2 Representation

Learning Objectives

After reading this chapter, you will

- Know that patterns can be represented as
  - Strings
  - Logical descriptions
  - Fuzzy and rough sets
  - Trees and graphs
- Have learnt how to classify patterns using proximity measures like
  - Distance measure
  - Non-metrics which include
    - k-median distance
    - Hausdorff distance
    - Edit distance
    - Mutual neighbourhood distance
    - Conceptual cohesiveness
    - Kernel functions
- Have found out what is involved in abstraction of data
- Have discovered the meaning of feature extraction
- Know the advantages of feature selection and the different approaches to it
- Know the parameters involved in evaluation of classifiers
- Understand the need to evaluate the clustering accomplished

A pattern is a physical object or an abstract notion. If we are talking about the classes of animals, then a description of an animal would be a pattern. If we are talking about various types of balls, then a description of a ball (which may include the size and material of the ball) is a pattern. These patterns are represented by a set of descriptions. Depending on the classification problem, distinguishing features of the patterns are used. These features are called attributes. A pattern is the representation of an object by the values taken by the attributes. In the classification problem, we have a set of objects for which the values of the attributes are known. We have a set of classes and each object belongs to one of these classes. The classes for the case where the patterns are animals could be mammals, reptiles etc. In the case of the patterns being balls, the classes could be football, cricket ball, table tennis ball etc. Given a new pattern, the class of the pattern is to be determined. The choice of attributes and representation of patterns is a very important step in pattern classification. A good representation is one which makes use of discriminating attributes and also reduces the computational burden in pattern classification.

2.1 Data Structures for Pattern Representation

2.1.1 Patterns as Vectors

An obvious representation of a pattern will be a vector. Each element of the vector can represent one attribute of the pattern. The first element of the vector will contain the value of the first attribute for the pattern being considered. For example, if we are representing spherical objects, (30, 1) may represent a spherical object with 30 units of weight and 1 unit diameter. The class label can form part of the vector. If spherical objects belong to class 1, the vector would be (30, 1, 1), where the first element represents the weight of the object, the second element, the diameter of the object and the third element represents the class of the object.

EXAMPLE 1

Using the vector representation, a set of patterns, for example can be represented as

1.0, 1.0, 1 ;	1.0, 2.0, 1
2.0, 1.0, 1 ;	2.0, 2.0, 1
4.0, 1.0, 2 ;	5.0, 1.0, 2
4.0, 2.0, 2 ;	5.0, 2.0, 2
1.0, 4.0, 2 ;	1.0, 5.0, 2
2.0, 4.0, 2 ;	2.0, 5.0, 2
4.0, 4.0, 1 ;	5.0, 5.0, 1
4.0, 5.0, 1 ;	5.0, 4.0, 1

where the first element is the first feature, the second element is the second feature and the third element gives the class of the pattern. This can be represented graphically as shown in Figure 2.1, where patterns of class 1 are represented using the symbol +, patterns of class 2 are represented using X and the square represents a test pattern.

2.1.2 Patterns as Strings

The string may be viewed as a sentence in a language, for example, a DNA sequence or a protein sequence.

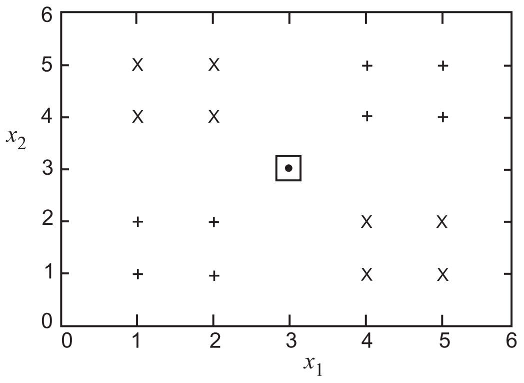

Figure 2.1 Example data set

As an illustration, a gene can be defined as a region of the chromosomal DNA constructed with four nitrogenous bases: adenine, guanine, cytosine and thymine, which are referred to by A, G, C and T respectively. The gene data is arranged in a sequence such as

GAAGTCCAG...

2.1.3 Logical Descriptions

Patterns can be represented as a logical description of the form

$$(x_1 = a_1..a_2) \wedge (x_2 = b_1..b_2) \wedge ...$$

where x_1 and x_2 are the attributes of the pattern and a_i and b_i are the values taken by the attribute.

This description actually consists of a conjunction of logical disjunctions. An example of this could be

$$(\text{colour} = \text{red} \vee \text{white}) \wedge (\text{make} = \text{leather}) \wedge (\text{shape} = \text{sphere})$$

to represent a cricket ball.

2.1.4 Fuzzy and Rough Pattern Sets

Fuzziness is used where it is not possible to make precise statements. It is therefore used to model subjective, incomplete and imprecise data. In a fuzzy set, the objects belong to the set depending on a membership value which varies from 0 to 1.

A rough set is a formal approximation of a crisp set in terms of a pair of sets which give the lower and the upper approximation of the original set. The lower and upper approximation sets themselves are crisp sets. The set X is thus represented by a tuple $\{\underline{X}, \overline{X}\}$ which is composed of the lower and upper approximation. The lower approximation of X is the collection of objects which can be classified with full certainty as members of the set X. Similarly, the upper approximation of X is the collection of objects that may possibly be classified as members of the set X.

The features of the fuzzy pattern may be a mixture of linguistic values, fuzzy numbers, intervals and real numbers. Each pattern X will be a vector consisting of linguistic values, fuzzy numbers, intervals and real values. For example, we may have linguistic knowledge such as "If X_1 is small and X_2 is large, then class 3". This would lead to the pattern (small, large) which has the class label 3. Fuzzy patterns can also be used in cases where there are uncertain or missing values. For example, the pattern maybe $X = (?, 6.2, 7)$. The missing value can be represented as an interval which includes its possible values. If the missing value in the above example lies in the interval [0,1], then the pattern can be represented as

$X = ([0, 1], 6.2, 7)$ with no missing values.

The values of the features may be rough values. Such feature vectors are called rough patterns. A rough value consists of an upper and a lower bound. A rough value can be used to effectively represent a range of values of the feature. For example, power may be represented as (230, 5.2, $(\underline{50}, \overline{49, 51})$), where the three features are voltage, current and frequency (represented by a lower and upper bound).

In some cases, hard class labels do not accurately reflect the nature of the available information. It may be that the pattern categories are ill-defined and best represented as fuzzy sets of patterns. Each training vector x_i is assigned a fuzzy label $u_i \in [0, 1]^c$ whose components are the grades of membership of that pattern to each class.

The classes to which the patterns belong may be fuzzy concepts, for example, the classes considered may be short, medium and tall.

2.1.5 Patterns as Trees and Graphs

Trees and graphs are popular data structures for representing patterns and pattern classes. Each node in the tree or graph may represent one or more patterns. The minimum spanning tree (MST), the Delauney tree (DT), the R-tree and the k-d tree are examples of this. The R-tree represents patterns in a tree structure which splits space into hierarchically nested and possibly overlapping minimum bounding rectangles or bounding boxes. Each node of an R-tree has a number of entries. A non-leaf node stores a way of identifying the node and the bounding box of all entries of nodes which are its descendants. Some of the important operations on an R-tree are appropriate updation (insertion, deletion) of the tree to reflect the necessary changes and searching of the tree to locate the nearest neighbours of a given pattern. Insertion and deletion algorithms use the bounding boxes from the nodes to ensure that the nearby elements are placed in the same leaf node. Search entails using the bounding boxes to decide whether or not to search inside a node. In this way most of the nodes in the tree need not be searched.

A set of patterns can be represented as a graph or a tree where following a path in the tree gives one of the patterns in the set. The whole pattern set is represented as a single tree. An example of this is the frequent pattern (FP) tree.

Minimum Spanning Tree

Each pattern is represented as a point in space. These points can be connected to form a tree called the minimum spanning tree (MST). A tree which covers all the nodes in a graph is called a spanning tree. If $d(X, Y)$ is the distance or dissimilarity between nodes X and Y, an MST is a spanning tree where the sum of the distances of the links (edges in the tree) is the minimum. Figure 2.2 shows an example of a minimum spanning tree.

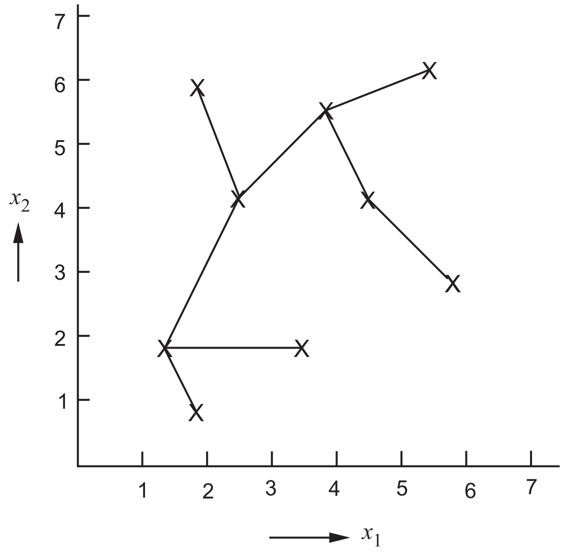

Figure 2.2 Example of a minimum spanning tree

The minimum spanning tree can be used for clustering data points. This is illustrated with the following example.

Example 2

In Figure 2.3, 8 patterns are considered. Figure 2.4 gives the minimum spanning tree for the 8 patterns.

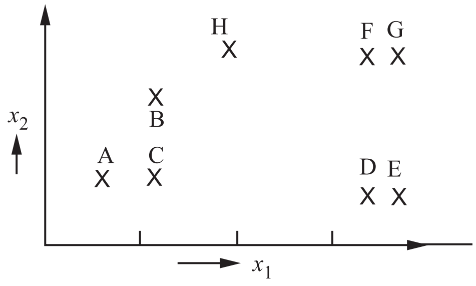

Figure 2.3 Patterns represented in feature space

The minimum spanning tree is used for clustering applications. The largest links in the MST are deleted to obtain the clusters. In Figure 2.4, if the largest link FD is deleted, it results in the formation of two clusters. The first cluster has points A, B, C, H, F and G, and the second cluster has the points D and E. In the first cluster, if the largest link HF is deleted, three clusters are formed. The first cluster has points A, B, C and H; the second cluster has points F and G; and the third cluster has points D and E.

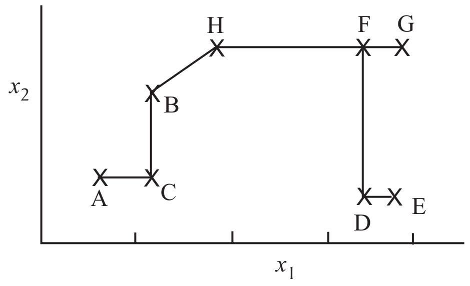

Figure 2.4 The MST for Figure 2.3

Frequent Pattern Trees

This data structure is basically used in transaction databases. The frequent pattern tree (FP tree) is generated from the transactions in the database. It is a compressed tree structure which is useful in finding associations between items in a transaction database. This means that the presence of some items in a transaction will probably imply the presence of other items in the same transaction. The FP growth algorithm used for efficient mining of frequent item sets in large databases uses this data structure.

The first step in constructing this tree is to determine the frequency of every item in the database and sort them from largest to smallest frequencies. Then each entry in the database is ordered so that the order matches the frequency just calculated from largest to smallest. The root of the FP tree is first created and labelled null. The first transaction is taken and the first branch of the FP tree is constructed according to the ordered sequence. The second transaction is then added according to the ordering done. The common prefix shared by this transaction with the previous transaction will follow the existing path, only incrementing the count by one. For the remaining part of the transaction, new nodes are created. This is continued for the whole database. Thus it can be seen that the FP tree is a compressed tree which has information about the frequent items in the original database.

EXAMPLE 3

Consider a 4 × 4 square, where each of the squares is a pixel to represent a digit. The squares are assigned an alphabet as shown in Figure 2.5. For example, digit 4 would be represented in the 4 × 4 square as shown in Figure 2.6 and denoted by a, e, g, i, j, k, l, o. The digits 0, 1, 7, 9 and 4 can be represented as shown in Table 2.1.

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

Figure 2.5 Square representing the pixels of a digit

1			
1		1	
1	1	1	1
		1	

Figure 2.6 Square representing the pixels of digit 4

Table 2.1 A transaction database

Digit	Transaction
0	a, d, e, h, i, l, m, p, b, c, n, o
1	d, h, l, p
7	a, b, c, d, h, l, p
9	a, b, c, d, i, j, k, l, p, e, h
4	a, e, g, i, j, k, l, o

A scan of the transaction database in Table 2.1 will yield the frequency of every item which when sorted from largest to smallest gives (l : 5), (a : 4), (d: 4), (p: 4), (h: 3), (i: 3), (c: 3), (e: 3). Only items which have a frequency of 3 and above are listed here. Note that ties are resolved arbitrarily.

Table 2.2 The transaction database ordered according to frequency of items

Sl.No.	Transaction
0	l, a, d, p, h, i, c, e
1	l, d, p, h
7	l, a, d, p, h
9	l, a, d, p, i, e
4	l, a, i

Table 2.2 shows the transaction database ordered according to the frequency of items. Items which have a support below a threshold are weeded out. In this case, items which have support of two or below are left out. Thus, e, m, b, c, j, k, g, f, n and o are removed. The items retained are l, a, d, p, h and i. Using the ordered database shown in Table 2.2, the FP tree given in Figure 2.7 is constructed.

The root node points to items which are the starting items of the transactions. Here, since all transactions start with l, the root node points to l. For the first transaction, the link is made from the root node to l, from l to a, from a to d, from d to p, from p to h and from h to j. A count of 1 is stored for each of these items. The second transaction is then processed. From the root node, it moves to node l, which already exists. Its count is increased by 1. Since node d is not the next node after l, another link is made from l to d, and from d to p and from p to h. The count for l will be 2, the count for d, p and h will be 1. The next transaction moves from root node to l and from l to a and then to d and then to p and then to h along the path which already exists. The count of the items along this path is increased by 1 so that the count for l will be 3 and the count for a, d, p and h will become 2. Taking the transaction for digit 9, there is a path from root node to l, to p passing through a and d. From p, a new link is made to node i. Now the count for l will be 4 and the count for a, d and p will be 3 and the count for i will be 1. For the last transaction, the path is from root node to the already existing l and a, so that the count of l will become 5 and the count for a will be 4. Then a new link is made from a to i, giving a count of 1 to i. As can be seen from Figure 2.7, the header node for i points to all the nodes with item i. Similarly, the header node for d, p and h also point to all the nodes having these items.

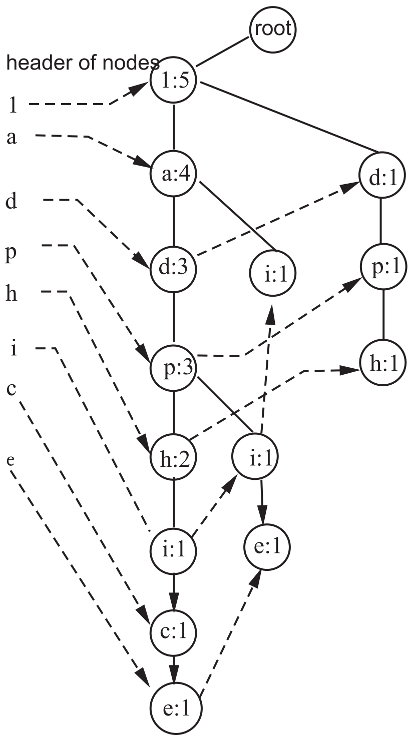

Figure 2.7 FP tree for the transaction database in Table 2.1

2.2 Representation of Clusters

Clustering refers to the process of grouping patterns with similar features together and placing objects with different features in separate groups. There are two data structures here. One is the partition P of the patterns and the other is the set of cluster representatives C.

In problems where the centroid is used as the representative of a group of patterns, P and C depend upon each other. So, if one of them is given, the other can be calculated. If the cluster centroids are given, any pattern can be optimally classified by assigning it to the cluster whose centroid is closest to the pattern. Similarly, if the partition is given, the centroids can be computed. If P is given, then C can be computed in $O(N)$ running time if there are N patterns. If C is given, then P can be generated in $O(NM)$ time if there are M centroids.

Clusters can therefore be represented either by P or C (in the case where the centroid is used as the representative of a group of patterns) or by both P and C.

2.3 Proximity Measures

In order to classify patterns, they need to be compared against each other and against a standard. When a new pattern is present and it is necessary to classify it, the proximity of this pattern to the patterns in the training set is to be found. Even in the case of unsupervised learning, it is required to find some groups in the data so that patterns which are similar are put together. A number of similarity and dissimilarity measures can be used. For example, in the case of the nearest neighbour classifier, a training pattern which is closest to the test pattern is to be found.

2.3.1 Distance Measure

A distance measure is used to find the dissimilarity between pattern representations. Patterns which are more similar should be closer. The distance function could be a metric or a non-metric. A metric is a measure for which the following properties hold :

1. *Positive reflexivity* $d(x,x) = 0$
2. *Symmetry* $d(x,y) = d(y,x)$
3. *Triangular inequality* $d(x,y) \leq d(x,z) + d(z,y)$

The popularly used distance metric called the Minkowski metric is of the form

$$d^m(X,Y) = \left(\sum_{k=1}^{d} \mid x_k - y_k \mid^m \right)^{\frac{1}{m}}$$

When m is 1 it is called the Manhattan distance or the L_1 distance. The most popular is the Euclidean distance or the L_2 distance which occurs when m is assigned the value of 2. We then get

$$d^2(X,Y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ...(x_d - y_d)^2}$$

In this way, L_∞ will be

$$d^\infty(X,Y) = \max_{k=1,..,d} \mid x_k - y_k \mid$$

While using the distance measure, it should be ensured that all the features have the same range of values, failing which attributes with larger ranges will be treated as more important. It will be like giving it a larger weightage. To ensure that all features are in the same range, normalisation of the feature values has to be carried out.

The Mahalanobis distance is also a popular distance measure used in classification. It is given by

$$d(X, Y)^2 = (X - Y)^T \Sigma^{-1} (X - Y)$$

where Σ is the covariance matrix.

Example 4

If $X = (4, 1, 3)$ and $Y = (2, 5, 1)$ then the Euclidean distance will be

$$d(X, Y) = \sqrt{(4-2)^2 + (1-5)^2 + (3-1)^2} = 4.9$$

2.3.2 Weighted Distance Measure

When attributes need to treated as more important, a weight can be added to their values. The weighted distance metric is of the form

$$d(X, Y) = \left(\sum_{k=1}^{d} w_k \times (x_k - y_k)^m \right)^{\frac{1}{m}}$$

where w_k is the weight associated with the kth dimension (or feature).

Example 5

If $X = (4, 2, 3)$, $Y = (2, 5, 1)$ and $w_1 = 0.3$, $w_2 = 0.6$ and $w_3 = 0.1$, then

$$d^2(X, Y) = \sqrt{0.3 \times (4-2)^2 + 0.6 \times (1-5)^2 + 0.1 \times (3-1)^2} = 3.35$$

The weights reflects the importance given to each feature. In this example, the second feature is more important than the first and the third feature is the least important.

It is possible to view the Mahalanobis distance as a weighted Euclidean distance, where the weighting is determined by the range of variability of the sample point expressed by the covariance matrix, where σ_i^2 is the variance in the ith feature direction, $i = 1, 2$. For example, if

$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$$

the Mahalanobis distance gives the Euclidean distance weighted by the inverse of the variance in each dimension.

Another distance metric is the Hausdorff distance. This is used when comparing two shapes as shown in Figure 2.8. In this metric, the points sampled along the shapes boundaries are compared. The Hausdorff distance is the maximum distance between any point in one shape and the point that is closest to it in the other. If there are two point sets (I) and (J), then the Hausdorff distance is

$$\max(\max_{i\in I}\min_{j\in J} \parallel i-j \parallel, \max_{j\in J}\min_{i\in I} \parallel i-j \parallel)$$

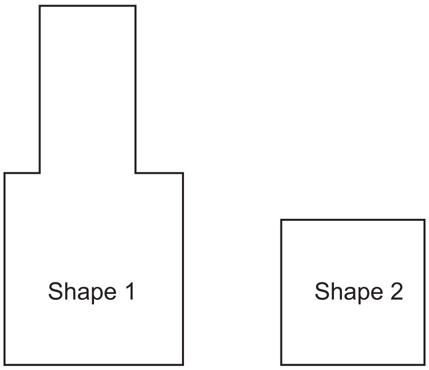

Figure 2.8 Shapes which can be compared by the Hausdorff distance

2.3.3 Non-Metric Similarity Function

Similarity functions which do not obey either the triangular inequality or symmetry come under this category. Usually these similarity functions are useful for images or string data. They are robust to outliers or to extremely noisy data. The squared Euclidean distance is itself an example of a non-metric, but it gives the same ranking as the Euclidean distance which is a metric. One non-metric similarity function is the k-median distance between two vectors. If $X = (x_1, x_2, ..., x_n)$ and $Y = (y_1, y_2, ..., y_n)$, then

$$d(X, Y) = k\text{-median}\{|x_1 - y_1|, ..., |x_n - y_n|\}$$

where the k-median operator returns the kth value of the ordered difference vector.

EXAMPLE 6

If X = (50, 3, 100, 29, 62, 140) and Y = (55, 15, 80, 50, 70, 170), then

Difference vector = {5, 12, 20, 21, 8, 30}

$$d(X,Y) = k\text{-median}\{5, 8, 12, 20, 21, 30\}$$

If $k = 3$, then $d(X,Y) = 12$

Another measure of similarity between two patterns X and Y is

$$S(X,Y) = \frac{X^tY}{||X||\,||Y||}$$

This corresponds to the cosine of the angle between X and Y. $S(X,Y)$ is the similarity between X and Y. If we view $1 - S(X,Y)$ as the distance, $d(X,Y)$, between X and Y, then $d(X,Y)$ does not satisfy the triangular inequality, it is not a metric. However, it is symmetric, because $\cos(\theta) = \cos(-\theta)$.

Example 7

If X, Y, and Z are two vectors in a 2-d space such that the angle between X and Y is 45 and that between Y and Z is 45, then

$$d(X,Z) = 1 - 0 = 1$$

whereas $d(X,Y) + d(Y,Z) = 2 - \sqrt{(2)} = 0.586$

So, triangular inequality is violated.

A non-metric which is non-symmetric is the Kullback–Leibler distance (KL distance). This is the natural distance function from a "true" probability distribution p, to a "target" probability distribution q. For discrete probability distributions if $p = \{p_1, ..., p_n\}$ and $q = \{q_1, ..., q_n\}$, then the KL distance is defined as

$$\mathrm{KL}(p,q) = \Sigma_i p_i \log_2 \left(\frac{p_i}{q_i}\right)$$

For continuous probability densities, the sum is replaced by an integral.

2.3.4 Edit Distance

Edit distance measures the distance between two strings. It is also called the Levenshtein distance. The edit distance between two strings s_1 and s_2 is defined as the minimum number of point mutations required to change s_1 to s_2. A point mutation involves any one of the following operations.

1. Changing a letter
2. Inserting a letter
3. Deleting a letter

The following recurrence relation defines the edit distance between two strings

$$\begin{aligned} d(\text{“ ”}, \text{“ ”}) &= 0 \\ d(s, \text{“ ”}) &= d(\text{“ ”}, s) = \|s\| \end{aligned}$$

$$d(s_1 + ch_1, s_2 + ch_2) = \min(d(s_1, s_2) + \{\text{if } ch_1 = ch_2 \text{ then } 0 \text{ else } 1\},$$
$$d(s_1 + ch_1, s_2) + 1, d(s_1, s_2 + ch_2) + 1)$$

If the last characters of the two strings ch_1 and ch_2 are identical, they can be matched for no penalty and the overall edit distance is $d(s_1, s_2)$. If ch_1 and ch_2 are different, then ch_1 can be changed into ch_2, giving an overall cost of $d(s_1, s_2)$ + 1. Another possibility is to delete ch_1 and edit s_1 into $s_2 + ch_2$, i.e., $d(s_1, s_2 + ch_2)$ + 1. The other possibility is $d(s_1 + ch_1, s_2)$ + 1. The minimum value of these possibilities gives the edit distance.

Example 8

1. If s = “TRAIN” and t = “BRAIN”, then edit distance = 1 because using the recurrence relation defined earlier, this requires a change of just one letter.
2. If s = “TRAIN” and t = “CRANE”, then edit distance = 3. We can write the edit distance from s to t to be the edit distance between “TRAI” and “CRAN” + 1 (since N and E are not the same). It would then be the edit distance between “TRA” and “CRA” + 2 (since I and N are not the same). Proceeding in this way, we get the edit distance to be 3.

2.3.5 Mutual Neighbourhood Distance (MND)

The similarity between two patterns A and B is

$$S(A, B) = f(A, B, \epsilon)$$

where ϵ is the set of neighbouring patterns. ϵ is called the *context* and corresponds to the surrounding points. With respect to each data point, all other data points are numbered from 1 to $N - 1$ in increasing order of some distance measure, such that the nearest neighbour is assigned value 1 and the farthest point is assigned the value $N - 1$. If we denote by NN(u,v), the number of data point v with respect to u, the mutual neighbourhood distance (MND), is defined as

$$\text{MND(u,v)} = \text{NN(u,v)} + \text{NN(v,u)}$$

This is symmetric and by defining NN(u,u)= 0, it is also reflexive. However, the triangle inequality is not satisfied and therefore MND is not a metric.

Example 9

Consider Figure 2.9. In Figure 2.9(a), the ranking of the points A, B and C can be represented as

	1	2
A	B	C
B	A	C
C	B	A

MND(A, B) = 2
MND(B, C) = 3
MND(A, C) = 4

In Figure 2.9(b), the ranking of the points A, B, C, D, E and F can be represented as

	1	2	3	4	5
A	D	E	F	B	C
B	A	C	D	E	F
C	B	A	D	E	F

MND(A, B) = 5
MND(B, C) = 3
MND(A, C) = 7

It can be seen that in the first case, the least MND distance is between A and B, whereas in the second case, it is between B and C. This happens by changing the context.

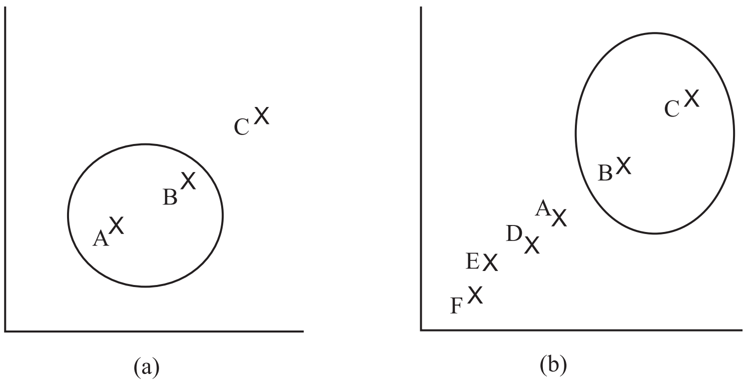

Figure 2.9 Mutual neighbourhood distance

2.3.6 Conceptual Cohesiveness

In this case, distance is applied to pairs of objects based on a set of concepts. A concept is a description of a class of objects sharing some attribute value. For example, (colour = blue) is a concept that represents a collection of blue objects. Conceptual cohesiveness uses the domain knowledge in the form of a set of concepts to characterise the similarity. The conceptual cohesiveness (similarity function) between A and B is characterised by

$S(A, B) = f(A, B, \epsilon, \mathcal{C})$, where $\mathcal{C}$ is a set of pre-defined concepts.

The notion of conceptual distance combines both symbolic and numerical methods. To find the conceptual distance between patterns A and B, A and B are generalised and the similar and dissimilar predicates will give the similarity $S(A, B, G)$ and $D(A, B, G)$. This depends on the generalisation $G(A, B)$. The distance function $f(A, B, G)$ is given by

$$f(A, B, G) = \frac{D(A, B, G)}{S(A, B, G)}$$

The generalisation is not unique—there may be other generalisations. So we can get $S(A, B, G')$ and $D(A, B, G')$ for another generalisation $G'(A, B)$, giving the distance function

$$f(A, B, G') = \frac{D(A, B, G')}{S(A, B, G')}$$

The minimum of these distance functions gives the conceptual distance. The reciprocal of the conceptual distance is called the *conceptual cohesiveness*.

2.3.7 Kernel Functions

The kernel function can be used to characterise the distance between two patterns x and y.

1. *Polynomial kernel* The similarity between x and y can be represented using the polynomial kernel function

   $$K(x, y) = \phi(x)^t \phi(y) = (x^t y + 1)^2$$

   where $\phi(x) = (x_1^2, x_2^2, 1, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2)$

   By using this, linearly dependent vectors in the input space get transformed to independent vectors in kernel space.

2. *Radial basis function kernel* Using the RBF kernel,

$$K(x, y) = \exp^{\frac{-||x-y||^2}{2\sigma^2}}$$

The output of this kernel depends on the Euclidean distance between x and y. Either x or y will be the centre of the radial basis function and σ will determine the area of influence over the data space.

2.4 Size of Patterns

The size of a pattern depends on the attributes being considered. In some cases, the length of the patterns may be a variable. For example, in document retrieval, documents may be of different lengths. This can be handled in different ways.

2.4.1 Normalisation of Data

This process entails normalisation so that all patterns have the same dimensionality. For example, in the case of document retrieval, a fixed number of keywords can be used to represent the document. Normalisation can also be done to give the same importance to every feature in a data set of patterns.

Example 10

Consider a data set of patterns with two features as shown below :

X_1	:	(2, 120)
X_2	:	(8, 533)
X_3	:	(1, 987)
X_4	:	(15, 1121)
X_5	:	(18, 1023)

Here, each line corresponds to a pattern. The first value represents the first feature and the second value represents the second feature. The first feature has values below 18, whereas the second feature is much larger. If these values are used in this way for computing distances, the first feature will be insignificant and will not have any bearing on the classification. Normalisation gives equal importance to every feature. Dividing every value of the first feature by its maximum value, which is 18, and dividing every value of the second feature by its maximum value, which is 1121, will make all the values lie between 0 and 1 as shown below :

$X_1^{'}$	:	(0.11, 0.11)
$X_2^{'}$	:	(0.44, 0.48)

X_3' : (0.06, 0.88)
X_4' : (0.83, 1.0)
X_5' : (1.0, 0.91)

2.4.2 Use of Appropriate Similarity Measures

Similarity measures can deal with unequal lengths. One such similarity measure is the edit distance.

2.5 Abstractions of the Data Set

In supervised learning, a set of training patterns where the class label for each pattern is given, is used for classification. The complete training set may not be used because the processing time may be too long but an abstraction of the training set can be used. For example, when using the nearest neighbour classifier on a test pattern with n training patterns, the effort involved will be $O(n)$. If m patterns are used and $(m << n)$, $O(m)$ effort will be sufficient. Depending on the abstraction, different classifiers are used.

1. *No abstraction of patterns* All the training patterns are used and abstraction of the data is not carried out. Examples of classifiers used here are the nearest neighbour (NN), the k-nearest neighbour (kNN) and the modified kNN (MkNN) classifiers. These methods use the neighbour(s) of the test pattern from all the training patterns.
2. *Single representative per class* All the patterns belonging to a class are represented by a single representative pattern. This single representative can be obtained in many ways. One option is to take the mean of all the patterns in the class. One could also take the medoid of all the samples. Here by medoid, we mean the most centrally located pattern. Other methods of obtaining a single representative, which may be typical of the domain to which the patterns belong, can also be used. A test pattern would be compared to the centroid of the patterns belonging to each class and classified as belonging to the class of the mean closest to it. This is the minimum distance classifier (MDC) .
3. *Multiple representatives per class*

   (a) *Cluster representatives as abstractions* The training patterns in the class are clustered and the cluster centre of each cluster is the representative of all the patterns in the cluster. The set of cluster centres is an abstraction of the whole training set.

(b) *Support vectors as representatives* Support vectors are determined for the class and used to represent the class. Support vector machines (SVMs) are described in Chapter 7.

(c) *Frequent item set based abstraction* In the case of transaction data bases, each pattern represents a transaction. An example of this is the departmental store where each transaction is the set of items bought by one customer. These transactions or patterns are of variable length. Item sets which occur frequently are called frequent item sets. If we have a threshold α, item sets which occur more than α times in the data set are the frequent item sets. Frequent item sets are an abstraction of the transaction database. An important observation is that any discrete-valued pattern may be viewed as a transaction.

EXAMPLE 11

In Figure 2.10, the cluster of points can be represented by its centroid or its medoid. Centroid stands for the sample mean of the points in the cluster. It need not coincide with one of the points in the cluster. The medoid is the most centrally located point in the cluster. Use of the centroid or medoid to represent the cluster is an example of using a single representative per class. It is possible to have more than one representative per cluster. For example, four extreme

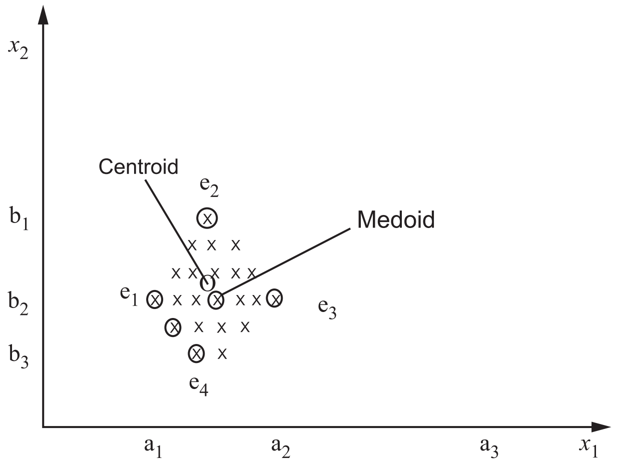

Figure 2.10 A set of data points

points labelled e_1, e_2, e_3, e_4 can represent the cluster as shown in Figure 2.10. When one representative point is used to represent the cluster, in the case of the nearest neighbour classification, only one distance needs to be computed from a test point instead of, say in our example, 22 distances. In the case of there being more than one representative pattern, if four representative patterns are there, only four distances need to be computed instead of 22 distances.

2.6 Feature Extraction

Feature extraction involves detecting and isolating various desired features of patterns. It is the operation of extracting features for identifying or interpreting meaningful information from the data. This is especially relevant in the case of image data where feature extraction involves automatic recognition of various features. Feature extraction is an essential pre-processing step in pattern recognition.

2.6.1 Fisher's Linear Discriminant

Fisher's linear discriminant projects high-dimensional data onto a line and performs classification in this space. If there are two classes, the projection maximises the distance between the means and minimises the variance within each class. Fisher's criterion which is maximised over all linear projections V can be defined as :

$$J(\mathrm{V}) = \frac{|\,\mathrm{mean}_1 - \mathrm{mean}_2|^2}{s_1^2 + s_2^2}$$

where mean_1 and mean_2 represent the mean of Class 1 patterns and Class 2 patterns respectively and s^2 is proportional to the variance. Maximising this criterion yields a closed form solution that involves the inverse of a covariance matrix.

In general, if x_i is a set of N column vectors of dimension D, the mean of the data set is

$$\mathrm{mean} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

In case of multi-dimensional data, the mean is a vector of length D, where D is the dimension of the data.

If there are K classes $\{C_1, C_2, ..., C_K\}$, the mean of class C_k containing N_k members is

$$\text{mean}_k = \frac{1}{N_k} \sum_{x_i \in C_k} x_i$$

The between class scatter matrix is

$$\sigma_B = \sum_{k=1}^{K} N_k(\text{mean}_k - \text{mean})(\text{mean}_k - \text{mean})^T$$

The within class scatter matrix is

$$\sigma_W = \sum_{k=1}^{K} \sum_{x_i \in C_k} (x_i - \text{mean}_k)(x_i - \text{mean}_k)^T$$

The transformation matrix that re-positions the data to be most separable is

$$J(V) = \frac{V^T \sigma_B V}{V^T \sigma_W V}$$

$J(V)$ is the criterion function to be maximised. The vector V that maximises $J(V)$ can be shown to satisfy

$$\sigma_B V = \lambda \sigma_W V$$

Let $\{v_1, v_2, ..., v_D\}$ be the generalised eigenvectors of σ_B and σ_W.

This gives a projection space of dimension D. A projection space of dimension $d < D$ can be defined using the generalised eigenvectors with the largest d eigenvalues to give $V_d = [v_1, v_2, ..., v_d]$.

The projection of vector x_i into a sub-space of dimension d is $y = V_d^T x$. In the case of the two-class problem,

$$\text{mean}_1 = \frac{1}{N_1} \sum_{x_i \in C_1} x_i$$

$$\text{mean}_2 = \frac{1}{N_2} \sum_{x_i \in C_2} x_i$$

$$\sigma_B = N_1(\text{mean}_1 - \text{mean})(\text{mean}_1 - \text{mean})^T + N_2(\text{mean}_2 - \text{mean})(\text{mean}_2 - \text{mean})^T$$

$$\sigma_W = \sum_{x_i \in C_1} (x_i - \text{mean}_1)(x_i - \text{mean}_1)^T + \sum_{x_i \in C_2} (x_i - \text{mean}_2)(x_i - \text{mean}_2)^T$$

$$\sigma_B V = \lambda \sigma_W V$$

This means that

$$\sigma_W^{-1}\sigma_B V = \lambda V$$

Since $\sigma_B V$ is always in the direction of $\text{mean}_1 - \text{mean}_2$, the solution for V is :

$$V = \sigma_W^{-1}(\text{mean}_1 - \text{mean}_2)$$

The intention here is to convert a d-dimensional problem to a one-dimensional one.

EXAMPLE 12

If there are six points namely $(2,2)^t$, $(4,3)^t$ and $(5,1)^t$ belonging to Class 1 and $(1,3)^t$, $(5,5)^t$ and $(3,6)^t$ belonging to Class 2, the means will be

$$m_{x1} = \begin{bmatrix} 3.66 \\ 2 \end{bmatrix}$$

$$m_{x2} = \begin{bmatrix} 3 \\ 4.66 \end{bmatrix}$$

The within class scatter matrix will be

$$\sigma_W = \begin{bmatrix} -1.66 \\ 0 \end{bmatrix} \times \begin{bmatrix} -1.66 & 0 \end{bmatrix} + \begin{bmatrix} 0.33 \\ 1 \end{bmatrix} \times \begin{bmatrix} 0.33 & -1 \end{bmatrix} + \begin{bmatrix} 1.33 \\ -1 \end{bmatrix}$$
$$\times \begin{bmatrix} 1.33 & -1 \end{bmatrix} + \begin{bmatrix} -2 \\ -1.66 \end{bmatrix} \times \begin{bmatrix} -2 & -1.66 \end{bmatrix} + \begin{bmatrix} 2 \\ 0.33 \end{bmatrix}$$
$$\times \begin{bmatrix} 2 & 0.33 \end{bmatrix} + \begin{bmatrix} 0 \\ 1.33 \end{bmatrix} \times \begin{bmatrix} 0 & 1.33 \end{bmatrix} = \begin{bmatrix} 12.63 & 2.98 \\ 2.98 & 6.63 \end{bmatrix}$$

$$S_W^{-1} = \frac{1}{74.88}\begin{bmatrix} 6.63 & -2.98 \\ -2.98 & 12.63 \end{bmatrix}$$

The direction is given by

$$V = \sigma_W^{-1}(\text{mean}_1 - \text{mean}_2) = \frac{1}{74.88}\begin{bmatrix} 6.63 & -2.98 \\ -2.98 & 12.63 \end{bmatrix} \times \begin{bmatrix} 0.66 \\ -2.66 \end{bmatrix}$$

$$V = \frac{1}{74.88}\begin{bmatrix} 12.30 \\ -34.2 \end{bmatrix} = \begin{bmatrix} 0.164 \\ -0.457 \end{bmatrix}$$

Note that $v^t x \geq -0.586$ if $x \in \text{Class } 1$ and $v^t x \leq -1.207$ if $x \in \text{Class } 2$.

2.6.2 Principal Component Analysis (PCA)

PCA involves a mathematical procedure that transforms a number of correlated variables into a smaller number of uncorrelated variables called principal components. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible. PCA finds the most accurate data representation in a lower dimensional space. The data is projected in the direction of maximum variance.

If x is a set of N column vectors of dimension D, the mean of the data set is

$$m_x = E\{x\}$$

The covariance matrix is

$$C_x = E\{(x - m_x)(x - m_x)^T\}$$

The components of C_x denoted by c_{ij} represent the covariances between the random variable components x_i and x_j. The component c_{ii} is the variance of the component x_i.

This is a symmetric matrix from which the orthogonal basis can be calculated by finding its eigenvalues and eigenvectors. The eigenvectors e_i and the corresponding eigenvalues λ_i are solutions of the equation

$$C_x e_i = \lambda_i e_i, i = 1, ..., n$$

By ordering the eigenvectors in the order of descending eigenvalues, an ordered orthogonal basis can be created with the first eigenvector having the direction of the largest variance of the data. In this way, we can find directions in which the data set has the most significant amounts of energy.

If A is the matrix consisting of eigenvectors of the covariance matrix as the row vectors formed by transforming a data vector x, we get

$$y = A(x - m_x)$$

The original data vector x can be reconstructed from y by

$$x = A^T y + m_x$$

Instead of using all the eigenvectors, we can represent the data in terms of only a few basis vectors of the orthogonal basis. If we denote the matrix having the K first eigenvectors as A_K, we get

$$y = A_K(x - m_x)$$

and

$$x = A_K^T y + m_x$$

The original data vector is projected on the coordinate axes having the dimension K and the vector is transformed back by a linear combination of the basis vectors. This minimises the mean-square error with the given number of eigenvectors used. By picking the eigenvectors having the largest eigenvalues, as little information as possible is lost. This provides a way of simplifying the representation by reducing the dimension of the representation.

Example 13

A data set contains four patterns in two dimensions. The patterns belonging to Class 1 are (1, 1) and (1, 2). The patterns belonging to Class 2 are (4, 4) and (5, 4). With these four patterns, the covariance matrix is

$$C_x = \begin{bmatrix} 4.25 & 2.92 \\ 2.92 & 2.25 \end{bmatrix}$$

The eigenvalues of C_x are

$$\lambda = \begin{bmatrix} 6.336 \\ 0.1635 \end{bmatrix}$$

where the two eigenvalues are represented as a column vector.

Since the second eigenvalue is very small compared to the first eigenvalue, the second eigenvector can be left out. The eigenvector which is most significant is

$$e_1 = \begin{bmatrix} 0.814 \\ 0.581 \end{bmatrix}$$

To transform the patterns on to the eigenvector, the pattern (1, 1) gets transformed to

$$\begin{bmatrix} 0.814 & 0.581 \end{bmatrix} \times \begin{bmatrix} -1.75 \\ -1.75 \end{bmatrix} = -2.44$$

Similarly, the patterns (1, 2), (4, 4) and (5, 4) get transformed to –1.86, 1.74 and 2.56.

When we try to get the original data using the transformed data, some information is lost. For pattern (1, 1), using the transformed data, we get

$$\begin{bmatrix} 0.814 & 0.581 \end{bmatrix}^T \times (-2.44) + \begin{bmatrix} 2.75 \\ 2.75 \end{bmatrix} = \begin{bmatrix} -1.99 & -1.418 \end{bmatrix}^T + \begin{bmatrix} 2.75 \\ 2.75 \end{bmatrix}$$

$$= \begin{bmatrix} 0.76 \\ 1.332 \end{bmatrix}$$

2.7 Feature Selection

The features used for classification may not always be meaningful. Removal of some of the features may give a better classification accuracy. Features which are useless for classification are found and left out. Feature selection can also be used to speed up the process of classification, at the same time, ensuring that the classification accuracy is optimal. Feature selection ensures the following:

1. *Reduction in cost of pattern classification and design of the classifier* Dimensionality reduction, i.e., using a limited feature set simplifies both the representation of patterns and the complexity of the classifiers. Consequently, the resulting classifier will be faster and use less memory.
2. *Improvement of classification accuracy* The improvement in classification is due to the following reasons:
   (a) The performance of a classifier depends on the inter-relationship between sample sizes, number of features, and classifier complexity. To obtain good classification accuracy, the number of training samples must increase as the number of features increase. The probability of misclassification does not increase as the number of features increases, as long as the number of training patterns is arbitrarily large. Beyond a certain point, inclusion of additional features leads to high probabilities of error due to the finite number of training samples. If the number of training patterns used to train the classifier is small, adding features may actually degrade the performance of a classifier. This is called the peaking phenomena and is illustrated in Figure 2.11. Thus it can be seen that a small number of features can alleviate the curse of dimensionality when the number of training patterns is limited.
   (b) It is found that under a broad set of conditions, as dimensionality increases, the distance to the nearest data point approaches the distance to the furthest data point. This affects the results of the nearest neighbour problem. Reducing the dimensionality is meaningful in such cases to get better results.

All feature selection algorithms basically involve searching through different feature sub-sets. Since feature selection algorithms are basically search procedures, if the number of features is large (or even above, say, 30 features), the number of feature sub-sets become prohibitively large. For optimal algorithms such as the exhaustive search and branch and bound technique , the computational efficiency comes down rather steeply and it is necessary to use sub-optimal procedures which are essentially faster.

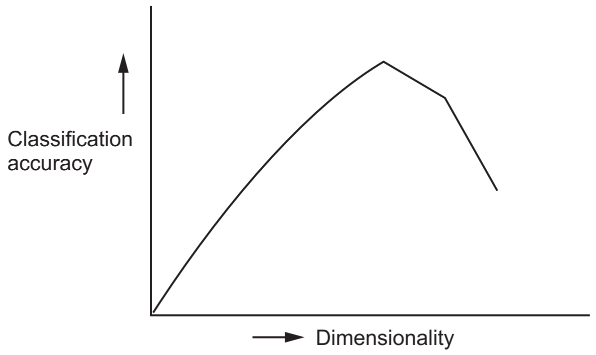

Figure 2.11 Peaking phenomenon or the curse of dimensionality

Feature sub-sets that are newly discovered have to be evaluated by using a criterion function. For a feature sub-set X, we have to find $J(X)$. The criterion function usually used is the classification error on the training set. Here $J = (1 - P_e)$, where $P(e)$ is the probability of classification error. This suggests that a higher value of J gives a better feature sub-set.

2.7.1 Exhaustive Search

The most straightforward approach to the problem of feature selection is to search through all the feature sub-sets and find the best sub-set. If the patterns consist of d features, and a sub-set of size m features is to be found with the smallest classification error, it entails searching all $\binom{d}{m}$ possible sub-sets of size m and selecting the sub-set with the highest criterion function $J(.)$, where $J = (1 - P_e)$. Table 2.3 shows the various sub-sets for a data set with 5 features. This gives sub-sets of three features. A 0 means that the corresponding feature is left out and a 1 means that the feature is included.

Table 2.3 Features selected in exhaustive search

Sl. No.	f1	f2	f3	f4	f5
1.	0	0	1	1	1
2.	0	1	0	1	1
3.	0	1	1	0	1
4.	0	1	1	1	0
5.	1	0	0	1	1
6.	1	0	1	0	1
7.	1	0	1	1	0
8.	1	1	0	0	1
9.	1	1	0	1	0
10.	1	1	1	0	0

This technique is impractical to use even for moderate values of d and m. Even when d is 24 and m is 12, approximately 2.7 million feature sub-sets must be evaluated.

2.7.2 Branch and Bound Search

The branch and bound scheme avoids exhaustive search by using intermediate results for obtaining bounds on the final criterion value. This search method assumes monotonicity as described below.

Let $(Z_1, Z_2, ..., Z_l)$ be the $l = d - m$ features to be discarded to obtain an m feature sub-set. Each variable Z_i can take on values in $\{1, 2, ..., d\}$. The order of the Z_is is immaterial and they are distinct, so we consider only sequences of Z_i such that $Z_1 < Z_2 < ... < Z_l$. The feature selection criterion is $J_l(Z_1, \ldots, Z_l)$. The feature sub-set selection problem is to find the optimum sub-set $Z_1^*, \ldots, Z_l^*$ such that

$$J_l(Z_1^*, ..., Z_l^*) = \max J_l(Z_1, ..., Z_l)$$

If the criterion J satisfies monotonicity, then

$$J_1(Z_1) \geq J_2(Z_1, Z_2) \geq ... \geq J_l(Z_1, ..., Z_l)$$

This means that a sub-set of features should not be better than any larger set that contains the sub-set.

Let B be a lower bound on the optimum value of the criterion $J_l(Z_1^*, ..., Z_l^*)$. That is

$$B \leq J_l(Z_1^*, ..., Z_l^*)$$

If $J_k(Z_1, ..., Z_k)(k \leq l)$ were less than B, then

$$J_l(Z_1, ..., Z_k, Z_{k+1}, ..., Z_l) \leq B$$

for all possible $Z_{k+1}, ..., Z_l$.

This means that whenever the criterion evaluated for any node is less than the bound B, all nodes that are successors of that node also have criterion values less than B, and therefore cannot lead to the optimum solution. This principle is used in the branch and bound algorithm. The branch and bound algorithm successively generates portions of the solution tree and computes the criterion. Whenever a sub-optimal partial sequence or node is found to have a criterion lower than the bound at that point in time, the sub-tree under that node is implicitly rejected and other partial sequences are explored.

Starting from the root of the tree, the successors of the current node are enumerated in the ordered list LIST(i). The successor for which the partial criterion $J_i(Z_1, ..., Z_i)$ is maximum is picked as the new current node and the algorithm moves on to the next higher level. The lists LIST(i) at each level i keeps track of the nodes that have been explored. The SUCCESSOR variables determine the number of successor nodes the current node will have at the next level. AVAIL keeps track of the feature values that can be enumerated at any level. Whenever the partial criterion is found to be less than the bound, the algorithm backtracks to the previous level and selects a hitherto unexplored node for expansion. When the algorithm reaches the last level l, the lower bound B is updated to be the current value of $J_l(Z_1, ..., Z_l)$ and the current sequence $(Z_1, ..., Z_l)$ is saved as $(Z_1^*, ..., Z_l^*)$. When all the nodes in LIST(i) for a given i are exhausted, the algorithm backtracks to the previous level. When the algorithm backtracks to level 0, it terminates. At the conclusion of the algorithm, $(Z_1^*, ..., Z_l^*)$ will give the complement of the best feature set.

The branch and bound algorithm is as follows:

Step 1: Take root node as the current node.

Step 2: Find successors of the current node.

Step 3: If successors exist, select the successor i not already searched which has the maximum J_i and make it the current node.

Step 4: If it is the last level, update the bound B to the current value of $J_l(Z_1, ..., Z_l)$ and store the current path $(Z_1, ..., Z_l)$ as the best path $(Z_1^*, ..., Z_l^*)$. Backtrack to previous levels to find the current node.

Step 5: If the current node is not the root, go to Step 2.

Step 6: The complement of the best path at this point $(Z_1^*, ..., Z_l^*)$ gives the best feature set.

This algorithm assumes monotonicity of the criterion function $J(.)$. This means that for two feature sub-sets χ_1 and χ_2, where $\chi_1 \subset \chi_2$, $J(\chi_1) < J(\chi_2)$. This is not always true.

The modified branch and bound algorithm (BAB$^+$) gives an algorithmic improvement on the BAB. Whenever the criterion evaluated for any node is not larger than the bound B, all nodes that are its successors also have criterion values not larger than B and therefore cannot be the optimal solution. BAB$^+$ does not generate these nodes and replaces the current bound with the criterion value which is larger than it and is held by the terminal node in the search procedure. The bound reflecting the criterion value held by the globally optimal solution node will never be replaced. This algorithm implicitly skips over the intermediate nodes and "short-traverses", thus improving the efficiency of the algorithm.

The relaxed branch and bound (RBAB) can be used even if there is a violation of monotonicity. Here we have a margin b and even if J exceeds the bound by an amount less than b, the search is continued. b is called the margin. On the other hand, the modified relaxed branch and bound algorithm gets rid of the margin and cuts branches below a node Z only if both Z and a parent of Z have a criterion value less than the bound.

Example 14

Consider a data set having four features f1, f2, f3 and f4. Figure 2.12 shows the tree that results when one feature at a time is removed. The numbering of the nodes shows the order in which the tree is formed. When a leaf node is reached, the criterion value of the node is evaluated. When node 3 is reached, since its criterion value is 25, the bound is set as 25. Since the bound of the next node 4 is 45 which is greater than 25, the bound is revised to 45. Node 5 has a smaller criterion than the bound and therefore the bound remains as 45. When node 6 is evaluated, it is found to have a criterion of 37. Since this is smaller than the bound, this node is not expanded further. Since monotonicity is assumed, nodes 7 and 8 are estimated to have criteria which are smaller than 37. Similarly node 9 which has a criterion of 43 is also not expanded further since its criterion is less than the bound. The features removed would therefore be f1 and f3. This is shown in Figure 2.13. Using relaxed branch and bound, if the margin b is 5, considering node 9, since the difference between its criterion 43 and the bound which is 45 is less than the margin, node 9 is further expanded to give node 10. If monotonicity is violated and node 10 has a criterion greater than 45, it will be chosen as the one with the best criterion which makes the two features to be removed as f3 and f4. This is shown in Figure 2.13 where the criterion value is 47.

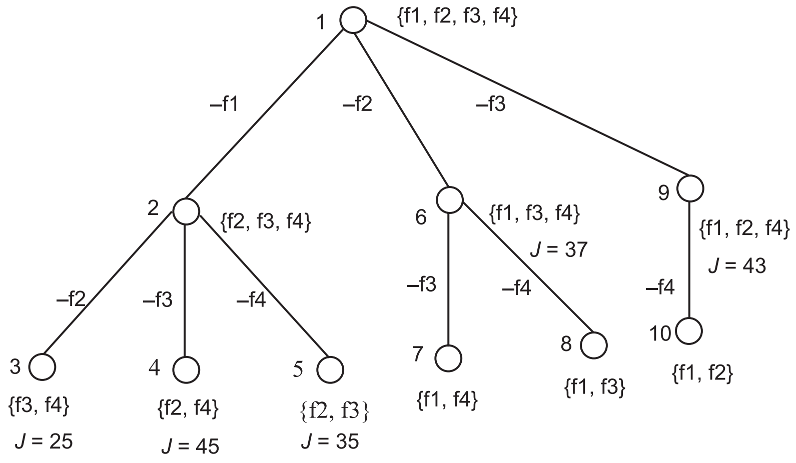

Figure 2.12 The solution tree for d = 4 and m = 2

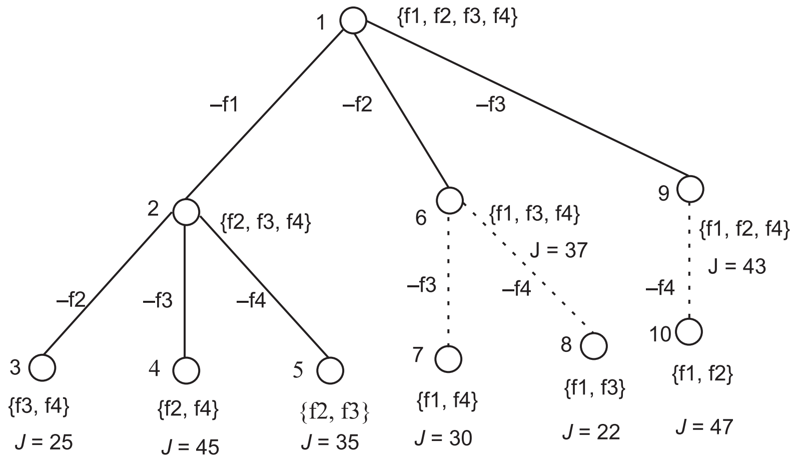

Figure 2.13 Using branch and bound algorithm

2.7.3 Selection of Best Individual Features

This is a very simple method where only the best features are selected. All the individual features are evaluated independently. The best m features are then selected. This method, though computationally simple, is very likely to fail since the dependencies between features also have to be taken into account.

2.7.4 Sequential Selection

These methods operate either by evaluating growing feature sets or shrinking feature sets. They either start with the empty set and add features one after the other; or they start with the full set and delete features. Depending on whether we start with an empty set or a full one, we have the sequential forward selection (SFS) and the sequential backward selection (SBS) respectively. Since these methods do not examine all possible sub-sets, the algorithms do not guarantee the optimal result (unlike the branch and bound and the exhaustive search techniques). Besides, these methods suffer from the "nesting effect" explained below.

Sequential Forward Selection (SFS)

The sequential method adds one feature at a time, each time checking on the performance. It is also called the "bottom-up approach" since the search starts with an empty set and successively builds up the feature sub-set. The method suffers from the "nesting effect" since features once selected cannot be later discarded. The number of feature sub-sets searched will be

$$\sum_{i=1}^{m}(d-i+1) = m[d - \frac{(m-1)}{2}]$$

The features are selected according to their significance. The most significant feature is selected to be added to the feature sub-set at every stage. The most significant feature is the feature that gives the highest increase in criterion function value as compared to that of the others before adding the feature. If U_0 is the feature added to the set X_k consisting of k features then the significance of this is

$$S_{k+0}(U_0) = J(X_k \bigcup U_0) - J(X_k)$$

The most significant feature with respect to the set X_k is

$$S_{k+0}(U_0^r) = \max_{1 \le i \le \phi} S_{k+0}(U_0^i)$$

i.e., $$J(X_k \bigcup U_0^r) = \max_{1 \le i \le \phi} J(X_k \bigcup U_0^i)$$

where ϕ is the number of all the possible 0-tuples.

The least significant feature with respect to set X_k is

$$S_{k+0}(U_0^r) = \min_{1 \le i \le \phi} S_{k+0}(U_0^i)$$

i.e., $$J(X_k \bigcup U_0^r) = \min_{1 \le i \le \phi} J(X_k \bigcup U_0^i)$$

Sequential Backward Selection (SBS)

This method first uses all the features and then deletes one feature at a time. It is also called the "top-down" approach, since the search starts with the complete set of features and successively discards features. The disadvantage is that features, once discarded, cannot be re-selected. Features are discarded successively by finding the least significant feature at that stage.

2.7.5 Sequential Floating Search

To take care of the "nesting effect" of SFS and SBS, "plus l take away r" selection was introduced, where the feature sub-set is first enhanced by l features using forward selection and then r features are deleted using backward selection. The main drawback of this method is that there is no theoretical way of predicting the values of l and r to achieve the best feature sub-set. The floating search method is an improvement on this, since there is no need to specify the parameters l and r. The number of forward and backward steps is determined dynamically while the method is running so as to maximise the criterion function. At each step, only a single feature is added or removed. The values of l and r is kept "floating", i.e., they are kept flexible so as to approximate the optimal solution as much as possible. Consequently, the dimensionality of the feature sub-set does not change monotonically but is actually "floating" up and down.

Sequential Floating Forward Search (SFFS)

The principle of SFFS is as follows:

STEP 1: Let $k = 0$.

STEP 2: Add the most significant feature to the current sub-set of size k. Let $k = k+1$.

STEP 3: Conditionally remove the least significant feature from the current sub-set.

STEP 4: If the current sub-set is the best sub-set of size $k-1$ found so far, let $k = k-1$ and go to Step 3. Else return the conditionally removed feature and go to Step 2.

EXAMPLE 15

Consider a data set of patterns with four features f1, f2, f3 and f4. The steps of the algorithm are as given below.

STEP 1: Let the sub-set of the features chosen be empty, i.e., $F = \phi$.

STEP 2: The most significant feature is found. Let it be f3. Then F = {f3}.

STEP 3: The least significant feature is found, i.e., it is seen if f3 can be removed.

STEP 4: The removal of f3 does not improve the criterion value and f3 is restored.

STEP 5: The most significant feature is found. Let it be f2. Then F = {f3, f2}

STEP 6: The least significant feature is found. It is f2. Condition fails.

STEP 7: The most significant feature is found. Let it be f1. Then F = {f3, f2, f1}

STEP 8: The least significant feature is found. Let it be f3. If it gives a better sub-set, then F = {f1, f2}.

The optimal set would be F = {f1, f2} if $m = 2$. It is noted that if in the last step, the least significant feature was found to be f1, F = {f2, f3} as in Step 5. This leads to looping.

Sequential Floating Backward Search (SBFS)

This method is similar to the SFFS except that backward search is carried out first and then the forward search. The principle of SBFS is as follows:

Step 1: $k = 0$.

Step 2: Remove the least significant feature from the current sub-set of size k. Let $k = k - 1$.

Step 3: Conditionally add the most significant feature from the features not in the current sub-set.

Step 4: If the current sub-set is the best sub-set of size $k - 1$ found so far, let $k = k + 1$ and go to Step 3. Else remove the conditionally added feature and go to Step 2.

The algorithm starts with all the features and then removes them one by one. Due to this, the method does not always give good results. If the dimensionality is large, i.e., d is large and if m is very small, it will be time-consuming and it is better to use the SFFS.

2.7.6 Max–Min Approach to Feature Selection

This method has a computational advantage over the other well-known methods due to the fact that instead of computationally time-consuming calculations in a multi-dimensional space, the max–min method requires calculations in two-dimensional space only. However, the results achieved by this method are rather unsatisfactory. It works as follows.

Let us denote

f_i = feature from the selected feature set $F_k = \{f_1, ..., f_k\}$ acquired in the ith step of the selection procedure.
g_j = jth feature from the set of features not selected.
$\delta J(y_j, x_i)$ = the absolute value of the difference between $J(g_j, f_i)$ and $J(f_i)$.

In the max–min method, the new feature y_j is chosen as the next feature if it yields

$$\text{max}_{\forall y_j} \text{min}_{\forall x_i} \Delta J(y_j, x_i)$$

The poor results of this method confirm that it is not possible to select a set of features in a high-dimensional space based on two-dimensional information without a substantial information loss.

EXAMPLE 16

Consider a data set with patterns having four features f1, f2, f3 and f4. The criterion function value using up to two features at a time is shown in Table 2.4.

Table 2.4 Criterion function using a sub-set of features

Feature	f1	f2	f3	f4
f1	10	30	35	55
f2	30	20	40	53
f3	35	40	30	42
f4	55	53	42	40

If J(f1) represents the criterion function using only feature f1, then

$$J\text{ (f1)} = 10;\ J\text{ (f2)} = 20;\ J\text{ (f3)} = 30;\ J\text{ (f4)} = 40$$

This is depicted by the diagonal entries in Table 2.4.

First considering f1, if J (f1, f2) = 30, J(f1, f3) = 35 and J (f1, f4) = 55, then

$$\Delta\ J\text{ (f1, f2)} = 10;\ \Delta\ J\text{ (f1, f3)} = 5;\ \Delta\ J\text{ (f1, f4)} = 15$$

The minimum of these is $\Delta\ J$ (f1, f3) = 5

Next considering f2, if J (f2, f1)=30, J (f2, f3)= 40 and J (f2,f4) = 53, then

$$\Delta\ J\text{ (f2, f1)} = 20;\ \Delta\ J\text{ (f2, f3)} = 10;\ \Delta\ J\text{ (f2, f4)} = 13$$

The minimum of these is $\Delta\ J$ (f2, f3) = 10

Next considering f3, if J (f3, f1) = 35, J (f3, f2) = 40 and J (f3, f4) = 42, then

$$\Delta\ J\text{ (f3, f1)} = 25;\ \Delta\ J\text{ (f3, f2)} = 20;\ \Delta\ J\text{ (f3, f4)} = 2$$

The minimum of these is $\Delta\ J$ (f3, f4) = 2

Next considering f4, if J (f4, f1) = 55, J (f4, f2) = 53 and J (f4, f3) = 42, then

$$\Delta\ J\text{ (f4, f1)} = 45;\ \Delta\ J\text{ (f4, f2)} = 33;\ \Delta\ J\text{ (f4, f3)} = 12$$

The minimum of these is $\Delta\ J$ (f4, f3) = 12

Finding the maximum of the four minimum values, we get f4 which is chosen as the next feature.

It can be seen that only two features are considered at a time. The features chosen could have been different if more number of features are considered at a time.

2.7.7 Stochastic Search Techniques

Genetic algorithm is a stochastic search technique which is often used for feature selection. The population in the GA consists of strings which are binary in nature. Each string (or chromosome) is of length d, with each position i being zero or one depending on the absence or presence of feature i in the set. This means that each feature sub-set is coded as a d-element bit string or binary valued vector. Each string in the population is a feature selection vector α, where each $\alpha = \alpha_1, ..., \alpha_d$ and α_i assumes a value 0 if the ith feature is excluded and 1 if it is present in the sub-set. To compute its fitness, each chromosome is evaluated by determining its performance on the training set.

2.7.8 Artificial Neural Networks

A multilayer feed-forward network with a back-propagation learning algorithm is used in this method. The approach considered here is to take a larger than necessary network and then remove unnecessary nodes. Pruning is carried out by eliminating the least salient nodes. It is based on the idea of iteratively eliminating units and adjusting the remaining weights in such a way that the network performance does not become worse over the entire training set. The pruning problem is formulated in terms of solving a system of linear equations using the optimisation technique.

The pruning of nodes corresponds to removing the corresponding features from the feature set. The saliency of a node is defined as the sum of the increase in error over all the training patterns caused by the removal of that node. The node pruning based feature selection first trains a network and then removes the least salient node. The reduced network is trained again, followed by the removal of the least salient node. This procedure is repeated to get the least classification error.

2.8 Evaluation of Classifiers

Before using a classifier to carry out a classification task, it is necessary to evaluate its performance. The various parameters of the classifier which requires to be taken into account are

1. *Accuracy of the classifier* The main aim of using a classifier is to correctly classify unknown patterns. Classification accuracy is an important parameter in evaluating a classifier.

2. *Design time and classification time* Design time is the time taken to build the classifier from the training data while classification time is the time taken to classify a pattern using the designed classifier. The nearest neighbour classifier does not require any design time. However, the classification time will be high since each test pattern has to be compared to all the patterns in the training set. On the other hand, the neural network classifier requires a high design time to train the weights in the network. But, the classification time will be low as it is only necessary to run the pattern through the trained network to get the class of the pattern.
3. *Space required* If an abstraction of the training set is carried out, the space required will be less. If no abstraction is carried out and the entire training data is required for classification, the space requirement is high. The nearest neighbour classifier requires the entire training data to be stored and therefore, the space required will be more. The neural network, decision tree and the rule-based classifier requires only the abstraction of the training set and therefore requires less space. For instance, the neural network classifier requires only the trained neural network to carry out the classification task. The training data set is not required.
4. *Explanation ability* If the reason for the classifier in choosing the class of a pattern is clear to the user, then its explanation ability is good. For instance, in the decision tree classifier, following the path from the root of the tree to the leaf node for the values of the features in the pattern will give the class of the pattern. Similarly, the user understands why a rule based system chooses a particular class for a pattern. On the other hand, the neural network classifier has a trained neural network and it is not clear to the user what the network is doing.
5. *Noise tolerance* This refers to the ability of the classifier to take care of outliers and patterns wrongly classified.

The accuracy of the classifier is the parameter usually taken into account. However, if the classification time is too large and it is required to carry out the classification task quickly, it may be better to sometimes sacrifice the accuracy and use a classifier which is faster. In places where there is a space limitation like in hand-held devices, it is good to use a classifier requiring little space. In crucial applications like medical diagnostics, explanation ability may be necessary in a classifier so that the doctors are sure that the classifier is doing the right thing.

To estimate how good a classifier is, an estimate can be made using the training set itself. This is known as re-substitution estimate. It assumes that the training data is a good representative of the data. Sometimes, a part of the training data is used as a measure of the performance of the classifier. Usually the training set is divided into smaller sub-sets. One of the sub-sets is used for training while the other is used for validation. The different methods of validation are as follows:

1. *Holdout method* The training set is divided into two sub-sets. Typically two-thirds of the data is used for training and one-thirds is used for validation. It could also be one-half for training and one-half for testing or any other proportion.

2. *Random sub-sampling* In this method, the holdout method is repeated a number of times with different training and validation data each time. If the process is repeated k times, the overall accuracy will be

   $$\mathrm{acc}_{\mathrm{overall}} = \frac{1}{k}\Sigma_{i=1}^{k}\mathrm{acc}_i$$

3. *Cross-validation* In cross-validation, each pattern is used the same number of times for training and exactly once for testing. An example of cross-validation is the two-fold cross-validation. Here the data set is divided into two equal sub-sets. First, one set is used for training and the other for testing. Then the roles of the sub-sets are swapped—the set for training becomes the set for validation and vice versa.

   In k-fold cross-validation, the data is divided into k equal sub-sets. During each run, one sub-set is used for testing and the rest of the sub-sets are used for training. This procedure is carried out k times so that each sub-set is used for testing once. A special case of the k-fold cross-validation is when $k = n$, where n is the number of patterns in the data set. This is called the *leave-one-out* approach, where each test set contains only one pattern. The procedure is repeated n times.

4. *Bootstrap procedure* Here a pattern is chosen randomly from the data set, while not deleting it from the data set. Another pattern is then randomly chosen. This is repeated till we have N patterns. These are used for training and the patterns not chosen are used for testing.

2.9 Evaluation of Clustering

It is also necessary to evaluate the clustering carried out by any method. Usually, if the dimensions are low in number, a manual inspection of the clustering graph gives a good idea if the clustering is okay. The quality of the clustering can be measured by examining the similarity between patterns. Patterns belonging to the same cluster should have high similarity and patterns belonging to different clusters should have low similarity.

Discussion

The representation of patterns is very important as a good representation makes use of the discriminating attributes of the objects. Use of good representation reduces the computation burden in pattern classification. There are a variety of data structures which can be used for representing patterns. The method chosen would depend on the problem. Feature selection is the process of selecting a sub-set of the features which is found to be more relevant and weeding out the features which really do not contribute much to the classification process. All feature selection methods depend on trying out different sub-sets of features and finding the best sub-set. Before deciding on the classifier to be used for a particular task, it is necessary to evaluate the classifier. The criteria to be used for evaluation of classifiers and how to go about evaluating classifiers has been discussed.

Further Reading

Huttenlocher (1993) describes the Hausdorff distance which is used as a similarity measure for patterns with images. The frequent pattern (FP) tree used to represent patterns in a compact way is given by Han et al. (2004).

A number of papers explain the different methods of feature selection. Narendra and Fukunaga (1977) explain the branch and bound algorithm for feature selection. An improvement on this algorithm is given Yu and Yuan (1993). Floating search methods for feature selection are described by Pudil et al. (1994) and Somol et al. (1999). Kuncheva and Jain (1999) and Siedlecki and Sklansky (1989) show how genetic algorithms can be used for feature selection. Kuncheva and Jain (1999) show how we can combine the process of feature selection and prototype selection.

Exercises

1. Find the centroid and medoid (most centrally located pattern) for the following set of patterns:

   (1, 1), (1, 3), (1, 4), (2, 2), (2, 3), (3, 1), (3, 4), (4, 2).

   Under what conditions will the medoid and the centroid be identical?
2. Under what conditions will a set of points give positive minimum variance value? Why is the centroid a good representative of a set of points?
3. Find the edit distance between the two strings "HOUSE" and "MOUND".
4. Show that the squared Euclidean distance is not a metric.
5. For non-binary data, show that $L_1 > L_2 > L_\infty$.
6. The similarity function $D(X, Y)$ between X and Y as given below is a non-metric.

$S(X,Y) = \frac{X^tY}{||X||\,||Y||}$

$D(X,Y) = 1 - S(X,Y)$

Show how this can be converted into a metric.

7. The Hausdorff distance between two points I and J is given by

   $\max(\max_{i \in I} \min_{j \in J} \parallel i - j \parallel, \max_{j \in J} \min_{i \in I} \parallel i - j \parallel)$

   This distance is not a metric. Which property is violated here, making it a non-metric?

8. Find the mutual neighbourhood distance (MND) between D and E given the following set of points:

   A = (1, 1); B = (2, 2); C = (2, 1); D = (3, 1);

   E = (4, 4); F = (5, 4); G = (6, 4); H = (6, 5)

9. Give the conditions under which the FP tree built on N d-dimensional binary patterns has the minimum number of nodes where each pattern has at most l ($< d$) 1s.

10. A data set consists of the following patterns:

    (1, 1, 1), (2, 2, 1), (1.5, 0.5, 1), (1, 3, 1), (4, 4, 2), (5, 5, 2), (4, 5, 2), (4, 6, 2)

    where each pattern consists of the x-coordinate, y-coordinate and the class. Find the direction of the W vector associated with Fisher's linear discriminant.

11. Given n d-dimensional patterns from two classes and $n < d$, comment on whether S_B and S_W used in Fisher's linear discriminant are singular or not.

12. Find the direction of W using Fisher's linear discriminant when $m_1 = m_2$ and $s_1 = s_2 = 0.5I$, where I is the identity matrix.

13. Given below is a set of patterns with three features X, Y and Z.

X	Y	Z	Output
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

Are any of the features redundant? Explain.

14. If there are ten features and it is necessary to reduce the number of features to six so that the best set of six features is chosen, what is the number of feature sub-sets to be evaluated to find the optimal set of six features in the exhaustive search?

Computer Exercises

1. Write a program to generate the FP tree of a set of patterns. Use the program to generate the FP tree for the data given in Table 2.1.
2. Extend the program of Computer Exercise 1 so as to get the nearest neighbour of any transaction using the FP tree and use it on the data given in Table 2.1.
3. Write a program to obtain the Fisher's linear discriminant for a training data set. Obtain the Fisher's linear discriminant for the two-dimensional data set :

   (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.
4. Write a program to implement the principal component analysis and use it on the data given in Computer Exercise 3.
5. Implement the branch and bound search for feature selection. Apply it to a data set having a number of features and find the set of features found by the program.
6. Implement the sequential floating forward search and apply it to a data set used for Computer Exercise 5. Compare the features found with that found using the branch and bound algorithm.
7. Implement the sequential floating backward search and apply it to the data set used for Computer Exercise 5. Compare the feature set found with that found using SFFS and branch and bound algorithm. Which algorithm gives the best set of features?

Bibliography

1. Han, Jaiwei, Jian Pei, Yiwen Yin and Runying Mao. Mining frequent patterns without candidate generation: A frequent pattern tree approach. *Data Mining and Knowledge Discovery* 8(1): 53–87. 2004.

2. Huttenlocher, D. P., G. A. Klanderman and W. A. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 15(9): 850–863. 1993.

3. Jain, A. K. and D. Zongker. Feature selection: Evaluation, application and small sample performance. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 19:153–157. 1997.

4. Kuncheva, L. and L. C. Jain. Nearest neighbor classifier: Simultaneous editing and feature selection. *Pattern Recognition Letters* 20:1149–1156. 1999.

5. Narendra, P. M. and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Trans. Computers* 26(9): 917–922. 1977.

6. Pudil, P., J. Novovicova and J. Kittler. Floating search methods in feature selection. *Pattern Recognition Letters* 15: 1119–1125. 1994.

7. Siedlecki, W. and J. Sklansky. A note on genetic algorithms for large-scale feature selection. *Pattern Recognition Letters* 10: 335–347. 1989.

8. Somol, P., P. Pudil, J. Novovicova, P. Paclik. Adaptive floating search methods in feature selection. *Pattern Recognition Letters* 20: 1157–1163. 1999.

9. Yu, Bin and Baozong Yuan. A more efficient branch and bound algorithm for feature selection. *Pattern Recognition* 26(6): 883–889. 1993.