2) K_n Nearest-Neighbor (NN) Density Estimation (DHS 4.4)

Parzen windows – $V_1, V_2, \ldots, V_j$ (representing the changing volumes)

Problem: How to choose V_j's

K_n-NN -> Let V_j be a function of the data.

Center each cell (region) at $\underline{x}$ and let it grow until it captures k_j samples. k_j is specified as some function of j.

If the density of samples is high near x, the cell will be small, and if the density is low, the cell will be large.

A common choice is:

$k_j = \sqrt{j}$

$p_j(\underline{x}) = (k_j/j) / V_j$

$k_j = \sqrt{j}$ => $p_j(\underline{x}) = 1/[\sqrt{j}\ V_j]$ => k-NN estimate

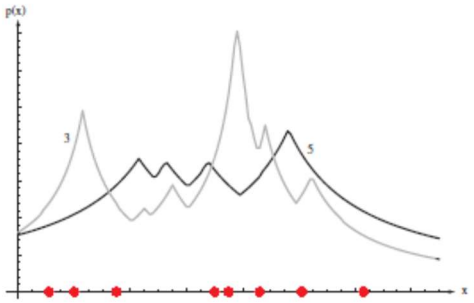

Figure 4.10: Eight points in one dimension and the k-nearest-neighbor density estimates, for $k = 3$ and 5. Note especially that the discontinuities in the slopes in the estimates generally occur *away* fom the positions of the points themselves.

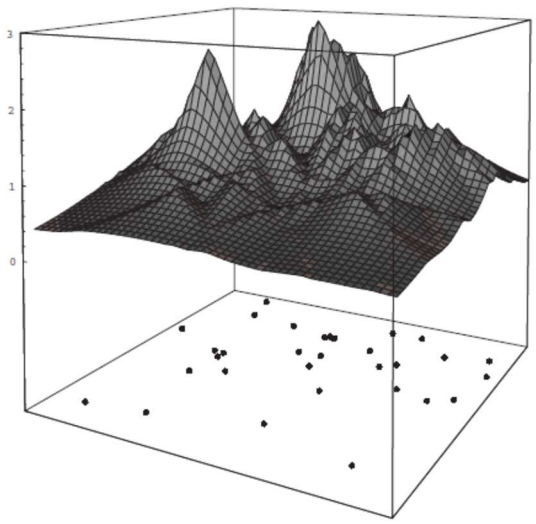

Figure 4.11: The k-nearest-neighbor estimate of a two-dimensional density for $k = 5$. Notice how such a finite n estimate can be quite "jagged," and that discontinuities in the slopes generally occur along lines away from the positions of the points themselves.

Example: $k_j = \sqrt{j}$ j=1 => k_j=1 (DHS 4.4.1)

$V_1=2|x-x_1|$

J=1: $p_1(x)=1/ [\sqrt{1}\ V_1] = 1/ [2|x-x_1|]$

Figure 4.12 (next page)

If $k_j=k_1\sqrt{j}$ (similar to the Parzen-window approach, $h_j = h_1/\text{sqrt}(j)$)

Different choices of k_1 give different estimates $p_j(x)$; all converge asymptotically to $p(\underline{x})$

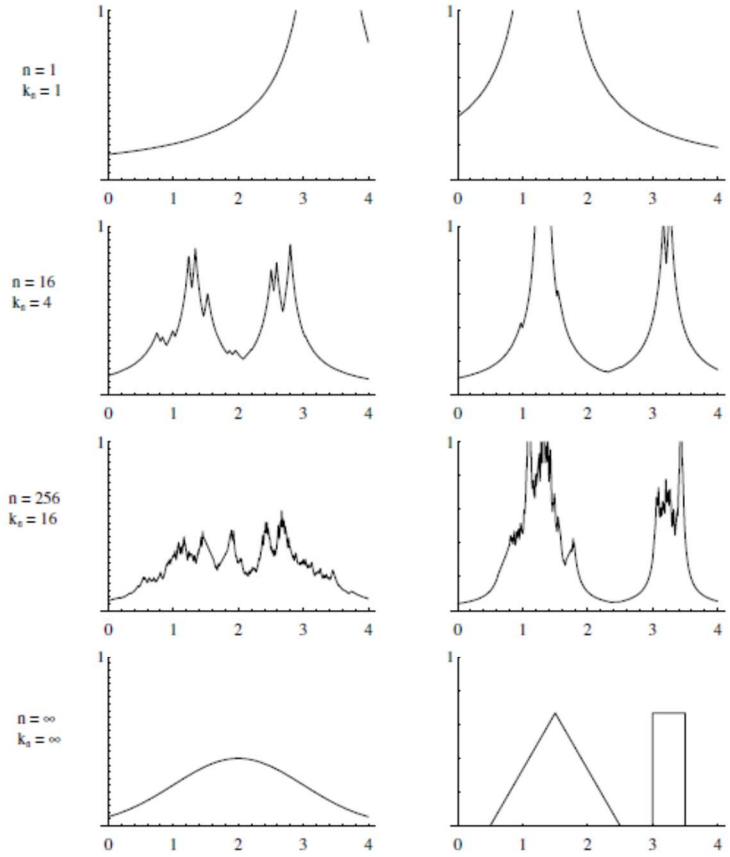

Figure 4.12: Several k-nearest-neighbor estimates of two unidimensional densities: a Gaussian and a bimodal distribution. Notice how the finite n estimates can be quite "spiky."

Estimation of a posteriori probabilities (DHS 4.4.2)

Use of the Parzen windows and k-nearest neighbor (k-NN) for estimation of posteriori probabilities $P(S_i|\underline{x})$

Place a cell, volume V around $\underline{x}$ and capture k samples.
Assume k_i of these samples belong to class S_i

Estimate of the joint probabilities density is:

$p_j(\underline{x},S_i) = (k_i/j) / V = P_j(S_i|\underline{x})p_j(\underline{x})$

$p_j(\underline{x}) = \sum_{i=1}^{K} p_j(\underline{x},S_i)$

$P_j(S_i|x) = p_j(x,S_i) / \sum_{i=1}^{K} p_j(\underline{x},S_i) = ((k_i/j) /V) / \sum_{l} ((k_l/j) / V) = k_i/k$

$P_j(S_i|\underline{x})=k_i/k$

Percentage of samples (in cell) that belong to S_i.

Cell size can be chosen either for Parzen window estimation (typically $V_j=1/\sqrt{j}$) or for NN

(typically $k=k_j=\sqrt{j}$).

Generally $k_i \neq k_j$
$P_j(S_i|x)=k_{ij}/k_j$ <- k-NN estimate or Parzen window estimate
where

- k_{ij}= Number of samples that fall in a cell (centered at x) at the j-th iteration, which belong to class S_i
- k_j=Total number of samples (over all classes) that fall into a cell (centered at x) at the j-th iteration.

That is, the estimate of a posteriori probability that S_i is the state of nature is merely the fraction of the samples within the cell that are labeled S_i.

As j goes to infinity an infinite number of samples will fall within the infinitely small cell. The fact that the cell volume could become arbitrarily small and yet contain an arbitrarily large number of samples would allow us to learn the unknown probabilities with virtual certainty and thus eventually obtain optimum performance.

Nearest Neighbor Rule (DHS 4.5)

Obtain comparable performance if we base our decision solely on the label of the single nearest neighbor of x.

$S^n=\{x_1, x_2, \ldots, x_n\}$ a set of n labeled prototypes
$x' \in S^n$ be the prototype nearest to a test point x.

Nearest neighbor rule for classifying x is to assign it the label associated with x'.

The nearest neighbor rule is suboptimal: lead to an error rate greater than the minimum possible, the Bayes rate.

With an unlimited number of proto-types the error rate is never worse than the twice the Bayes rate.

Voronoi Tessellation

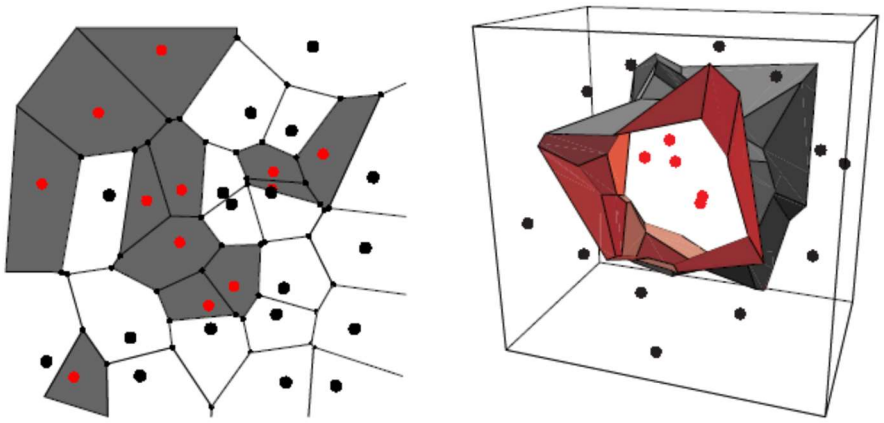

Figure 4.13: In two dimensions, the nearest-neighbor algorithm leads to a partitioning of the input space into Voronoi cells, each labelled by the category of the training point it contains. In three dimensions, the cells are three-dimensional, and the decision boundary resembles the surface of a crystal.

k-Nearest Neighbor Rule (DHS 4.5.4)

Extension of the nearest neighbor rule is the k-nearest neighbor rule.

This rule classifies x by assigning it the label most frequently represented among the k nearest samples; in other words, a decision is made by examining the labels on the k nearest neighbors and taking a vote

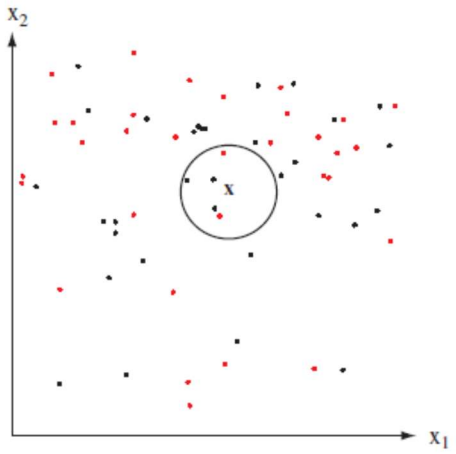

Figure 4.15: The k-nearest-neighbor query starts at the test point and grows a spherical region until it encloses k training samples, and labels the test point by a majority vote of these samples. In this $k = 5$ case, the test point x would be labelled the category of the black points.

Metrics and Nearest Neighbor Classification (DHS 4.6)

Properties of Metrics

non-negativity: $D(\mathbf{a}, \mathbf{b}) \geq 0$

reflexivity: $D(\mathbf{a}, \mathbf{b}) = 0$ if and only if $\mathbf{a} = \mathbf{b}$

symmetry: $D(\mathbf{a}, \mathbf{b}) = D(\mathbf{b}, \mathbf{a})$

triangle inequality: $D(\mathbf{a}, \mathbf{b}) + D(\mathbf{b}, \mathbf{c}) \geq D(\mathbf{a}, \mathbf{c})$.

* Scaling Problem (DHS Fig. 4.18)

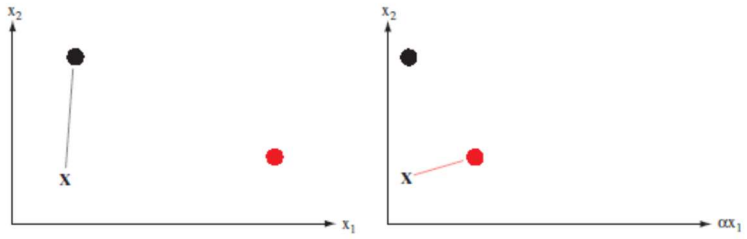

Figure 4.18: Even if each coordinate is scaled by some constant, the resulting space still obeys the properties of a metric. However, a nearest-neighbor classifier would have different results depending upon such rescaling. Consider the test point x and its nearest neighbor. In the original space (left), the black prototype is closest. In the figure at the right, the x_1 axis has been rescaled by a factor 1/3; now the nearest prototype is the red one. 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, and this is equivalent to changing the metric in the original space.

Metrics

- Minkowski metric (a.k.a. L_k norm) $$L_k(\mathbf{a}, \mathbf{b}) = \left(\sum_{i=1}^{d} |a_i - b_i|^k \right)^{1/k},$$
  - k=2 → Euclidean distance (L_2 norm)
- Manhattan distance with k=1 (L_1 norm)
- Tanimoto metric $$D_{Tanimoto}(\mathcal{S}_1, \mathcal{S}_2) = \frac{n_1 + n_2 - 2n_{12}}{n_1 + n_2 - n_{12}},$$ n_1 and n_2 are the no. of elements in each set of S_1 and S_2. n_{12} is the number in both sets.