CHAPTER 2

Nearest Neighbor-Based Models

Learning Objectives

At the end of this chapter, you will be able to:

- Explain the proximity measures of classification
- Give a brief overview of distance measures
- Describe the ML models that use the nearest neighbor concept

2.1 Introduction to Proximity Measures ✓

Proximity measures are used to quantify the degree of similarity or dissimilarity between two or more pattern vectors. These pattern vectors can represent documents, images or even entire audio or video files. Proximity measures are often used by machine learning (ML) algorithms to compare and classify or group or make predictions using these patterns. There are many types of proximity measures and the popular ones are:

- ✓ **Euclidean distance:** This the most popular distance as it is intuitively appealing. It measures the straight-line distance between two points in a multi-dimensional space. So, it is also called *as the crow flies* distance.
- **Cosine similarity:** This measures the cosine of the angle between two vectors and is often used in text analysis to compute similarity between a pair of documents.
- **Jaccard similarity:** This measures the ratio of the cardinalities of intersection over union of two sets and is often used in recommendation systems to compare users' preferences.
- **Hamming distance:** This measures the number of positions at which two binary strings differ and is often used in error correction codes.

In the following section, we will discuss various distance measures.

2.2 Distance Measures ✓

A distance measure is used to find the dissimilarity between patterns represented as vectors. Patterns which are more similar should be closer. The distance function (d) could be a metric or a non-metric. The most popularly used family of distance metrics is called the **Minkowski metric**.

A **metric** is a type of measure that possesses three key attributes: positive reflexivity, symmetry and triangular inequality:

- *Positive reflexivity:* $d(x, x) = 0$
- *Symmetry:* $d(x, y) = d(y, x)$
- *Triangular inequality:* $d(x, y) \leq d(x, z) + d(z, y)$, where x, y, z are any three patterns.

The following sections describe different types of dissimilarity/similarity (proximity) measures between pattern vectors.

2.2.1 Minkowski Distance

The Minkowski metric is a commonly used family of distance metrics which can be expressed as

$$d^r(p, q) = \left(\sum_{k=1}^{L}(|p_k - q_k|^r)\right)^{\frac{1}{r}},$$

where r is a parameter that determines the type of metric being used and p and q are l-dimensional vectors. Some variations based on selecting the value of r are:

- L_∞ *norm:* Here, $r = \infty$ and $d(p, q) = maximum_k(|p(k) - q(k)|)$, $k \in \{1, \ldots, L\}$.
- L_2 *norm:* In this case, $r = 2$ and $d(p, q) = (\sum_{k=1}^{L}(|p(k) - q(k)|^2))^{\frac{1}{2}}$ is the Euclidean distance; this is the most popular variation.
- L_1 *norm:* In this case, $r = 1$ and $d(p, q) = (\sum_{k=1}^{L}(|p(k) - q(k)|))$ is the city-block distance.
- *Fractional norm*: It is possible that r is a fraction. In such a case, the resulting distance is called fractional norm. It is not a metric as it violates the triangle inequality.

The importance of different norms will be examined while explaining the nearest neighbor classifiers. It is important to ensure that all features used in the distance measure have the same range of values, as attributes with larger ranges may gain undue advantage. **Normalisation** of feature values can help to ensure that they are in the same range.

The **Mahalanobis distance** is another popular distance measure that is used in classification, and it is computed using the covariance matrix. The squared Mahalanobis distance is given by

$$d^2(x, y) = (x - y)^t \Sigma^{-1} (x - y)$$

EXAMPLE 1: If $x = (5, 2, 4)$ and $y = (3, 4, 2)$, the Euclidean distance between them is $d(x, y) = \sqrt{(5-3)^2 + (2-4)^2 + (4-2)^2}$, which equals 3.46.

2.2.2 Weighted Distance Measure

To assign greater importance to certain attributes, a weight can be applied to their values in the weighted distance metric. This metric takes the form

$$d(x, y) = \left(\sum_{k=1}^{L} w_k \times (x(k) - y(k))^r\right)^{\frac{1}{r}},$$

where w_k represents the weight associated with the k^{th} dimension or feature.

EXAMPLE 2: If $x = (5, 2, 4)$ and $y = (3, 4, 2)$, with weights assigned as $w_1 = 0.3$, $w_2 = 0.5$ and $w_3 = 0.2$, then $d(x, y)$ is calculated as $0.3 \times (5-3)^2 + 0.5 \times (2-4)^2 + 0.2 \times (4-2)^2 = 4$.

The weights determine the significance of each feature, with the second feature being more important than the first, and the third feature being the least significant in this example.

2.2.3 Non-Metric Similarity Functions ✓

This category includes similarity functions that do not obey the triangular inequality or symmetry. They are commonly used for image or string data and they are resistant to outliers or extremely noisy data. The squared Euclidean distance is an example of a non-metric, but it provides the same ranking as the Euclidean distance metric.

One example of a non-metric similarity function is the k-median distance between two vectors. Given $x = (x(1), x(2), \ldots, x(l))$ and $y = (y(1), y(2), \ldots, y(l))$, the formula for the k-median distance is

$$d(x, y) = k\text{-median}\{|x(1) - y(1)|, \ldots, |x(n) - y(n)|\},$$

where the k-median operator returns the k^{th} value of the ordered difference vector.

Another similarity measure is

$$S(x, y) = \frac{x^t y}{\|x\|\|y\|},$$

which corresponds to the cosine of the angle between the vectors x and y. It is symmetric because $cos(\theta) = cos(-\theta)$ and it represents the similarity between x and y. A possible distance function corresponding to the cosine similarity, $S(x, y)$ is

$$d(x, y) = 1 - S(x, y)$$

Note that $d(x, y)$ is symmetric as $S(x, y)$ is symmetric. However, $d(x, y)$ violates the triangular inequality. Example 3 demonstrates how it violates the inequality.

EXAMPLE 3: Let x, y and z be three vectors in a two-dimensional space, where the angle between x and z is 45 and the angle between z and y is 45. Here, $d(x, y) = 1 - cos(\pi/2) = 1$, while $d(x, z) + d(z, y) = 1 - cos(\pi/4) + 1 - cos(\pi/4) = 2 - \frac{2}{\sqrt{(2)}} = 0.586$. So, $d(x, z) + d(z, y) < d(x, y)$, violating the triangle inequality.

2.2.4 Levenshtein Distance / edit distance

The Levenshtein distance, also known as edit distance, is a measure of the distance between two strings. It is determined by calculating the minimum number of mutations needed to transform string $s1$ into string $s2$, where a mutation can be one of three operations: changing a letter, inserting a letter or deleting a letter. The edit distance can be defined using the following recurrence relation:

- $d(\text{“ ”}, \text{“ ”}) = 0$, *(two empty strings match)*
- $d(s, \text{“ ”}) = d(\text{“ ”}, s) = \|s\|$, *(distance from an empty string)*
- $d(s1 + ch1, s2 + ch2) = min\ [d(s1, s2) + \{\text{if ch1} = \text{ch2 then 0 else 1}\}, d(s1 + ch1, s2) + 1, d(s1, s2 + ch2) + 1]$

If the last characters of the two strings are identical, they can be matched without penalty, resulting in an edit distance of $d(s1, s2)$. If they are different, $ch1$ can be changed into $ch2$ with an overall cost of $d(s1, s2) + 1$, or $ch1$ can be deleted and $s1$ edited into $s2 + ch2$, resulting in a cost of $d(s1, s2 + ch2) + 1$, or $ch1$ can be inserted into $s2$ resulting in a cost of $d(s1 + ch1, s2) + 1$. The minimum value of these possibilities gives the edit distance. For instance, if we consider the strings "CAT" and "RAT", the edit distance between them is 1 because only one letter needs to be changed. However, if we consider the strings "CAT" and "LAN", the edit distance between them is 2 because multiple changes are required to transform one into the other.

2.2.5 Mutual Neighborhood Distance (MND)

In this case, the function used to measure the similarity between two patterns, x and y, is defined as $S(x, y) = f(x, y, \epsilon)$, where ϵ denotes the context, that is, the surrounding points. In this context, all other data points are labelled in increasing order of some distance measure, starting with the nearest neighbor as 1 and ending with the farthest point as $N - 1$. The label of x with respect to y is denoted by NN (x, y), and the mutual neighborhood distance (MND) is calculated as MND (x, y) = NN (x, y)+ NN (y, x). MND is symmetric, with NN (x, x) set to 0; it is also reflexive. However, it does not satisfy the triangle inequality and is not a metric.

Example 4 explains how the context, the collection of points in the vicinity, changes the MND between points p, q and r.

EXAMPLE 4: Consider Fig. 2.1.

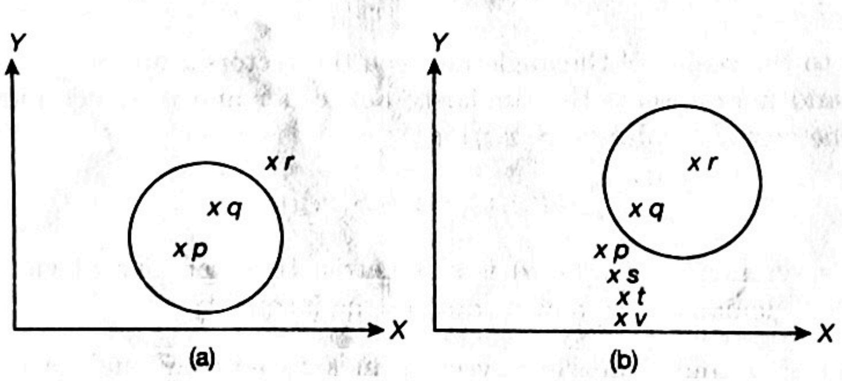

FIG. 2.1 Mutual neighborhood distance illustration

The ranking of the points p, q and r can be represented as shown in Table 2.1 and their mutual nearest neighbor distances are shown in Table 2.2.

TABLE 2.1 Relative positional ranking

	1	2
p	q	r
q	p	r
r	q	p

TABLE 2.2 Mutual distances

MND$(p, q) = 2$
MND$(q, r) = 3$
MND$(p, r) = 4$

Corresponding to Fig. 2.1 (b), the ranking of the points p, q, r, s, t and v can be represented as shown in Table 2.3 and their mutual nearest neighbor distances are shown in Table 2.4. It can be seen that in the first case, the least MND is between p and q, whereas in the second case, it is between q and r. This occurs by changing the context or introducing more points in the vicinity of one of the points.

TABLE 2.3 Relative positional ranking

	1	2	3	4	5
p	s	t	v	q	r
q	p	r	s	t	v
r	q	p	s	t	v

TABLE 2.4 Mutual distances

$\text{MND}(p,q) = 5$
$\text{MND}(q,r) = 3$
$\text{MND}(p,r) = 7$

2.2.6 Proximity Between Binary Patterns

We can view l-dimensional binary patterns as binary strings of length l. Let p and q be two l-bit binary strings. Some of the popular proximity measures on such binary patterns are:

- **Hamming Distance (HD):** If $p(i) = q(i)$, we say that p and q match on their i^{th} bit, else $(p(i) \neq q(i))$, and p and q mismatch on the i^{th} bit. Hamming distance is the number of mismatching bits of the l-bit locations.

  EXAMPLE 5: Consider the two 10-bit patterns p and q given by
  $p = 1000001000$
  $q = 0000001001$
  The Hamming distance is 2 as they mismatch in bit positions 1 and 10.

- **Simple Matching Coefficient (SMC):** Let us define the following:
  M_{01} is the number of bits where p is 0 and q is 1
  M_{10} is the number of bits where p is 1 and q is 0
  M_{00} is the number of bits where p is 0 and q is 0
  M_{11} is the number of bits where p is 1 and q is 1
  Now we define SMC as follows:

$$\text{SMC}(p,q) = \frac{M_{11} + M_{00}}{M_{00} + M_{01} + M_{10} + M_{11}}$$

- **Jaccard Coefficient (JC):** It is defined as

$$\text{JC}(p,q) = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}$$

  EXAMPLE 6: Consider again
  $p = 1000001000$
  $q = 0000001001$
  Here, $M_{11} = 1,\ M_{00} = 7,\ M_{01} = M_{10} = 1$. So,
  $\text{SMC}(p,q) = \frac{1+7}{7+1+1+1} = \frac{8}{10} = 0.8$ and
  $\text{JC}(p,q) = \frac{1}{1+1+1} = \frac{1}{3} = 0.33$

2.3 Different Classification Algorithms Based on the Distance Measures

Many classification algorithms (classifiers) inherently depend upon some distance between a pair of patterns. We discuss them in this section.

2.3.1 Nearest Neighbor Classifier (NNC)

Here, a test pattern x is classified based on its nearest neighbor (NN) in the training data. Specifically, let

$$\mathcal{X} = \{(x_1, l_1), (x_2, l_2), \ldots, (x_n, l_n)\}$$

be the labelled training data set of n patterns, where each element is a tuple; the first component of the tuple is the pattern vector and the second component is its class label.

Let each pattern be a vector in some l-dimensional space. Here, x_i, $i = 1, 2, \ldots, n$ is the i^{th} training pattern and l_i is its class label. So, if there are p classes with their labels coming from the set

$$\mathcal{L} = \{C_1, C_2, \ldots, C_p\},$$

then $l_i \in \mathcal{L}$, for $i = 1, 2, \ldots, n$. Now the nearest neighbor of the test pattern x is given by

$$\text{NN}(x) = \arg\min_{x_j \in \mathcal{X}} d(x, x_j),$$

where x_j is the j^{th} training pattern and $d(x, x_j)$ is the distance between x and x_j. Intuitively, NN(x) is in the proximity of x; so NN(x) is at a minimum distance from x and is maximally similar to x. The NN rule assigns x the class label of NN(x).

EXAMPLE 7: Let the training set consist of the following two-dimensional patterns with associated labels:

TABLE 2.5 Example data set

$x_1 = (0.7, 0.7)$, $l_1 = 1$;	$x_2 = (0.8, 0.8)$, $l_2 = 1$;	$x_3 = (1.1, 0.7)$, $l_3 = 1$
$x_4 = (0.7, 1.1)$, $l_4 = 1$;	$x_5 = (1.1, 1.1)$, $l_5 = 1$;	$x_6 = (3.0, 2.0)$, $l_6 = 2$
$x_7 = (3.7, 2.7)$, $l_7 = 2$;	$x_8 = (4.1, 2.7)$, $l_8 = 2$;	$x_9 = (3.7, 3.1)$, $l_9 = 2$
$x_{10} = (4.1, 3.1)$, $l_{10} = 2$;	$x_{11} = (4.3, 2.7)$, $l_{11} = 2$;	$x_{12} = (4.3, 3.1)$, $l_{12} = 2$
$x_{13} = (3.1, 0.3)$, $l_{13} = 3$;	$x_{14} = (3.1, 0.6)$, $l_{14} = 3$;	$x_{15} = (3.7, 0.4)$, $l_{15} = 3$
$x_{16} = (3.4, 0.6)$, $l_{16} = 3$;	$x_{17} = (3.9, 0.9)$, $l_{17} = 3$;	$x_{18} = (3.9, 0.6)$, $l_{18} = 3$

For the i^{th} pattern x_i, the class label is l_i and $l_i \in \{1, 2, 3\}$ for $i = 1, 2, \ldots, 18$. This can be seen in Fig. 2.2. Here '▲' corresponds to Class 1, '+' corresponds to Class 2 and '*' corresponds to Class 3. Now if there is a test pattern $T = (2.1, 0.7)$, it is necessary to find the distance from T to all the training patterns.

Let the distance between a training pattern x and T be the Euclidean distance

$$d(x, T) = \sqrt{(x(1) - T(1))^2 + (x(2) - T(2))^2}$$

The distance from a point T to every point in the training set can be computed using the above formula. For $T = (2.1, 0.7)$, the distance to x_1 is

$$d(x_1, T) = \sqrt{(0.7 - 2.1)^2 + (0.7 - 0.7)^2} = 1.4$$

We find, after calculating the distance from all the 18 training points to T, that the closest neighbor of T is x_3, which has a distance of 1.0 from T and x_3, which belongs to Class 1. Hence, T is classified as belonging to Class 1.

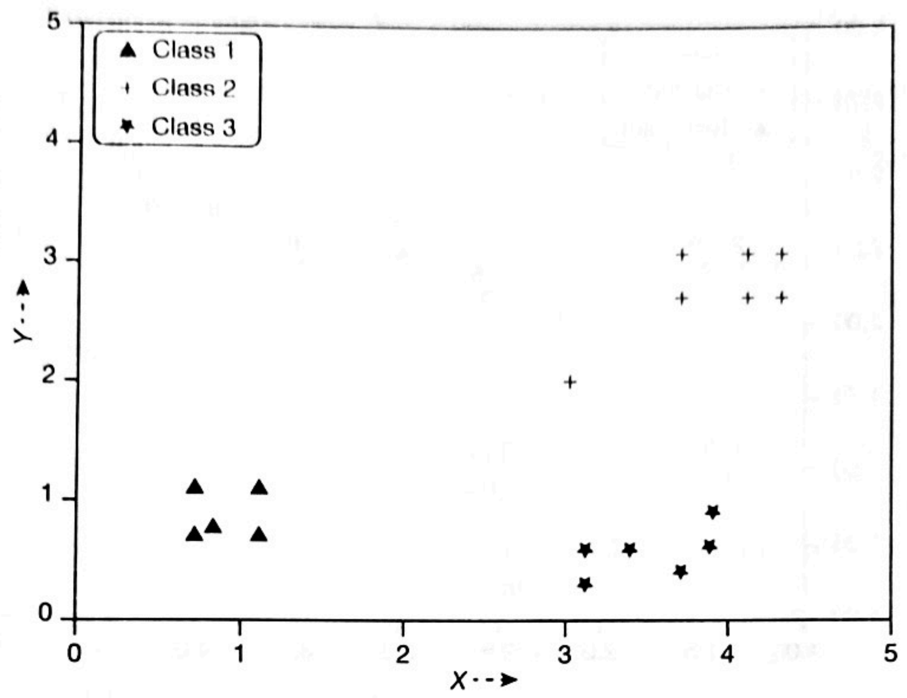

FIG. 2.2 Example data set in graphical form

2.3.2 K-Nearest Neighbor Classifier

In the k-nearest neighbor (KNN) algorithm, we find the k nearest neighbors of a test pattern x from the training data $\mathcal{X}$, and then assign the majority class label among the k neighbors to x. To determine the k nearest neighbors of x, it is necessary to calculate the distance between x and each of the n training patterns in the l-dimensional space. The distance metric used depends on the specific problem at hand, and can be Euclidean distance, Manhattan distance or cosine distance, among others. The class label of x is then determined based on the majority class label among its k nearest neighbors.

Assuming that Fig. 2.2 is a visual representation of the KNN algorithm being applied to a test pattern T, if the value of k is set to 5, the five nearest neighbors of T are x_3, x_{14}, x_{13}, x_5 and x_{16}. Among these five patterns, the majority class is Class 3. By using this method of selecting the majority class label among the k nearest neighbors, the errors in classification can be reduced, especially when the training patterns are noisy. While the closest pattern to the test pattern may belong to a different class, considering the number of neighbors and the majority class label increases the likelihood of correct classification. Figure 2.3 visually demonstrates this phenomenon.

It indicates that the test point T is closest to point 5, which is an outlier in Class 1 and is represented as 'x'. If the KNN algorithm is used, the point T will be classified as part of Class 2, which is represented by '+'. The selection of k is a crucial aspect of this algorithm. In the case of large data sets, k can be increased to decrease the error. The value of k can be determined through experimentation by keeping aside a subset of the training data as the validation data and classifying patterns from the validation set using different values of k using the training patterns to compute the neighbors. The value of k can be selected based on the lowest error observed in classification.

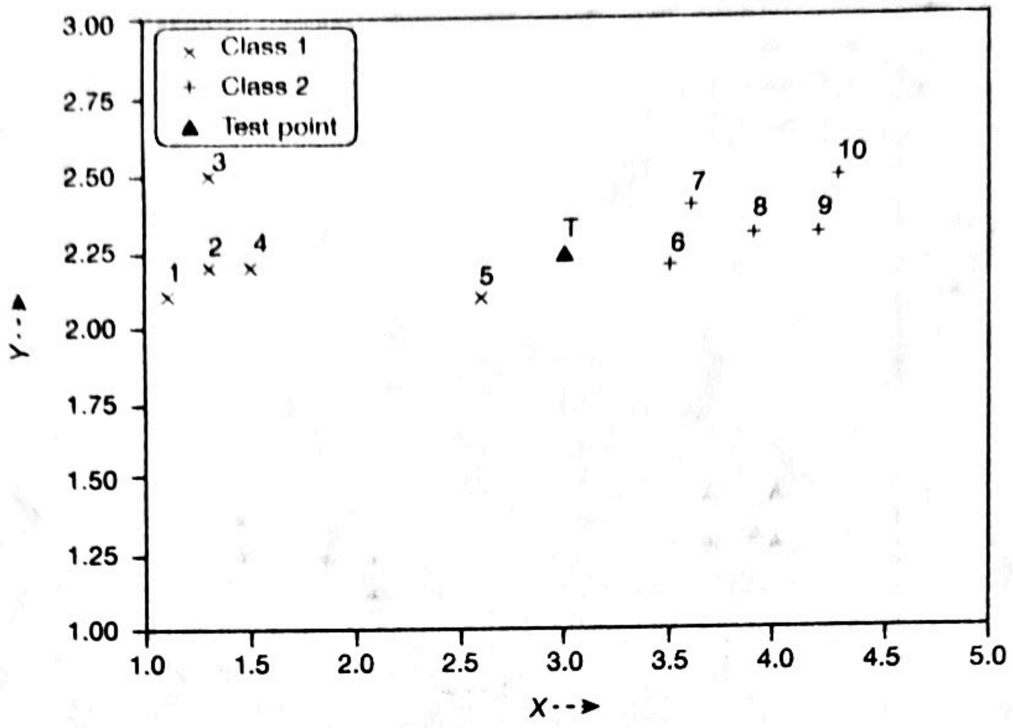

FIG. 2.3 Classification of T using the KNN classifier

EXAMPLE 8: Based on Fig. 2.2, if the pattern T is (2.1, 0.7), its nearest neighbor is x_3, and it would be classified as part of Class 1 if the nearest neighbor algorithm is employed. However, if the five nearest neighbors are considered, they are x_3 and x_5, both belonging to Class 1, and x_{14}, x_{13} and x_{16}, belonging to Class 3. Following the majority class rule, the pattern T would be classified as part of Class 3.

2.3.3 Weighted *K*-Nearest Neighbor (WKNN) Algorithm

This algorithm is similar to the KNN algorithm, as it also considers the k nearest neighbors. However, this algorithm takes into account the distance of each of the k neighbors from the test point by weighting them accordingly. It is also known as the distance-weighted k-nearest neighbor algorithm. Each neighbor is associated with a weight w, which is determined by the following formula:

$$w_j = \begin{cases} \frac{(dk-dj)}{(dk-d1)} & if \quad (d_k \neq d_1) \\ 0 & if \quad (d_k = d_1) \end{cases}$$

Here, j represents the neighbor's index in the list of k nearest neighbors, while d_k and d_j are the distances between the test point and the k^{th} neighbor and the j^{th} neighbor, respectively. The value of w_j ranges from 1 for the nearest neighbor to 0 for the k^{th} (most distant) neighbor. Using these weights, the WKNN algorithm assigns the test pattern to the class with the highest total weight of its representative neighbors among the k nearest neighbors. Unlike the traditional KNN algorithm, the WKNN algorithm employs a weighted majority rule, which takes into account the effect of outlier patterns in classification.

EXAMPLE 9: Consider $T = (2.1, 0.7)$ in Fig. 2.2. For the five nearest points, the distances from T are

$d(T, x_3) = 1.0$; $d(T, x_{14}) = 1.01$; $d(T, x_{13}) = 1.08$; $d(T, x_5) = 1.08$; $d(T, x_{16}) = 1.30$

The weight values will be

$w_3 = 1.0$

$w_{14} = \frac{(1.30-1.01)}{(1.30-1.00)} = 0.97$

$w_{13} = \frac{(1.30-1.08)}{(1.30-1.00)} = 0.73$

$w_5 = \frac{(1.30-1.08)}{(1.30-1.00)} = 0.73$

$w_{16} = 0$

Summing up for each selected class, Class 1 to which x_3 and x_5 belong sums to 1.73, and Class 3 to which x_{14}, x_{13} and x_{16} belong sums to 1.7. Therefore, the point T belongs to Class 1.

It is possible that the KNN and WKNN algorithms assign the same pattern a different class label. This is also illustrated in the following example.

EXAMPLE 10: In Fig. 2.2, when $T = (1.9, 2.4)$, the five nearest patterns are x_6, x_5, x_4, x_7 and x_3. The distances from T to these patterns are

$d(T, x_6) = 1.17$; $d(T, x_5) = 1.53$; $d(T, x_4) = 1.77$; $d(T, x_7) = 1.83$; $d(T, x_3) = 1.88$

The weight values are

$w_6 = 1$

$w_5 = \frac{(1.88-1.53)}{(1.88-1.17)} = 0.493$

$w_4 = \frac{(1.88-1.77)}{(1.88-1.17)} = 0.155$

$w_7 = \frac{(1.88-1.83)}{(1.88-1.17)} = 0.07$

$w_3 = 0$

Summing up for each class, Class 2 to which x_6 and x_7 belong sums to 1.07 and Class 1 to which x_5, x_4 and x_3 belong sums to 0.648, and therefore, T is classified as belonging to Class 2 by WKNN. Note that the same pattern is classified as belonging to Class 1 when we use the KNN algorithm with $k = 5$.

2.3.4 Radius Distance Nearest Neighbor Algorithm

This algorithm is an alternative to the KNN algorithm that considers all the neighbors within a specified distance r of the point of interest. This algorithm can be described as follows:

1. Given a point T, identify the subset of data points that fall within the radius r centred at T, denoted by
$$Br(T) = \{x_i \in \mathcal{X} \ s.t. \ \|T - Xi\| \leq r\}$$
2. If $Br(T)$ is empty, output the majority class of the entire data set.
3. If $Br(T)$ is not empty, output the majority class of the data points within $Br(T)$.

This algorithm is useful for identifying outliers, as any pattern that does not have similarity with the patterns within the chosen radius can be identified as an outlier. The choice of the value of radius r is critical as it can affect the performance of the algorithm.

By considering all neighbors within the specified radius, this algorithm can be more effective than the traditional KNN algorithm, especially when the nearest neighbor is too far away to be relevant.

EXAMPLE 11: For the example shown in Fig. 2.2, from point $T = (2.1, 0.7)$, the patterns which are located within a radius of 1.45 are x_1, x_2, x_3, x_5, x_{13}, x_{14} and x_{16}. The majority of these patterns belong to Class 1. T is therefore assigned to Class 1.

2.3.5 Tree-Based Nearest Neighbor Algorithm

This section explores how to find nearest neighbors in the transaction databases. Transaction databases store data collected from various sources such as businesses, scientific experiments, physical systems monitoring or supermarket transactions. Also known as market basket data, these databases contain the transactions made by each customer, with each transaction consisting of items bought by the customer. The transactions can differ in size, and the objective of analysing this data is to establish a relationship between certain items in the transactions. This process, known as **association rule mining**, aims to identify the occurrence of one item based on the occurrence of other items. To simplify and expedite the process, only frequently occurring items are considered, with a minimum support value chosen to exclude items occurring less than the minimum support. One of the tree data structures used to represent the processed transactional database is the frequent pattern (FP) tree.
To construct the FP tree:

1. The first step is to determine the frequency of each item in the transaction database. Frequency of an item is the number of transactions in which it occurs in the transaction database. Consider only those items whose frequency is greater than or equal to the user-defined minimum support; sort them in descending order of frequency.
2. Each entry in the transaction database is then arranged in the same order of frequency of items from largest to smallest, ignoring the infrequent items.
3. The root of the FP tree is created and labelled *null*. The first transaction in the database is used to construct the first branch of the FP tree based on the ordered sequence.
4. The second transaction is added to the tree using the same order. The common prefix between the second and first transaction is added to the existing path, increasing the count by one. For the remaining part of the transaction, new nodes are created. This process is repeated for the entire database.

The following example shows how the FP tree can be constructed from a transaction database, which is in the form of a table of size 4 × 4. Consider a 4 × 4 square to represent digits, with each square serving as a pixel. The squares are assigned a positional value, as illustrated in Table 2.6. For instance, the digit 0 can be represented using the 4 × 4 square depicted in Table 2.7 and denoted by the positional values 1, 2, 3, 4, 5, 8, 9, 12, 13, 14, 15 and 16. Table 2.8 displays how the digits 0, 1, 4, 6 and 7 can be represented using this method.

TABLE 2.6 Positional representation of a pixel 4 × 4 table

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

TABLE 2.7 Representation of '0' in a 4 × 4 table

1	1	1	1
1			1
1			1
1	1	1	1

The frequency of each item can be determined while scanning the transaction database presented in Table 2.8. Sorting the items from the highest to the lowest frequency results in the following list: (12:5), (1:4), (16:4), (4:3), (5:3), (8:3), (9:3) and (15:3). With the minimum support set to 3, only items with a frequency of 3 or more are included in this list. It is important to note that any ties that may arise are settled arbitrarily.

TABLE 2.8 Transaction database

Digit	Transaction (Positional information of a digit)
0	1, 2, 3, 4, 5, 8, 9, 12, 13, 14, 15, 16
1	4, 8, 12, 16
4	1, 5, 7, 9, 10, 11, 12, 15
6	1, 5, 9, 10, 11, 12, 13, 14, 15, 16
7	1, 2, 3, 4, 8, 12, 16

The transaction database in Table 2.9 shows the transactions with the items ordered based on their frequencies. With support value set to 3, only items with a frequency of three or more are retained in this table. The items with a support of two or less are removed.

TABLE 2.9 Transaction database with transactions ordered according to frequency of items

Digit	Transaction (After removing non-frequent items)
0	12, 1, 16, 4, 5, 8, 9, 15
1	12, 16, 4, 8
4	12, 1, 5, 9, 15
6	12, 1, 6, 5, 9, 15
7	12, 1, 16, 4, 8

Using this ordered database, an FP tree is constructed, as shown in Fig. 2.4.

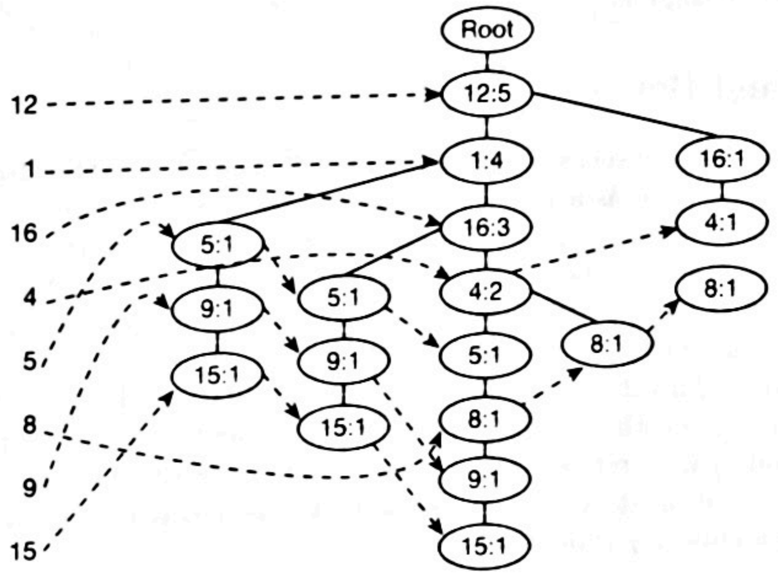

FIG. 2.4 FP tree for the transaction database in Table 2.8

The root node points to the starting item of the transactions, which in this case is 12. Each transaction is processed sequentially, and the corresponding nodes are added to the tree. The count of each item is incremented along the path that already exists, and new nodes are created for the remaining items in the transaction.

For example, in the first transaction, the path goes from the root node to 12, then to 1, 16, 4, 5, 8, 9 and 15. Each of these nodes has a count of 1. In the second transaction, since the next item after 12 is not 1, a new branch is created from 12 to 16, and the counts of 12, 16, 4 and 8 are incremented by 1. This process continues for each transaction until the entire database has been processed.

The resulting FP tree is a compressed representation of the frequent items in the transaction database. Each node in the tree corresponds to an item, and the count of each item is stored at the corresponding node. The header node for each item points to all the nodes in the tree that contain that item.

FP Tree-Based Nearest Neighbor: To identify the nearest neighbor of a test pattern in a transaction database, the FP tree can be utilised with the following approach:

1. Firstly, remove items in the test pattern that are below the minimum support threshold. Then, arrange the remaining items in the pattern according to their order in the FP tree.
2. Next, search the tree from the root node for the branch containing the first item in the test pattern. If this branch exists, continue to search for the next item in the pattern, and so on.
3. In the event that an item does not exist, examine all the branches from that point and choose the one with the maximum number of common items with the test pattern. This will determine the nearest neighbor.

EXAMPLE 12: Suppose we have an FP tree as shown in Fig. 2.4 for the transaction data set in Table 2.8. Let us consider a test pattern with features 1, 2, 3, 4, 6, 7, 8, 12 and 16. Removing items below the minimum support threshold, we are left with 1, 4, 8, 12, 16. By arranging these items in the order in which they appear in the FP tree, we get 12, 1, 16, 4, 8. Starting from the root node of the FP tree (12), we can compare the remaining items in the test pattern. It is observed that the test pattern has the maximum number of items in common with digit 7. Therefore, it can be classified as belonging to digit 7.

2.3.6 Branch and Bound Method

The branch and bound method seeks to efficiently find the nearest neighbor by taking advantage of an ordered data structure such as a tree-like structure. By clustering the data into representative groups with the smallest possible radius, we can search for the nearest neighbor while avoiding branches that cannot possibly have a closer neighbor than the current best value found. Lower bounds can be computed for the distances in the other clusters, allowing us to eliminate clusters that cannot possibly contain the nearest neighbor.

This recursive method involves clustering the data points hierarchically into subsets until there are clusters of one point, and then finding the nearest neighbor to a new point T. This is done by computing lower bond b_j with reference to cluster j and recursively branching to the cluster with the smallest b_j until the nearest neighbor is found or the bound is not satisfied.

Note that b_j for a cluster j is obtained by

$$b_j = d(T, \mu_j) - r_j,$$

where μ_j is the centre of the cluster j and r_j is its radius.

The branch and bound method has shown significant improvements over the standard nearest neighbor algorithm on an average.

For the patterns shown in Table 2.5, the branch and bound method is demonstrated using Fig. 2.5. The first step involves clustering the points into subsets. In this example, the points of Class 1 form Cluster 1, the points of Class 2 form Cluster 2, and the points of Class 3 form Cluster 3. Further sub-clusters are formed by clustering Cluster 1 into Clusters 1a and 1b, Cluster 2 into Clusters 2a and 2b, and Cluster 3 into Clusters 3a and 3b. At the next level, each point is taken as a sub-cluster. Figure 2.5 also shows the centres and radii of the clusters created for the patterns shown in Table 2.5.

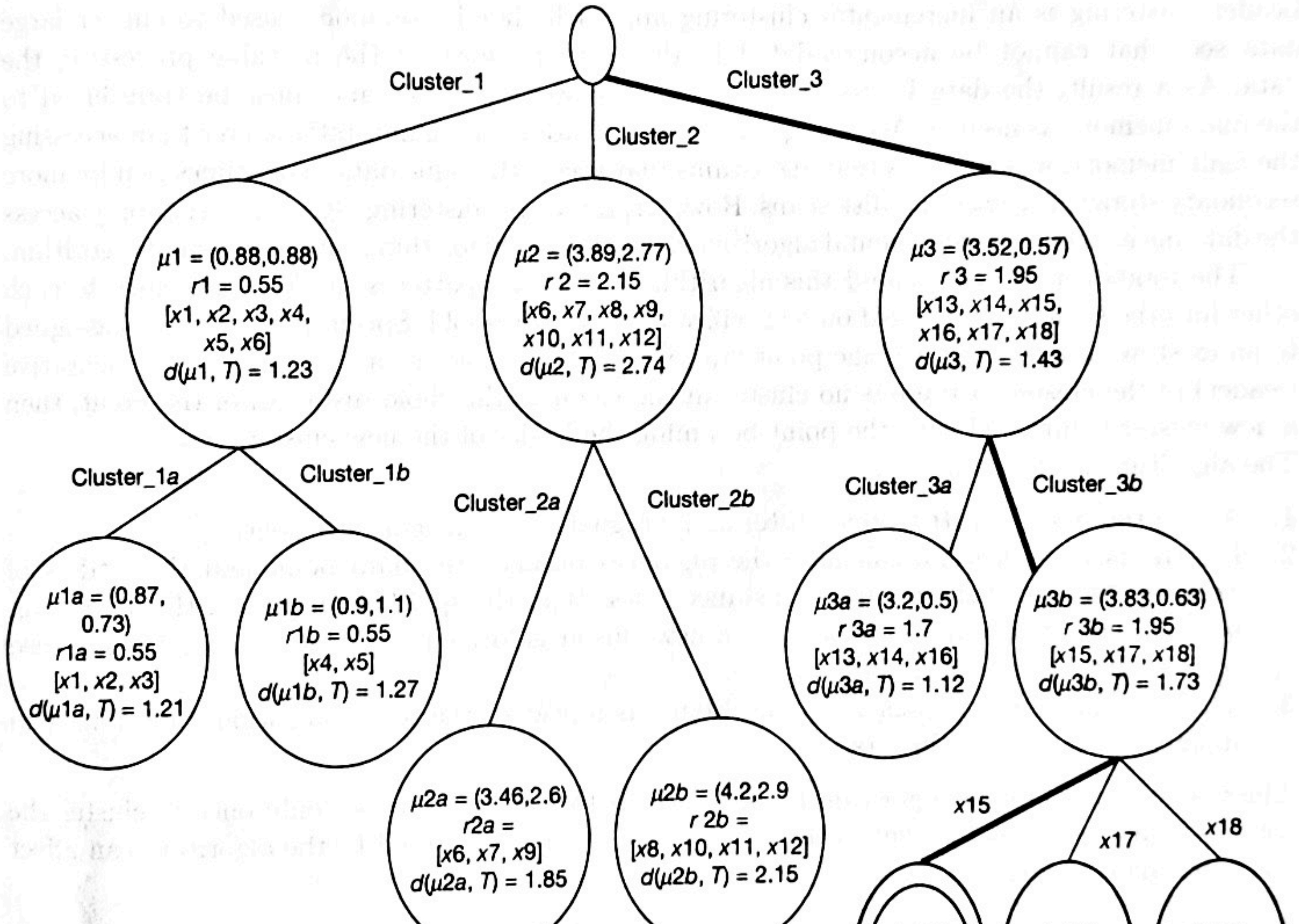

FIG. 2.5 Working of branch and bound algorithm for Table 2.5

EXAMPLE 13: To find the nearest neighbor of a new point $T = (2.1, 0.7)$, the lower bounds b_j for each cluster j are computed.

For Cluster 1, $b_1 = d(T, \mu_1) - r_1 = 0.68$. For Cluster 2, $b_2 = 0.59$. For Cluster 3, $b_3 = 0.52$.

Since b_3 is the smallest, the sub-clusters of Cluster 3 are searched. The centre of Cluster 3a is (3.2, 0.5) with a radius of 1.7, and the centre of Cluster 3b is (3.83, 0.63) with a radius of 1.95. This gives rise to $b_{3a} = 0.58$ and $b_{3b} = 0.22$. The second sub-cluster of Cluster 3 is searched and point x_{15} is found to be the closest point to T.

The bound d is calculated as the distance from T to point x_{15}, which is 1.63. Since b_1 and b_2 are greater than d, Clusters 1 and 2 need not be searched. Therefore, x_{15} is declared as the nearest neighbor of T.

2.3.7 Leader Clustering

Leader clustering is an incremental clustering approach that is commonly used to cluster large data sets that cannot be accommodated in the main memory of the machine processing the data. As a result, the data is stored in a secondary storage device and must be transferred to the main memory as needed. Accessing the secondary storage is significantly slower than accessing the main memory, which means that algorithms that access the same data many times require more secondary storage accesses and disk scans. However, there are clustering algorithms that only access the data once, known as incremental algorithms. The Leader algorithm is an incremental algorithm.

The fundamental idea behind this algorithm is to group patterns in close proximity to each other into the same cluster based on a specified distance threshold. Specifically, a point is assigned to an existing nearest cluster if the point falls within a threshold distance from the representative (leader) of the cluster; if there is no cluster in the vicinity (threshold distance) of the point, then a new cluster is initiated with the point becoming the leader of the new cluster.
The algorithm works as follows:

1. Assign the first data item to a cluster and designate it as the cluster leader.
2. For the next data item, calculate the distances between the data point and the leaders of existing clusters. If the minimum distance is less than the specified threshold, the data point is assigned to that cluster. Otherwise, a new cluster is formed, and the data point is assigned to it as the new cluster leader.
3. The next data item is considered and Step 2 is repeated; the process continues until all data items are assigned to clusters.

The main strength of the algorithm is that it needs to scan the data set only once to cluster the set. It should be noted that the order in which the data is presented to the algorithm can affect the resulting clusters.

EXAMPLE 14: Consider the data given in Fig. 2.5. Let the data be processed in the order $x_1, x_2, \ldots, x_{18}$ and the threshold T be set to 1.5. To start with, x_1 is assigned to Cluster 1 and is the leader of Cluster 1. Then x_2, x_3, x_4 and x_5 are assigned to Cluster 1 since the Euclidean distance from each one of them is below the threshold 1.5 (that is, 0.1, 0.4, 0.4, 0.6, respectively) from x_1.

The Euclidean distance from x_1 to x_6 is 2.6 (which is more than the threshold) and hence forms a new cluster, Cluster 2, for which x_6 is the Cluster leader; x_7, x_8 and x_9 are assigned to Cluster 2, since the Euclidean distance from them to x_6 is below the threshold (that is, 1.0, 1.3, and 1.3, respectively, from x_6). The Euclidean distance from x_{10} is more than 1.5 with respect to x_1 and x_6 (which are, respectively, 4.2 and 1.6). So, a new cluster, Cluster 3 is formed with x_{10} as the cluster leader; x_{11} and x_{12} are assigned to Cluster 3, since the Euclidean distance from x_{10} to them is below the threshold (that is, 0.4 and 0.2, respectively, from x_{10}).

The Euclidean distance from x_{13} is more than 1.5 with respect to x_1, x_6 and x_{10} (which are, respectively, 2.4, 1.7 and 2.9). So, a new cluster, Cluster 4 is formed with x_{13} as the cluster leader; x_{14}, x_{15}, x_{16}, x_{17} and x_{18} are assigned to Cluster 4, since the Euclidean distance from x_{13} to them is below the threshold (that is, 0.3, 0.6, 0.4, 1.0 and 0.9, respectively, from x_{13}).

Figure 2.6 shows the clusters formed using the Leader algorithm with a threshold value of 1.5.

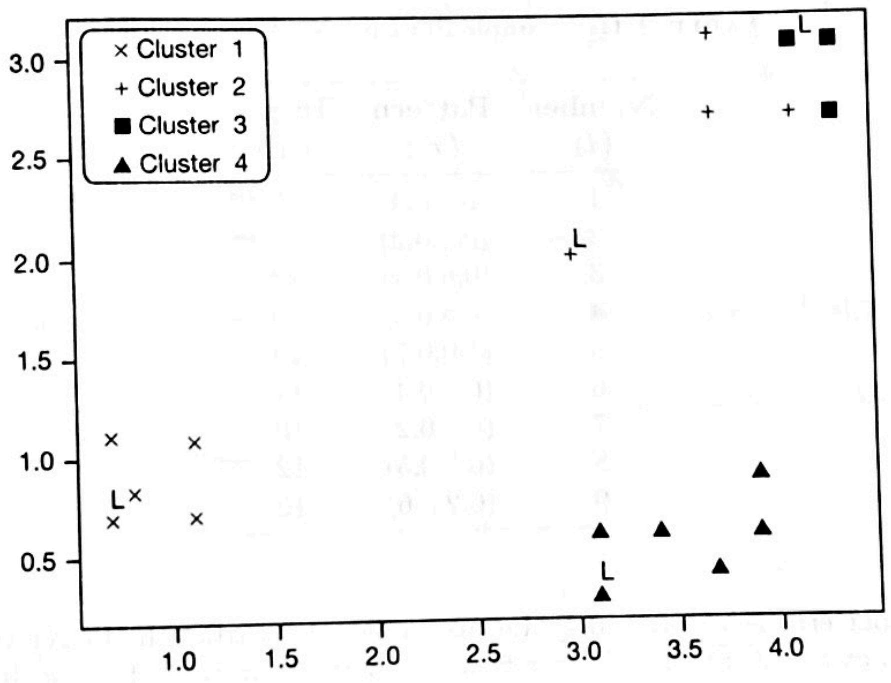

FIG. 2.6 Clusters formed using the Leader algorithm for the data set in Table 2.5 with leaders shown as 'L'

2.4 KNN REGRESSION

It is possible to use KNN for regression also. So, in this case we are given a set $\mathcal{X}$ of n labelled examples, where

$$\mathcal{X} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

Here x_i, $i = 1, 2, \ldots, n$ is a data vector and y_i is a scalar. It is possible for y_i also to be a vector in some applications. However, we restrict our attention to scalar y_is. The regression model needs to use $\mathcal{X}$ to find the value of y for a new vector x. In the case of regression based on KNN, we perform the following:

1. Find the k nearest neighbors of x from n data vectors. Let them be $x^1, x^2, \ldots, x^k$.
2. Consider the y values associated with these x's. Let them be $y^1, y^2, \ldots, y^k$.

3. Take the average of these y's and declare this average value to be the predicted value of y associated with x. So, the predicted value of y, call it $\hat{y}$ is,

$$\hat{y} = \frac{1}{k}(y^1 + y^2 + \cdots + y^k)$$

We will illustrate it using the following example.

EXAMPLE 15: Consider the data shown in Table 2.10.

TABLE 2.10 Example data for KNN regression

Number (i)	**Pattern** (x_i)	**Target** (y_i)
1	(0.2,0.4)	8
2	(0.4,0.2)	8
3	(0.6,0.4)	12
4	(0.8,0.6)	16
5	(1.0,0.7)	19
6	(0.8,0.4)	14
7	(0.6,0.2)	10
8	(0.5,0.5)	12
9	(0.2,0.6)	10

Let the new pattern be $x = (0.3, 0.4)$. Let us see how to predict the target value for x using KNN regression. Let $k = 3$; the 3 NNs of x from the patterns in the table are (0.2,0.4), (0.4,0.2) and (0.5,0.5). The corresponding target values observed in the table are 8, 8 and 12, respectively. The average of these values is $\frac{8+8+12}{3} = 9.33$. So, the predicted target value for $x = (0.3, 0.4)$ is 9.33.

Further, we are given that the underlying model is a linear model such that $y_i = f(x_i) = ax_i(1) + bx_i(2) + c$, where $x_i(1)$ is the first component and $x_i(2)$ is the second component of the vector x_i.

If we use the first three entries in the table and solve for a, b and c, we get $a = b = 10$ and $c = 2$. Note that the remaining 6 rows satisfy this linear model. Using the model, the target value for $x = (0.3, 0.4)$ is 9. The value predicted by KNN regression is 9.33. The squared error is $(9 - 9.33)^2 = 0.1111$ that is obtained by using the predicted value (9.33) and the value given by ground truth (9).

We will now examine the role of KNN regression on a real-world data set.

We consider the Boston Housing data set, it contains information collected by the U.S. Census Service concerning housing in the Boston area. It has 13 features including crime rate in the town and the number of rooms in the dwelling. The target is the median value of owner-occupied homes in \$1000s. We consider the first 250 pattern vectors in the set for this experiment. We use 200 patterns for training and the remaining 50 patterns for testing. The true target values and the predicted target values of the 50 test patterns are shown in Fig. 2.7.

The corresponding MSE values for different values of k are plotted in Fig. 2.8 where MSE is the average of the squared errors across the collection of patterns.

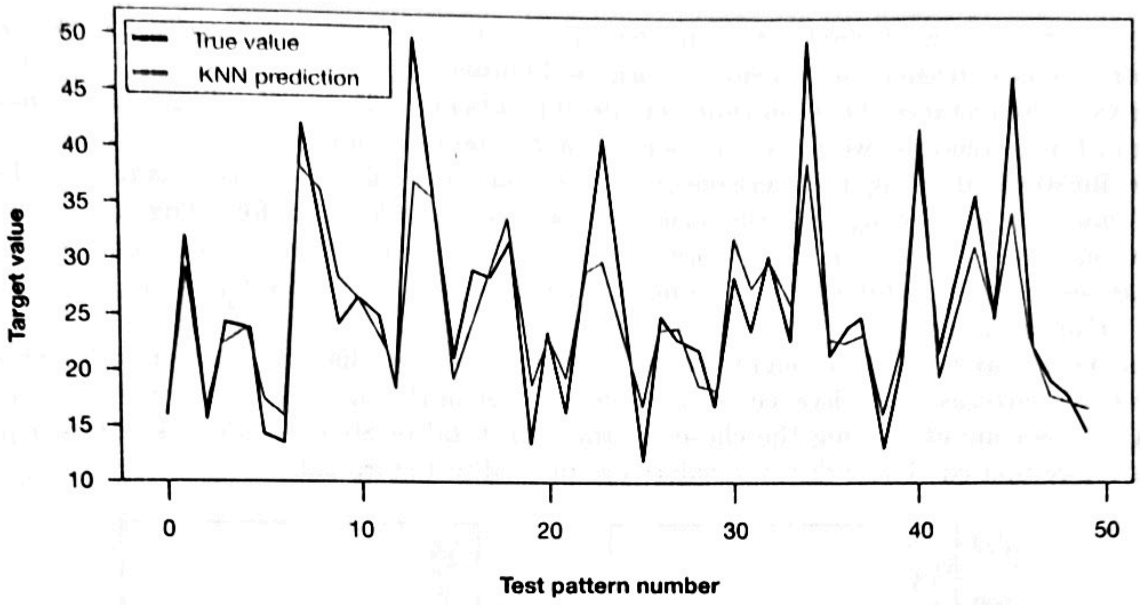

FIG. 2.7 KNN regression: results on the Boston Housing data (for colour figure, please see Colour Plate 1)

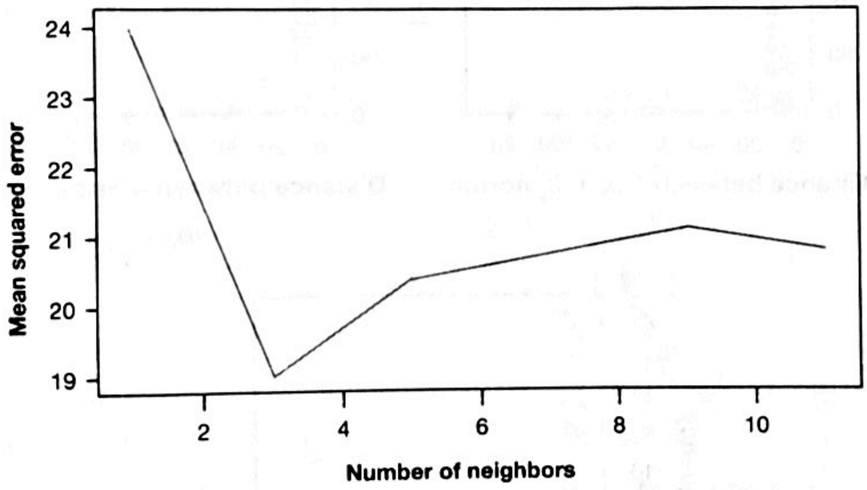

FIG. 2.8 KNN regression: MSE values on the Boston Housing data

2.5 CONCENTRATION EFFECT AND FRACTIONAL NORMS

A major difficulty encountered while using some of the popular distance measures like the Euclidean distance is that the distance values, between various pairs of points, may not show much dynamic range. Consider the following example.

EXAMPLE 16: Let $p = (4, 2)$ and $q = (2, 4)$ be two points in a two-dimensional space. Values of the distance using some popular distance norms are:

- L_∞ norm or the Max norm: $Max(|4-2|, |2-4|) = 2$.
- L_2 norm or Euclidean distance: $2\sqrt{2}$
- L_1 norm or City-Block distance: $|4-2| + |2-4| = 4$.

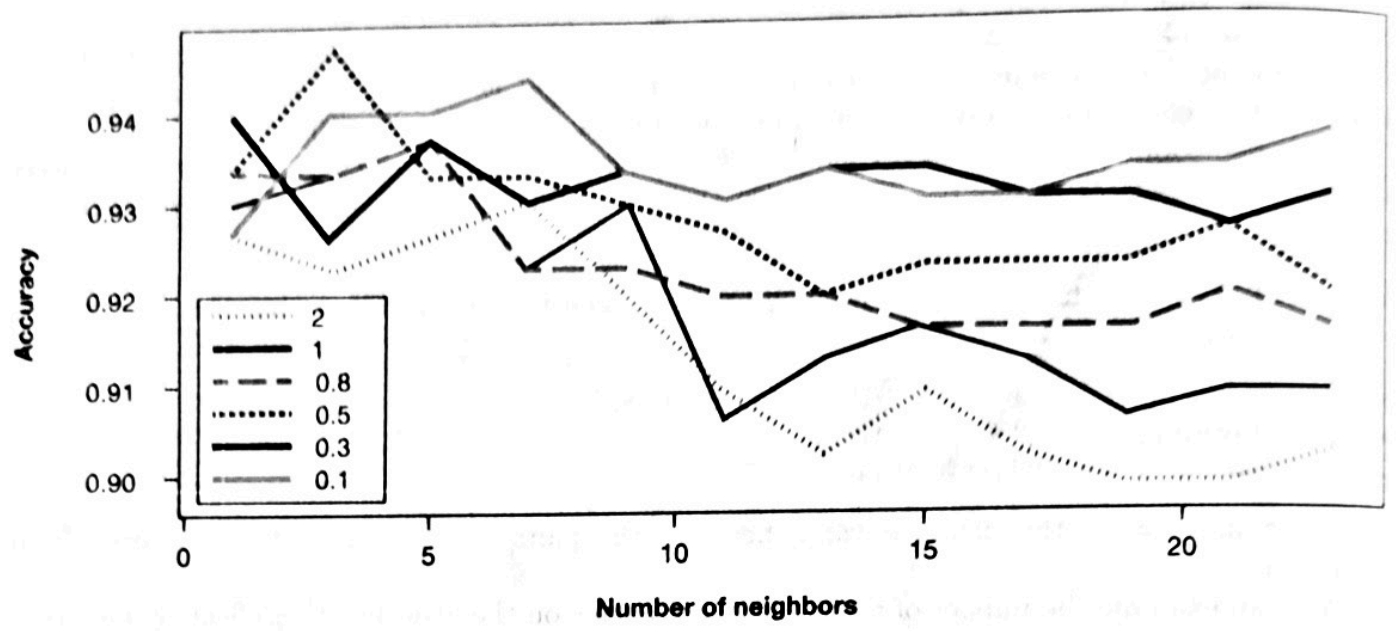

FIG. 2.11 Accuracy of KNN using different norms (for colour figure, please see Colour Plate 1)

2.6 PERFORMANCE MEASURES

There are several measures used to evaluate the performance of ML models. We will consider some of the popular measures in this section.

2.6.1 Performance of Classifiers

- **Classification Accuracy:** Let n be the total number of patterns that are classified by a classification algorithm. Let n_c be the number of correctly classified patterns. Then the classification accuracy is

$$\textit{Classification accuracy} = \frac{n_c}{n}$$

- **Confusion Matrix:** It captures the results in a compact form. This data can be compacted further to better analyse the results. We illustrate this with an example.

EXAMPLE 18: Let there be three classes C_1, C_2 and C_3. Let there be 200, 100 and 50 patterns from classes C_1, C_2 and C_3, respectively. Let a classifier classify these patterns into three classes, as shown in Table 2.11.

TABLE 2.11 Confusion matrix for a three-class problem

True/Predicted	C_1	C_2	C_3
C_1	180	15	5
C_2	5	85	10
C_3	3	2	45

The first row of the table shows how the 200 patterns from C_1 are classified: 180 of them are assigned to C_1, 15 are assigned to C_2 and 5 are assigned C_3. Similarly row 2 accounts for 100 patterns from C_2 and row 3 shows the assignment of 50 patterns from C_3.
Note that classification accuracy is obtained by considering the correctly classified patterns. Note that 180 patterns from C_1, 85 patterns from C_2 and 45 patterns from C_3 are correctly classified, as indicated by the diagonal entries in the matrix. They add up to 310. So, classification accuracy is $\frac{310}{350} \approx 0.8857$. The percentage accuracy is obtained by multiplying this number by 100. So, ~~the~~ percentage accuracy is 88.57%.
It is possible to compact the confusion table by looking at the entries with respect to C_1. The compact table of size 2 × 2 is shown in Table 2.12.

TABLE 2.12 Compact confusion matrix for C_1

True/Predicted	C_1	$\bar{C}_1$ [C_2+C_3]
C_1	180	20
$\bar{C}_1$	8	142

This compact table gives us useful information that could be used in calculating other evaluation measures. Now we are concerned with C_1 and $\bar{C}_1$ or not C_1 (classes other than C_1). The first row of this table shows that 180 patterns of C_1 are correctly classified. This entry corresponds to the correctly classified number of C_1 patterns to Class C_1. So, these are called true positives or TP. TP = 180.
The second column in the first row has 20 patterns. These patterns are from C_1 that are not classified as belonging to C_1; they are assigned to $\bar{C}_1$. So, they are called false negatives or FN. Here, FN = 20.
The first entry in the second row shows that 8 patterns from the other classes are assigned to Class C_1. So, they are called false positives or FP. So, FP = 8. The second entry in the second row corresponds to 142 patterns from $\bar{C}_1$ that are assigned to $\bar{C}_1$. These are called true negatives or TN. So, TN = 182. 142
We can use the confusion matrix entries, specifically TP, TN, FP and FN, to define additional evaluation measures. They are:

- *Precision*: It is the ratio of TP to the total number of predicted positives (TP + FP). So, precision in this example is $\frac{180}{188} \approx 0.9574$.
- *Recall*: It is the ratio of TP to the total number of positives in the training data (TP + FN). So, in this example, it is $\frac{180}{200} = 0.9$.
- *F1 Score*: It is the harmonic mean of precision and recall. More specifically, it is $\frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$. In this example, F1 Score ≈ 0.9278.

We can have one more popular measure called **area under the ROC (receiver operating characteristic) curve** based on some additional quantities derived from the table:

- True positive rate or TPR is given by

$$TPR = \frac{TP}{TP + FN}$$

- False positive rate or FPR is given by

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

- Area under the curve (AUC) is obtained by plotting a graph between FPR on the X-axis and TPR on the Y-axis. This is called the receiver operating characteristic (ROC). We illustrate this with the help of the following example.

EXAMPLE 19: Consider the entries in Table 2.12. The values are TP = 180, FP = 8, FN = 20 and TN = 182. So, TPR = $\frac{180}{180+20}$ = 0.9 and FPR = $\frac{8}{190}$ = 0.042. So, (0.042, 0.9) give us a point for the ROC plot.
We obtain multiple points by computing probabilities and thresholding them as explained next. Let there be two classes with labels 1 and 0. Let the training patterns from each class be given by
Class label 1: (1,1), (2,2), (3,3), (4,4), (5,5)
Class label 0: (4,1), (5,2), (6,3), (7,4), (8,5)
Let the validation patterns be given by
Class label 1: (2,1), (1,3)
Class label 0: (8,4), (7,5)

If we find $k = 5$ nearest neighbors for each validation pattern from the training set, we obtain details as shown in Table 2.13. Note that the table has 7 columns. The first column corresponds to the validation pattern considered. There are 4 rows in the table corresponding to the 4 validation patterns. The second column corresponds to the class label of these validation patterns. Note that the first two are from Class 1 and the remaining two are from Class 2.

TABLE 2.13 Computing probabilities and thresholding

Pattern	Class	5 NNs	Probability	Th>0.3	Th>0.5	Th>0.7
(2,1)	1	(1,1),(2,2),(4,1),(3,3),(5,2)	0.6	1	1	0
(1,3)	1	(2,2),(1,1),(3,3),(4,4),(4,1)	0.8	1	1	1
(8,4)	0	(8,5),(7,4),(6,3),(5,5),(5,2)	0.2	0	0	0
(7,5)	0	(8,5),(7,4),(5,5),(6,3),(4,4)	0.4	1	0	0

The third column lists ($k =$) 5 NNs, out of the 10 training patterns, of the validation pattern given in column 1. The fourth column gives the probability that the validation pattern belongs to Class 1. This probability is the ratio of the number of neighbors, out of 5, from Class 1 to the total number of NNs considered, that is, 5.

For example, for the validation pattern (2,1), three patterns, namely, (1,1), (2,2), (3,3) are from Class 1. So, the probability is $\frac{3}{5} = 0.6$. Similarly, probabilities are calculated for the remaining three validation patterns.

In columns 5 to 7, we use thresholds (Th) to convert the probability into 1 or 0. For example in column 5, if the probability is more than (Th =) 0.3, we put a 1, otherwise, a 0. Similarly, we get a 1, 0, 1 in rows 2, 3 and 4 based on the threshold (Th) value of 0.3. Entries in columns 6 and 7 are obtained as in column 5 but with thresholds 0.5 and 0.7, respectively.

For each of the thresholdings, we have a possible assignment of the four validation patterns. By comparing with the true class label given in column 2 of the table, we get an (FPR, TPR) pair as follows:

- For Th > 0.3, we get TP $= 2$, FN $= 0$, FP $= 1$, TN $= 1$. So, FPR $= \frac{1}{1+1} = 0.5$ and TPR $= \frac{2}{2+0} = 1$.
- For Th > 0.5, we get TP $= 2$, FN $= 0$, FP $= 0$, TN $= 2$. So, FPR $= \frac{0}{0+2} = 0$ and TPR $= \frac{2}{2+0} = 1$.
- For Th > 0.7, we get TP $= 1$, FN $= 1$, FP $= 0$, TN $= 2$. So, FPR $= \frac{0}{0+2} = 0$ and TPR $= \frac{1}{1+1} = 0.5$.

So, by thresholding, we obtain different (FPR, TPR) pairs that are used in plotting the ROC curve. The standard rule is to convert the class decisions into probabilities and then threshold them to get FPR and TPR pairs.

Note that schemes for computing probabilities can differ when other classifiers are used. However, a similar thresholding scheme can be used to obtain FPR and TPR pairs once the probabilities are available.

relation b/w dependent & → independent Variable

2.6.2 Performance of the Regression Algorithms

- **Mean squared error (MSE):** It is the most popular metric used for regression. It is defined by

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2,$$

where n is the number of patterns, y_i is the target value for the i^{th} pattern and $\hat{y}_i$ is the value predicted by the regression model.

- **Mean absolute error (MAE):** It is the average of the difference between the target and the predicted values. It is given by

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

2.7 Area Under the ROC Curve for the Breast Cancer Data Set

We consider the Breast Cancer data set for this experiment.

- The total number of patterns is 569 and each is a 30-dimensional vector.
- There are two classes: *Benign* and *Malignant*.
- We use KNN for classification with the value of $k = 15$.
- We use 171 patterns for validation and 398 patterns for training.
- We used the sklearn platform for conducting the experiment.
- We show the ROC curve in Fig. 2.12.
- Note the broken line in the figure. This indicates the possible lower bound.
- Ideally the grey line, the ROC curve, should go vertical on the Y-axis from 0 to 1 and then go horizontal at $Y = 1$, parallel to the X-axis from $X = 0$ to $X = 1$.

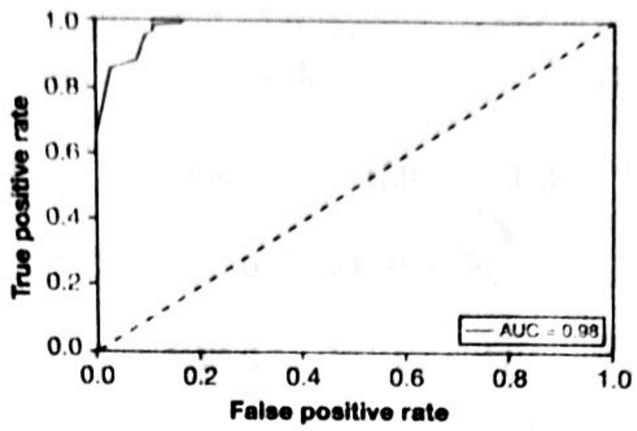

Fig. 2.12 ROC curve for Breast Cancer data using KNN (for colour figure, please see Col
Plate 1)

- So, in the ideal case, the area under the ROC should be 1.
- The ROC curve is almost ideal, but for the top-left corner.
- Note that the area under the curve, AUC, is 0.98.

Summary

- We have considered neighborhood-based ML models in this chapter. They include:
  - KNN classifier; it is the NNC when $k = 1$. Evaluation of its performance is based classification accuracy.
  - Regression or function value estimation using KNN. Evaluation of the KNN-based es mation scheme is by using the MSE.
  - Clustering based on spherical neighborhood using a threshold value as the radius of t sphere. In the Leader clustering algorithm, each of the resulting clusters is represent by its leader.
- We have identified the role of proximity measures in neighborhood models. We have explor several proximity measures that are popularly used.
- We have detailed a tree-based NNC and a branch and bound-based NNC for efficiency.
- We have studied the concentration problem that results as a consequence of using the popul integral norms including the Euclidean distance. We have discussed how fractional norms c overcome this problem by expanding the dynamic range of the distance values.
- We have shown experimental results on the Wisconsin Breast Cancer data set for classificati and the Boston Housing data set for regression.
- We have detailed the performance evaluation measures that can be used to analyse vario ML models. Classification accuracy and MSE are the most popular for classification a regression, respectively.
- When there is class imbalance, that is, a class has a smaller number of patterns and t other class has a very large number of patterns, then accuracy is not the right choice. Oth measures like F1 score need to be used.

- **Explanation Capability:** These neighborhood ML models are based on
  - The intuitively appealing fact that similar things exist in close proximity of each other.
  - NNC is a simple and transparent classifier as the decision to assign a class label to a test pattern is based on the class label of its nearest pattern. It is not difficult for experts in an application domain to appreciate and understand how it works.
  - Even KNN and regression based on KNN is very transparent for the respective domain experts to understand the decision-making process based on extended neighborhood.
  - The Leader clustering algorithm is the simplest clustering algorithm. It is order dependent; it gives different clusterings and sets of leaders for different orders in which the data is presented to the algorithm. However, it is transparent and easy to appreciate as the leaders are good representatives for clusters.

EXERCISES

Use the Euclidean distance for computing the distance unless specifically stated otherwise.

1. Consider two l-bit binary strings p and q. How are $\text{HD}(p,q)$ and $\text{SMC}(p,q)$ related?
2. Derive the condition under which $\text{SMC}(p,q) > \text{JC}(p,q)$ for l-bit strings p and q.
3. Let $p = (4,2,5)$ and $q = (2,1,-7)$. Find the distance between these two patterns using
   a. Euclidean distance
   b. City-block distance
   c. Max norm (L_∞)
   d. Fractional norm ($r = 0.25$)
4. Consider the points $p = (3,0), q = (0,3)$ and $s = (0,0)$. Consider the fractional norm with $m = 0.5$. Is it a metric?
5. Give a two-dimensional example having 2 classes, where NNC is better than KNN with $k = 3$. This example illustrates that NNC can perform better than KNN ($k > 1$) in some cases.
6. Consider the set of two-dimensional patterns: ((1, 1.5), 1), ((1, 2.5), 1), ((1, 3.5), 1), ((2, 1.5), 1), ((2, 2.5), 1), ((2, 3.5), 1), ((2, 4),1), ((2.5, 2.5), 1),((3.5, 1.5), 1), ((3.5, 2.5), 1), ((3.5, 3.5), 2), ((3.5, 4.5), 2), ((4.5, 1.5), 2), ((4.5, 2.5), 2), ((4.5, 3.5), 2),((5, 4.5), 2), ((5, 5.5),2),((6, 3.5), 2), ((6, 4.5), 2), ((6, 5.5), 2), where each pattern is represented by feature 1, feature 2 and the class.
   a. If a test pattern T is at (2.6, 5.5), find the class of T using the nearest neighbor algorithm.
   b. Find the class of T using the KNN algorithm, where k is 3.
   c. Find the class of T using the WKNN algorithm, where k is 3.
   d. Find the class of T using the radius distance nearest neighbor algorithm with the radius = 2.5.
7. Consider two sets of two-dimensional points from two classes:
   Class 1: (3.5,3), (4.5,1), (1,5), (2,6), (3.5,4), (4.5,3)
   Class 2: (2,3.5), (2.5,1), (3.5,2), (3,2), (2.5,2), (2,1)
   Consider a variant defined using the centroid data set. Let Centroid1 be the mean of the six vectors in Class 1. Similarly Centroid2 is the mean of the six vectors in Class 2. Note that

the mean vector, Centroid, is defined on a collection of n vectors $X_1, X_2, \ldots, X_n$ as

$$\text{Centroid} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

Now the centroids are used as the training data, that is, each of the new classes have only the centroid patterns $NewClass_1 : Centroid1$; $NewClass_2 : Centroid2$. Using this training data, classify the 12 patterns given using NNC.

a. This classifier is called the minimal distance classifier (MDC). What is the advantage of using MDC?

b. What is the classification accuracy of MDC in this problem on the 12 patterns?

8. Consider the two-dimensional vectors from two classes. Plot the decision boundaries for NNC and MDC. These decision boundaries are piece-wise linear and they are obtained by considering pairs of points, one from Class 1 and the other from Class 2. One side of the decision boundary will have Class 1 patterns and the other side will have Class 2 patterns.

Class 1	Class 2
(1,0)	(−2,0)
(−1,0)	(0,1)
(0,0)	(2,0)

9. Consider the data given in the following table.

Pattern	Vector (x)	Class label	Function $f(x)$
1	1 1	C_1	3
2	1 2	C_1	2
3	2 2	C_1	5
4	3 2	C_1	8
5	4 2	C_2	11
6	4 3	C_2	10
7	4 4	C_2	9
8	5 3	C_2	13

a. Use the Leader clustering algorithm on the 8 two-dimensional vectors using a threshold value of 3. Each vector has two components. So, you need not consider the last two columns for this problem. Give the leaders and the clusters obtained.

b. For this problem, we need to consider all the columns in the data other than the class labels.

i. Given that $f((x_1, x_2)) = ax_1 + bx_2 + c$, obtain the values of a, b and c using the first three patterns and the respective f values. Verify whether the remaining 5 patterns also satisfy this equation or not.

ii. Consider the pattern (3, 1).

A. Obtain its f value using $f((3, 1)) = 3a + b + c$ where a, b and c are obtained in part (i) of the problem.

B. Find its (k =) 3 nearest neighbors from the 8 patterns in the table.

C. Let $\hat{f}((3, 1))$ be the average of the f values of these 3 nearest neighbors. Obtain the value of squared error for (3, 1) by using $(f((3, 1)) - \hat{f}((3, 1)))^2$.

10. Cluster the 12 patterns given in Q7 using the Leader algorithm. Use a threshold of 1.5 units to cluster.
    a. Cluster the patterns in each class separately using the order in which the patterns are given. Let S_1^L be the set of leaders obtained. Give S_1^L.
    b. Cluster all the 12 patterns without using the class label. Let S_2^L be the set of leaders obtained. Assume that each leader belongs to a class if the cluster represented by the leader has a majority of patterns from that class. What is S_2^L?
    c. Is there a difference between S_1^L and S_2^L?
11. Consider the data in Q9(a) and reduce the data set size as follows. Take a training pattern x. Find out 3 ($k = 3$) nearest neighbors of x; break any ties in favour of the class to which x belongs. If all the k neighbors are from the same class as that of x, then mark x. Perform this over all training patterns and then drop all the marked patterns and report the reduced set.
12. Consider the following process called bootstrapping:
    Consider a pattern x from the data given in Q9(a) and find 2 nearest neighbors x^1, x^2 from the same class as x; compute x' by the mean of the $k+1$ (3) vectors, that is, x and its 2 NNs. Repeat this process on all the training patterns in the collection. Replace the training data points with the x values obtained. What is the effect of this bootstrapping process? What happens if x values are added to the training set instead of replacing it?
13. Consider a 4 × 4 square to represent a digit, with each square assigned a positional value as illustrated in Table 2.6. The table given below depicts the digits represented using this method.

Transaction database

Digit	Transaction (Positional information of a digit)
0	1, 2, 3, 4, 5, 8, 9, 12, 13, 14, 15, 16
1	4, 8, 12, 16
7	1, 2, 3, 4, 8, 12, 16
9	1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 16
4	1, 5, 7, 9, 10, 11, 12, 15

    Construct the FP tree for this table with minimum support = 3.
14. Consider the FP tree constructed in Q13, identify the label (digit) for the test pattern, T = 1, 5, 9, 10, 11, 12, 15.
15. Consider the data given in Table 2.12. Use $k = 3$ to compute the (FPR, TPR) pairs for the thresholds Th > 0.3, Th > 0.5 and Th > 0.7 on the probabilities obtained.

PRACTICAL EXERCISES

1. Randomly generate 100 values of x in the range [0,1]. Let them be $x_1, x_2, \ldots, x_{100}$. Perform the following based on the data set generated.
    a. Label the first 50 points $\{x_1, \ldots, x_{50}\}$ as follows: if ($x_i \leq 0.5$), then $x_i \in Class_1$, else $x_i \in Class_2$.
    b. Classification.
        i. Classify the remaining points, $x_{51}, \ldots, x_{100}$ using KNN. Perform this for $k = 1, 2, 3, 4, 5, 20, 30$.

ii. Classify the remaining points, $x_{51}, \ldots, x_{100}$ using WKNN. Perform this for $k = 1, 2, 3, 4, 5, 20, 30$.

iii. Classify the remaining points, $x_{51}, \ldots, x_{100}$ using radius-based NNC. Perform this for $k = 1, 2, 3, 4, 5, 20, 30$.

c. Compute the classification accuracy in all three cases and report. [Note: Classification accuracy $= \frac{nc}{n}$, where $n = 50$, nc = number correctly classified]

2. Cluster the entire set of 100 points (as mentioned in Q1) using Leader clustering. Choose different values for threshold (T) and carry out the clustering.
3. Let the clustering obtained using some threshold, T_i be $C_i = \{Cluster_1, Cluster_2, \ldots, Cluster_{il}\}$ Computer the purity value for each clustering, which is given by

$$Purity(C_i) = \sum_{j=1}^{il} maximum(|Cluster_j \cap Cluster_1|, |Cluster_j \cap Cluster_2|, \ldots, |Cluster_j \cap Cluster_{il}|)$$

4. Use the Digits data set available under sklearn: **`https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load\_digits.html`**
Consider 10% of the data for training (179 samples). Each pattern is an 8×8-sized character where each value is an integer in the range 0 to 16. Convert it into binary form by replacing a value below 8 by 0 and other values (≥ 8) by 1.

a. Use these 179 patterns with labels and the remaining without labels for this subtask. Use KNN and label the patterns without labels. Obtain the % classification accuracy. Perform this task with k values from the set $\{1, 3, 5, 10, 20\}$.

b. Obtain the frequent itemsets for these 179 patterns using FP-growth by viewing each binary pattern as a transaction of 64 items. Repeat this task with different *minsup* values from $\{0.1, 0.3, 0.5, 0.7\}$.

Bibliography

- Andoni, A and Indyk, P. 2008. Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions, *Communications of the ACM*, 51(1): 117–122.
- Beyer, K, Goldstein, J, Ramakrishnan, R and Shaft, U. 1998. When is "Nearest Neighbour" Meaningful?. *Proceedings of ICDT*, 217–235.
- Flexer, A and Schnitzer, D. 2015. Choosing l^p norms in high-dimensional spaces based on Hub Analysis. *Neurocomputing*, 169: 281–287.
- Murty, MN and Susheela Devi, V. 2015. *Introduction to Pattern Recognition and Machine Learning*, World Scientific Publishing Co. Pte. Ltd.: Singapore.
- Duda, RO, Hart, PE and Stork, DG. 2007. *Pattern Classification*, New York: John Wiley & Sons.
- Han, JW, Kamber, M and Pei, J. 2012. *Data Mining Concepts and Techniques, Third Edition*, Morgan Kaufmann Publishers, Waltham.