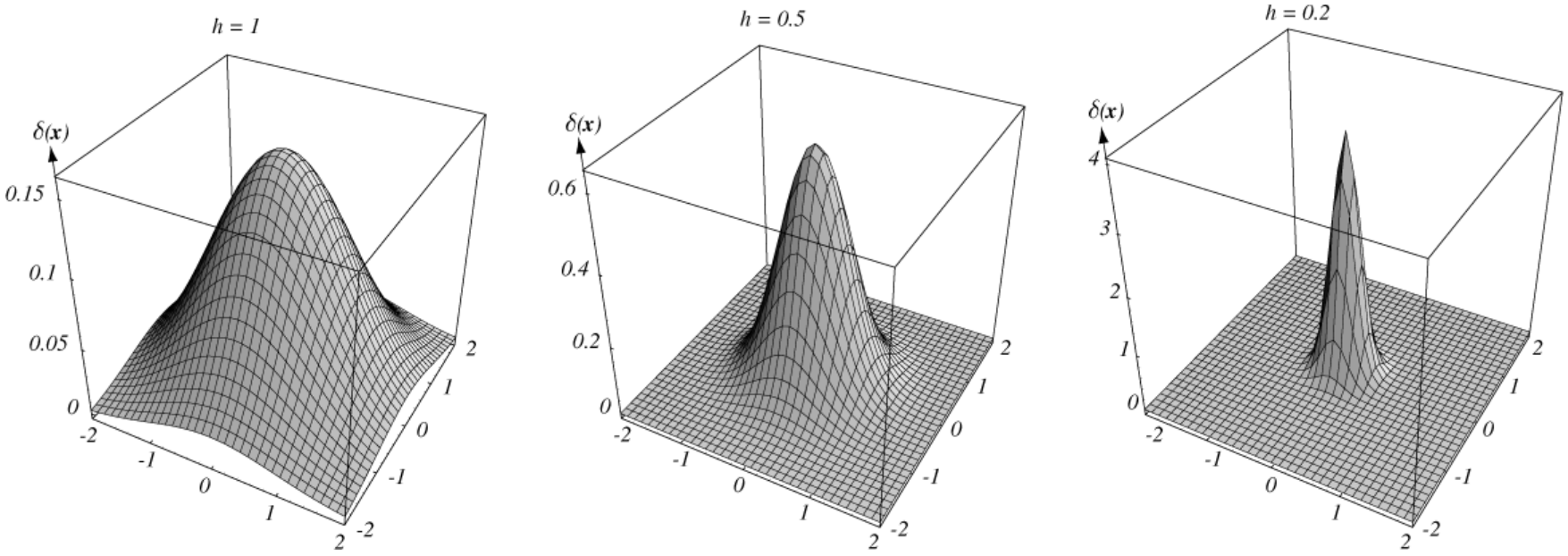
$$p_n(\mathbf{x}) = \frac{k_n/n}{V_n}$$

  - The window function is being used for interpolation – each sample contributing to the estimate in accordance with its distance from $\mathbf{x}$.
- The effect of the *window width* h_n on $p_n(\mathbf{x})$

Let's define $\delta_n(\mathbf{x})$ $\quad \delta_n(\mathbf{x}) = \frac{1}{V_n}\varphi\left(\frac{\mathbf{x}}{h_n}\right)$

$$\Rightarrow \quad p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \delta_n(\mathbf{x} - \mathbf{x}_i)$$

Parzen Windows

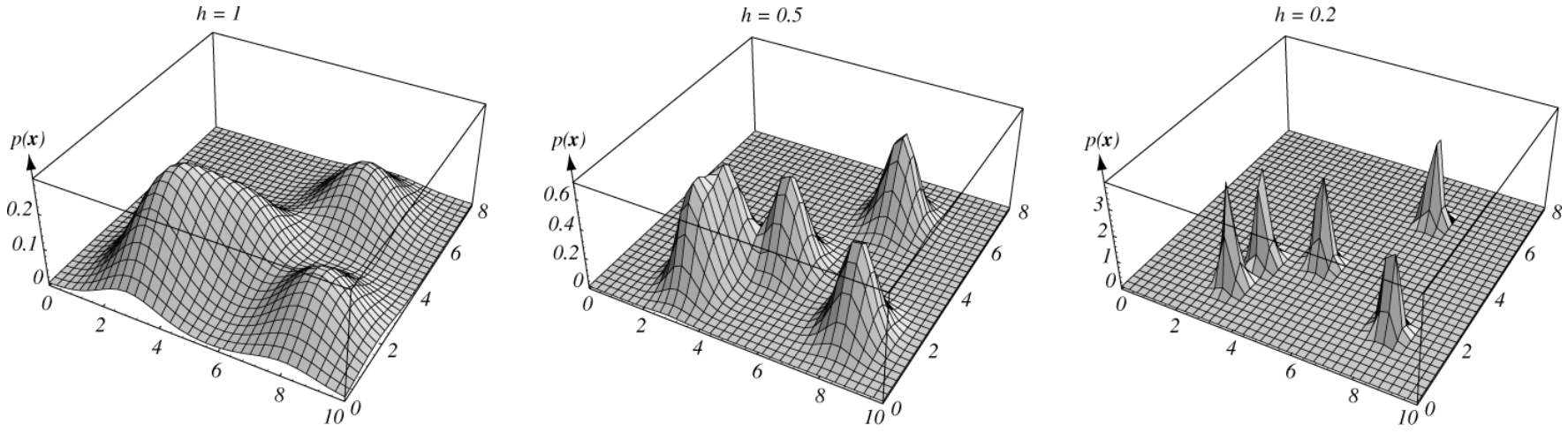


2-D circularly symmetric normal Parzen windows.
Note that because the $\delta(\mathbf{x})$ are normalized, different vertical scales must be used to show their structure.

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

$$\int \delta_n(\mathbf{x}-\mathbf{x}_i)d\mathbf{x} = \int \frac{1}{V_n}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)d\mathbf{x} = \int \varphi(\mathbf{u})d\mathbf{u} = 1$$

Parzen Windows



Parzen-window density estimates based on the same set of five samples. The vertical axes have been scaled to show the structure of each distribution.

$$p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \delta_n(\mathbf{x} - \mathbf{x}_i)$$ $p_n(\mathbf{x})$ is the superposition of n functions.

Parzen Windows

- The choice of h_n (or V_n) has an important effect on $p_n(\mathbf{x})$.
  - Too large: the estimate suffers from too little resolution
  - Too small: the estimate suffers from too much statistical variability.
  - With a limited number of samples: acceptable compromise is needed.
  - With an unlimited number of samples, let V_n slowly approach zero as n increases and have $p_n(\mathbf{x})$ converge to the $p(\mathbf{x})$.

- Convergence

$$\lim_{n\to\infty} \bar{p}_n(\mathbf{x}) = p(\mathbf{x})$$

and

$$\lim_{n\to\infty} \sigma_n^2(\mathbf{x}) = 0$$

Parzen Windows

- Conditions needed to prove convergence
  - Continuity of $p(\bullet)$ at $\mathbf{x}$ is required, and the conditions imposed by Eqs. 12 and 13 are invoked.
  - Additional conditions assure convergence:

$$\sup_{\mathbf{u}} \varphi(\mathbf{u}) < \infty$$

$$\lim_{\|\mathbf{u}\|\to\infty} \varphi(\mathbf{u}) \prod_{i=1}^{d} u_i = 0$$

keep $\varphi(\bullet)$ well-behaved, and they are satisfied by most density functions.

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

$$\lim_{n\to\infty} nV_n = \infty$$

The volume V_n must approach zero, but at a rate slower than $1/n$.

Parzen Windows

- Convergence of the Mean

$$\begin{aligned}\bar{p}_n(\mathbf{x}) &= E[p_n(\mathbf{x})] \\ &= \frac{1}{n}\sum_{i=1}^{n} E\left[\frac{1}{V_n}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)\right] \\ &= \int \frac{1}{V_n}\varphi\left(\frac{\mathbf{x}-\mathbf{v}}{h_n}\right)p(\mathbf{v})d\mathbf{v} \\ &= \int \delta_n(\mathbf{x}-\mathbf{v})\,p(\mathbf{v})d\mathbf{v}\end{aligned}$$

  - The expected value of the estimate is an averaged value of the unknown density
    - A convolution of the unknown density and the window function.
  - A blurred version of $p(\mathbf{x})$ as seen through the averaging window.
  - As V_n approaches zero, $\delta_n(\mathbf{x}-\mathbf{v})$ approaches a delta function centered at $\mathbf{x}$. Thus, if p is continuous at $\mathbf{x}$, the mean will approach $p(\mathbf{x})$ as n approaches infinity.

Parzen Windows

- Convergence of the Variance

$$\sigma_n^2(\mathbf{x}) = \sum_{i=1}^{n} E\left[\left(\frac{1}{nV_n}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right) - \frac{1}{n}\bar{p}_n(\mathbf{x})\right)^2\right]$$

$$= nE\left[\frac{1}{n^2V_n^2}\varphi^2\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)\right] - \frac{1}{n}\bar{p}_n^2(\mathbf{x}) \quad \Rightarrow \quad \sigma_n^2(\mathbf{x}) \le \frac{\sup(\varphi(\cdot))\bar{p}_n(\mathbf{x})}{nV_n}$$

$$= \frac{1}{nV_n}\int \frac{1}{V_n}\varphi^2\left(\frac{\mathbf{x}-\mathbf{v}}{h_n}\right)p(\mathbf{v})d\mathbf{v} - \frac{1}{n}\bar{p}_n^2(\mathbf{x})$$

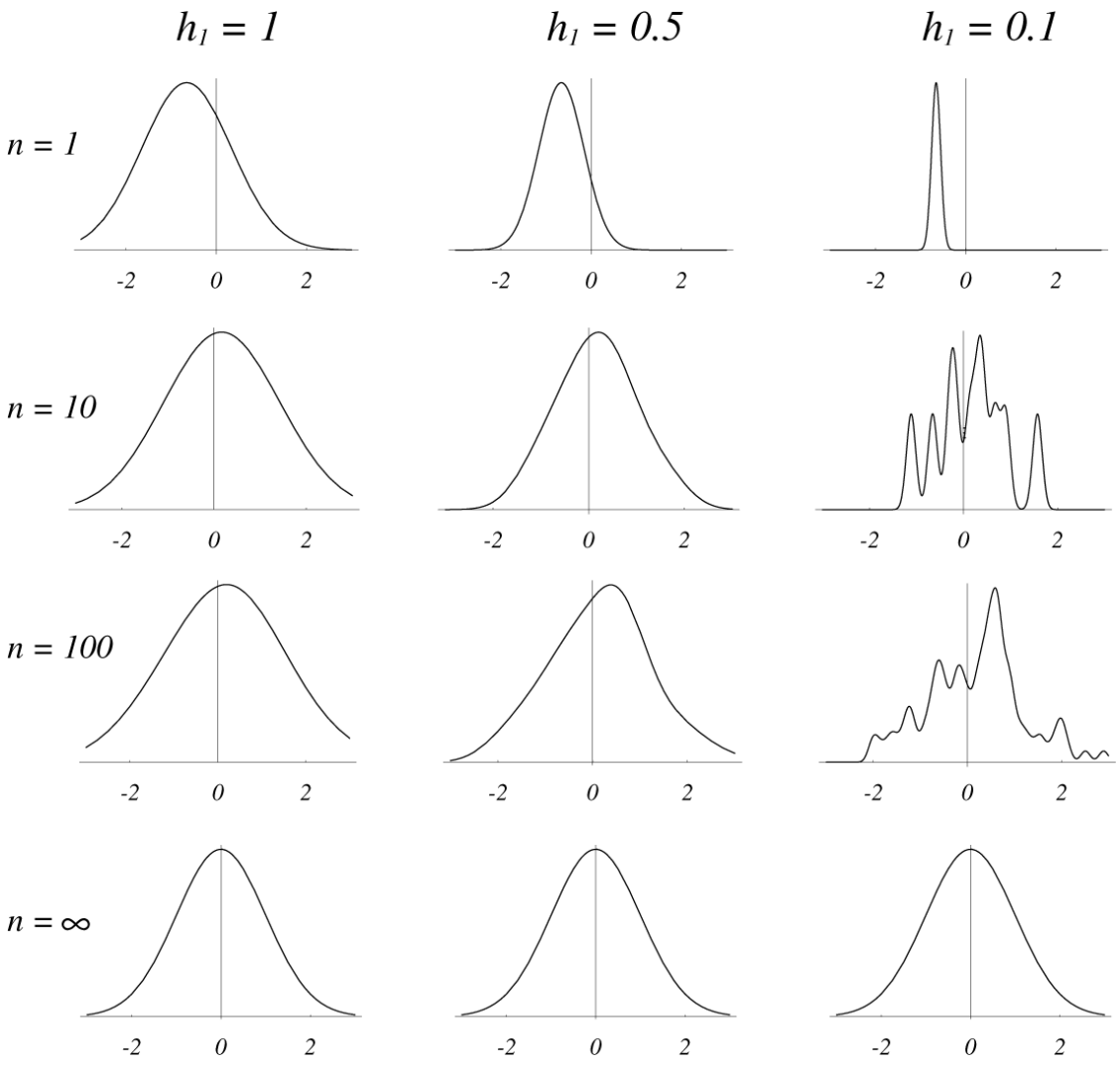
  - To obtain a small variance we want a large value for V_n. *But* a large V_n smoothes out the local variations in density.
  - However, because the numerator stays finite as n approaches infinity, we can let V_n approach zero and still obtain zero variance, provided that nV_n approaches infinity.

$$V_n = V_1/\sqrt{n} \text{ or } V_n = V_1/\ln n$$

Parzen Windows

- Illustration

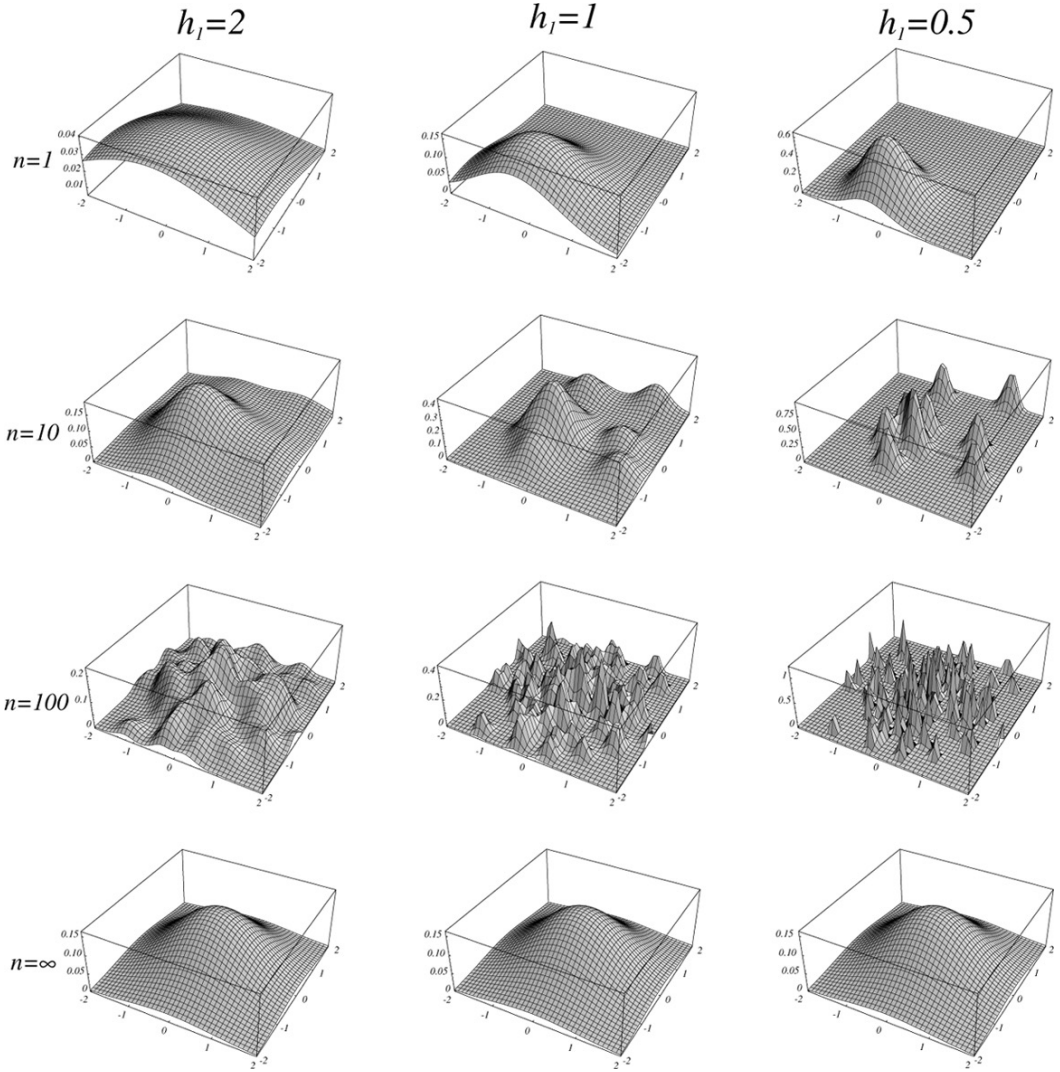
$$\varphi(u) = \frac{1}{\sqrt{2\pi}} e^{-u^2/2}$$



a univariate normal density using different window widths and number of samples.

Parzen Windows

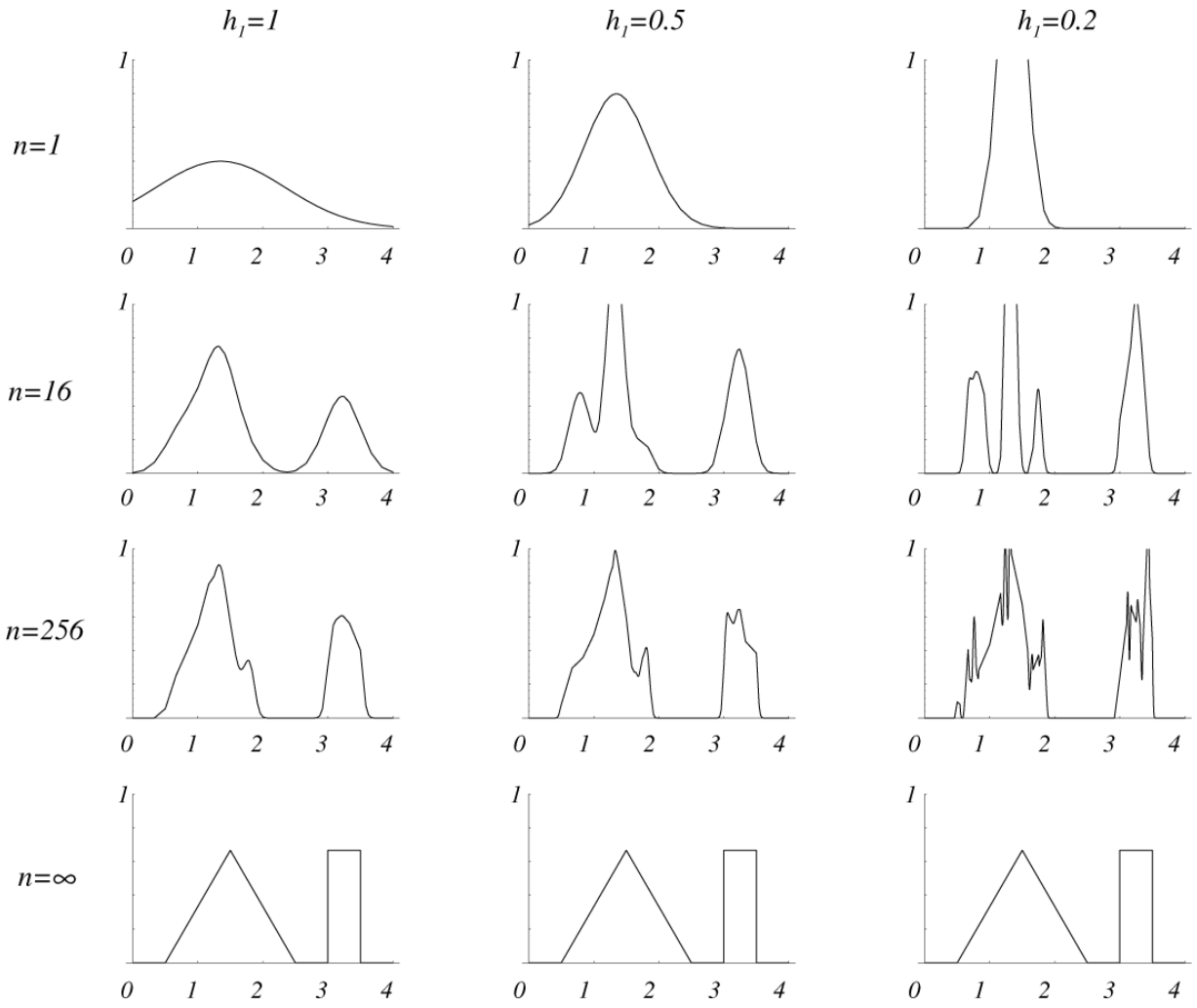
- Illustration



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

Parzen Windows

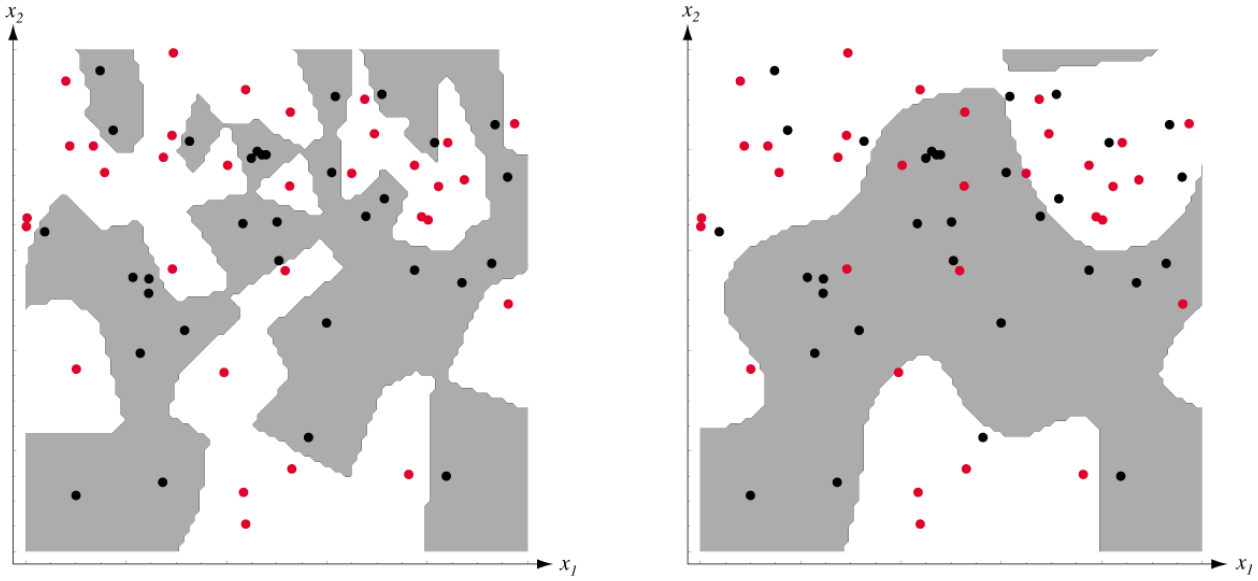
- Illustration



Parzen-window estimates of a bimodal distribution using different window widths and numbers of samples.

Parzen Windows

- Classification Example



2-D Parzen-window dichotomizer: 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.

- The decision regions for a Parzen-window classifier depend on the choice of window function.
- The training error can be made arbitrarily low by making the window width sufficiently small. However, a low *training error* does not guarantee a small *test error*.
- The demand for a large number of samples grows exponentially with the dimensionality of the feature space – "curse of dimensionality"

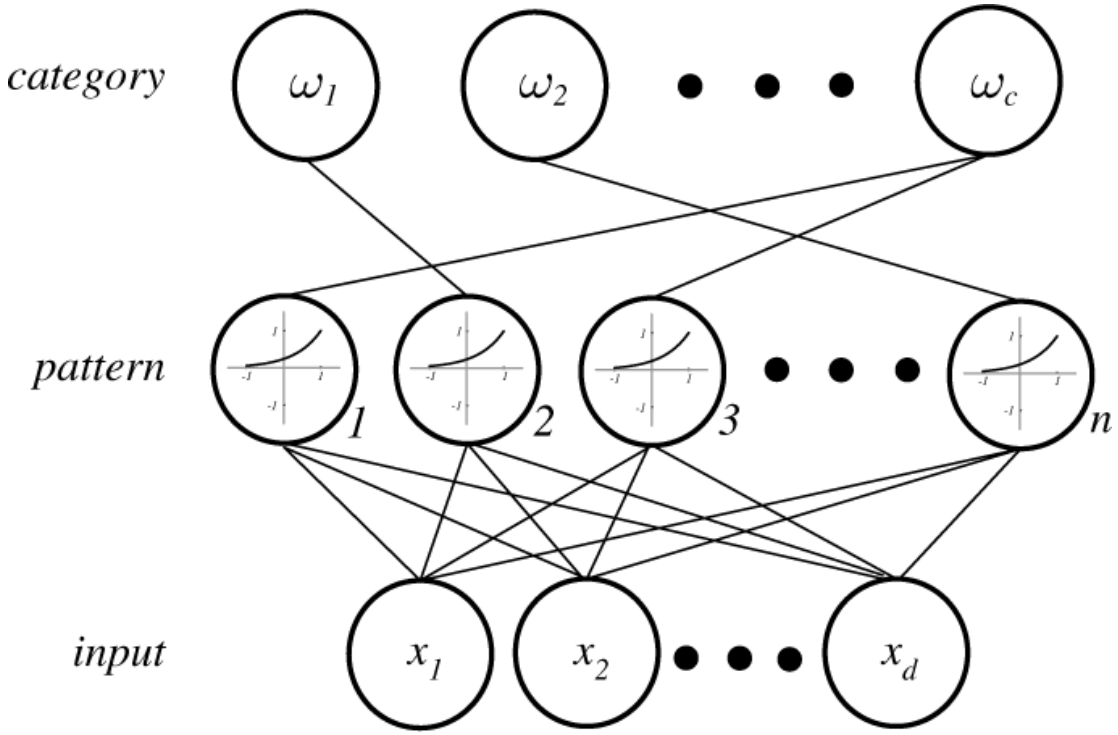
Parzen Windows

p169

- Advantages: generality
  - The same procedure is used for the unimodal normal case and the bimodal mixture case.
  - No assumption on distribution is needed.
  - With enough samples, convergence to an arbitrarily complicated target density is assured.
- Limitations
  - 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.
    - Requirements for computation time and storage
  - Curse of dimensionality

Parzen Windows

- Probabilistic Neural Networks (PNNs)
  - Parallel implementation
    - Trades space complexity for time complexity.



PNN consists of d input units, n pattern units, and c category units.

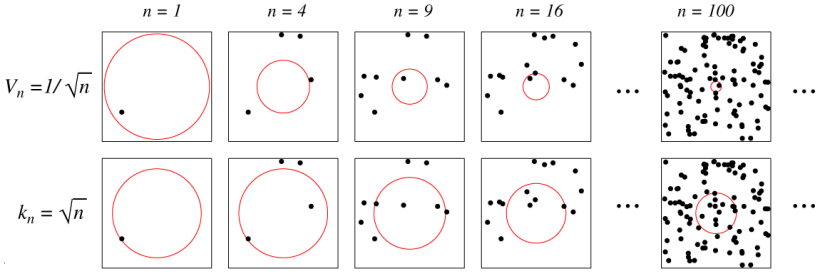
Parzen Windows

- Choosing the Window Function
  - The choice of the sequence of cell-volume size V_1, V_2, …or overall window size.
  - If we take $V_n = V_1/\sqrt{n}$, the results for any finite n will be very sensitive to the choice for the initial volume V_1.
    - If V_1 is too small, most of the volumes will be empty, and the estimate $p_n(\mathbf{x})$ will be very erratic.
    - If V_1 is too large, spatial variations in $p(\mathbf{x})$ may be lost due to averaging over the cell volume.
    - It may be the case that a cell volume appropriate for one region of the feature space might be entirely unsuitable in a different region.
  - Chapter 9: general methods, including cross-validation

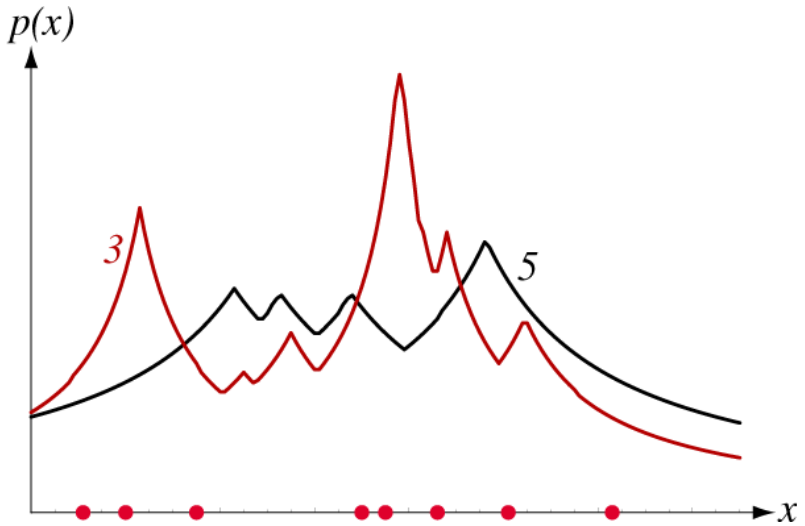
k_n-Nearest-neighbor Estimation

- A potential remedy for the problem of the unknown "best" window function
  - let the cell volume be a function of the training data, rather than some arbitrary function of the overall number of samples.
    - To estimate $p(\mathbf{x})$ from n training samples we can center a cell about $\mathbf{x}$ and let it grow until it capture k_n samples – k_n nearest-neighbors of $\mathbf{x}$.
  - If we take $$p_n(\mathbf{x}) = \frac{k_n/n}{V_n}$$

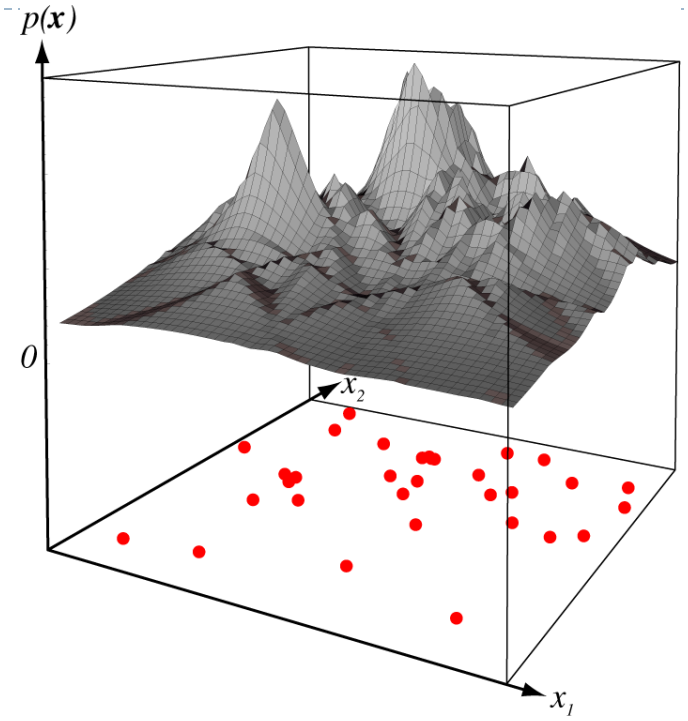
$\lim_{n\to\infty} k_n = \infty$ and $\lim_{n\to\infty} k_n/n = 0$ are necessary and sufficient for $p_n(\mathbf{x})$ to converge to $p(\mathbf{x})$ in probability at all points where $p(\mathbf{x})$ is continuous.



k_n-Nearest-neighbor Estimation



Eight points in one dimension and 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.



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

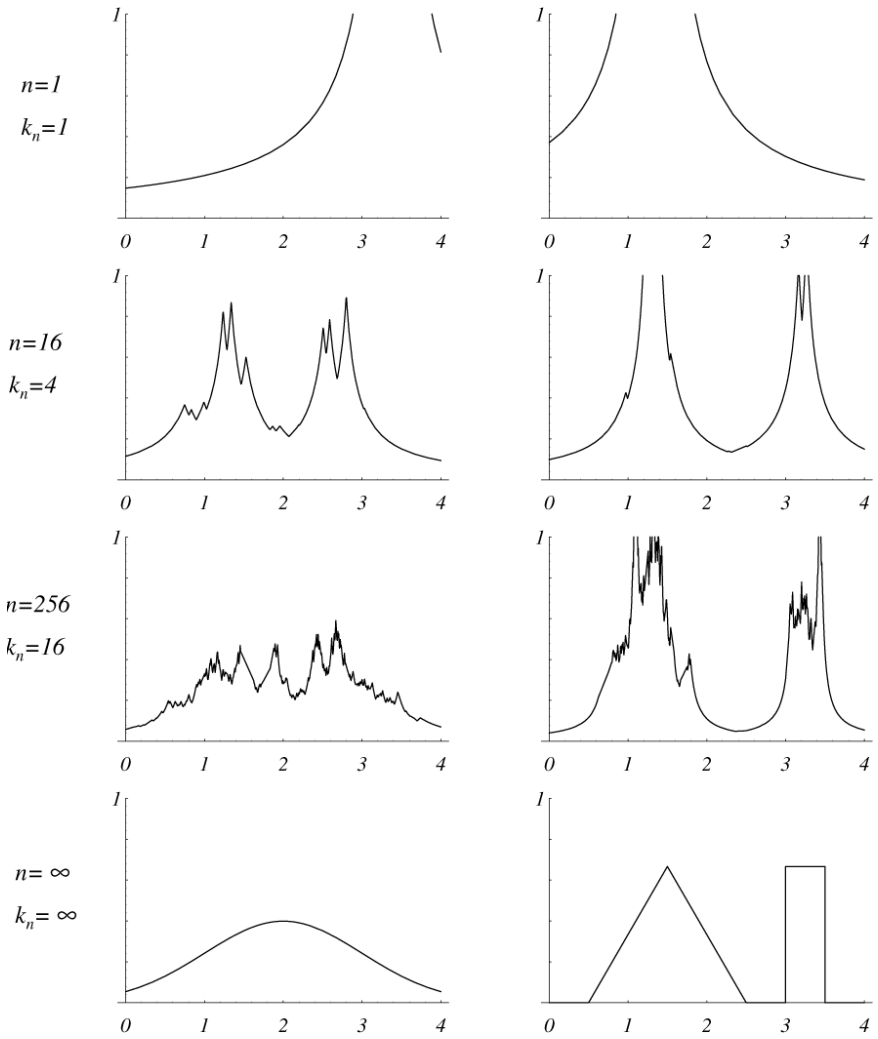
k_n-Nearest-neighbor Estimation

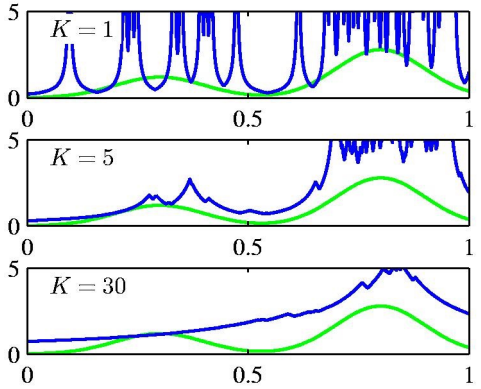
- k_n-Nearest-neighbor and Parzen-window estimation
  - performance comparison with $n = 1$ and $k_n = \sqrt{n} = 1$: the estimate becomes

$$p_n(x) = \frac{1}{2|x - x_1|}$$

    - poor estimate of $p(\mathbf{x})$, with its integral diverging to infinity.
    - The estimate becomes considerably better as n gets larger, even the integral of the estimate remains infinite.
    - For classification, one popular method is to adjust the window width until the classifier has the lowest error on a separate set of samples.

k_n-Nearest-neighbor Estimation





several k-nearest-neighbor estimates of two unidimensional densities: a Gaussian and a binomial distribution. Notice how the finite n estimates can be quite spiky.

k_n-Nearest-neighbor Estimation

- Estimation of *A Posteriori* Probabilities
  - Estimation of the a posteriori probabilities from a set of n labeled samples by using the samples to estimate the densities involved.

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

    - A cell of volume V around $\mathbf{x}$ and capture k samples, k_i of which turn out to be labeled ω_i.
  - A reasonable estimate for $P(\omega_i|\mathbf{x})$

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

    - The state of nature is merely the fraction of the samples within the cell that are labeled ω_i.
    - If there are enough samples and if the cell is sufficiently small, it can be shown that this will yield performance approaching the best possible.

The Nearest-Neighbor Rule

A set of n labeled prototypes $D^n = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$

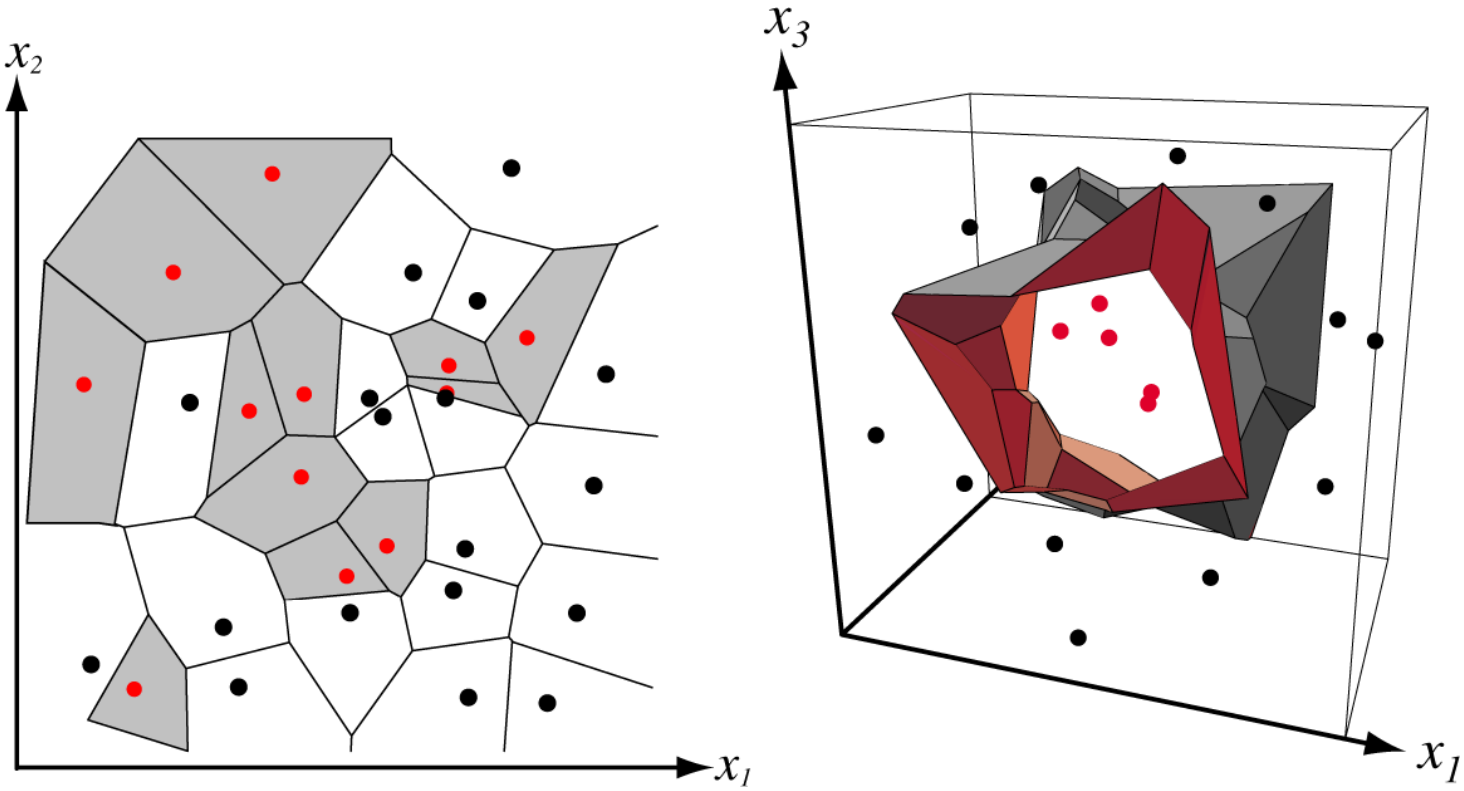
The prototype nearest to a test point $\mathbf{x}$: $\mathbf{x}' \in D^n$

- The nearest-neighbor rule for classifying $\mathbf{x}$ is to assign it the label associated with $\mathbf{x}'$ $P(\omega_i|\mathbf{x}') \approx P(\omega_i|\mathbf{x})$
  - A suboptimal procedure
    - leads to an error rate greater than the Bayes rate.
    - But, with an unlimited number of prototypes the error rate is never worse than twice the Bayes rate.
  - If we define $\omega_m(\mathbf{x})$ by

    $$P(\omega_m|\mathbf{x}) = \max_i P(\omega_i|\mathbf{x})$$

    then the Bayes decision rule always select ω_m.
  - This allows us to partition the feature space into cells consisting of all points closer to a given training point $\mathbf{x}'$ than to any other training points. – *Voronoi tesselation*

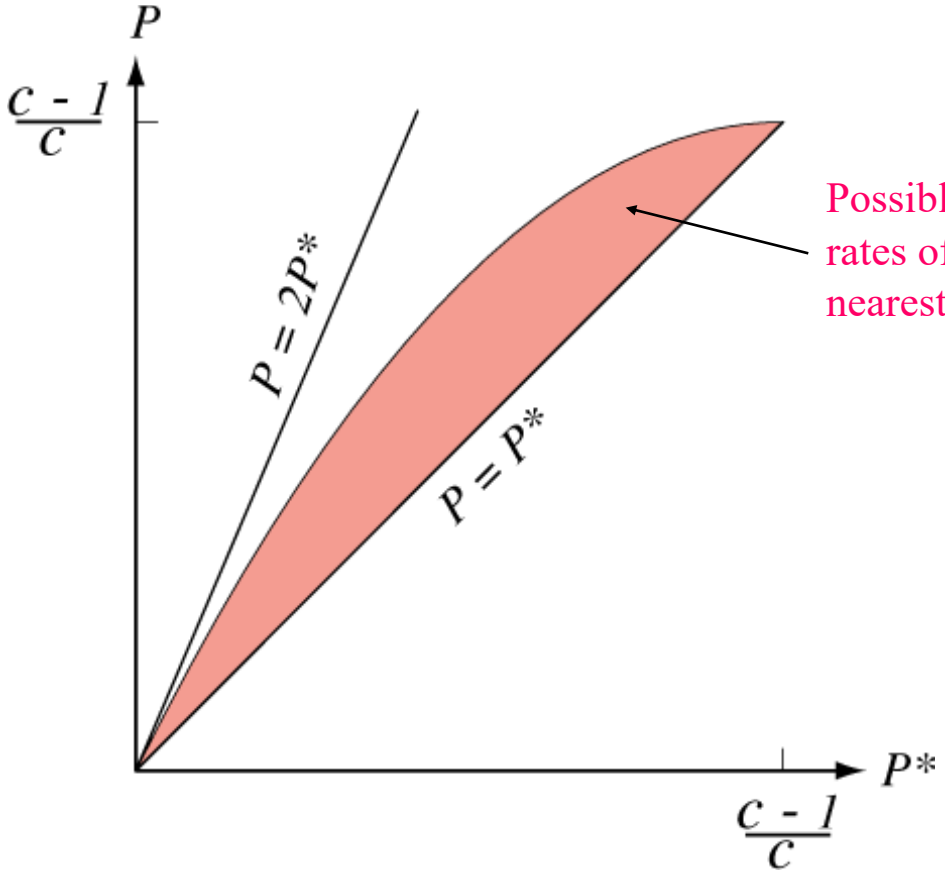
The Nearest-Neighbor Rule



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.

The Nearest-Neighbor Rule

- Error Bounds

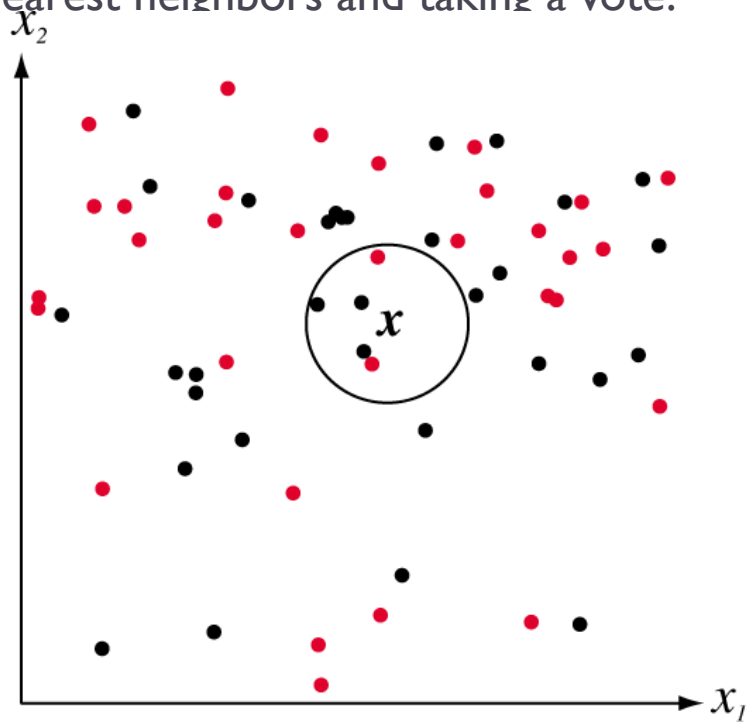


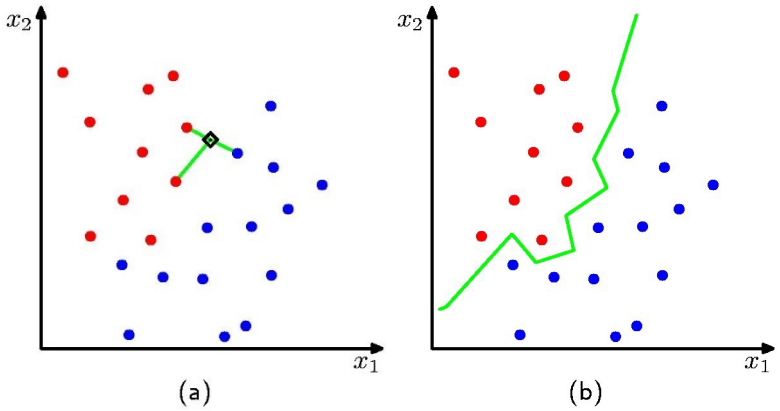
Possible asymptotic error rates of the c-category nearest-neighbor classifier

Bounds on the nearest-neighbor error rate P in a c-category problem given infinite training data, where P* is the Bayes error. At low error rates, the nearest-neighbor error rate is bounded above by twice the Bayes rate.

The Nearest-Neighbor Rule

- The k-Nearest-Neighbor rule
  - An extension of the nearest-neighbor rule
  - The rule classifies $\mathbf{x}$ by assigning it the label most frequently represented among the k nearest samples: that is, a decision is made by examining the labels on the k nearest neighbors and taking a vote.

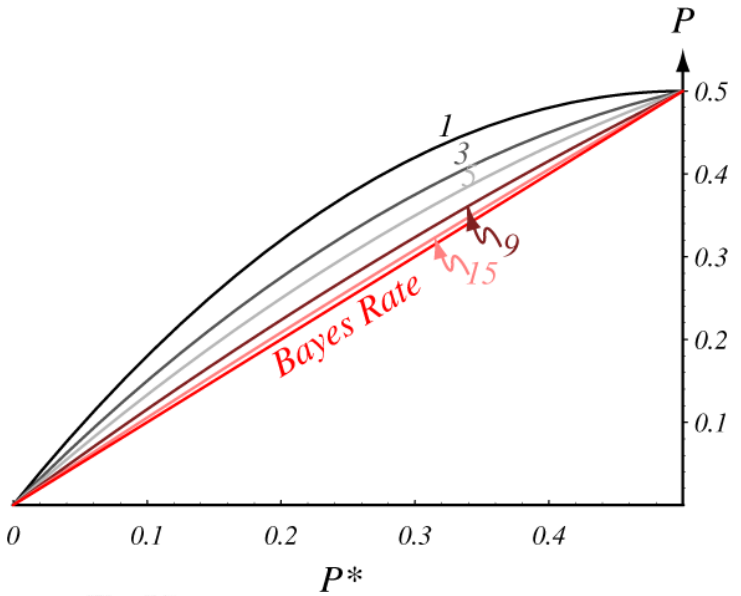


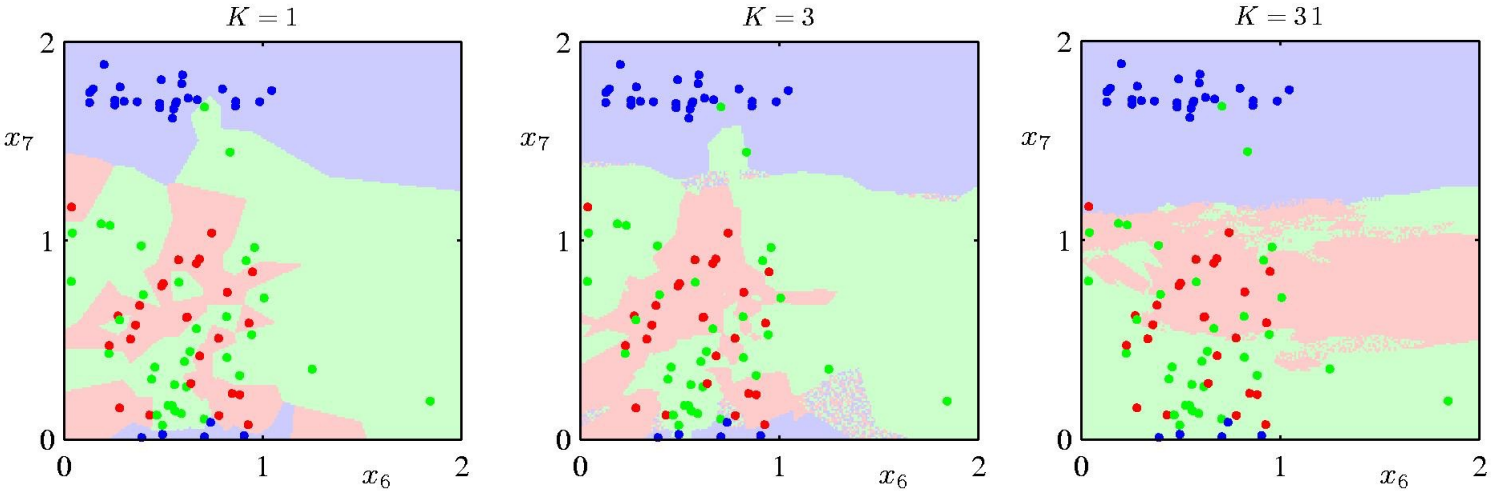


The k-nearest-neighbor query starts at the test point $\mathbf{x}$ and grows a spherical region until it encloses k training samples, and it labels the test point by a majority vote of these samples. In this $k = 5$ case, the point $\mathbf{x}$ would be labeled the category of the **black** points.

The Nearest-Neighbor Rule

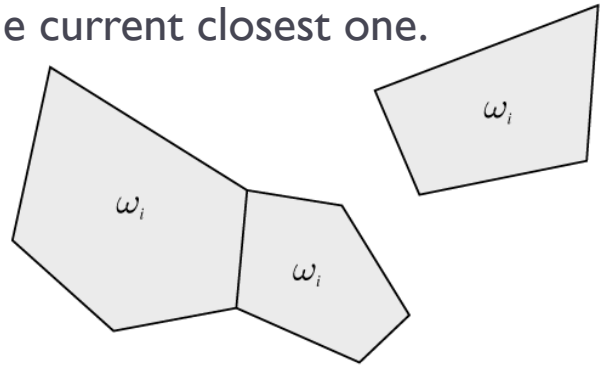
- The k-Nearest-Neighbor rule
  - The error rate for the k-nearest-neighbor rule for a two-category problem.

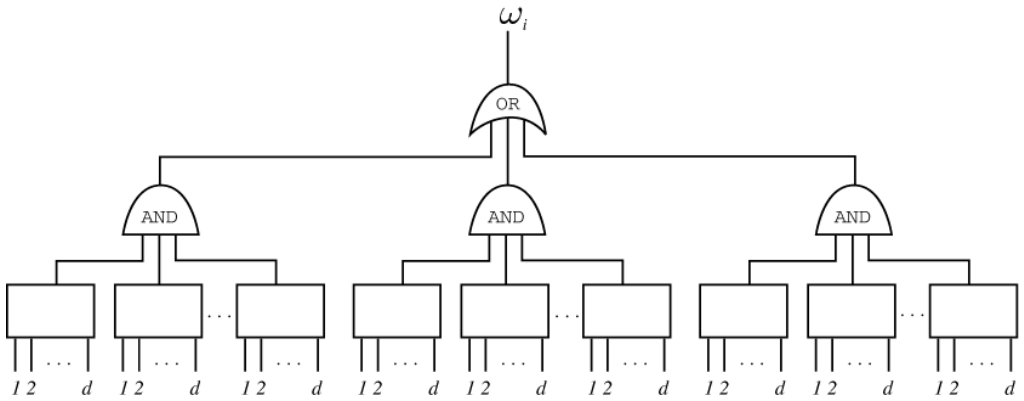




The Nearest-Neighbor Rule

- Computational complexity of the k-Nearest-Neighbor rule
  - Seeking the (single) closest to a test point $\mathbf{x}$ ($k = 1$) from n labeled training samples in d dimensions.
  - The most naïve approach is to inspect each stored point in turn, calculate its distance to $\mathbf{x}$, retaining the identity only of the current closest one.





A parallel NN circuit can perform search in constant time. The d-dim test pattern $\mathbf{x}$ is presented to each box, which calculates which side of a cell's face $\mathbf{x}$ lies on. If it is on the close side of every face of a cell, it lies in the Voronoi cell of the stored pattern, and receives its label.

The Nearest-Neighbor Rule

- Three general algorithmic techniques for reducing the computational burden in NN searches:
  - Computing partial distance
    - Calculating the distance using some subset r of the full d dimensions, and if this partial distance is too great, we do not compute further.

$$D_r(\mathbf{a}, \mathbf{b}) = \left(\sum_{k=1}^{r} (a_k - b_k)^2 \right)^{1/2}$$

  - Prestructuring
    - Some form of *search tree*, in which prototypes are selectively linked, is created.
    - During the classification, we compute the distance of the test point to one of a few stored *entry* or *root* prototypes and then consider only the prototypes linked to it.
    - If the tree is properly structured, we will reduce the total number of prototypes that need to be searched.

The Nearest-Neighbor Rule

- Editing the stored prototypes
  - Eliminating useless prototypes during training
  - Editing, pruning, condensing
  - A simple method: to eliminate prototypes that are surrounded by training points of the same category label. – This leaves the decision boundaries unchanged, while reducing recall time.

- We can combine these three complexity reduction methods – first edit the prototypes, then form a search tree during training, and finally compute partial distances during classification.

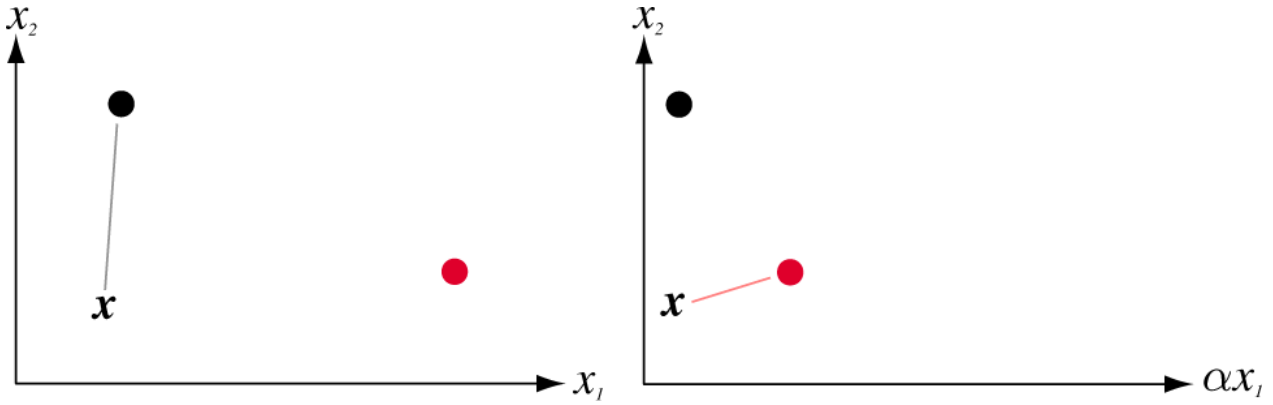
Metrics and Nearest-Neighbor Classification

- The nearest-neighbor classifier relies on a metric or "distance" function between patterns.
  - The notion of a metric is far more general.
- Properties of Metrics
  - Nonnegativity: $D(\mathbf{a}, \mathbf{b}) \geq 0$
  - Reflexivity: $D(\mathbf{a}, \mathbf{b}) = 0$ iff $\mathbf{a} = \mathbf{b}$
  - Symmetry: $D(\mathbf{a}, \mathbf{b}) = D(\mathbf{b}, \mathbf{a})$
  - Triangle inequality: $D(\mathbf{a}, \mathbf{b}) + D(\mathbf{b}, \mathbf{c}) \geq D(\mathbf{a}, \mathbf{c})$
  - Euclidean formula for distance in d dimensions possesses the properties of metric.

$$D(\mathbf{a}, \mathbf{b}) = \left(\sum_{k=1}^{d} (a_k - b_k)^2 \right)^{1/2}$$

Metrics and Nearest-Neighbor Classification

- Properties of Metrics (cont.)
  - Euclidean distance relationships in the transformed space
  - Scale changes can have a major impact on nearest-neighbor classifiers.
  - If there is a large disparity in the ranges of the full data in each dimension, a common procedure is to rescale all the data to equalize such ranges.



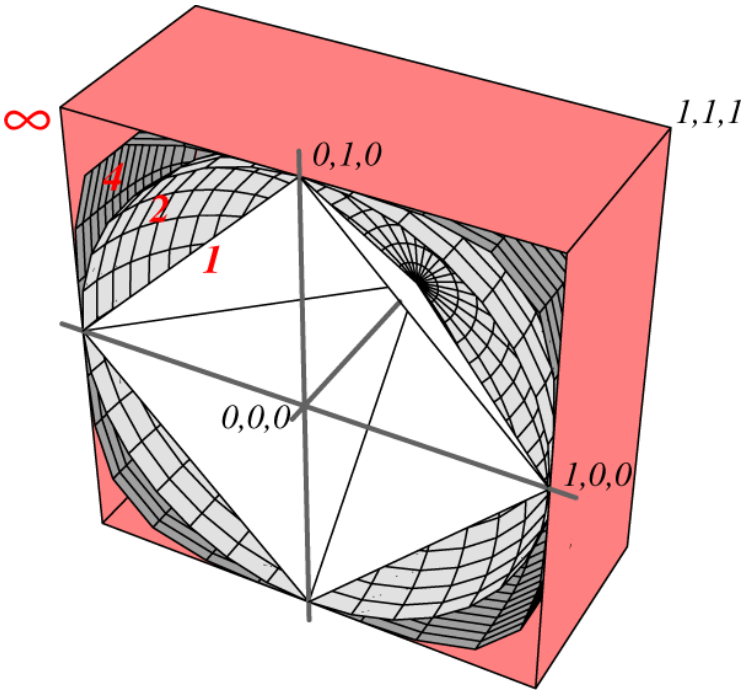
Scaling the coordinates of a feature space can change the distance relationships computed by the Euclidean metric.

Metrics and Nearest-Neighbor Classification

- Minkowski Metric
  - One general class of metrics for d-dimensional patterns :
    - L_k norm

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

    - L_2 norm: Euclidean distance
    - L_1 norm: Manhattan or city block distance
    - Suppose we compute the distances between the projections of **a** and **b** onto each of the d coordinate axes. L_∞ distance between **a** and **b** corresponds to the maximum of these projected distances



Metrics and Nearest-Neighbor Classification

- Tanimoto Metric

$$D_{Tanimoto}(S_1, S_2) = \frac{n_1 + n_2 - 2n_{12}}{n_1 + n_2 - n_{12}}$$

where n_1 and n_2 are the number of elements in sets S_1 and S_2, respectively, and n_{12} is the number that is in both sets.

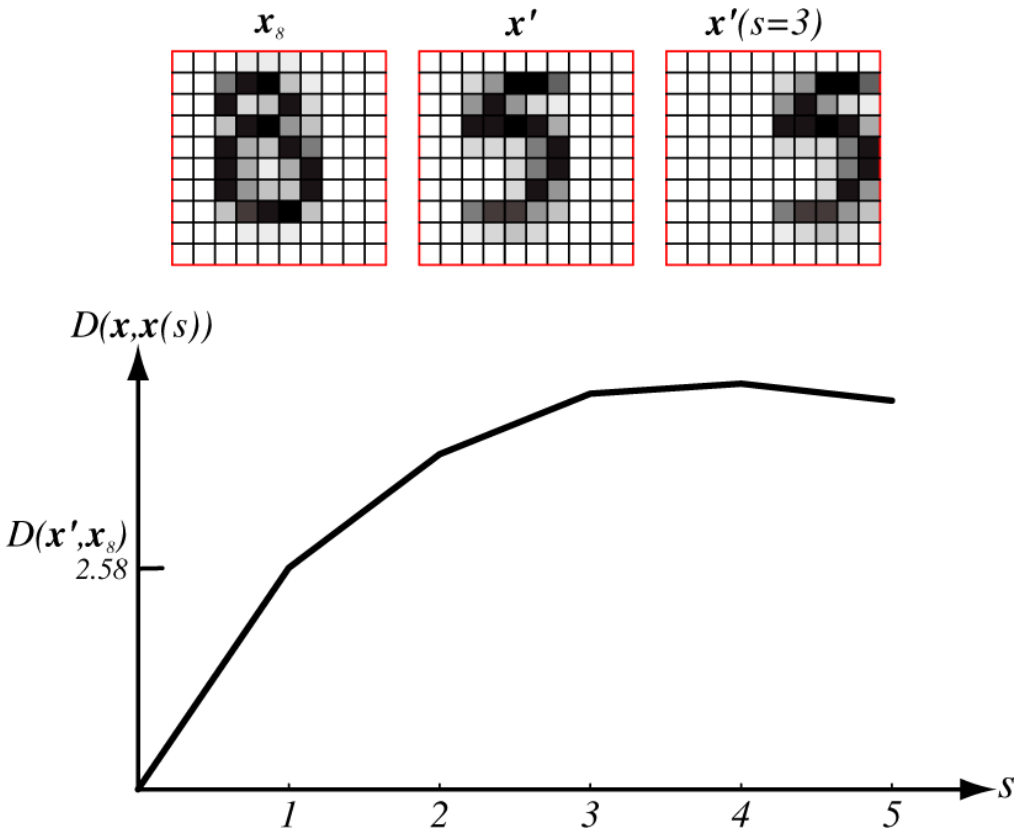
  - The metric finds greatest use for problems in which two *patterns* or *features* are either the same or different.

- The selection among metrics is generally dictated by computational concerns, and it is hard to base a choice on prior knowledge about the distributions.

Metrics and Nearest-Neighbor Classification

- Tangent Distance
  - Drawbacks inherent in the uncritical use of a particular metric in nearest-neighbor classifiers
    - can be overcome by the careful use of more general distance measure.
    - Invariance
      - Consider a 100 −dimensional pattern representing a 10×10 pixel grayscale image of a handwritten 5.
      - Even if the relative shift is a mere three pixels, the Euclidean distance grows very large.
      - Ideally, we would not compute the distance between two patterns until we had transformed them to be as similar to one another as possible.
        - The computational complexity of such transformations is often quite high.

Metrics and Nearest-Neighbor Classification



The uncritical use of Euclidean metric cannot address the problem of translation invariance.

Metrics and Nearest-Neighbor Classification

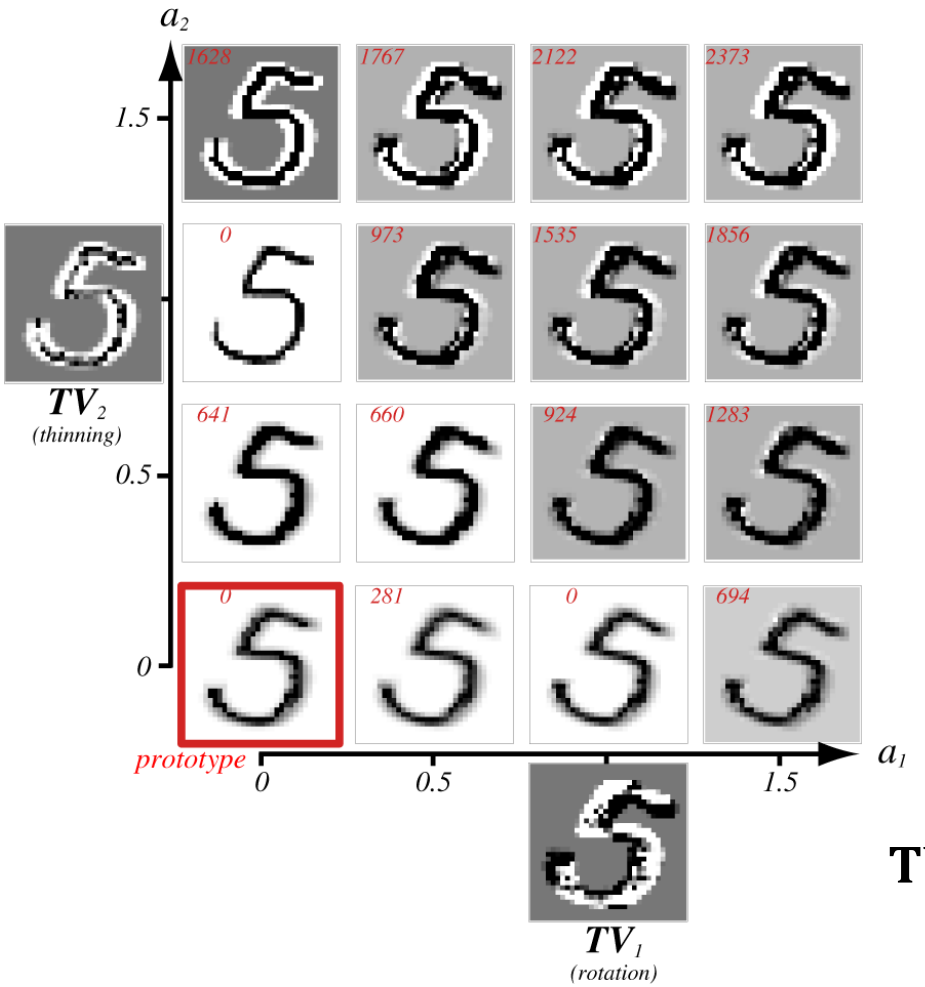
- Tangent Distance (cont.)
  - The general approach in tangent distance classifiers is to use a novel measure of distance and a linear approximation to the arbitrary transforms.
  - During construction of the classifier we take each stored prototype and perform each of the transformations $F_i(\mathbf{x}'; \alpha_i)$ on it.
  - A tangent vector $\mathbf{TV}_i$ for each transformation is constructed.

$$\mathbf{TV}_i = F_i(\mathbf{x}'; \alpha_i) - \mathbf{x}'$$

  - For each prototype $\mathbf{x}'$ an $r \times d$ matrix $\mathbf{T}$, consisting of the tangent vectors at each prototype, is formed.

Metrics and Nearest-Neighbor Classification

- Tangent Distance (cont.)

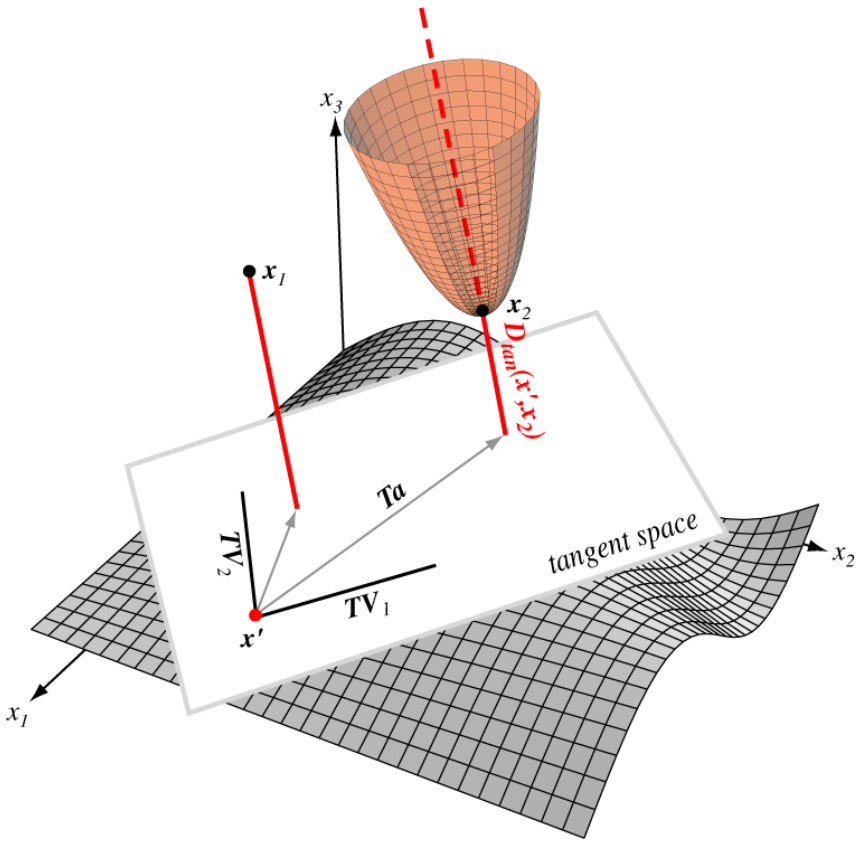


$$\mathbf{TV}_i = F_i(\mathbf{x}'; \alpha_i) - \mathbf{x}'$$

Metrics and Nearest-Neighbor Classification

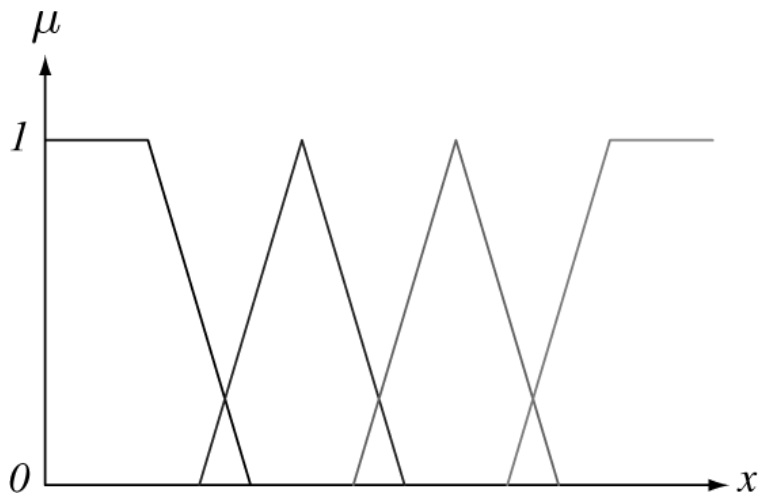
- Tangent Distance (cont.)

$$D_{tan}(\mathbf{x}', \mathbf{x}) = \min_{\mathbf{a}}[\|(\mathbf{x}' + \mathbf{Ta}) - \mathbf{x}\|]$$

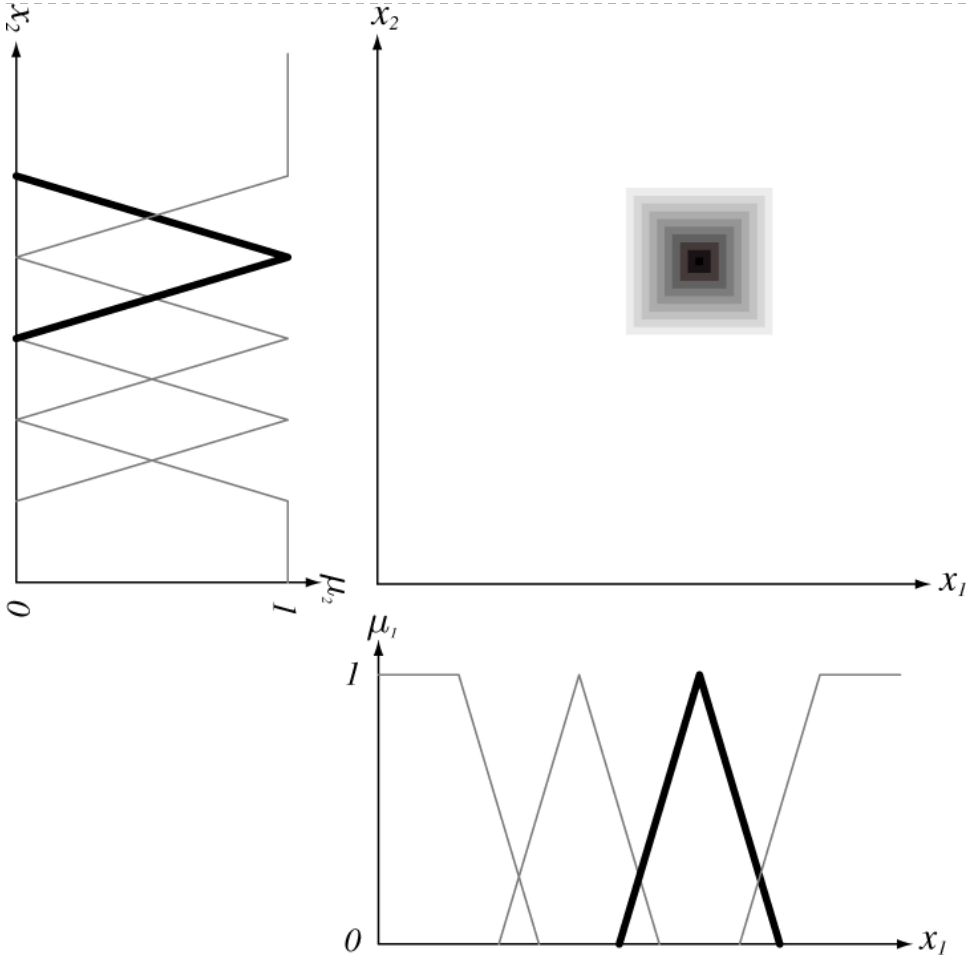


Fuzzy Classification

- To create "fuzzy category memberships functions," which convert an objectively measurable parameter into a subjective *category memberships*.
  - The term categories used in this context refers not to the final class, but instead to overlapping ranges of feature values.
  - Lightness – dark, medium-dark, medium, medium-light, and light



Fuzzy Classification



$$\mu_x(x) \cdot \mu_y(y)$$

"Category membership" functions and a conjunction rule based on the designer's prior knowledge lead to discriminant functions. The resulting discriminant function for the final category is indicated by the gray scale in the middle: the greater the discriminant, the darker.

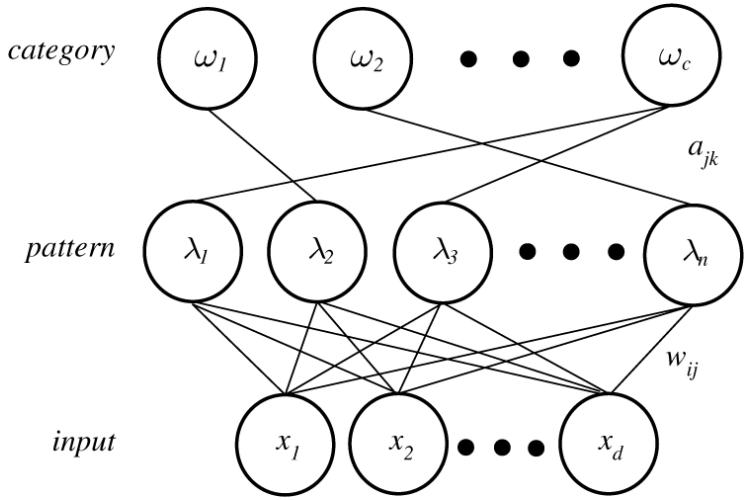
Fuzzy Classification

- The limitations
  - Fuzzy methods are cumbersome to use in high dimensions or on complex problems or in problems with dozens or hundreds of features.
  - The amount of information the designer can be expected to bring to a problem is quite limited – the number, positions, and widths of "category memberships."
  - Because of their lack of normalization, pure fuzzy methods are poorly suited to problems in which there is a changing cost matrix.
  - Pure fuzzy methods do not make use of training data. When such pure fuzzy methods have unacceptable performance, it has been traditional to try to graft on adaptive(e.g., neuro-fuzzy) methods.

Reduced Coulomb Energy Networks

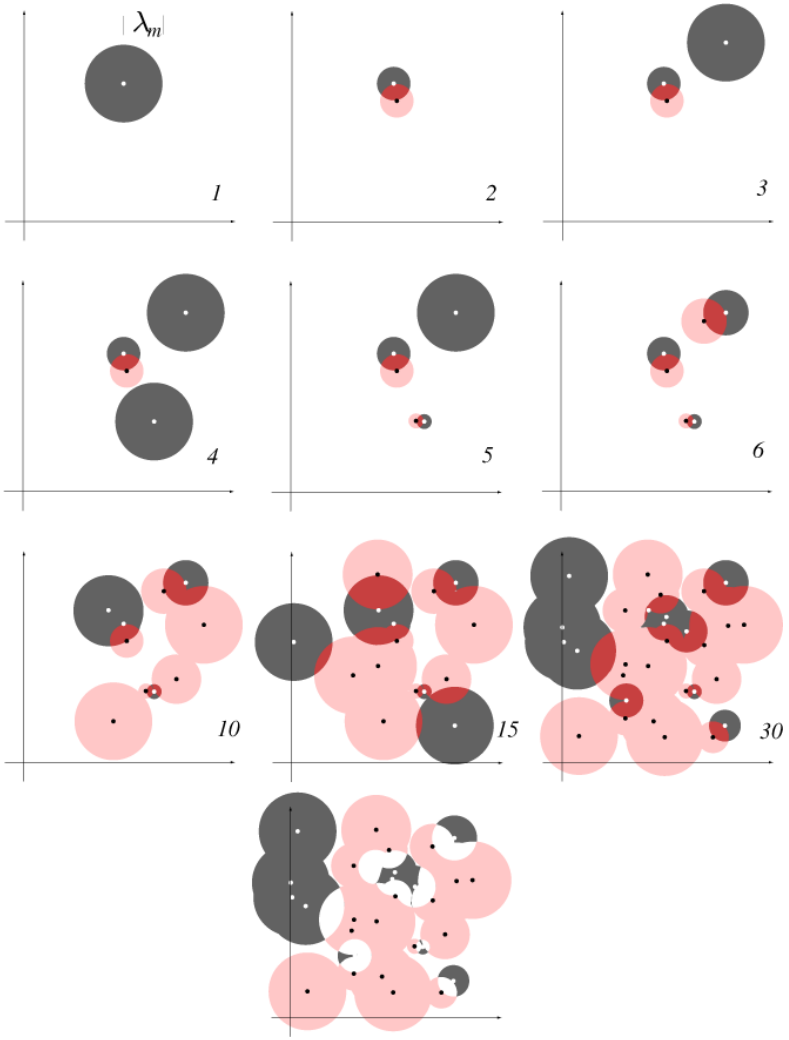
- Parzen-window method: uses a fixed window throughout the feature space.
- However, in some regions a small window width would be appropriate, while elsewhere a large width would be appropriate.
  - The k-nearest-neighbor method.
- An approach that is intermediate between these two is to adjust the size of the window during training according to the distance to the nearest point of different category.
  - The reduced Coulomb energy (RCE) network
    - A neural network approach for such region adjustment.

Reduced Coulomb Energy Networks



- In RCE network, each pattern unit has an adjustable parameter that corresponds to the radius of a d −dimensional sphere in the input space.
- During training, each radius is adjusted so that each pattern unit covers a region as large as possible without containing a training point from another category.
- During classification, a normalized test point is classified by the associated label obtained through training. Any region overlapped is considered ambiguous.

Reduced Coulomb Energy Networks



Summary

- Two approaches to nonparametric estimation for pattern classification:
  - The densities are estimated
    - Parzen windows and their hardware implementation, PNNs
  - The category is chosen directly
    - *k*-nearest-neighbor and several forms of relaxation networks.
- Fuzzy classification methods
  - Employ heuristic choices of "category membership" function and heuristic conjunction rules to obtain discriminant functions.
- Relaxation methods
  - "basins of attraction" surrounding training prototypes
  - RCE networks