If a problem has only a few (two or three) important features, then classification is usually an easy problem. For example, with two parameters you can often simply make a scatterplot of the feature values and can determine graphically how to divide the plane into homogeneous regions where the objects are of the same classes. The classification problem becomes very hard, though, when there are many parameters to consider. Not only is the resulting high-dimensional space difficult to visualize, but there are so many different combinations of parameters that techniques based on exhaustive searches of the parameter space rapidly become computationally infeasible. Practical methods for classification always involve a heuristic approach intended to find a “good-enough” solution to the optimization problem.

**Nearest-Neighbor Classifiers**

A very simple classifier can be based on a nearest-neighbor approach. In this method, you simply find in the \( N \)-dimensional feature space the closest object from the training set to an object being classified. Since the neighbor is nearby, it is likely to be similar to the object being classified and so is likely to be the same class as that object.

Classification by a nearest-neighbor algorithm searches for the closest value. Several issues, though, are associated with the use of the algorithm: (1) the inclusion of irrelevant variables lowers the classification accuracy; (2) the algorithm works primarily on numerical variables; categorical variables can be handled but must be specially treated by the algorithm; and (3) if the scales of variables are not in proportion to their importance, classification accuracy will be degraded.

The \( k \)-nearest-neighbor algorithm looks for the closest data point in the data set. The \( k \)-parameter specifies how many nearest neighbors to consider (an odd number is usually chosen to prevent ties). The “closeness” is defined by the difference (“distance”) along the scale of each variable, which is converted to a similarity measure. This distance is defined as the Euclidian distance. Alternatively, the Manhattan Distance can be used, which is defined for a plane with a data point \( p_1 \) at coordinates \((x_1, y_1)\) and its nearest neighbor \( p_2 \) at coordinates \((x_2, y_2)\) as

\[
\text{Manhattan Distance} = |x_1 - x_2| + |y_1 - y_2|.
\]  

(Eq. 1)

An analogous relationship can be defined in a higher-dimensional space.

Nearest-neighbor methods have the advantage that they are easy to implement. They can also give quite good results if the features are chosen carefully (and if they are weighted carefully in the computation of the distance). There are several serious disadvantages of the nearest-neighbor methods. First, they (like the neural networks) do not simplify the distribution of objects in parameter space to a comprehensible set of features. Instead, the training set is retained in its entirety as a description of the object distribution. (There are some thinning methods that can be used on the training set, but the result still does not usually constitute a compact description of the object distribution.) The method is also rather slow if the training set has many examples. The most serious shortcoming of nearest-neighbor methods is that they are very sensitive to the presence of irrelevant...