3 Nearest Neighbour Based Classifiers

Learning Objectives

This chapter is an introduction to nearest neighbour classifiers. At the end of the chapter, you will understand:

- What a nearest neighbour(NN) algorithm is
- The different variants of NN algorithms like
    - The k nearest neighbour(kNN) algorithm
    - The modified k nearest neighbour (MkNN) algorithm
    - The fuzzy kNN algorithm
    - The r near algorithm
- The use of efficient algorithms
- What is meant by data reduction
- The different methods of prototype selection used in classification like
    - The minimal distance classifier(MDC)
    - The condensed nearest neighbour(CNN) algorithm
    - The modified condensed nearest neighbour (MCNN) algorithm
    - Editing methods

One of the simplest decision procedures that can be used for classification is the nearest neighbour (NN) rule. It classifies a sample based on the category of its nearest neighbour. When large samples are involved, it can be shown that this rule has a probability of error which is less than twice the optimum error—hence there is less than twice the probability of error compared to any other decision rule. The nearest neighbour based classifiers use some or all the patterns available in the training set to classify a test pattern. These classifiers essentially involve finding the similarity between the test pattern and every pattern in the training set.

3.1 Nearest Neighbour Algorithm

The nearest neighbour algorithm assigns to a test pattern the class label of its closest neighbour. Let there be n training patterns, $(X_1, \theta_1), (X_2, \theta_2), ..., (X_n, \theta_n)$, where X_i is of dimension d and θ_i is the class label of the ith pattern. If P is the test pattern, then if

$$d(\mathrm{P}, \mathrm{X}_k) = \min\{d(\mathrm{P}, \mathrm{X}_i)\}$$

where $i = 1 \ldots n$. Pattern P is assigned to the class θ_k associated with X_k.

Example 1

Let the training set consist of the following three dimensional patterns:

$$\begin{array}{lll}
\mathrm{X}_1 = (0.8, 0.8, 1), & \mathrm{X}_2 = (1.0, 1.0, 1), & \mathrm{X}_3 = (1.2, 0.8, 1) \\
\mathrm{X}_4 = (0.8, 1.2, 1), & \mathrm{X}_5 = (1.2, 1.2, 1), & \mathrm{X}_6 = (4.0, 3.0, 2) \\
\mathrm{X}_7 = (3.8, 2.8, 2), & \mathrm{X}_8 = (4.2, 2.8, 2), & \mathrm{X}_9 = (3.8, 3.2, 2) \\
\mathrm{X}_{10} = (4.2, 3.2, 2), & \mathrm{X}_{11} = (4.4, 2.8, 2), & \mathrm{X}_{12} = (4.4, 3.2, 2) \\
\mathrm{X}_{13} = (3.2, 0.4, 3), & \mathrm{X}_{14} = (3.2, 0.7, 3), & \mathrm{X}_{15} = (3.8, 0.5, 3) \\
\mathrm{X}_{16} = (3.5, 1.0, 3), & \mathrm{X}_{17} = (4.0, 1.0, 3), & \mathrm{X}_{18} = (4.0, 0.7, 3)
\end{array}$$

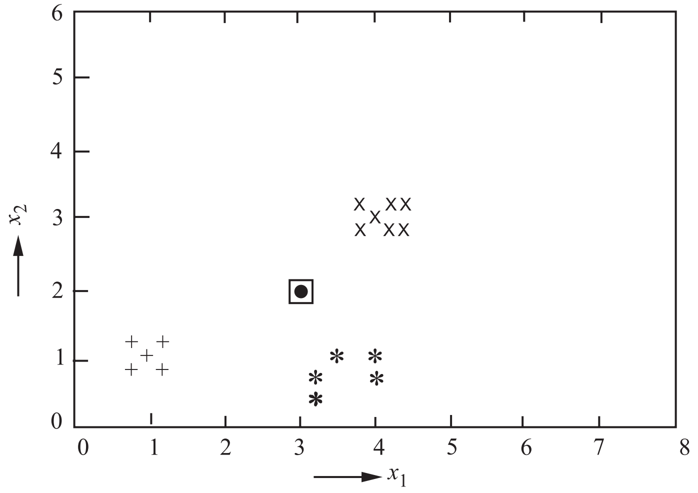

Figure 3.1 Example data set

For each pattern, the first two numbers in the triplets gives the first and second features, and the third number gives the class label of the pattern.

This can be seen plotted in Figure 3.1. Here "+" corresponds to Class 1, "X" corresponds to Class 2 and "*" corresponds to Class 3.

Now if there is a test pattern P = (3.0, 2.0), it is necessary to find the distance from P to all the training patterns.

Let the distance between X and P be the Euclidean distance

$$d(\mathrm{X}, \mathrm{P}) = \sqrt{(X[1] - P[1])^2 + (X[2] - P[2])^2}$$

The distance from a point P to every point in the set can be computed using the above formula. For P = (3.0, 2.0), the distance to X_1 is

$$d(\mathrm{X}_1, \mathrm{P}) = \sqrt{(0.8 - 3.0)^2 + (0.8 - 2.0)^2} = 2.51$$

We find, after calculating the distance from all the training points to P, that the closest neighbour of P is X_{16}, which has a distance of 1.12 from P and belongs to Class 3. Hence P is classified as belonging to Class 3.

3.2 Variants of the NN Algorithm

3.2.1 k-Nearest Neighbour (kNN) Algorithm

In this algorithm, instead of finding just one nearest neighbour as in the NN algorithm, k neighbours are found. The majority class of these k nearest neighbours is the class label assigned to the new pattern. The value chosen for k is crucial. With the right value of k, the classification accuracy will be better than that got by using the nearest neighbour algorithm.

EXAMPLE 2

In the example shown in Figure 3.1, if k is taken to be 5, the five nearest neighbours of P are X_{16}, X_7, X_{14}, X_6 and X_{17}. The majority class of these five patterns is class 3.

This method will reduce the error in classification when training patterns are noisy. The closest pattern of the test pattern may belong to another class, but when a number of neighbours are obtained and the majority class label is considered, the pattern is more likely to be classified correctly. Figure 3.2 illustrates this.

EXAMPLE 3

It can be seen from Figure 3.2 that the test point P is closest to point 5 which is an outlier in Class 1 (represented as a cross). If kNN algorithm is used, the point P will be classified as belonging to Class 2 represented by circles.

Choosing k is crucial to the working of this algorithm. For large data sets, k can be larger to reduce the error. The value of k can be determined by experimentation, where a number of patterns taken out from the training set(validation set) can be classified using the remaining training patterns for different values of k. It can be chosen as the value which gives the least error in classification.

EXAMPLE 4

In Figure 3.1, if P is the pattern (4.2, 1.8), its nearest neighbour is X_{17} and it would be classified as belonging to Class 3 if the nearest neighbour algorithm is used. If the 5 nearest neighbours are taken, it can be seen that they are X_{17} and X_{16}, both belonging to Class 3 and X_8, X_7 and X_{11}, belonging to Class 2. Following the majority class rule, the pattern would be classified as belonging to Class 2.

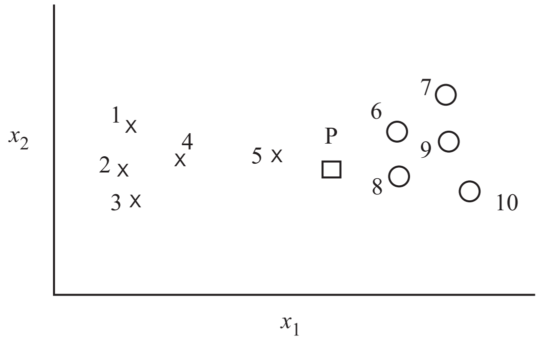

Figure 3.2 P can be correctly classified using the kNN algorithm

3.2.2 Modified k-Nearest Neighbour (MkNN) Algorithm

This algorithm is similar to the kNN algorithm, inasmuch as it takes the k nearest neighbours into consideration. The only difference is that these k nearest neighbours are weighted according to their distance from the test point. It is also called the distance-weighted k-nearest neighbour algorithm. Each of the neighbours is associated with the weight w which is defined as

$$w_j = \begin{cases} \frac{d_k - d_j}{d_k - d_1} & \text{if } d_k \neq d_1 \\ 1 & \text{if } d_k = d_1 \end{cases}$$

where $j = 1, .., k$. The value of w_j varies from a maximum of 1 for the nearest neighbour down to a minimum of zero for the most distant. Having computed the weights w_j, the MkNN algorithm assigns the test pattern P to that class for which the weights of the representatives among the k nearest neighbours sums to the greatest value.

Instead of using the simple majority rule, it can be observed that MkNN employs a weighted majority rule. This would mean that outlier patterns have lesser effect on classification.

Example 5

Consider P = $(3.0, 2.0)$ in Figure 3.1. For the five nearest points, the distances from P are

$$d(\mathrm{P}, \mathrm{X}_{16}) = 1.2;\ d(\mathrm{P}, \mathrm{X}_7) = 1.13;\ d(\mathrm{P}, \mathrm{X}_{14}) = 1.32;$$

$$d(\mathrm{P}, \mathrm{X}_6) = 1.41;\ d(\mathrm{P}, \mathrm{X}_{17}) = 1.41;$$

The values of w will be

$$w_{16} = 1$$

$$w_7 = \frac{(1.41 - 1.13)}{(1.41 - 1.12)} = 0.97$$

$$w_{14} = \frac{(1.41 - 1.32)}{(1.41 - 1.12)} = 0.31$$

$$w_6 = 0$$

$$w_{17} = 0$$

Summing up for each class, Class 1 sums to 0, Class 2 to which X_7 and X_6 belong sums to 0.97 and Class 3 to which X_{16}, X_{14} and X_{17} belong sums to 1.31. Therefore, the point P belongs to Class 3.

It is possible that kNN and MkNN algorithms assign the same pattern a different class label. This is illustrated in the next example.

Example 6

In Figure 3.1, when P = (4.2, 1.8), the five nearest patterns are X_{17}, X_8, X_{11}, X_{16} and X_7. The distances from P to these patterns are

$$d(\mathrm{P}, \mathrm{X}_{17}) = 0.83;\ d(\mathrm{P}, \mathrm{X}_8) = 1.0;\ d(\mathrm{P}, \mathrm{X}_{11}) = 1.02$$

$$d(\mathrm{P}, \mathrm{X}_{16}) = 1.06;\ d(\mathrm{P}, \mathrm{X}_7) = 1.08$$

The value of w will be

$$w_{17} = 1$$

$$w_8 = \frac{(1.08 - 1.0)}{(1.08 - 0.83)} = 0.32$$

$$w_{11} = \frac{(1.08 - 1.02)}{(1.08 - 0.83)} = 0.24$$

$$w_{16} = \frac{(1.08 - 1.06)}{(1.08 - 0.83)} = 0.08$$

$$w_7 = 0$$

Summing up for each class, Class 1 sums to 0, Class 2 to which X_8, X_{11} and X_7 belong sums to 0.56 and Class 3 to which X_{17} and X_{16} belong sums to 1.08 and therefore, P is classified as belonging to Class 3. Note that the same pattern is classified as belonging to Class 2 when we used the k nearest neighbour algorithm with $k = 5$.

3.2.3 Fuzzy kNN Algorithm

In this algorithm, the concept of fuzzy sets, whose elements have a degree of membership, is used. In classical set theory, an element either belongs or does not belong to a set. In fuzzy sets, the elements of the set have a membership function attached to them which is in the real unit interval [0, 1].

In fuzzy kNN algorithm, the fuzzy sets are the c classes. Each pattern belongs to every class i with a membership value which depends on the class of its k neighbours.

The fuzzy kNN algorithm finds the k nearest neighbours of the test point and assigns a class membership for every point to all the classes. A new sample P has membership $\mu_i(\mathrm{P})$ for class i which is given by

$$\mu_i(\mathrm{P}) = \frac{\Sigma_{j=1}^{K}\mu_{ij}(\frac{1}{d(\mathrm{P},\mathrm{X}_j)^{\frac{2}{m-1}}})}{\Sigma_{j=1}^{K}(\frac{1}{d(\mathrm{P},\mathrm{X}_j)^{\frac{2}{m-1}}})}$$

μ_{ij} gives the membership in the ith class of the jth vector of the training set. One way of assigning this membership is to give complete membership in their known class and zero membership in all other classes. Another way to assign this membership is to base it on the distance of the sample from the class mean for all the classes. Here m is a constant and its value is provided by the user.

Each pattern has a membership of 1 to the classes which figure in the k nearest neighbours and a membership of 0 to the other classes. The test pattern is given a membership to every class and the class to which it has the highest membership value is the one it is assigned to.

Example 7

In the example shown in Figure 3.1, if we take P = (3.0, 2.0) and k = 5, the 5 nearest neighbours will be 16, 7, 14, 6 and 17. When we consider Class 1, since there are no patterns from Class 1, $\mu_1(P)$ will be 0. For Class 2, μ_{ij} will be 1 for patterns 6 and 7, and 0 for patterns 16, 14 and 17. For Class 3, μ_{ij} will be 1 for patterns 16, 14 and 17, and 0 for patterns 6 and 7. Taking $m = 2$, we get

$\mu_1(\mathrm{P}) = 0$

$$\mu_2(\mathrm{P}) = \frac{\dfrac{1}{1.414^2} + \dfrac{1}{1.131^2}}{\dfrac{1}{1.414^2} + \dfrac{1}{1.131^2} + \dfrac{1}{1.118^2} + \dfrac{1}{1.315^2} + \dfrac{1}{1.414^2}} = 0.406$$

$$\mu_3(\mathrm{P}) = \frac{\dfrac{1}{1.118^2} + \dfrac{1}{1.315^2} + \dfrac{1}{1.414^2}}{\dfrac{1}{1.118^2} + \dfrac{1}{1.315^2} + \dfrac{1}{1.414^2} + \dfrac{1}{1.414^2} + \dfrac{1}{1.131}} = 0.594$$

Therefore, we assign the pattern P to class 3.

3.2.4 *r* Near Neighbours

Instead of looking at nearest neighbours, we can also look at all the neighbours within some distance r of the point of interest. The algorithm is as follows.

STEP 1: Given the point P, determine the sub-set of data that lies in the ball of radius r centred at P,

$$B_r(\mathrm{P}) = \{\mathrm{X}_i \in \mathrm{X} \text{ s.t. } ||\mathrm{P} - \mathrm{X}_i|| \leq r\}$$

STEP 2: If $B_r(\mathrm{P})$ is empty, then output the majority class of the entire data set.

STEP 3: If $B_r(\mathrm{P})$ is not empty, output the majority class of the data points in it.

If the nearest neighbour of the point of interest is very far away, then that neighbour would have little to do with the point. It is from this logic that we get the r near neighbours algorithm.

This algorithm can be used to identify outliers. If a pattern does not have any similarity with the patterns within the radius chosen, it can be identified as an outlier. The choice of the radius r is crucial to the algorithm.

EXAMPLE 8

In the case of the example shown in Figure 3.1, from the point P = (3.0, 2.0), patterns which are in a radius of 1.45, are X_6, X_7, X_8, X_9, X_{14}, X_{16} and X_{17}. The majority of these patterns belong to Class 2. P is therefore assigned to Class 2.

3.3 Use of the Nearest Neighbour Algorithm for Transaction Databases

Data collected from scientific experiments, or monitoring of physical systems such as telecommunications networks or from transactions at a supermarket, are stored in transaction databases. It is also called market-basket data and consists of transactions made by each customer. In the case of supermarket data, each transaction contains items bought by a customer. Each transaction in the database could be of a different size. The goal of using this data is to see if occurrence of certain items in the transactions can be used to deduce the occurrence of other items. In other words, it is necessary to find the associative relationship between items. This is called association rule mining. To simplify and speed up the process, only items which occur frequently are considered. A minimum support value is chosen and items which occur less than the minimum support are removed. These transaction databases can be represented using the FP tree described in Section 2.1.5.

The FP tree can be used to find the nearest neighbour to a test pattern in a transaction data. The following procedure is used :

Step 1: Remove items in the test pattern which are below the minimum support .

Step 2: Arrange the remaining items in order according to that in the FP tree.

Step 3: Search from the root node for the branch having the first item in the test pattern. If it exists, look for the next item in the test pattern and so on. When the item does not exist, then look at all the branches from that point and choose the branch having maximum items in common with the test pattern. This gives the nearest neighbour.

Example 9

Consider Example 3 in Chapter 2. Figure 2.7 shows the FP tree for the transaction database shown in Table 2.1. Now consider a test pattern a, b, c, d, f, g, h, l, p.

Removing items which are below the minimum support, we get a, d, h, l, p. Arranging these items according to the items in the FP tree, we get l, a, d, p, h. Starting from the root node of the FP tree l, if we compare the items remaining in the test pattern, we can see that it has maximum items common with 7 and therefore can be classified as belonging to digit 7.

3.4 Efficient Algorithms

The nearest neighbour classification method is very time consuming especially when the number of training patterns is large. To overcome this, many efficient algorithms have been proposed which aim to find the nearest neighbour quickly. Some of these algorithms are detailed below. Many of them entail using a pre-processing step which requires additional computing time, but has to be done only once however large the number of test vectors to be classified is. Pre-processing might require a preliminary ordering of the prototypes based on intra-set distances, together with an application of the triangle inequality during the examination stage. This might require considerable time but since it needs to be done only once, it does save time in the long run.

3.4.1 The Branch and Bound Algorithm

If the data are stored in an ordered fashion, it is possible that we may not have to look at all the points to find the nearest neighbour. For example, if the data is stored in a tree-like data structure, the search for the nearest neighbour can be performed efficiently by not having to search down branches where one can obtain a lower bound for the distance that is higher than the current best value found. The data is clustered into representative groups S_j, where each group has the smallest possible radius. Associated with each cluster is the centre of the cluster μ_j and the radius of the cluster, r_j. After searching one of the clusters, point X* is found which is the nearest neighbour from the cluster to the point of interest P. Let this distance be $d(\mathrm{P}, \mathrm{X}_j^*) = d$. Lower bounds can be computed for the distances in the other clusters. Consider the point $\mathrm{X}_k \in \mathrm{S}_j$, then by triangular inequality,

$$d(\mathrm{P}, \mu_j) \leq d(\mathrm{P}, \mathrm{X}_k) + d(\mathrm{X}_k, \mu_j) \leq d(\mathrm{P}, \mathrm{X}_j) + r_j$$

Therefore, a lower bound b_j can be computed for the distance between any $\mathrm{X}_k \in S_j$ and P as follows:

$$\min_{\mathrm{X}_k \in \mathrm{S}_j} d(\mathrm{P}, \mathrm{X}_k) \geq b_j = d(\mathrm{P}, \mu_j) - r_j$$

Clusters which satisfy the inequality $b_j \geq d$ need not be searched. If each cluster has an exactly analogous cluster structure within the clusters themselves, we get the following recursive algorithm.

STEP 1: Cluster the data points into L sets S_i, $i = 1$ to L. These are first level clusters. For each cluster, maintain the centre μ_i and the radius r_i. Within each cluster, S_i, cluster the data into L sub-clusters $S_{i,j}$ for $j = 1$ to L. Maintain the centre $\mu_{i,j}$ and the radius $r_{i,j}$. These are second level clusters. Continue this recursive clustering until there are clusters of one point.

STEP 2: To find the nearest neighbour to a new point P,

1. *Branch step* First compute b_j. Then find the nearest neighbour in the cluster with the smallest b_j. If the clusters have clusters, this step is done recursively.
2. *Bound step* If that cluster does not satisfy the bound, look into the cluster with the next highest b_j. Otherwise stop and output the result.

It can be shown that on an average, there is considerable improvement over the standard nearest neighbour algorithm.

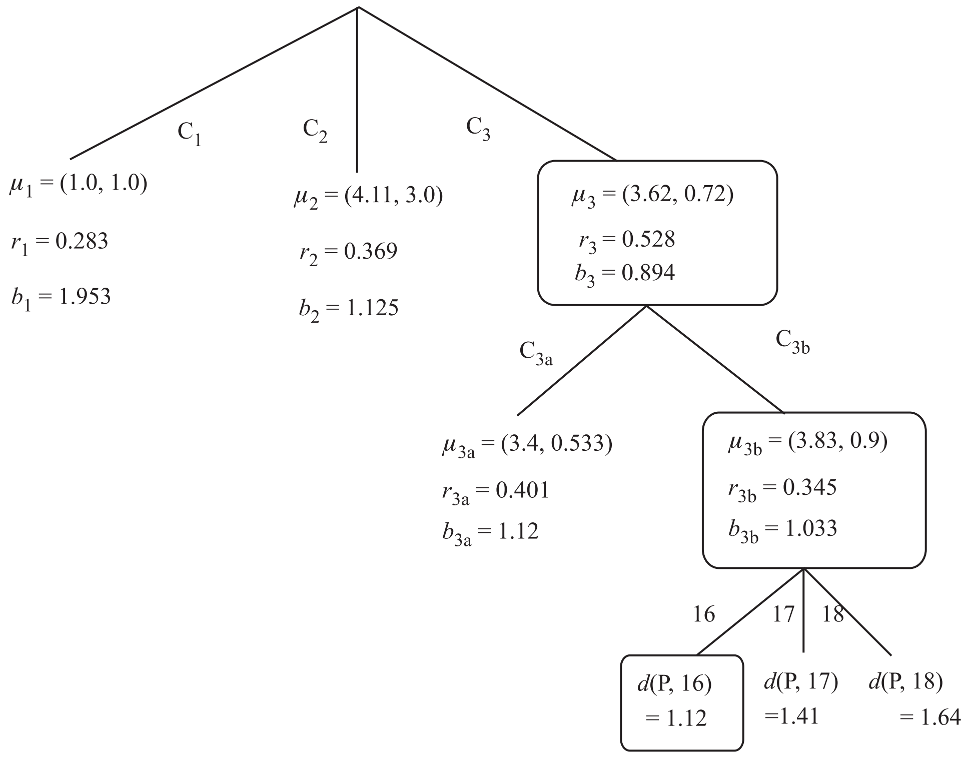

Figure 3.3 Working of the branch and bound algorithm for clustering

EXAMPLE 10

Let us take the example in Figure 3.1. The working of the algorithm is shown in Figure 3.3. Clustering of the points has to be first carried out. Let the points of Class 1 be taken as one cluster, points of Class 2 as the second cluster and the points of Class 3 as the third cluster. Further Cluster 1 is sub-clustered as Cluster 1a and 1b, Cluster 2 is sub-clustered as Cluster 2a and 2b and Cluster 3 is sub-clustered as 3a and 3b. At the next level, each point is taken to be a sub-cluster. This clustering of the points is shown in Figure 3.4.

The centre of Cluster 1 is (1.0, 1.0) and the radius is 0.283. The centre of Cluster 2 is (4.11, 3.0) and the radius is 0.369. The centre of Cluster 3 is (3.62, 0.72) and the radius is 0.528. Taking a point P = (3.0, 2.0), at the first level, b_j for each cluster j can be found. $b_1 = d(\mathrm{P}, \mu_1) - r_1 = 1.953$. Similarly $b_2 = 1.125$ and $b_3 = 0.894$. Since b_3 is the smallest, the sub-clusters of Cluster 3 are searched, and we get the centre of Cluster 3a as (3.4, 0.533) and the radius to be 0.401. The centre of Cluster 3b is (3.83, 0.9) and the radius 0.345. This gives rise to $b_{3a} = 1.12$ and $b_{3b} = 1.033$. The second sub-cluster of Cluster 3 is therefore searched and point 16 is found to be the point closest to point P. The bound d is calculated as the distance from P to point 16 which is 1.12 . Since b_1 and b_2 are greater than d, the clusters 1 and 2 need not be searched and therefore X_{16} is the closest point to P. X_{16} is declared as the nearest neighbour of P.

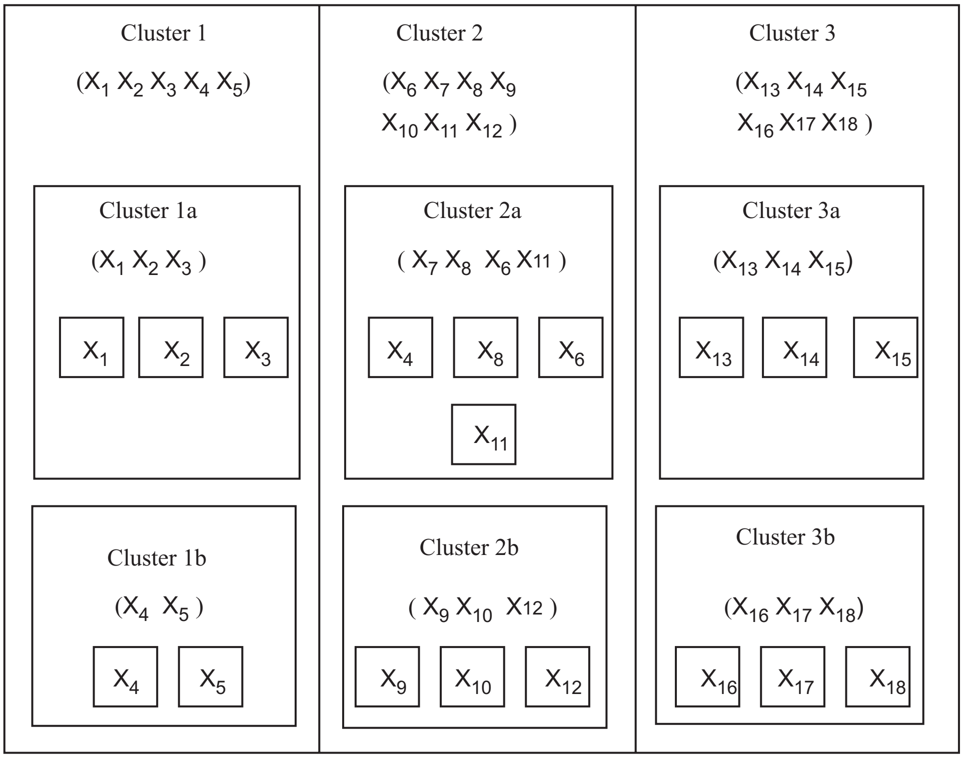

Figure 3.4 Clustering of points using branch and bound method

3.4.2 The Cube Algorithm

In this algorithm, during the pre-processing stage, the patterns are projected onto each axis and a side length l for the initial hypercube to be inspected is specified. Figure 3.5 illustrates in two dimensions the square C of side l centered at P $= (p_1, p_2)$ and its projection onto both axes. By appropriately incrementing or decrementing l in an iterative manner, a hypercube containing exactly k points can be quickly found. The algorithm is described below:

STEP 1: Form $B_i(j)$ and $K_i(j)$, i = 1,..., d and $j = 1$,..., n, where d is the number of dimensions (or features) and n gives the number of patterns. B_i is a real array containing the n ordered values of feature i. K_i is an integer array of indexes, where $K_i(J)$ contains the index of the pattern represented by $B_i(j)$. Let the integer array Z contain n elements each of which is initialised to 1.

STEP 2: Let P = $(p_1, ..., p_d)$ be the test vector.

STEP 3: For each axis i, set pointer ptr$_{i1}$ to the location in A_i of the first element $\geq p_i - \frac{l}{2}$; set pointer ptr$_{i2}$ to the location in B_i of the last element $\leq p_i + \frac{l}{2}$.

STEP 4: For each axis i, go sequentially through the array K_i starting with $K_i(P1_i)$ and ending with $K_i(\text{ptr}_{i2})$. At each location in K_i, shift left 1 bit, the contents of the cell in Z pointed to by that location. In other words, at each location in K_i, multiply by 2, the contents of the cell pointed to by that location.

STEP 5: Count the locations in Z with the value 2^d, and let L equal this count. This count gives the points which fall within the hypercube.

STEP 6: If $L = k$, stop. If $L < k$ (where k = number of neighbours required), increase l appropriately. Reset pointers, repeat the scanning and shifting procedure only on the newly included cells of K_i, and go to Step 4. If $L > k$, decrease l appropriately. Reset pointers, scan only the newly excluded cells of K_i and shift right 1 bit, the proper cells of Z. Go to Step 4.

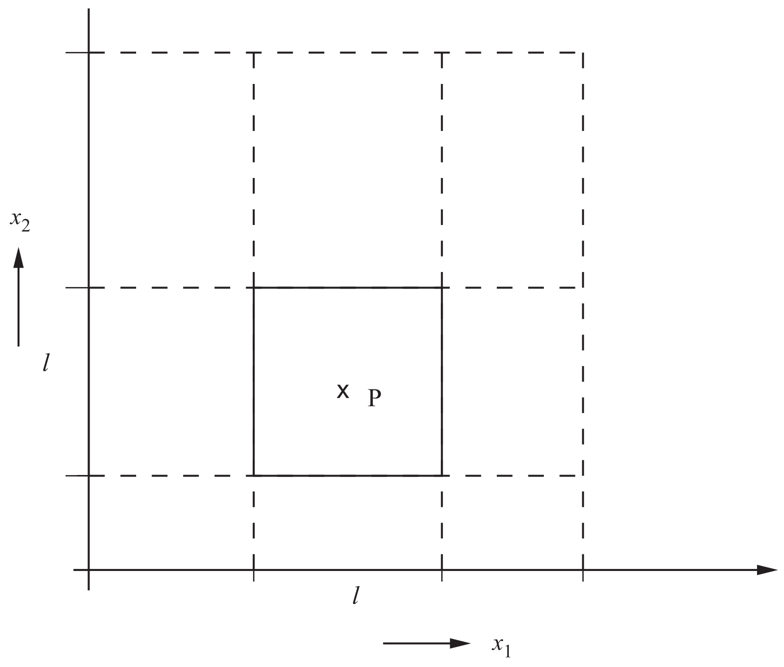

Figure 3.5 Projection of a hypercube

Example 11

Figure 3.6 shows a set of two-dimensional points. The points 1, 2, 3, 4, 5 and 6 belong to Class 1 and the points 7, 8, 9, 10, 11 and 12 belong to Class 2. It is required to find k nearest neighbours of P, where $k = 5$. If l is chosen to be 2, the points that fall within the square will obey the conditions $2 \leq p_1 \leq 4$ and $2 \leq p_2 \leq 4$. It is found that 3 points fall within the square. l is then increased by 0.5 and it is found that 5 points fall within the square which give the 5 nearest neighbours. The majority class labels of these five points can be used to find the class label of the point P. Since 4, 5 and 6 belong to Class 1 and 7 and 9 belong to Class 2, among the 5 closest patterns, three belong to Class 1 and two belong to Class 2. P is therefore assigned to Class 1.

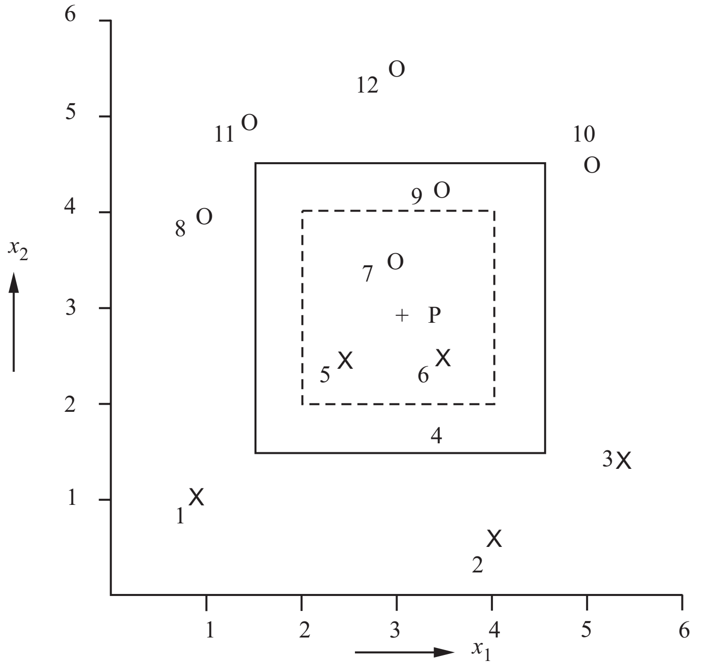

Figure 3.6 Example of clustering using the cube algorithm

This algorithm requires a much steeper cost for storage. It requires the storage of N sorted feature values and their indexes for each axis, or $2dN$ words. Processing time involves the sorting of dN word arrays which is of the order of $dN \log N$ comparisons.

3.4.3 Searching for the Nearest Neighbour by Projection

Various projection algorithms are available for performing nearest neighbour searches. In the simplest case of one-dimensional space, if the point set is ordered, we can find the nearest neighbours of point P by locating P's position in the order. Finding its successor and predecessor and taking the closest of these two points, will give the nearest neighbour.

In the two-dimensional case, the points are sorted in increasing order using the first coordinate value. To find the nearest neighbour to point P, we locate P's position in the sorted list of the first coordinate value. We then search out the nearest neighbours of P. We examine the point that is closer to P in x-distance first and remember the closest point to P so far observed (which we will call Q). The search can be pruned as soon as we know that any other point we would visit is necessarily further from P than Q is. Specifically, if only the distance in the x-direction from P to the next point to be searched is greater than the distance to Q, then the search can be stopped and Q can be reported as P's nearest neighbour. Figure 3.7 shows this graphically.

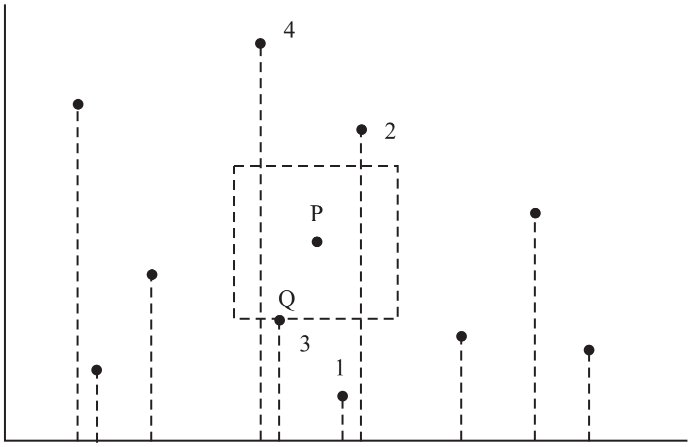

Figure 3.7 Searching for the nearest neighbour by projection on the x-axis

This process is modified to project the points onto both the x- and y-axes, instead of only the x-axis. Now the search is carried out both in the x- and y-axes in parallel. In other words, the first step is on the x-axis, the second on the y-axis, the third on the x-axis etc. The search stops as soon as the search in either direction stops. Figure 3.8 shows the nearest neighbour search using two projections.

Example 12

In the example given in Figure 3.1, if the test point is P = (3, 2), all the data points are arranged according to the value in the x-axis. The order in increasing order of x-axis values is X_{13}, X_{14}, X_{16}, X_7, X_9, X_6, X_{17}, X_{18}, X_8, X_{10} etc. The distance of these points in the x-direction is 0.2, 0.2, 0.5, 0.8, 0.8, 1.0, 1.0, 1.0, 1.2 and 1.2. These points are taken in the order given, and their actual distance to P is found. At each point, the closest point to P so far is kept in memory. The distances from X_{13}, X_{14}, X_{16}, X_7, X_9, X_{15}, X_6, X_{17} and X_{18} to P are 1.162, 1.315, 1.118, 1.131, 1.44, 1.7, 1.414, 1.414 and 1.64 . Starting from X_{13}, the actual distance is checked and the minimum so far is noted. When X_{16} is reached, the minimum distance changes to 1.118. When X_{18} is reached, the closest point is X_{16} with a distance of 1.118 . The next point X_8 has a distance in the x-direction of 1.2 which is larger than the actual distance of the closest point X_{16}. Then it is not necessary to check any more points, and X_{16} is the closest point to P.

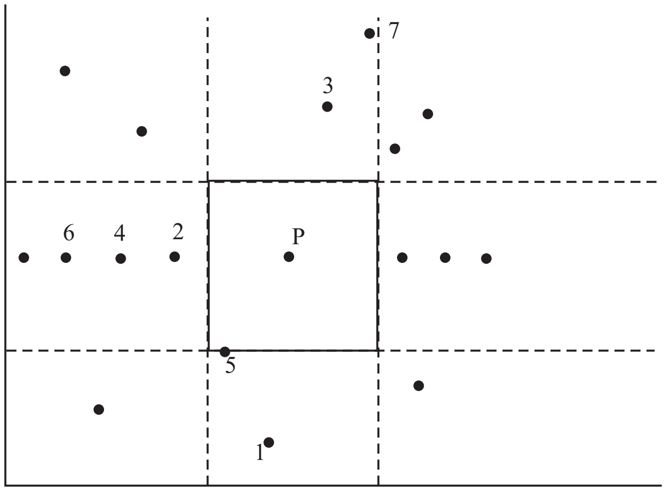

Figure 3.8 Searching for the nearest neighbour using two projections

3.4.4 Ordered Partitions

A pre-processing technique called the ordered partition method is used to reduce the computational requirements for finding the k nearest neighbours to a test sample among n training samples in a d-dimensional feature space. The training patterns are partitioned by their first coordinate values along each axis such that the ordering characteristic is preserved between the partitions. The width of the partitions (or blocks) are adjusted so that each partition contains the same number of samples.

The search starts from level 0 and the squared distance r_l^2 is found for each branch of the tree. This distance is defined recursively as

$$r_l^2 = \begin{cases} r_{l-1}^2 + \min((x_l - a_l)^2, (x_l - b_l)^2), & \text{for } l \neq d \\ r_{d-1}^2 + (x_d - a_d)^2, & \text{for } l = d \end{cases}$$

At each level l, the squared distance r_l^2 is found from the test sample. The node with the smallest distance at that level is chosen to continue the search. When a terminal node is found, that distance is noted and the smallest distance d is kept in memory. Any node whose distance is greater than d at any level need not be searched.

This procedure finds the k nearest neighbours in a constant expected time.

EXAMPLE 13

Consider the following three-dimensional patterns:

$$\begin{array}{lll} X_1 = (1, 1, 1) & X_2 = (1.2, 2.0, 4.0) & X_3 = (2.0, 1.5, 3) \\ X_4 = (2.5, 3.0, 5.0) & X_5 = (3.0, 7.0, 6.0) & X_6 = (3.5, 2.5, 3.5) \\ X_7 = (4.0, 6.0, 2.5) & X_8 = (4.5, 5.5, 4.5) & X_9 = (5.0, 1.5, 2.0) \\ X_{10} = (5.5, 6.5, 1.5) & X_{11} = (6.0, 8.0, 7.5) & X_{12} = (7.0, 9.0, 8.0) \end{array}$$

Figure 3.9 shows the partitioned data when these patterns are first partitioned according to the first dimension, then the second dimension and then the third dimension.

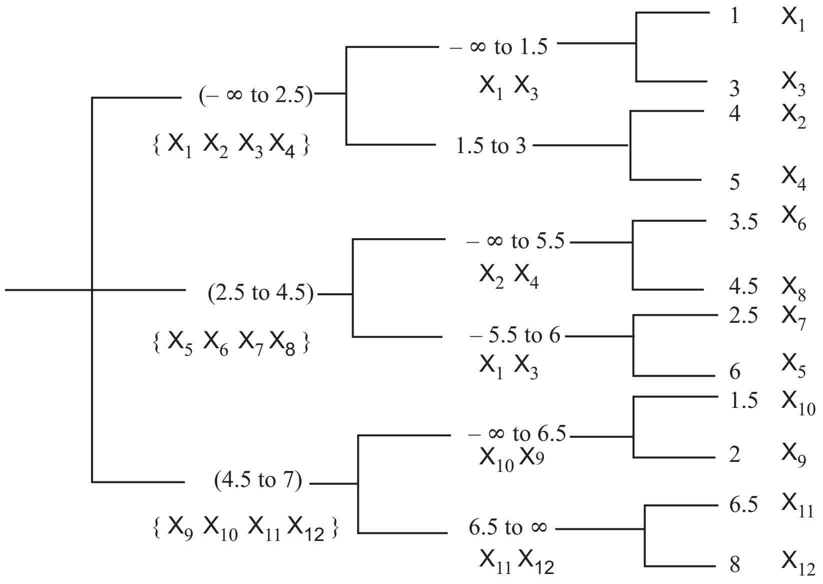

Figure 3.9 Partitioning of data for search using ordered partitions

If the point P is (7.0, 7.0, 7.0), searching through the partitioned data, first the last branch, i.e., the first dimension being between 4.5 to 7 is chosen. From there, the second branch is chosen for the second dimension, i.e., 6.5 to ∞. From there, X_{11} is chosen as the closest point up to that point. This will give a distance of 0 + 0.25 + 0.25 = 0.5 . The other branch in the second dimension namely $-\infty$ to 6.5 is also searched. But the other two branches in the first dimension are not searched as the squared distance in the first level itself exceeds 0.5 . This distance is 20.25 for the first branch and 6.25 for the second branch. Therefore, X_{11} is the closest point to point P.

3.4.5 Incremental Nearest Neighbour Search

This incremental search algorithm finds the next nearest neighbour more efficiently by eliminating redundant computations. The collection of n training data is stored in a k-d tree. The k-d tree is a binary tree structure for storing and performing operations on data containing k-dimensional keys. The root of the k-d tree represents the k-dimensional space containing the collection of data. Each node represents a sub-space containing a sub-set of the collection of data. Partitioning a non-terminal node's sub-space into two parts by a hyper-plane perpendicular to one of the k coordinate axes gives the two children of the node. The position of the hyper-plane is chosen so that each of the children contains approximately half the parents' data patterns. The discriminating axis is determined by choosing the axis along which the parent's data are most dispersed. Children that contain less than a threshold number of data points become leaf nodes of the tree. Each leaf node is represented in the k-d tree as a hyper-volume. Figure 3.10 shows the construction of the k-d tree for two-dimensional data. Here each leaf node is represented by a rectangle in the k-d tree.

The nearest neighbour search using the k-d tree is carried out by first locating the test pattern in the k-d tree created using the training data. By descending the tree, the leaf node which spatially contains the test point is located. The distance of all the data points in the leaf node from the test point is found and ordered in increasing order. The data point closest to the test point is a possible candidate for the nearest neighbour. The distance r from the test point to this data point is an upper bound for the distance to the actual nearest neighbour. If we draw a hyper-sphere S_r centered at the test point with radius r and if S_r is completely enclosed by the hyper-volume of the leaf node, then only the points in the leaf node need be searched to find the nearest neighbour. If S_r is not entirely within the leaf node, the tree is ascended one level and then descended one level to the other children. Only leaf nodes which spatially intersect S_r need to be considered. Here again the distance to all the data points in this leaf node are computed and the data point which is closest in these distances and those computed for the earlier leaf node are considered. The closest point defines the new value of r. This process of ascending the tree and evaluating new data points is continued till S_r is entirely enclosed by the space defined by the current non-terminal node.

Example 14

Figure 3.10 shows a k-d tree drawn for the set of points. The closest neighbour for the point P is found within the same leaf node and a circle drawn. Since this circle lies completely within the square which represents a leaf node, the point found is the closest neighbour. In the case of the point Q, the circle also includes other leaf nodes. It is therefore necessary to search those leaf nodes also to find the nearest neighbour.

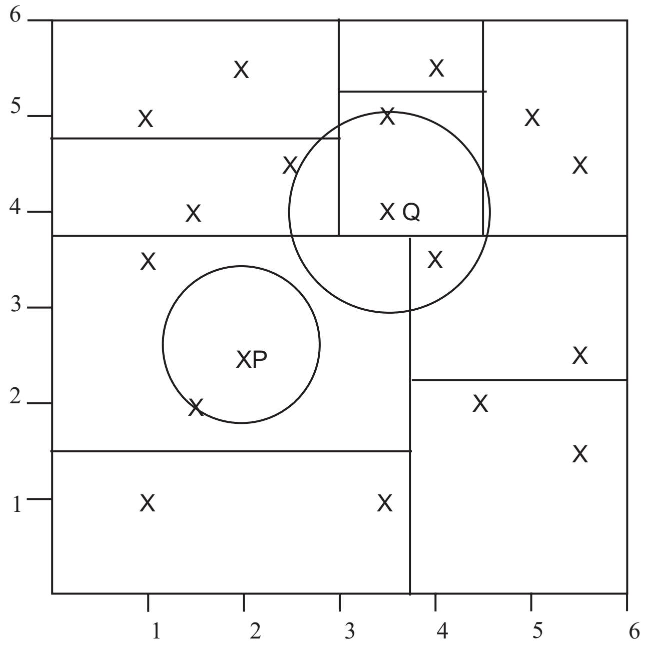

Figure 3.10 Construction of the k-d tree and search for the closest node

3.5 Data Reduction

In supervised learning, the efficacy of the classification is dependent on the training data. A larger training data may lead to better classification but may require a longer time for the classification process. While using classification tools such as neural networks or decision trees, it is only the design time which increases. Once the design is complete, classification of new samples will be fast. But in the case of techniques based on the nearest neighbour rule where every new sample has to be compared to all the training patterns, the time to classify each sample is large. In such cases, reducing the number of training patterns will definitely lead to faster classification. This reduction has to be carried out without sacrificing the classification accuracy.

Another option to reduce classification time is feature selection where the dimensionality of the patterns is reduced so that a sub-set of the features which gives good classification accuracy is used.

Let $F = \{f_1, ..., f_d\}$ be the set of features describing objects as d-dimensional vectors in R^n and let $X = \{X_1, ..., X_n\}$, $X_j \epsilon R^n$ be the data set. Associated with each X_j, $j = 1, ..., n$, is a class label from the set $C = 1, ..., c$. Prototype selection and feature selection work in the following way. For prototype selection, instead of X, we use a sub-set $S_1 \subset$ X and for feature selection, instead of F, we use $S_2 \subset F$. Feature selection has already been described in Chapter 2. Prototype selection is described in the next section.

3.6 Prototype Selection

Given a set of samples, a prototype would be a sample which would represent or be a typical sample for a large number of samples in the set. For example, given a set of samples, the centroid of the set could be called a prototype of the set. Instead of a single sample, we can have a number of prototypes which collectively represent a set of samples. Prototype selection essentially entails finding the sample or set of samples to represent a larger set of samples.

In spite of the quality of performance of the nearest neighbour rule, it has some practical disadvantages. The performance of the nearest neighbour algorithm increases with increase in the number of training patterns. However, if the training set is too large, use of the nearest neighbour and related techniques will require each test sample to be compared with every sample in the training set. Time and space requirements increase linearly with increase in the number of training patterns. The computational burden is significant and makes this method inapplicable for large training sets. Prototype selection provides a solution to this problem.

Prototype selection refers to the process of reducing the training set used for classification. This can either mean that a representative sub-set of the training patterns is selected or some prototypes based on the training set are chosen.

Formally, let $\chi = \{(\mathrm{X}_1, \theta_1), (\mathrm{X}_2, \theta_2), ..., (\mathrm{X}_n, \theta_n)\}$ be the given labelled training set. Prototype selection refers to the process of obtaining $\chi^{'} = \{\mathrm{X}^1, \mathrm{X}^2, ..., \mathrm{X}^k\}$ from χ such that (i) $k < n$ and (ii) either $\chi^{'}$ is a sub-set of χ or each $\mathrm{X}^i, 1 \leq i \leq k$ is obtained from patterns in χ.

This selection has to be carried out so that there is no significant decrease in classification accuracy. Based on criteria such as classification accuracy, number of prototypes etc., the optimal prototype set has to be obtained. One question that needs to be answered before doing this process is to decide on how many prototypes need to be selected.

3.6.1 Minimal Distance Classifier (MDC)

One simple prototype selection strategy is to use the minimum distance classifier. Each class is represented by the sample mean or centroid of all the samples in the class. This method selects only one prototype to represent a class. If $X_{i1}, X_{i2},..., X_{iN}$ are the N training patterns for the class i, then the representative sample will be

$$C_i = \frac{\Sigma_{j=1}^{N} X_{ij}}{N}$$

In order to classify a test pattern P, if C_k is the centroid closest to P, it is assigned the class label k of which C_k is the representative pattern.

The major advantage of MDC is that the time complexity is $O(n + mC)$, where $O(n)$ is the time required for computing the centroids and $O(mC)$ is the time required to search for the nearest centroid of the m test patterns. C is the number of classes. The MDC results are the same as the optimal classifier when the classes are normally distributed with a diagonal covariance matrix and the variances in different directions are the same (isotropic classes).

Example 15

Consider the example data set given in Figure 3.1. The training set consists of 5 patterns of Class 1, 7 examples of Class 2 and 6 examples of Class 3. The centroid for Class 1 can be obtained by taking the patterns X_1, X_2, X_3, X_4 and X_5 and finding the mean for each coordinate. In this case, the centroid will be (1.0,1.0). Similarly, the centroid for Class 2 will be (4.11, 3) and the centroid for Class 3 will be (3.62, 0.72). This is represented in Figure 3.11.

Consider a test pattern P at (3.0, 2.0). To determine its class, we find the distance of P from the centroids of the three classes.

From P to the centroid of Class 1, the distance is 2.24.

From P to the centroid of Class 2, the distance is 1.49.

From P to the centroid of Class 3, the distance is 1.42.

Since P is closest to the centroid of Class 3, it is classified as belonging to Class 3 according to MDC.

Using MDC, only the distance of P to three points have to be computed whereas using the nearest neighbour classifier, the distance of P to 18 points have to be computed. In general, C distances have to be computed using MDC and n distances computed using the nearest neighbour classifier. It is to be noted that $C << n$ in large-scale applications.

Prototype selection methods can be classified according to the following criteria:

1. Algorithms which select prototypes already available in the training set, i.e., $\chi^{'} \subset \chi$.
2. Algorithms which make use of the training set and produce prototypes which are different from those in the training set, i.e., $\chi^{'} \not\subset \chi$.

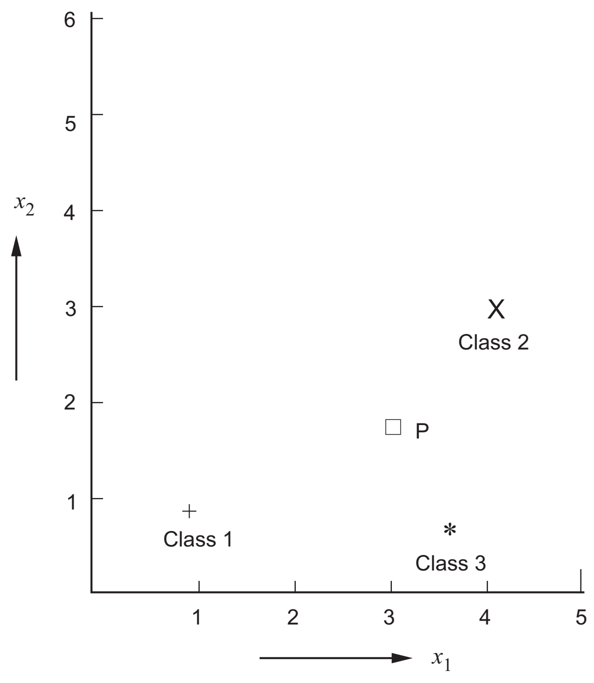

Figure 3.11 Classification using MDC

It should be noted that the second type can be changed to the first type if the nearest sample in the training set is taken as the prototype from the one generated using the algorithm.

Depending on the type of algorithm used, prototype selection can be basically divided into three types.

1. The first type of algorithm determines a sub-set of the training patterns by classifying the training set itself (or part of it) by using the prototypes selected. These are called condensation algorithms. With these algorithms, we obtain a reduced and consistent set of prototypes without significantly degrading the performance of the original data set.
2. Another type of algorithm involves editing to obtain a sub-set of the data set giving improved performance by removing "bad" prototypes.
3. The third type of algorithm clusters the training patterns and uses representative patterns, where one pattern represents each cluster.

In this section, we will discuss the first two methods in detail.

3.6.2 Condensation Algorithms

Even though the MDC is a time-efficient classifier, it fails to give good results when the centroid is not a good representative of the patterns in a class.

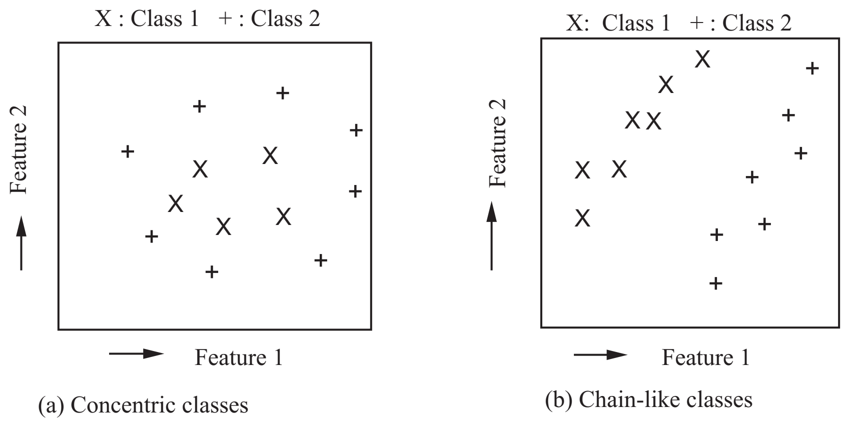

Figure 3.12 Example of data sets having concentric and chain-like classes

This happens often in nature when the classes are chain-like and elongated in one direction or when two or more classes are concentric which means that patterns of different classes have almost identical sample means. It is illustrated in Figure 3.12. In such cases, it may be meaningful to use more than one representative pattern from each class. One of the first studies on prototype selection is the condensed nearest neighbour (CNN) rule. Other methods include iterative condensation algorithms (ICA), the reduced nearest neighbour (RNN) and modified condensed nearest neighbour (MCNN) algorithms. These methods aim to preserve the integrity of the process by ensuring that the condensed set is consistent with the original set. This means that all the original samples are correctly classified by the condensed set under the nearest neighbour rule. Many of the methods ensure consistency but not minimality of the condensed sub-set. In fact, CNN, which is sensitive to the initial ordering of the training set, can end up with the whole training set (though it is unlikely).

Condensed Nearest Neighbour Algorithm

One of the first and most popularly used methods of prototype selection is the condensed nearest neighbour (CNN) algorithm. In this algorithm, a single pattern is put in the condensed set first. Then each pattern is considered and its nearest neighbour in the condensed set is found. If its label is the same as that of the pattern in the condensed set, it is left out; otherwise the new pattern is included in the condensed set. After one pass through the training patterns, another iteration is carried out where each training pattern is classified using the condensed set already formed. These iterations are carried out till no more patterns are added to the condensed set. At this stage, the condensed set is consistent. The algorithm is given below.

Let $Train$ be the set of N training pattern pairs given by

$$Train = (\mathrm{X}_1, \theta_1), (\mathrm{X}_2, \theta_2), \cdots, (\mathrm{X}_N, \theta_N).$$

Each pair has a pattern and its class label as the two components. Let $Condensed$ be a set which is empty initially.

STEP 1: Select the first pair from $Train$ and add it to $Condensed$. Let *Reduced* be the set given by $Train - Condensed$.

STEP 2: Select the first pair from *Reduced* and find the nearest neighbour of the pattern, in the selected pair, in $Condensed$. If the class label associated with the nearest neighbour and the selected pattern are different, then add the selected pair to $Condensed$. Delete the selected pair from *Reduced*.

STEP 3: Repeat Step 2 till *Reduced* is empty.

STEP 4: Set *Reduced* $= Train - Condensed$. Repeat Steps 2 and 3.

STEP 5: Stop if there is no change in $Condensed$ (or *Reduced*) during two successive iterations of Step 4 else iterate through Steps 2, 3 and 4.

EXAMPLE 16

Consider the following set of training samples belonging to three classes, Class 1 represented by a cross, Class 2 represented by a circle and Class 3 represented by a plus.

$\mathrm{X}_1 = (1.0, 1.0, 1)$,	$\mathrm{X}_2 = (1.0, 2.0, 1)$,	$\mathrm{X}_3 = (1.5, 1.5, 1)$
$\mathrm{X}_4 = (2.0, 2.0, 1)$,	$\mathrm{X}_5 = (3.0, 2.0, 2)$,	$\mathrm{X}_6 = (4.0, 2.0, 2)$
$\mathrm{X}_7 = (4.0, 3.0, 2)$,	$\mathrm{X}_8 = (5.0, 2.5, 2)$,	$\mathrm{X}_9 = (2.0, 3.0, 3)$
$\mathrm{X}_{10} = (3.0, 3.5, 3)$,	$\mathrm{X}_{11} = (2.0, 4.0, 3)$	

This data set is shown in Figure 3.13.

Let the patterns be sent to the algorithm in order from X_1 to X_{11}. First X_1 is put into the set $Condensed$. Since only X_1 is in $Condensed$, X_2, X_3 and X_4 are closest to X_1 and since they have the same class label as X_1, nothing is done. Since X_5 has a class label different from the class label of X_1, it is added to $Condensed$. Now X_6 is compared to X_1 and X_5. Since it is closer to X_5 which has the same label as X_6, nothing is done.

X_7 and X_8 are also closer to X_5 which has the same label, and hence they are not included in *Condensed*. The distance of X_9 from X_1 is 2.236 and the distance of X_9 from X_5 is 1.414. Hence X_9 is closest to X_5 of all the patterns in *Condensed*. But since X_9 has a different label from X_5, it is included in *Condensed*. Now *Condensed* has the patterns X_1, X_5 and X_9. The patterns X_{10} and X_{11} are closest to X_9 in *Condensed* and are therefore not included in *Condensed*. This completes one iteration.

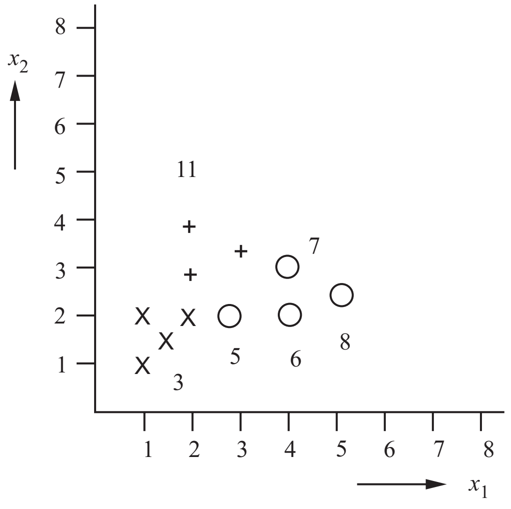

Figure 3.13 Example data set

In the second iteration, X_2 and X_3 are closest to X_1 which has the same label. Therefore, they are not included in *Condensed*. X_4 is equidistant from X_5 and X_9. If X_4 is taken to be closest to X_5 to break the tie, since its label is different from X_5, it is included in *Condensed*. X_6, X_7 and X_8 are closest to X_5 and are not included in *Condensed*. The patterns X_{10} and X_{11} are closer to X_9 which has the same label and therefore are not included in the condensed set.

In the next iteration, since there is no change in *Condensed*, the condensed set consists of the patterns X_1, X_4, X_5 and X_9.

Obtaining the condensed data set is a time-consuming process but once it is available, classification is much faster compared to using the entire training data set. CNN is typically good at reducing the size of the training data set. But this is achieved at the cost of some reduction in the classification accuracy. Also, CNN is *order-dependent*. The set *Condensed* obtained varies in size and contents based on the order in which the training patterns are presented to the above algorithm.

It is very likely that there will be considerable savings in computational expense during the classification phase if this method is used. The trade-off involved is the computational cost of deriving the condensed set, which in itself can be considerable.

But as it involves design time and not the on-line time required for classification, it may not be so significant. This method is sensitive to the initial ordering of the input training data set. Both the contents and the size of the resulting condensed set are likely to change with a different ordering.

Example 17

In Figure 3.14, there are six points. The points 1: (1, 2), 2: (2, 2), and 3: (2, 1) belong to one class and the points 4: (2.7, 2), 5: (4, 2) and 6: (4, 1) belong to another class. If the initial ordering of the data is 1, 2, 3, 4, 5 and 6, then the points obtained in the condensed set will be 1, 4 and 2. If the initial ordering is 2, 3, 1, 4, 5 and 6, the condensed set will have 2 and 4. Thus the condensed set is different and of different sizes in the two cases.

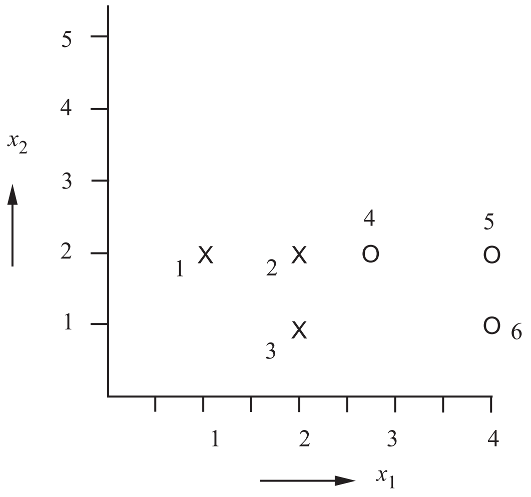

Figure 3.14 Example data set of six points

Modified Condensed Nearest Neighbour Algorithm

In any problem, especially one of high dimensionality, the boundaries of each class are very difficult to determine. The modified condensed nearest neighbour(MCNN) algorithm attempts to partition the region of a class into simpler non-overlapping regions. This is done in an incremental manner, adding prototypes to a representative prototype set till finally all the training patterns are classified correctly using this set of representative prototypes. At this stage, the region pertaining to each class is resolved into approximate Voronoi regions.

$$R_j = \bigcup_{i=1}^{n} V_{ji},\ j = 1, \ldots, c$$

where n is the number of regions in class j. The total number of classes is c.

In this method, a set of prototypes is obtained in an incremental manner. The algorithm starts with a basic set of prototypes comprising one pattern from each class. The training set is classified using these prototypes. Based on the misclassified samples, a representative prototype for each class is determined and added to the set of basic prototypes. Now the training set is classified again with the augmented set of prototypes. Representative prototypes for each class are again determined based on the misclassified samples. This process is repeated till all patterns in the training set are classified correctly. Determining the representative pattern for the misclassified samples in each class is also done in an iterative manner.

The method used for finding a single representative sample of a group of patterns depends on the data set used. One simple method of doing this is to use the centroid as the representative of the group of patterns. Of course, this works well for patterns having a Gaussian distribution with the covariance matrix equal and diagonal. The centroid can also be used if the class can be split up into regions with the above property.

The algorithm for MCNN is given below:

```
Step 1:  Let the training samples be the set $Train$. Set $Prototype = \phi$. Set $Typical = \phi$.
Step 2:  Find a typical pattern for each class from $Train$ and put the patterns into
         $Typical$.
Step 3:  $Prototype = Prototype \bigcup Typical$
Step 4:  With $Prototype$, classify $Train$ using the nearest neighbour algorithm.
Step 5:  Set $Sub\text{-}group = \phi$. Put the misclassified samples into $Sub\text{-}group$.
Step 6:  If $Sub\text{-}group = \phi$, stop. $Prototype$ gives the set of prototypes finally chosen.
Step 7:  Set $Typical = \phi$. Find typical patterns for each class in $Sub\text{-}group$ and put
         them into $Typical$.
Step 8:  Use $Typical$ to classify $Sub\text{-}group$.
Step 9:  Find the correctly classified patterns. Set $Sub\text{-}group = \phi$. Put correctly
         classified patterns in $Sub\text{-}group$ and if misclassified samples exist, go to
         Step 7.
Step 10: Go to Step 3.
```

The way the typical patterns are found is shown in Steps 5–9. They are found in each class to represent and classify the misclassified patterns. Only the correctly classified samples are kept in *Sub-group*. Typical patterns are found again. This is done till there are no misclassified samples when the typical patterns are used to classify the samples in *Sub-group*.

Using this algorithm, the number of misclassified samples from the training set keeps coming down till all the training patterns are classified correctly by the condensed set (as happens in the case of CNN).

Example 18

Consider the example given in Figure 3.15. The centroid C_i is found for each class i. We get

$C_1 = (1.375, 1.625)$

$C_2 = (4.0, 3.0)$

$C_3 = (2.33, 4.0)$

C_1 is closest to X_3, C_2 is closest to X_7 and C_3 is closest to X_{11}.

The prototype set is thus taken as X_3, X_7 and X_{11}. If the training set is classified using this prototype set, all the patterns are correctly classified except X_{10} which is equidistant from X_7 and X_{11}. Since X_7 occurs in the training set before X_{11}, if X_{10} is classified as belonging to Class 2, it is misclassified. The misclassified patterns in each class are used to find the next set of typical patterns which are added to the prototype set. At each iteration, the prototype set keeps building up till all the training patterns are classified correctly.

Unlike the CNN, MCNN is order-independent which means that the algorithm gives the same set of prototypes whatever may be the order of the patterns in the data set. While the design time is higher than that of CNN, the classification accuracy obtained using this method is found to be fairly good.

Example 19

In the CNN algorithm, the first pattern in the data set is always put into the condensed set. Then any pattern of the same class as the first pattern is not included into the condensed set. Only when a pattern of another class is encountered, is it put into the condensed set. The patterns which are subsequently put into the condensed set in this iteration and other iterations depend on the patterns already in the condensed set. Therefore, the CNN is an order-dependent algorithm. In the MCNN algorithm, if mean is the typical pattern for a cluster, the mean is found using all the patterns and the closest pattern to the mean is taken as the pattern to be put into the condensed set. This will not depend on the order in which the patterns are presented to the algorithm and makes the MCNN algorithm order-independent. In Figure 3.16, if the closest point to the centroid is taken as the typical sample, then the two patterns included in the prototype set will be 2 and 5. This is true for this data set in whatever order the data is available. The final prototype set will be 2 and 5.

3.6.3 Editing Algorithms

Editing removes samples which are outliers and reduces overlap among classes, thereby giving an improved performance compared to the original data set. Editing algorithms basically estimate the classification error using classification rules, and discards prototypes that are misclassified. The generalised editing scheme can be described as follows:

1. Using the error estimator ϵ, obtain an estimate of the classification error for the rule δ training with the set S. Let R be the set of misclassified samples.
2. Let $S = S - R$.
3. If the terminating condition is satisfied then stop, else go to Step 1.

The performance of the edited prototype set is dependent on the way the editing is done. Some editing methods partition the training set to get two independent sets for designing and testing.

One method of editing finds the k nearest neighbours of every sample and discards the sample if its label does not agree with the label of the largest number of its k neighbours.

The MULTIEDIT is an iterative procedure which uses the principles of "diffusion" and "confusion". It can be described as follows:

STEP 1: *Diffusion* Make a random partition of the data S into N sub-sets S_1,..., S_N.

STEP 2: *Classification* Classify the samples in S_i using the nearest neighbour rule with $S_{(i+1) \bmod N}, i = 1, ..., N$.

STEP 3: *Editing* Discard all the samples that were misclassified.

STEP 4: *Confusion* Pool all the remaining data to constitute the new set S.

STEP 5: *Termination* If Step 3 does not produce any editing, exit with the final solution, else go to Step 1.

EXAMPLE 20

Consider the data set given in Figure 3.15. This consists of the patterns

$X_1 = (1.0, 1.0, 1)$, $X_2 = (1.0, 2.0, 1)$, $X_3 = (1.5, 1.5, 1)$
$X_4 = (2.0, 2.0, 1)$, $X_5 = (1.0, 3.0, 2)$, $X_6 = (3.0, 2.0, 2)$
$X_7 = (4.0, 2.0, 2)$, $X_8 = (5.0, 2.5, 2)$, $X_9 = (3.0, 3.5, 2)$
$X_{10} = (2.0, 3.0, 3)$, $X_{11} = (2.0, 4.0, 3)$, $X_{12} = (3.0, 4.5, 3)$
$X_{13} = (4.0, 3.0, 3)$

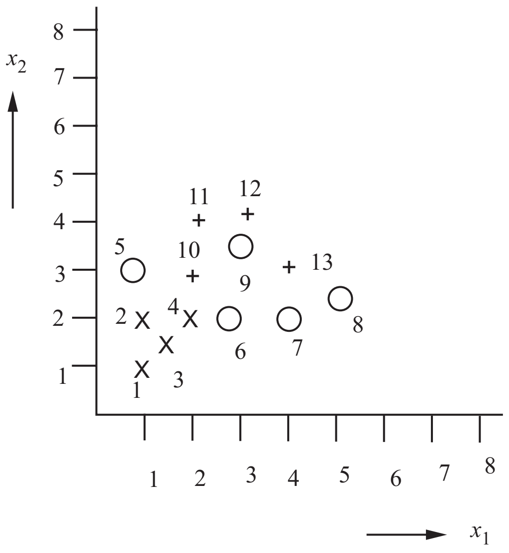

Figure 3.15 Example data set

Each triplet consists of the x-coordinate, y-coordinate and the class label. Let us make three groups S_1, S_2 and S_3 so that

$$S_1 = (X_1, X_2, X_5, X_7, X_{12})$$

$$S_2 = (X_3, X_6, X_8, X_{10})$$

$$S_3 = (X_4, X_9, X_{13}, X_{11})$$

Now if we classify S_1 using S_2 and S_3, the misclassified samples are X_5 and X_{12}. If we classify S_2 using S_1 and S_3, the misclassified sample is X_6. If we classify S_3 using S_1 and S_2, the misclassified samples are X_9 and X_{13}

So the patterns X_5, X_{12}, X_6, X_9 and X_{13} are discarded. This has the effect of removing outliers and making the boundaries of the classes more far apart and towards getting a linear boundary. The remaining patterns are retained giving

$$S = (X_1, X_2, X_3, X_4, X_7, X_8, X_{10}, X_{11})$$

Again, let us make three groups.

$$S_1 = (X_1, X_4, X_{10})$$

$$S_2 = (X_2, X_7, X_{11})$$

$$S_3 = (X_3, X_8)$$

Now we need to repeat the procedure of classifying S_1 using S_2 and S_3, classifying S_2 using S_1 and S_3 and classifying S_3 using S_1 and S_2. The rest of the procedure is left as an exercise.

3.6.4 Clustering Methods

We can obtain representative patterns from a given data set by selecting clusters of high density data regions in multi-dimensional space and replacing each such region by its representative like the centroid of the data in the region. Clustering sub-divides the patterns into groups so that patterns in the same group are similar while patterns belonging to different groups are dissimilar according to some criteria. There are many ways of clustering and they are described in Chapter 9.

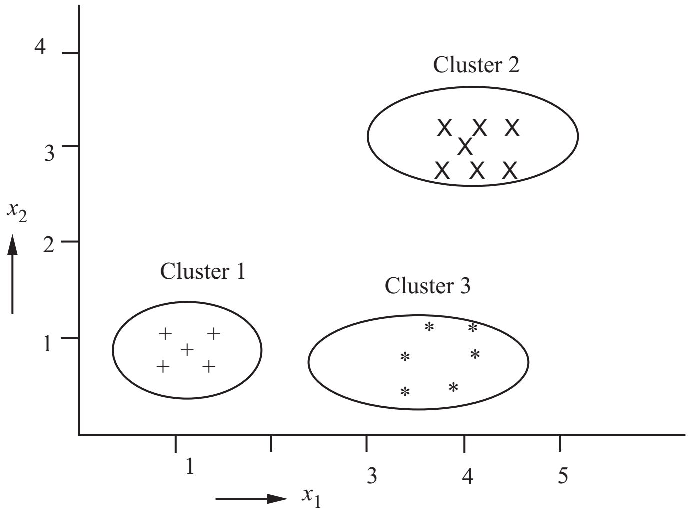

Figure 3.16 Clustering of patterns given in Figure 3.1

EXAMPLE 21

The patterns in Figure 3.1 can be divided into three clusters, one for each class. This is shown in Figure 3.16. If the centroid is the representative pattern of a cluster, Cluster 1 can be represented by the point (1.0,1.0), Cluster 2 can be represented by (4.11, 3) and Cluster 3 can be represented by (3.62, 0.72).

$$C_1 = (1.0, 1.0)$$

$$C_2 = (4.11, 3)$$

$$C_3 = (3.62, 0.72)$$

These centroids can be used as the representative pattern in the class. Now if there is a test pattern P at (4.2,1.8), the distance to the three centroids is

$$d(\mathrm{C}_1, \mathrm{P}) = 3.30$$

$$d(\mathrm{C}_2, \mathrm{P}) = 1.20$$

$$d(\mathrm{C}_3, \mathrm{P}) = 1.23$$

Using the nearest neighbour rule, the pattern P will be classified as belonging to Class 2.

It is also possible to have more clusters for each class. This is shown in Figure 3.17. Each cluster can be represented by one typical pattern.

The centroids of the six clusters are

$$\begin{array}{ll} \mathrm{C}_{11} = (1.0, 0.867) & \mathrm{C}_{12} = (1.0, 1.2) \\ \mathrm{C}_{21} = (3.8, 3.0) & \mathrm{C}_{22} = (4.24, 3.0) \\ \mathrm{C}_{31} = (3.43, 0.65) & \mathrm{C}_{32} = (4.0, 0.85) \end{array}$$

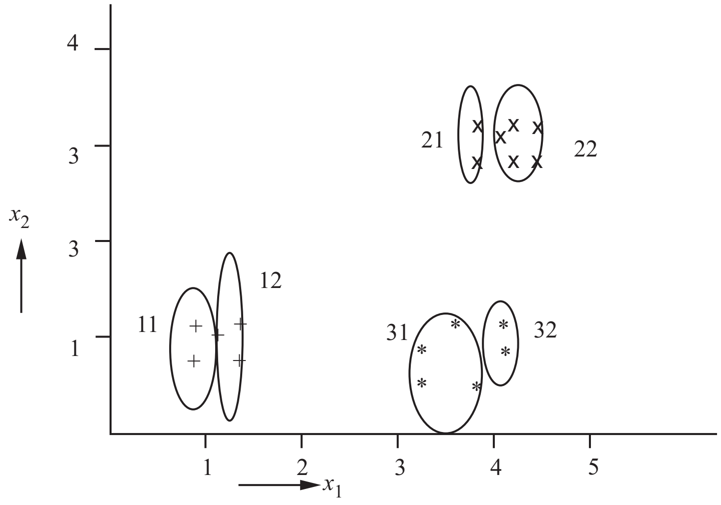

Figure 3.17 Further clustering of the patterns given in Figure 3.1

Taking the distance of the test point P from these centroids, we get

$$\begin{array}{ll} d(\mathrm{C}_{11}, \mathrm{P}) = 3.33 & d(\mathrm{C}_{12}, \mathrm{P}) = 3.26 \\ d(\mathrm{C}_{21}, \mathrm{P}) = 1.26 & d(\mathrm{C}_{22}, \mathrm{P}) = 1.20 \\ d(\mathrm{C}_{31}, \mathrm{P}) = 1.38 & d(\mathrm{C}_{32}, \mathrm{P}) = 0.97 \end{array}$$

By the nearest neighbour rule, P is closest to the centroid C_{32} and therefore has the same label as the patterns in the cluster which is Class 3.

3.6.5 Other Methods

There are some variations on the CNN in which the algorithms are slightly different or in which different concepts of neighbourhood like the mutual neighbourhood value are used. Some methods use the concept of graph neighbours.

Stochastic techniques can also be applied to the task of prototype selection. The genetic algorithm can be used to find an optimal sub-set of prototypes. In this method, binary chromosomes of length equal to the number of patterns in the original set are used. A ‘‘1’’ indicates that the corresponding exemplar is to be included in the sub-set and a ‘‘0’’ means that the corresponding exemplar is not included. The classification accuracy obtained by using the reduced set suggested by the chromosome, with a separate validation set is used as the fitness function. A genetic algorithm can be used to do simultaneous editing and feature selection. Here the binary string consists of $n + d$ bits, where n is the number of training patterns and d is the dimensionality of the samples.

Simulated annealing or the Tabu search can be used instead of the genetic algorithm. Here the current solution would be the binary string as described for editing by the genetic algorithm. The evaluation of the strings would also be done in the same way.

Discussion

The nearest neighbour classifiers are intuitively appealing and effective. The drawback is in the implementation since considerable computation time and space will be required for large training data. To reduce the computation time, various approaches have been proposed which result in retrieving the nearest neighbours in a short time. Many of these methods require pre-processing of the training set. This chapter discussed the nearest neighbour classifiers and the various approaches to efficiently find the nearest neighbours to the test patterns.

Prototype selection forms an abstraction of the training set which reduces the time and storage requirement by using the reduced prototype set instead of the complete training set. Various methods for prototype selection were discussed in this chapter.

Further Reading

The nearest neighbour algorithm is described well by Cover and Hart (1967). Patrick and Fischer (1970) proposed the kNN algorithm. The mkNN algorithm is aptly described by Dudani (1976) while the fuzzy kNN algorithm is discussed by Jozwick (1983).

A number of papers have been published on finding efficient algorithms to find the nearest neighbours of a pattern. Fukunaga and Narendra (1975) explain the branch and bound algorithm to find k nearest neighbours. Miclet and Dabouz (1983)propose an improvement on the branch and bound algorithm. Yunck (1976) identifies the nearest neighbours by projecting the patterns on to a hypercube. Friedman et al. (1975) show how nearest neighbour search can be carried out using projection. Papadimitriou and Bentley (1980) analyse an algorithm for nearest neighbour search using projection. Finding the nearest neighbours using ordered partitions is described by Kim and Park (1986). Broder (1990) describes incremental nearest neighbour search. Other papers on fast algorithms to find nearest neighbours include those by Zhang and Srihari (2004), McNames (2001) and Lai et al. (2007).

The condensed nearest neighbour algorithm which is one of the earliest methods of prototype selection was proposed by Hart (1968). Devi and Murty (2002) describe the modified condensed nearest neighbour algorithm. Dasarathy (1994) found a minimum consistent set (MCS). Gates (1972) describes the reduced nearest neighbour algorithm. The concept of a mutual nearest neighbour is used to find the prototype set by Gowda and Krishna (1979). Construction and use of proximity graphs for finding the prototypes is described by Sanchez (1995). Kuncheva (1995) uses a genetic algorithm to carry out editing of the training patterns. Kuncheva and Jain (1999) use a genetic algorithm to carry out both editing and feature selection simultaneously. Swonger (1972) gives an algorithm for condensation of the training data. Dejiver and Kittler (1980), Tomek (1976) and Wilson (1972) use editing for prototype selection. Chang (1974) and Lam, Keung and Ling (2002; 2002) have also written papers on prototype selection.

Exercises

1. Give an example of a data set for which the kNN classifier gives better results than the NN classifier. Is there an example for which the NN classifier gives better results than the kNN classifier?
2. Give an example where the MkNN gives the correct classification as compared to the kNN classifier.
3. Discuss the performance of the kNN classifier with different values of k. Give an example and show the results obtained as k varies. What happens when $k = n$?
4. Discuss the performance of the MkNN classifier with different values of k. Give an example and show the results obtained as k varies.
5. Show with an example that the CNN algorithm is order-dependent.

6. Give an example where the minimum distance classifier gives good results. Also give an example where the minimum distance classifier gives poor results.

7. Consider the following set of two-dimensional vectors corresponding to classes w_1 and w_2.

w_1	w_2
$(1, 0)$	$(0, 0)$
$(0, 1)$	$(0, 2)$
$(0, -1)$	$(0, -2)$
$(0, -2)$	$(-2, 0)$

   (a) Plot the decision boundary corresponding to the minimum distance classifier.

   (b) Plot the decision boundary corresponding to the nearest neighbour algorithm for classification.

8. Consider the set of two-dimensional patterns :

   (1, 1, 1), (1, 2, 1), (1, 3, 1), (2, 1, 1), (2, 2, 1), (2, 3, 1), (2, 3.5,1), (2.5, 2, 1),(3.5, 1, 1), (3.5, 2, 1), (3.5, 3, 2), (3.5, 4, 2), (4.5, 1, 2), (4.5, 2, 2), (4.5, 3, 2),(5, 4, 2), (5, 5,2),(6, 3, 2), (6, 4, 2), (6, 5, 2)

   where each pattern is represented by feature 1, feature 2 and the class.

   (a) If a test pattern P is at (3.8, 3.1), find the class of P using the nearest neighbour algorithm.

   (b) Find the class of P using the kNN algorithm, where k is 3.

   (c) Find the class of P using MkNN algorithm, where k is 3.

9. Consider the set of two-dimensional patterns:

   (1, 1, 1), (1, 2, 1), (1, 3, 1), (2, 1, 1), (2, 2, 1), (2, 3, 1), (2, 3.5, 1), (2.5, 2, 1),(3.5, 1, 1), (3.5, 2, 1), (3.5, 3,2), (3.5, 4,2), (4.5, 1, 2), (4.5, 2, 2), (4.5, 3, 2), (5, 4, 2), (5, 5, 2), (6, 3, 2),(6, 4,2), (6, 5, 2)

   where each pattern is represented by feature 1, feature 2 and the class. Find the centroid of the two classes. Use the minimum distance classifier to find the class of a point P at $(3.8, 3.1)$.

10. Consider the set of two-dimensional patterns:

    (1, 1, 1), (1, 2, 1), (1, 3, 1), (2, 1, 1), (2, 2, 1), (2, 3, 1), (2, 3.5, 1), (2.5, 2, 1), (3.5, 1, 1), (3.5, 2, 1), (3.5, 3, 2), (3.5, 4, 2), (4.5, 1, 2), (4.5, 2, 2), (4.5, 3, 2), (5, 4, 2), (5, 5, 2), (6, 3, 2), (6, 4, 2), (6, 5, 2)

    where each pattern is represented by feature 1, feature 2 and the class.

    (a) Find the condensed set using condensed nearest neighbour algorithm.

    (b) If the patterns of Class 2 appear first, giving the set
    (3.5, 3, 2), (3.5, 4, 2), (4.5, 1, 2), (4.5, 2, 2), (4.5, 3, 2), (5, 4, 2), (5, 5, 2),(6, 3, 2),(6, 4, 2), (6, 5, 2), (1, 1, 1), (1, 2, 1), (1, 3, 1), (2, 1, 1), (2, 2, 1), (2, 3, 1), (2, 3.5,1), (2.5, 2, 1),(3.5, 1, 1), (3.5, 2, 1)
    find the condensed set using condensed nearest neighbour algorithm.

11. Consider the data set given in Problem 8. Find the condensed set using the modified condensed nearest neighbour algorithm.

Computer Exercises

1. Implement the nearest neighbour algorithm. Use it to classify several test patterns generated randomly for the data set in Figure 3.1.
2. Implement the kNN algorithm. Use it to classify several test patterns generated randomly for the data set in Figure 3.1.
3. Implement the MkNN algorithm. Use it to classify several test patterns generated randomly for the data set in Figure 3.1.
4. Use the NN, kNN and MkNN algorithm to classify several test patterns generated randomly for the data given in Exercise Problems 8, 9 and 10.
5. Use the MDC to classify the data given in Exercise Problems 8, 9 and 10.
6. Implement the CNN algorithm. Use it on the data sets in Exercise Problems 8, 9 and 10. Classify randomly generated test patterns.
7. Choose a large data set divided into training data and test data. Use the training data and classify the test patterns using NN, kNN, and MkNN algorithms. In each case, find the classification accuracy obtained on the test data.
8. Choose a large data set divided into training data and test data. Condense the data set using CNN algorithm. Classify the test patterns using the condensed data and find the classification accuracy.

9. Choose a large data set divided into training data and test data. Condense the data set using MCNN algorithm. Classify the test patterns using the condensed data and find the classification accuracy.

Bibliography

1. Broder, A. J. Strategies for efficient incremental nearest neighbour search. *Pattern Recognition* 23(1/2): 171–178. 1990.
2. Chang, C. L. Finding prototypes for nearest neighbour classifiers. *IEEE Trans. on Computers* C-23(11): 1179–1184. 1974.
3. Cover, T. M. and P. E. Hart. Nearest neighbor pattern classification *IEEE Trans. on Information Theory* IT-13: 21–27. 1967.
4. Dasarathy, Belur V. Minimal consistent set (MCS) identification for optimal nearest neighbour decision system design. *IEEE Trans. on Systems, Man and Cybernetics* 24(3). 1994.
5. Dejiver, P. A. and J. Kittler. On the edited nearest neighbour rule. *Proceedings of the 5th International Conference on Pattern Recognition*. pp. 72–80. 1980.
6. Dudani, S. A. The distance-weighted k nearest neighbor rule. *IEEE Trans. on SMC* SMC-6(4): 325–327. 1976.
7. Friedman, J. H., F. Baskett and L. J. Shustek. An algorithm for finding nearest neighbours. *IEEE Trans on Computers* C-24(10): 1000–1006. 1975.
8. Fukunaga, K. and P. M. Narendra. A branch and bound algorithm for computing k nearest neighbours. *IEEE Trans. on Computers*. pp. 750–753. 1975.
9. Gates, G. W. The reduced nearest neighbour rule. *IEEE Trans. on Information Theory* IT-18(3): 431–433. 1972.
10. Gowda, K.C. and G. Krishna. Edit and error correction using the concept of mutual nearest neighbourhood. *International Conference on Cybernetics and Society*. pp. 222–226. 1979.
11. Hart, P. E. The condensed nearest neighbor rule. *IEEE Trans. on Information Theory* IT-14(3): 515–516. 1968.
12. Jozwik, A. A learning scheme for a fuzzy kNN rule. *Pattern Recognition Letters* 1(5/6): 287–289. 1983.
13. Kim, B. S. and S. B. Park. A fast nearest neighbour finding algorithm based on the ordered partition. *IEEE Trans on PAMI* PAMI-8(6): 761–766. 1986.

14. Kuncheva, L. Editing for the k nearest neighbours rule by a genetic algorithm. *Pattern Recognition Letters* 16(8): 809–814. 1995.
15. Kuncheva, L. and L. C. Jain. Nearest neighbor classifier: Simultaneous editing and feature selection. *Pattern Recognition Letters* 20: 1149–1156. 1999.
16. Lai, Jim Z. C., Yi-Ching Liaw and Julie Liu. Fast k nearest neighbour search based on projection and triangular inequality. *Pattern Recognition* 40: 351–359. 2007.
17. McNames, James. A fast nearest neighbour algorithm based on a principal axis search tree. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 23(9): 964–976. 2001.
18. Miclet, L. and M. Dabouz. Approximative fast nearest neighbour recognition. *Pattern Recognition Letters* 1: 277–285. 1983.
19. Papadimitriou, C. H. and J. L. Bentley. A worst-case analysis of nearest neighbour searching by projection. *Lecture Notes in Computer Science* 85: 470–482. 1980.
20. Patrick, E. A., and F. P. Fischer. A generalized k nearest neighbor rule. *Information and Control* 16: 128–152. 1970.
21. Sanchez, J. S., F. Pla and F. J. Ferri. Prototype selection for the nearest neighbour rule through proximity graphs. *Pattern Recognition Letters* 18(6): 507–513. 1995.
22. Devi, V. Susheela and M. Narasimha Murty. An incremental prototype set building technique. *Pattern Recognition* 35: 505–513. 2002.
23. Swonger, C. W. Sample set condensation for a condensed nearest neighbor decision rule for pattern recognition. *Frontiers of Pattern Recognition.* 511–519. 1972.
24. Tomek, I. A generalization of the kNN rule. *IEEE Trans. on SMC* SMC-6(2): 121–126. 1976.
25. Tomek, I. An experiment with the edited nearest neighbour rule. *IEEE Trans. on SMC* SMC-6(6): 448–452. 1976.
26. Lam, Wai, Chi-Kin Keung and Charles X. Ling. Learning good prototypes for classification using filtering and abstraction of instances. *Pattern Recognition* 35: 1491–1506. 2002.

27. Lam, Wai, Chi-Kin Keung and Danyu Liu. Discovering useful concept prototypes for classification based on filtering and abstraction. *IEEE Trans PAMI* 24(8): 1075–1090. 2002.

28. Wilson, D. L. Asymptotic properties of nearest neighbour rules using edited data. *IEEE Trans. SMC* SMC-2(3): 408–421. 1972.

29. Yunck, Thomas P. A technique to identify nearest neighbours. *IEEE Trans. SMC* SMC-6(10): 678–683. 1976.

30. Zhang, Bin and Sargur N. Srihari. Fast k nearest neighbour classification using cluster-based trees. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 26(4): 525–528. 2004.