LECTURE NOTES
ON
PATTERN RECOGNITION

PREPARED BY

DR. PRASHANTA KUMAR PATRA

COLLEGE OF ENGINEERING AND TECHNOLOGY,
BHUBANESWAR

INTRODUCTION

- This course deals with pattern recognition. A pattern is either a physical object, for example a book or a chair or an abstract notion, like style of talking, or style of writing. It is also a shared property of a set of objects; for example, chairs, rectangles, or blue colored objects. We illustrate using ellipses and rectangles shown in Figure 1.

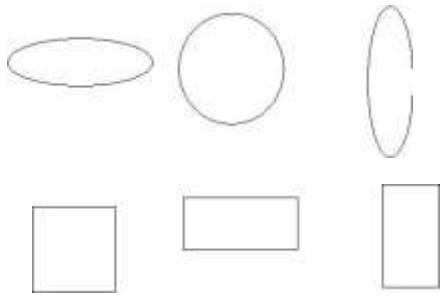

Figure 1: Ellipses and Rectangles

Cognition is the act of seeing or perceiving, whereas recognition means as having seen or perceived. There are three ways of appreciating the activity of pattern recognition in a simple manner:

1. Classification: Assign a pattern to one of the already known (semantically labelled) classes. For example, consider the two classes of physical objects shown in Figure 1: ellipses and rectangles where ellipse and rectangle are class labels. Now the classification problem, prominent in pattern recognition, involves:
   (a) Either learn a model or directly use the training data set (collection of labelled patterns) and
   (b) assign a class label to a new pattern (test pattern) or equiv-alently assign the test pattern to one of the known classes. That is, with respect to objects in Figure 1, given a new object we would like to classify it as either an ellipse or a rectangle.

The Classification problem: *We are given a collection of se-mantically labelled patterns,* X*, where*

$$X = \{(X_1, C^1), (X_2, C^2), \cdots, (X_n, C^n)\}$$

We need to note the following here:

- *The number of classes,* K*, is fixed and finite. The value of* K *is known* a priori. *Let the class labels be* $C_1, C_2, \cdots, C_K$.
- *The set* X *is finite and is of size (cardinality)* n*. Further,* X_i *represents the* i^{th} *pattern and* C^i *is the corresponding seman-tic class label for* $i = 1, \cdots, n$*. So, observe that* $C^i \in C = \{C_1, C_2, \cdots, C_K\}$.
- *Let* X *be a test pattern. Then, either we use the training set* X *directly or models* $M_1, M_2, \cdots, M_K$ *learnt from* X *to assign a class label, out of* $C_1, C_2, \cdots, C_K$*, to* X*. Here, model* M_i *is learnt from the training patterns drawn from class* C_i*, for* $i = 1, 2, \cdots, K$.

An Example: *Let us say that we are given the following collec-tion of* chairs *and* humans *as the training set.*

$X = \{(X_1, \text{chair}), (X_2, \text{chair}), (X_3, \text{human}), (X_4, \text{human}), (X_5, \text{human}), (X_6, \text{chair}), (X_7, \text{human}), (X_8, \text{chair}), (X_9, \text{human}), (X_{10}, \text{human})\}$

Now the problem is, given a test pattern X*, classify* X *as either* chair *or* human. *In other words, assign one of the two class labels to* X.

2. Clustering: *Assign a pattern to one of the* syntactically labelled classes *or clusters. For example, consider two clusters of patterns, labelled* C_1 *and* C_2*. Given a new pattern, assign it to either* C_1 *or* C_2 *based on the similarity between the pattern and the collection. Here, the labels are syntactic because we can switch the labels of the two collections without affecting the results.* Clustering *is concerned with grouping of patterns based on similarity. Patterns in a cluster are similar to each other whereas patterns in different clusters are dissimilar.*

Clustering Problem: *We are given a collection,* X *, of syntacti-cally labelled patterns, where*

$$X = \{X_1, X_2, \cdots, X_n\}.$$

Note that the patterns are syntactically labelled using different subscripts. The problem is to partition the set X *into some finite number of blocks or clusters. In other words, we partition* X *, so that*

$$X = C_1 \cup C_2 \cup C_3 \cdots \cup C_K$$

where C_i *is the* i^{th} *cluster. Clustering is done so that none of the* K *clusters is empty and any pair of clusters do not overlap, which means*

$$C_i \neq \emptyset, \text{ and } C_i \cap C_j = \emptyset \text{ for } i \neq j \text{ and } i, j \in \{1, 2, \cdots, K\}.$$

An Example of Clustering: *Consider a collection of patterns*

$$X = \{X_1, X_2, \cdots, X_{10}\}.$$

A possible partition, of X *having two clusters is*
$C_1 = \{X_1, X_2, X_4, X_5, X_7, X_8\}$ *and* $C_2 = \{X_3, X_6, X_9, X_{10}\}$. *Typically, a notion of* similarity *or* matching *is used to partition*
X . *Patterns in* C_1 *are similar to other patterns in* C_1 *and patterns in* C_2 *are similar to other patterns in* C_2*; a pattern, say* X_2*, in* C_1 *is dissimilar to a pattern, say* X_9*, in* C_2*. In clustering, it is possi-ble to switch the labels; for example, we have the same partition as above if*

$C_1 = \{X_3, X_6, X_9, X_{10}\}$
$C_2 = \{X_1, X_2, X_4, X_5, X_7, X_8\}$

3. Semi-Supervised Classification: *Here, we are given a small collection of semantically labelled patterns and a large collection of syntactically labelled patterns. The problem is to assign a new pattern (test pattern) to one of the classes or equivalently assign a semantic label to the test pattern.*

Semi-Supervised Classification Problem: We are given a collection, X , given by

$$X = \{(X_1, C^1), \cdots (X_l, C^l), X_{l+1}, \cdots X_{l+u}\}$$

where l patterns are semantically labelled and u patterns are syn-tactically labelled. The problem is to build models $M_1, M_2, \cdots M_K$ corresponding to classes $C_1, C_2, \cdots, C_K$ respectively. Now given a new pattern, X , classify it to one of the K classes using the models built.

An Example: Given a set, X , of patterns given by

$$X = \{(X_1, \text{human}), (X_2, \text{chair}), X_3, X_4, X_5, X_6, X_7\}$$

the problem is to assign a class label of chair or human to a new pattern (test pattern) X .

The popularity of pattern recognition (PR) may be attributed to its application potential; there are several important applications. For example,

– document recognition: there are a variety of applications including classification and clustering of
  * email messages and web documents; one requirement is to recognize whether a mail is spam or not.
  * fingerprints, face images, and speech signals which form an important variety of documents used in biometrics.
  * health records which may include x-ray images, ultrasound images, ECG charts and reports on various tests, diagnosis, and prescriptions of medicines.
  * legal records including judgments delivered, petitions and ap-peals made.
– remote sensed data analysis: for example, images obtained using satellite or aerial survey are analysed to discriminate healthy crops from deceased crops.
– bioinformatics: Here, classification and clustering of DNA and protein sequences is an important activity.

– semantic computing: Knowledge in different forms is used in clustering and classification to facilitate natural language under-standing, software engineering, and information retrieval.

– There are plenty of other areas like agriculture, education, and economics where pattern recognition tools are routinely used.

Abstractions

In machine recognition of patterns, we need to process patterns so that their representations can be stored on the machine. Not only the pattern representations, but also the classes and clusters need to be represented ap-propriately. In pattern recognition, inputs are abstractions and the outputs also are abstractions.

- As a consequence, we do not need to deal with all the specific details of the individual patterns.
- It is meaningful to summarize the data appropriately or look for an apt abstraction of the data.
- Such an abstraction is friendlier to both the human and the machine.
- For the human it is easy for comprehension and for the machine it reduces the computational burden in the form time and space required for processing.
- Generating an abstraction from examples is a well-known paradigm in machine learning.
- Specifically, learning from examples or supervised learning and learning from observations or clustering are the two important machine learning paradigms that are useful here.
- In artificial intelligence, the machine learning activity is enriched with the help of domain knowledge; abstractions in the form of rule-based systems are popular in this context.
- In addition data mining tools are useful when the set of training pat-terns is large.

- So, naturally pattern recognition overlaps with machine learning, arti-ficial intelligence and data mining.

Two popular paradigms for pattern recognition are:

- statistical pattern recognition: In this case, vector-spaces are used to represent patterns and collections of patterns. Vector-space representations are popular in information retrieval, data mining, and statistical machine learning. Abstractions like vectors, graphs, rules or probability distributions are used to represent clusters and classes.

- syntactic pattern recognition: In this case, patterns are viewed as sentences in a formal language like mathematical logic. So, it is useful in describing classes and clusters of well-structured patterns. This paradigm is popular as linguistic or structural pattern recognition.

- Readers interested in some of these applications may refer to popu-lar journals such as Pattern Recognition (www.elsevier.com/locate/pr) and IEEE Transactions on Pattern Analysis and Machine Intelligence (www.computer.org/tpami) for details. Similarly, for specific applica-tion areas like bioinformatics refer to Bioinformatics (http://bioinformatics.oxfordjournals.org/) and for semantic comput-ing refer to International Journal of Semantic Computing (www.worldscinet.com/ijsc/). An excellent introduction to syntactic pattern Recognition is providede by Syntactic Pattern Recognition: An Introduction by RC Gonzalez and MG Thomason, Addision-Wesley, 1978.

Assignment

Solve the following problems:

1. Consider the data, of four adults indicating their health status, shown in the following table. Devise a simple classifier that can properly classify all the four patterns. How is the fifth adult having a weight of 65 KGs classified using the classifier?

Weight of Adults in KGs	Class label
50	Unhelathy
60	Healthy
70	Healthy
80	Unhealthy

2. Consider the data items bought in a supermarket. The features include cost of the item, size of the item, colour of the object and the class label. The data is shown in the following table. Which feature would you like to use for classification? Why?

item no	cost in Rs.	volume in cm^3	colour	Class label
1	10	6	blue	inexpensive
2	15	6	blue	inexpensive
3	25	6	blue	inexpensive
4	150	1000	red	expensive
5	215	100	red	expensive
6	178	120	red	expensive

Different Paradigms for Pattern Recognition

- There are several paradigms in use to solve the pattern recognition problem.
- The two main paradigms are
    1. Statistical Pattern Recognition
    2. Syntactic Pattern Recognition
- Of the two, the statistical pattern recognition has been more popular and received a major attention in the literature.
- The main reason for this is that most of the practical problems in this area have to deal with noisy data and uncertainty and statistics and probability are good tools to deal with such problems.
- On the other hand, formal language theory provides the background for syntactic pattern recognition. Systems based on such linguistic tools, more often than not, are not ideally suited to deal with noisy environments. However, they are powerful in dealing with well-structured domains. Also, recently there is a growing interest in statistical pattern recognition because of the influence of statistical learning theory.
- This naturally prompts us to orient material in this course towards statistical classification and clustering.

Statistical Pattern Recognition

- In statistical pattern recognition, we use vectors to represent patterns and class labels from a label set.
- The abstractions typically deal with probability density/distributions of points in multi-dimensional spaces, trees and graphs, rules, and vec-tors themselves.
- Because of the vector space representation, it is meaningful to talk of subspaces/projections and similarity between points in terms of dis-tance measures.

- There are several soft computing tools associated with this notion. Soft computing techniques are tolerant of imprecision, uncertainty and ap-proximation. These tools include neural networks, fuzzy systems and evolutionary computation.
- For example, vectorial representation of points and classes are also employed by
  - neural networks,
  - fuzzy set and rough set based pattern recognition schemes.
- In pattern recognition, we assign labels to patterns. This is achieved using a set of semantically labelled patterns; such a set is called the training data set. It is obtained in practice based on inputs from ex-perts.
- In Figure 1, there are patterns of Class 'X' and Class '+'.

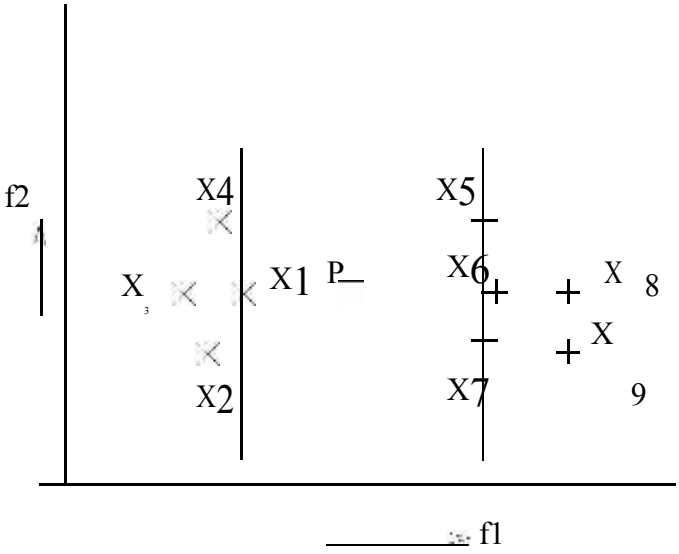

Figure 1: Example set of patterns

- The pattern P is a new sample (test sample) which has to be assigned either to Class 'X' or Class '+'. There are different possibilities; some of them are

  – The nearest neighbour classifier (NNC): Here, P is assigned to the class of its nearest neighbour. Note that pattern X_1 (labelled 'X') is the nearest neighbour of P. So, the test pattern P is assigned the class label 'X'. The nearest neighbour classifier is explained in Module 7.

  – The K-Nearest neighbour classifier (KNNC) is based on the class labels of K nearest neighbours of the test pattern P . Note that patterns X_1 (from class 'X'), X_6 (from class '+') and X_7 (from class '+') are the first three (K=3) neighbours.A majority (2 out of 3) of the neighbours are from class '+'. So, P is assigned the class label '+'. We discuss the KNNC in module 7.

  – Decision stump classifier: In this case, each of the two features is considered for splitting; the one which provides the best separation between the two classes is chosen. The test pattern is classified based on this split. So, in the example, the test pattern P is classified based on whether its first feature (x-coordinate) value is less than A or not. If it is less than A, then the class is 'X', else it is '+'. In Figure 1, P is assigned to class 'X'. A generalization of the decision stump called the decision tree classifier is studied in module 12.

  – Separating line as decision boundary: In Figure 1, the two classes may be characterized in terms of the boundary patterns falling on the support lines. In the example, pattern X_1 (class 'X') falls on one line (say line1) and patterns X_5 and X_7 (of class '+') fall on a parallel line (line2). So, any pattern closer to line 1 is assigned the class label 'X' and similarly patterns closer to line2 are assigned class label '+'. We discuss classifiers based on such linear discriminants in module 12. Neural networks and support vector machines (SVMs) are members of this category. We discuss them in module 13.

  – It is possible to use a combinations of classifiers to classify a test pattern. For example, P could be classified using weighted nearest

neighbours. Suppose such a weighted classifier assigns a weight of 0.4 to the first neighbour (pattern X_1, labelled 'X'), a weight of 0.35 to the second neighbour (pattern X_6 from class '+') and a weight of 0.25 to the third neighbour (pattern X_7 from class '+'). We first add the weights of the neighbours of P coming from the same class. So, the sum of the weights for class 'X', W_X is 0.4 as only the first neighbour is from 'X'. The sum of the weights for class '+', W_+ is 0.6 (0.35 ' + ' 0.25) corresponding the remaining two neighbours (8 and 6) from class '+'. So, P is assigned class label '+'. We discuss combinations of classifiers in module 16.

– In a system that is built to classify humans into tall, medium and short, the abstractions, learnt from examples, facilitate assigning one of these class labels (tall, medium or short) to a newly en-countered human. Here, the class labels are semantic; they convey some meaning.

– In the case of clustering, we can group a collection of unlabelled patterns also; in such a case, the labels assigned to each group of patterns is syntactic, simply the cluster identity.

– Several times, it is possible that there is a large training data which can be directly used for classification. In such a context, clustering can be used to generate abstractions of the data and use these abstractions for classification. For example, sets of patterns corresponding to each of the classes can be clustered to form sub-classes. Each such subclass (cluster) can berepresented by a single prototypical pattern; these representative patterns can be used to build the classifier instead of the entire data set. In Modules 14 and 15, a discussion on some of the popular clustering algorithms is presented.

Importance of Representation

- It is possible to directly use a classification rule without generating any abstraction, for example by using the NNC.
- In such a case, the notion of proximity/similarity (or distance) is used to classify patterns.

- Such a similarity function is computed based on the representation of patterns; the representation scheme plays a crucial role in classification.
- A pattern is represented as a vector of feature values.
- The features which are used to represent patterns are important. We illustrate it with the help of the following example.

Example

Consider the following data where humans are to be categorized into tall and short. The classes are represented using the feature Weight. If a newly

Weight of human (in Kilograms)	Class label
40	tall
50	short
60	tall
70	short

encountered person weighs 46 KGs, then he/she may be assigned the class label short because 46 is closer to 50. However, such an assignment does not appeal to us because we know that weight and the class labels TALL and SHORT do not correlate well; a feature such as Height is more appropriate. Module 2 deals with representation of patterns and classes.

Overview of the course

- Modules 3-6 deal with representation of patterns and classes. Also, proximity between patterns is discussed in these modules.
- Various classifiers are discussed in modules 7 to 13 and module 16.
  - The most popular and simple classifier is based on the NNC. In such a classification scheme, we do not have any training phase. A detailed discussion on nearest neighbor classification is presented in Module 7, 8, and 9.

– It is important to look for theoretical aspects of the limits of clas-sifiers under uncertainty. Bayes classifier characterizes optimality in terms of minimum error-rate classification. It is discussed in Module 10.

– A decision tree is a transparent data structure to deal with clas-sification of patterns employing both numerical and categorical features. We discuss decision tree classifiers in Module 11.

– Using linear decision boundaries in high-dimensional spaces has gained a lot of prominence in the recent past. Support vector machines (SVMs) are built based on this notion. In Module 12 and 13, the role of SVMs in classification is explored.

– It is meaningful to use more than one classifier to arrive at the class label of a new pattern. Such combination of classifiers forms the basis for Module 16.

- In Modules 14 a discussion on some of the popular clustering algorithms is presented.
- There are several challenges faced while clustering large datasets. In module 15 some of these challenges are outlined and algorithms for clustering large datasets are presented.
- Finally we consider an application of document classification and re-trieval in module 17.

Assignment

- Consider a collection of data items bought in a supermarket. The features include cost of the item, size of the item and the class label. The data is shown in the following table. Consider a new item with cost = 34 and volume = 8. How do you classify this item using the NNC? How about KNNC with K = 3?
- Consider the problem of classifying objects into triangles and rectangles. Which paradigm do you use? Provide an appropriate representation.
- Consider a variant of the previous problem where the classes are small circle and big circle. How do you classify such objects?

item no	cost in Rs.	volume in cm^3	Class label
1	10	6	inexpensive
2	15	6	inexpensive
3	25	6	inexpensive
4	50	10	expensive
5	45	10	expensive
6	47	12	expensive

What is a pattern?

- A pattern represents a physical object or an abstract notion. For example, the pattern may represent physical objects like balls, animals or furniture. Abstract notions could be like whether a person will play tennis or not(depending on features like weather etc.).
- It gives the description of the object or the notion.
- The description is given in the form of attributes of the object.
- These are also called the features of the object.

What are classes?

- The patterns belong to two or more classes.
- The task of pattern recognition pertains to finding the class to which a pattern belongs.
- The attributes or features used to represent the patterns should be discriminatory attributes. This means that they help in classifying the patterns.
- The task of finding the discriminatory features is called feature extrac-tion/selection.

What is classification?

- Given a pattern, the task of identifying the class to which the pattern belongs is called classification.
- Generally, a set of patterns is given where the class label of each pattern is known. This is known as the training data.
- The information in the training data should be used to identify the class of the test pattern.

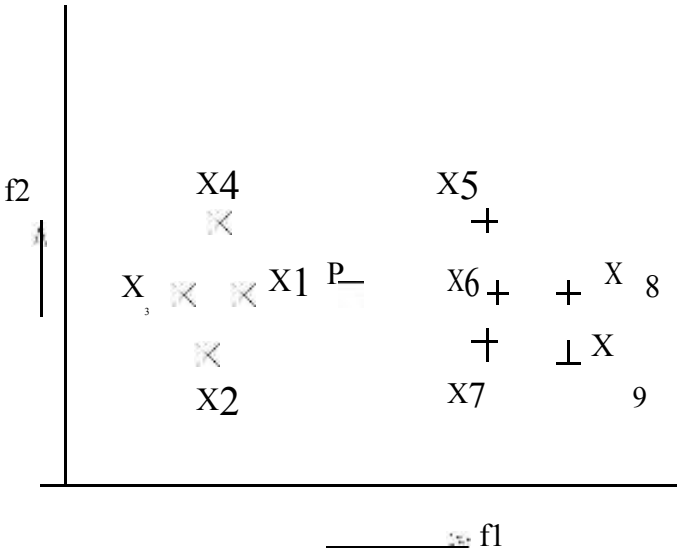

Figure 1: Dataset of two classes

- This type of classification where a training set is used is called super-vised learning. In supervised learning, we can learn about the values of the features for each class from the training set and using this infor-mation, a given pattern is classified.

  Consider the patterns of two classes given in Figure 1. This is the training data.

  Using the training data, we can classify the pattern P. The information of the two classes available in the training data can be used to carry out this classification. There are a number of classifiers which carry out supervised classification like nearest neighbour and related algorithms, Bayes classifier, decision trees, SVM, neural networks, etc which are discussed in later modules.

Representation of patterns

- Patterns can be represented in a number of ways.
- All the ways pertains to giving the values of the features used for that particular pattern.
- For supervised learning, where a training set is given, each pattern in the training set will also have the class of the pattern given.

Representing patterns as vectors

- The most popular method of representing patterns is as vectors.
- Here, the training dataset may be represented as a matrix of size (nxd), where each row corresponds to a pattern and each column represents a feature.
- Each attribute/feature/variable is associated with a domain. A domain is a set of numbers, each number pertains to a value of an attribute for that particular pattern.
- The class label is a dependent attribute which depends on the 'd' in-dependent attributes.

Example The dataset could be as follows :

	f_1	f_2	f_3	f_4	f_5	f_6	Class label
Pattern 1:	1	4	3	6	4	7	1
Pattern 2:	4	7	5	7	4	2	2
Pattern 3:	6	9	7	5	3	1	3
Pattern 4:	7	4	6	2	8	6	1
Pattern 5:	4	7	5	8	2	6	2
Pattern 6:	5	3	7	9	5	3	3
Pattern 7:	8	1	9	4	2	8	3

In this case, n=7 and d=6. As can be seen,each pattern has six attributes(or features). Each attribute in this case is a number between 1 and 9. The last number in each line gives the class of the pattern. In this case, the class of the patterns is either 1, 2 or 3.

2. If the patterns are two- or three-dimensional, they can be plotted.

3. Consider the dataset

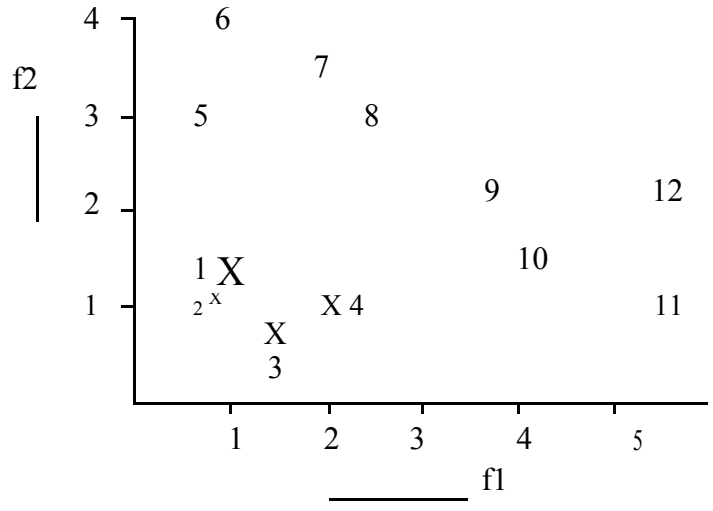

Figure 2: Dataset of three classes

Pattern 1 : (1,1.25,1)
Pattern 2 : (1,1,1)
Pattern 3 : (1.5,0.75,1)
Pattern 4 : (2,1,1)
Pattern 5 : (1,3,2)
Pattern 6 : (1,4,2)
Pattern 7 : (1.5,3.5,2)
Pattern 8 : (2,3,2)
Pattern 9 : (4,2,3)
Pattern 10 : (4.5,1.5,3)
Pattern 11 : (5,1,3)
Pattern 12 : (5,2,3)

Each triplet consists of feature 1, feature 2 and the class label. This is shown in Figure 2.

Representing patterns as strings

X Here each pattern is a string of characters from an alphabet.

X This is generally used to represent gene expressions.

X For example, DNA can be represented as

GTGCATCTGACTCCT...

RNA is expressed as

GUGCAUCUGACUCCU....

This can be translated into protein which would be of the form

VHLTPEEK ....

\endash Each string of characters represents a pattern. Operations like pattern matching or finding the similarity between strings are carried out with these patterns.

\endash More details on proteins and genes can be got from [1].

Representing patterns by using logical operators

\endash Here each pattern is represented by a sentence(well formed formula) in a logic.

\endash An example would be
if (beak(x) = red) and (colour(x) = green) then parrot(x)
This is a rule where the antecedent is a conjunction of primitives and the consequent is the class label.

\endash Another example would be
if (has-trunk(x)) and (colour(x) = black) and (size(x) = large) then elephant(x)

Representing patterns using fuzzy and rough sets

\endash The features in a fuzzy pattern may consist of linguistic values, fuzzy numbers and intervals.

\endash For example, linguistic values can be like tall, medium, short for height which is very subjective and can be modelled by fuzzy membership values.

X A feature in the pattern maybe represented by an interval instead of a single number. This would give a range in which that feature falls. An example of this would be the pattern

(3, small, 6.5, [1, 10])

The above example gives a pattern with 4 features. The 4th feature is in the form of an interval. In this case the feature falls within the range 1 to 10. This is also used when there are missing values. When a particular feature of a pattern is missing, looking at other patterns, we can find a range of values which this feature can take. This can be represented as an interval.

The example pattern given above has the second feature as a linguistic value. The first feature is an integer and the third feature is a real value.

X Rough sets are used to represent classes. So, a class description will consist of an upper approximate set and a lower approximate set. An element y belongs to the lower approximation if the equivalence class to which y belongs is included in the set. On the other hand y belongs to the upper approximation of the set if its equivalence class has a non-empty intersection with the set. The lower approximation consists of objects which are members of the set with full certainty. The upper approximation consists of objects which may possibly belong to the set.

X For example, consider Figure 3. This represents an object whose loca-tion can be found by the grid shown. The object shown completely cov-ers (A3,B2), (A3,B3), (A4,B2) and (A4,B3). The object falls partially in (A2,B1),(A2,B2),(A2,B3), (A2,B4),(A3,B1),(A3,B4),(A4,B1),(A4,B4), (A5,B2), and (A5,B3). The pattern can be represented as a rough set where the first four values of the grid gives the lower approximation and the rest of the values of the grid listed above form the upper ap-proximation.

X Not just the features, each pattern can have grades of membership to every class instead of belonging to one class. In other words, each

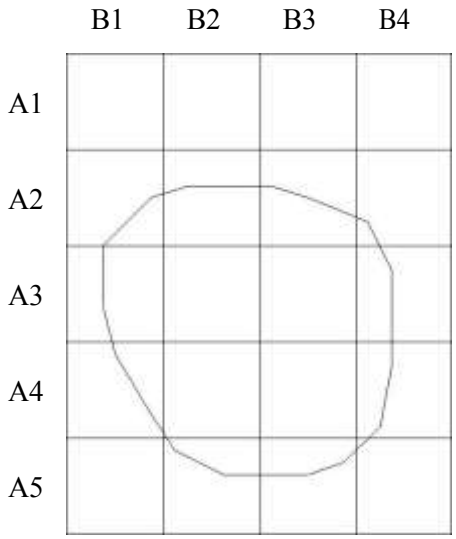

Figure 3: Representation of an object

pattern has a fuzzy label which consists of c values in [0,1] where each component gives the grade of membership of the pattern to one class. Here c gives the number of classes. For example, consider a collection of documents. It is possible that each of the documents may be associated with more than one category. A paragraph in a document, for instance, may be associated with sport and another with politics.

3. The classes can also be fuzzy. One example of this would be to have linguistic values for classes. The classes for a set of patterns can be small and big. These classes are fuzzy in nature as the perception of small and big is different for different people.

Representing a Dataset as a Tree

- Each attribute in a pattern can be represented as an edge in a tree.
- Examples of this would be the Frequent Pattern tree (FP tree) and the Pattern Count tree (PC tree). Here the attributes are assumed to be categorical, that is, the domain is a finite collection of values. The FP-tree has been explained in this lesson.
- Each pattern may be represented as a node in a tree. Some of the trees used are Minimum Spanning Tree (MST), Delauney Tree (DT), R-tree and the K-D tree. The MST and the K-D tree has been described in this lesson.

Minimum Spanning Tree

- The complete set of patterns can be represented using the minimum spanning tree.
- Consider a graph where the cost of each edge is known.
- A spanning tree is a tree which consists of a subset of all the edges in the graph which spans all the nodes of the graph.
- The spanning tree for a graph is not unique and a number of spanning tree can be found.
- The minimum spanning tree is the spanning tree which gives the total minimum cost.
- It is possible to have multiple minimum spanning trees for some datasets.
- One method of obtaining the MST is by using the Prim's algorithm.
- The algorithm is as follows :

  Choose one vertex v as the starting point.

  Find the shortest outgoing edge from v and add it to the MST.

  Then keep adding the shortest edge from the vertices in the MST so that the edge added does not form a cycle with the existing edges.

- Stop the algorithm when there are n-1 edges where n is the number of vertices.

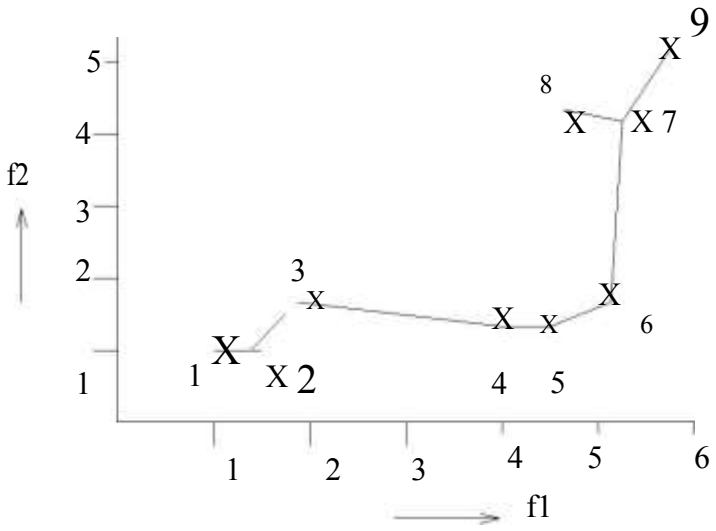

Figure 1: Minimum Spanning Tree

- Figure 1 shows a pattern set of 9 points. The minimum spanning tree is shown for the 9 points.

- MST can be used for clustering. Find the edge with the maximum distance in the MST and delete that edge. Here 3-4 would be deleted. Then find the edge with the maximum distance in the resulting tree and delete it. Now 6-7 would be deleted. If we stop deleting edges at this point, we get three separate components. Each of these would be one cluster. If we want four clusters, we can again find the edge with the maximum distance and delete it. Here, through appropriate deletion, we have divided the patterns into three clusters. Cluster 1 having patterns 1, 2 and 3, Cluster 2 having 4, 5 and 6 and Cluster 3 having 7, 8 and 9.

- An important property is that each pattern is a numerical vector and an edge is characterized by the distance between two such vectors.

Frequent Pattern Trees(FP trees)

- FP tree is used to represent the patterns in a transaction database.
- An example of a transaction database is the collection of transactions at a supermarket.
- The items bought by each customer at the supermarket is a transaction in the database. This is a pattern in the database. It can be seen that the length of each pattern in this database is not of the same size.
- In this database, it is necessary to find the association between items in the database. In other words, the presence of some item may imply the presence of some other items in the transactions.
- It is necessary to first find the frequent items in the database. Then the FP-tree can be constructed to help in efficiently mine the frequent itemsets in the database.
- The FP tree is a compressed tree structure where only items which occur more frequently are considered. The number of times an item occurs in the transaction database is the support of that item. So when we say that only frequently occurring items are included, it means that items which have a support below a threshold are left out.
- The items in the transactions are then reordered so that more frequent items occur earlier and the less frequent items occur later in the trans-action.
- The first transaction is drawn with the items as nodes and links con-necting the nodes. Each item has a count of one.
- The second transaction is then drawn. If it has a overlap with the existing link then it is not redrawn but the count is increased of those items.
- When the item does not correspond to any of the existing, those links are drawn.
- Let us consider an example.

Consider transactions in a supermarket. For simplicity, let us consider only 15 items A to O. Let the items picked up by four customers be

Transaction 1 : a,b,c,g,e,d,i,h
Transaction 2 : b,c,d,e,f,o,k,m
Transaction 3 : e,j,n,o,c,n,g,i
Transaction 4 : b,f,o,m,n,k,c,d,e,g,i,j,l

The frequency of each item will be a:1, b:3, c: 4, d: 3, e: 4,f: 1, g: 3,

- 1, i: 2, j: 2, k: 2, l: 1, m: 2, n: 2, o: 3 . Here a:1 means A occurs in one transaction. Similarly, d:3 stands for D occurring in 3 transactions, namely transaction 1, transaction 2 and transaction 4.

If only items with frequency 3 and above are considered, then the items being considered are d:3, e:4, c:4, g:3, b:3 and o:3. Then the transac-tions become

Transaction 1 : b,c,g,e,d
Transaction 2 : b,c,d,e,o
Transaction 3 : e,c,o,g
Transaction 4 : b,c,o,d,e,g

Ordering the transactions so that items which are more frequent occur first we get

Transaction 1 : e,c,b,d,g
Transaction 2 : e,c,b,d,o
Transaction 3 : e,c,g,o
Transaction 4 : e,c,b,d,g,o

Items which have the same frequency are ordered arbitrarily. The above items are then used to construct the FP-tree.

Figure 2 shows the FP-tree for the example above.

- All transactions are drawn starting from the root. The largest trans-action is taken first. A link is drawn from the root to e, then from e

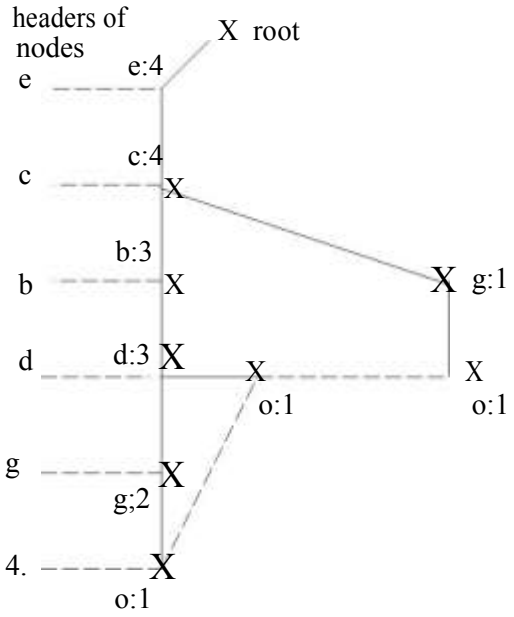

Figure 2: Frequent-Pattern Tree(FP-tree)

to c, c to b and so on, till o is reached. For each node, a count is kept of how many times that node is utilised. Now each of the nodes will get a count of 1. So we get e:1, c:1, h:1, d:1, g:1 and o:1. Next, taking transaction 2, the first element is e. We need to see if there is already an existing link from the root. In this case, since it exists, the count of e is incremented by 1. So we get e:2. Next from e, we see if there is already an existing link to the next element of transaction 2 which is c. Since it exists, we only increment c to get c:2. Similarly we increment b,d and g to get b:2, d:2 and g:2. We next consider transaction 3. Using the same procedure, we get e:3, c:3, b:3, d:3 and then we draw a new link from d to o and the count of o will be o:1. Then we consider transaction 4. We increment e and c to get e:4 and c:4. We then draw a new link from c to g and set the count of g to 1 giving g:1. From g, we again draw a new link to o giving o:1. This completes the FP-tree.

- An important observation is that any categorical pattern may be viewed as a transaction. Each feature of the pattern is equivalent to an item in a transaction. For example, consider Figure 3. Let each square be

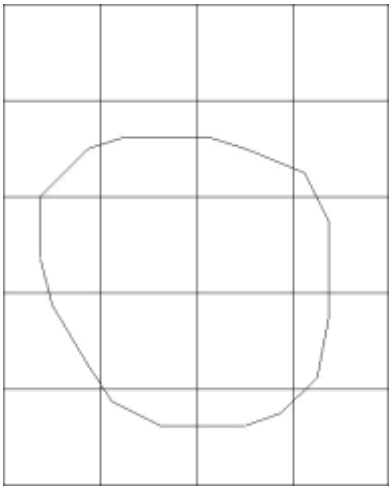

Figure 3: Representation of an object

represented by a letter as shown in the following table. Then the object

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

in Figure 3 can be represented as

(e f g h i l m p r s)

Another object can be represented by another combination of the En-glish alphabet.

We therefore get patterns which are of diff erent lengths and each letter in a pattern is equivalent to an item in a transaction. Thus it can be seen that, each pattern in this case can be viewed as a transaction.

\endash Further an association rule may be viewed as a conjunctive logical formula. An association rule will be of the form

If (item1) and (item2) and (item3) then (item3)

In the case of the supermarket where each transaction gives the items bought by one customer, the association rule will be of the form

If (toothpaste) and (soap) and (powder) then (milk)

In the example shown in Figure 3 the association rule will be of the form

If (a) and (q) and (e) then (d)

It can, therefore, be seen that the association rule is a conjunctive logical formula.

K-d Tree

\endash K-d tree is another name for the k-dimensional tree.

\endash It is a data structure for storing patterns in k-dimensional space by partitioning the space.

\endash The k-d tree is a binary tree structure for storing and performing op-erations on data containing k-dimensional keys.

\endash The K-d tree is used to represent a set of patterns which have been partitioned. Each region of the K-d tree represents a subset of the patterns.

\endash Consider the set of patterns represented by Figure 4.

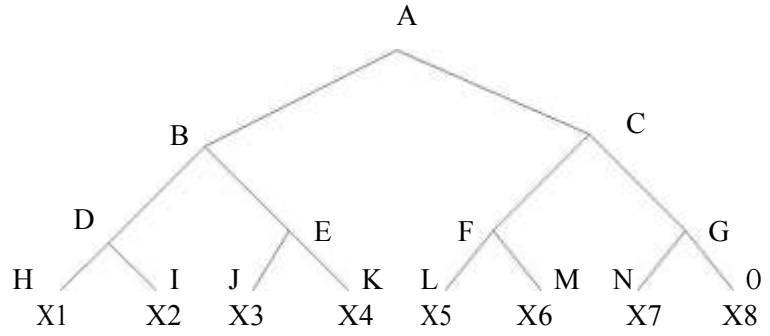

Figure 4: Partition of patterns for K-d tree

\endash A represents the complete set of patterns. At each point, the set of patterns is partitioned into two blocks giving rise to a binary tree.

\endash A is partitioned into B and C. B is partitioned into D and E. C is partitioned into F and G. D is partitioned into H and I . E is parti-tioned into J and K. F is partitioned into L and M . G is partitioned into N and O.

\endash Figure 5 gives the K-d tree for the above data.

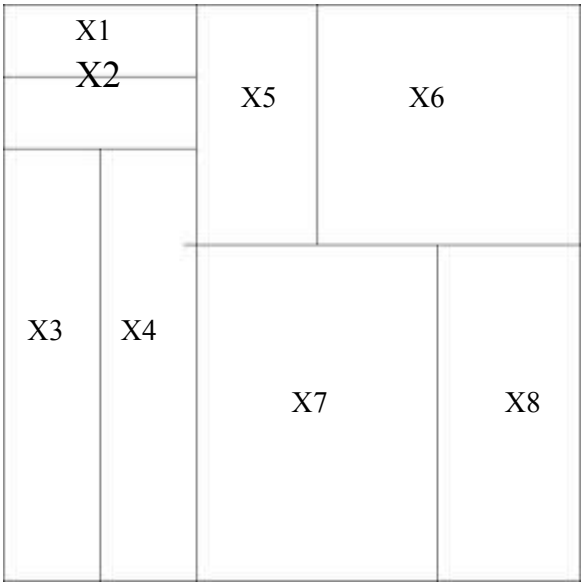

Figure 5: Partition of patterns for K-d tree

Assignment

X Consider the data shown in the following table where each pattern is represented by 4 binary features F_1, F_2, F_3, AND F_4. Suggest:

a vector-space representation

a logic based representation

Which of the above two representations is suited well for the data? Why?

pattern no	F_1	F_2	F_3	F_4	Class label
1	1	0	1	0	1
2	0	1	0	1	2
3	1	0	1	0	1
4	1	0	1	0	1
5	0	1	0	1	2

4. Consider the table below of 4 transactions in a supermarket where we consider only 4 items, I_1, I_2, I_3, AND I_4. There are two classes. Obtain the frequent patterns by considering items with frequency 2 and above. Is it possible to use the resultant FP-Tree in classification?

Transaction no	I_1	I_2	I_3	I_4	Class label
1	1	0	1	0	inexpensive
2	0	1	0	1	expensive
3	1	0	1	0	inexpensive
4	0	1	0	1	expensive

X Consider a two-class problem where the classes are labeled pen drive and laptop. Suggest a set of features that could be used to discriminate between these two classes of objects. What is the corresponding vector-space representation?

X For what kind of datasets does the FP-Tree be large in size? Provide such a dataset.

X Given a dataset, will the corresponding MST be unique? Why is it so? If not provide a counterexample.

X Consider a dataset in a d-dimensional space. What is the worst-case size of a K-d tree built on such a dataset?

Proximity measures

- Whenever classification is carried out, it is done according to some similarity of the test pattern to the training patterns.
- For clustering also, patterns which are similar to each other are to be put into the same cluster while patterns which are dissimilar are to be put into different clusters.
- To determine this similarity/dissimilarity, proximity measures are used.
- Some of the proximity measures are metrics while some are not.
- The distance between two patterns is used as a proximity measure. If this distance is smaller, then the two patterns are more similar.
- For the distance measure to be metric, the following properties should hold :

Positive Reflexivity : $d(X,X) = 0 \ \forall \ X$

Symmetry : $d(X,Y) = d(Y,X) \ \forall X, Y$ and

Triangular Inequality $d(X,Z) + d(Z,Y) \geq d(X,Y) \ \forall X, Y, Z$

where $d(X,Y)$ gives the distance between X and Y.

Distance measure

- These measures find the distance between points in a d-dimensional space, where each pattern is represented as a point in the d-space.
- The distance is inversely proportional to the similarity. If $d(X,Y)$ gives the distance between X and Y, and $s(X,Y)$ gives the similarity between X and Y, then

$$d(X, Y) \propto \frac{1}{s(X, Y)}$$

- The Euclidean distance is the most popular distance measure. If we have two patterns X and Y, then the euclidean distance will be

$$d_2(X, Y) = \sqrt{\sum_{K=1}^{D} (x_K - y_K)^2}$$

- The generalization of this equation gives the Minkowski distance which is

$$d_M(X, Y) = \left(\sum_{K=1}^{D} |x_K - y_K|^M\right)^{\frac{1}{M}}$$

where $m=1,2,\ldots,\infty$

- Depending on the value of m, we get different distance measures.

Euclidean distance(ED)

- When $m=2$ is substituted in Minkowski metric, we get the euclidean distance.
- This is also called the L_2 norm.
- This can be written as

$$d_2(X, Y) = \sqrt{\sum_{K=1}^{D} (x_K - y_K)^2}$$

- Euclidean distance is popularly used in pattern recognition. This is because Euclidean distance is easy for human comprehension and ro-tation and translation invariant.

Manhattan distance

- When $m=1$ is substituted in the equation, we get the Manhattan or the city block distance.

- This is also called the L_1 norm.
- This can be written as

$$d(X, Y) = \left(\sum_{K=1}^{D} | x_K - y_K |\right)$$

L_∞ norm or Max norm

- Similarly, the L_∞ norm is

$L_\infty = \max | x_K - y_K |$ f or k = 1, 2, ...d

where d is the number of dimensions or features.

Weighted distance measure

- When some features are more significant than others, the **weighted distance measure** is used.
- The weighted distance is as follows :

$$d(X, Y) = \left(\sum_{K=1}^{D} w_K (x_K - y_K)^M\right)^{\frac{1}{M}}$$

5. The weight for each feature depends on its importance.
6. The greater the significance of the feature, the larger the weight.
7. The weighted distance measure using Euclidean distance(m=2) would be

$$d_2(X, Y) = w_1 * (x_1 - y_1)^2 + w_2 * (x_2 - y_2)^2 + \ldots + w_D * (x_D - y_D)^2$$

- A generalized version of the weighted distance is the squared Maha-lanobis distance(MD). Let a class be represented by $N(\mu, \Sigma)$, where μ is the mean and Σ is the covariance matrix, then the squared Maha-lanobis distance between a pattern X and the class is

  $$(X - \mu)^T \Sigma^{-1} (X - \mu)$$

  If $\Sigma = I$ (identity matrix), then MD=ED

- As an example, consider If X = (5,3,7,5,1) and Y = (6,5,2,9,5), then

Manhattan distance or L_1 norm

$$d(X, Y) = |5 - 6| + |3 - 5| + |7 - 2| + |5 - 9| + |1 - 5| = 16$$

Euclidean distance or L_2 norm

$$d_2(X, Y) = \sqrt{(5 - 6)^2 + (3 - 5)^2 + (7 - 2)^2 + (5 - 9)^2 + (1 - 5)^2} = 7.87$$

L_∞ norm

$$d_\infty(X, Y) = \max(1, 2, 5, 4, 4) = 5$$

Weighted euclidean distance

Let the weight for the feature w_1 be 0.5

Let the weight for the feature w_2 be 0.1
Let the weight for the feature w_3 be 0.3
Let the weight for the feature w_4 be 0.7
Let the weight for the feature w_5 be 0.4

Then the weighted euclidean distance will be

$$d(X, Y) = \sqrt{(0.5 * (5-6)^2) + (0.1 * (3-5)^2) + (0.3 * (7-2)^2) + (0.7 * (5-9)^2) + (0.4 * (1-5)^2)}$$
$$= 5.1$$

There are other dissimilarity measures besides the metric distance mea-sure described above.

Many of them are useful for image data or string data.

Many of these measures do not exhibit all the properties of a metric and are therefore non-metric dissimilarity measures.

One example of a non-metric measure is the squared Euclidean dis-tance.

k-Median Distance

The k-Median distance is a non-metric distance function.

This finds the distance between two vectors.

The difference between the values for each element of the vector is found.

Putting this together, we get the difference vector.

If the values in the difference vector are put in non-decreasing order, the kTH value gives the k-Median distance.

If the two vectors are

{p1, p2, ...pN} and {q1, q2, ..., qN},

Then the difference vector will be

D = {| p1 − q1 |, | p2 − q2 |, ..., | pN − qN |}

Then d(P,Q) = k-median (D)

As an example, let us take

P = (10, 56, 23, 32, 78, 99, 65, 83, 1 , 53) and Q = (34, 87, 94, 21, 49, 81, 23, 77, 34, 61)

Then the difference vector will be

=(| 10 − 34 |, | 56 − 87 |, | 23 − 94 |, | 32 − 21 |, | 78 − 49 |, | 99 − 81 |, | 65 − 23 |, | 83 − 77 |, | 1 − 34 |, | 53 − 61 |)
= (24, 31, 71, 11, 29, 18, 42, 6, 33, 8)

If D is arranged in non-decreasing order we get

D' = (6, 8, 11, 24, 29, 21, 31, 33, 42, 71)

If k = 5, then d(P,Q) = 29

Feature Extraction

Feature extraction refers to the process of identifying and combining certain features of patterns. Given a collection F of features, the ex-tracted features are linear/non-linear combinations of elements of F .

This is a pre-processing step of pattern recognition.

Before pattern classification is carried, it is necessary to decide what attributes of the patterns are to be measured and recorded.

The features chosen should be discriminating features of the pattern.

It is an important stage of pre-processing, as the feature extraction influences the quality of the pattern classification carried out.

Two methods explained in this module are :

Fisher's Linear Discriminant :

This is used in supervised clas-sification. Supervised classification refers to classification being carried out where labeled training examples are available to learn the classifier.

Principal Component Analysis : This is an unsupervised learning activity. Unsupervised learning activity uses the underlying dis-tribution of the patterns to carry out classification. No labeled examples are available. A typical unsupervised learning activity is clustering.

Fisher's Linear Discriminant

Fisher's Linear Discriminant is used to map a d-dimensional data to one dimensional data using a projection vector V such that it maps a vector X to a scalar V T X ; note that V T X is a scalar.

Classification is then carried out in the one dimensional space.

Let us consider a two-class problem.

If the mean of class 1 is m1 and the mean of class 2 is m2, and if v1 is proportional to the variance of class 1 patterns and v2 is proportional to the variance of class 2 patterns, then the Fisher's criterion is

$$J(V) = \frac{|M1 - M2|2}{V12 + V22}$$

J (V) is the criterion function to be maximized and it is necessary to find V such that that J (V) is maximized.

• If there are n patterns, then the mean of the patterns is

$$= \frac{1}{n} \sum_{K=1}^{n} XK$$

If there are k classes, then the mean of the patterns of Class i is

$$mI = \frac{1}{nI} \sum_{XI \in C LASS I} XI$$

The between class scatter matrix is

$$= \sum_{I=1}^{K} nI\,(mI - m)(mI - m)T$$

The within class scatter matrix is

$$= \sum_{I=1}^{K} \sum_{XI \in C LASS I} (XI - mI)(XI - mI)T$$

The criterion function to be maximized is

$$(V) = \frac{J\ V\ \ V)}{V\ \ \ V)}$$

The problem of maximizing J (V) can be written as the following con-strained optimization problem.

min −12 V T ∗ B ∗ V

s.t. V T ∗ W ∗ V = 1

which corresponds to the Lagrangian

L=−12 V T ∗ B ∗ V + 12 ∗ λ ∗ (V T ∗ W ∗ V − 1)

From the KKT conditions1

we can see that the following equation has to hold.

B ∗ V = λ ∗ W ∗ V

This means that

W −1 ∗ B ∗ V = λ ∗ V

Let v1, v2, ..., vD gives the generalized eigenvectors of B and W , which gives a projection space of dimension d.

We need to get a projection space of dimension m < d giving the eigen-vectors

VM = (v1, v2, ..., vM)

The projection of vector XI into a subspace of dimension m is

= WDT X

Principal Component Analysis(PCA)

• Principal Component Analysis is a procedure by which the number of variables are reduced.

• The attempt is to find a smaller number of variables which are uncor-related.

• The first principal component is the most important and accounts for as much of the variability as possible. The second principal comes next etc.

• The number of variables are reduced by projecting the data in the direction of maximum variance.

• The method involves finding the eigenvectors and the corresponding eigenvalues of the covariance matrix.

• If the eigenvectors are ordered in descending order of the eigenvalues, the first eigenvector gives the direction of the largest variance of the data.

• By excluding the directions giving very low eigenvalues, we can reduce the number of variables being considered.

• If X = X1, X2 , ..., Xn is the set of n patterns of dimension d, the sample mean of the data set is

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

The sample covariance matrix is

• = (X − m) (X − m)

•

• C is a symmetric matrix. The orthogonal basis can be calculated by finding the eigenvalues and eigenvectors.

• The eigenvectors gi and the corresponding eigenvalues λi are solutions of the equation

$C * gi = \lambda i * gi, i = 1, ..., d$

• The eigenvector corresponding to the largest eigenvalue gives the di-rection of the largest variance of the data.

• By ordering the eigenvectors according to the eigenvalues, the directions along which there is maximum variance can be found.

• If E is the matrix consisting of eigenvectors as row vectors, we can transform the data • to get Y .

= E(X − m)

• The original data X can be got from Y as follows :

X = EtY + m

• Instead of using all d eigenvectors, the data can be represented by using the first k eigenvectors where k < d.

• If only the first k eigenvectors are used represented by EK , then

= EK (X − m)

and

X ′ = EKt Y + m

- This reconstructed X' will not exactly match the original data X. This shows that some information is lost when only the first k eigenvectors are considered.
- By choosing the eigenvectors having largest eigenvalues, as little infor-mation as possible is lost.
- At the same time, the dimensionality of the data is reduced so as to simplify the representation.
- As an example, let us consider two patterns of class 1. The patterns are (2,2) and (3,1). Let us consider two patterns of class 2. The patterns are (5,4) and (7,4).

The mean of the patterns will be

- #

4.25
mean =
2.75

The covariance matrix is

" #

5.25 2.96
C =
2.96 2.25

The eigenvalues of C are

$\lambda 1 = 6.98$ and $\lambda 2 = 0.52$

The first eigenvalue is very much larger than the second eigenvalue.

The eigenvector corresponding to this is

- #

0.8633
eigen1 =
0.5046

The pattern (2,2) gets transformed to

- #

$$\begin{bmatrix} 0.8633 \\ 0.5045 \end{bmatrix} * \begin{bmatrix} -2.25 \\ -0.75 \end{bmatrix} = -2.32$$

Similarly, the patterns (3,1), (5,4) and (7,4) get transformed to -1.96, 1.71 and 2.57 .

When we try to get the original data from the transformed, some in-formation gets lost. After transformation, pattern (2,2) becomes,

8. #

4.25

0.8633

h0.5046 i * (-2.32) + 2.75 =

X #

4.25

h−2.00 −1.17 i + 2.75 =

2.25

" 1.58 #

showing that there is some loss in information.

Feature Selection

- The features which are chosen to represent a set of patterns may be large or redundant in some way. In such cases, it may be good to reduce the features. This process is called feature selection or dimensionality reduction.
- If the dimensionality is large, it is necessary to have a large training set to get a good classification accuracy.
- This is due to the curse of dimensionality or peaking phenomenon.

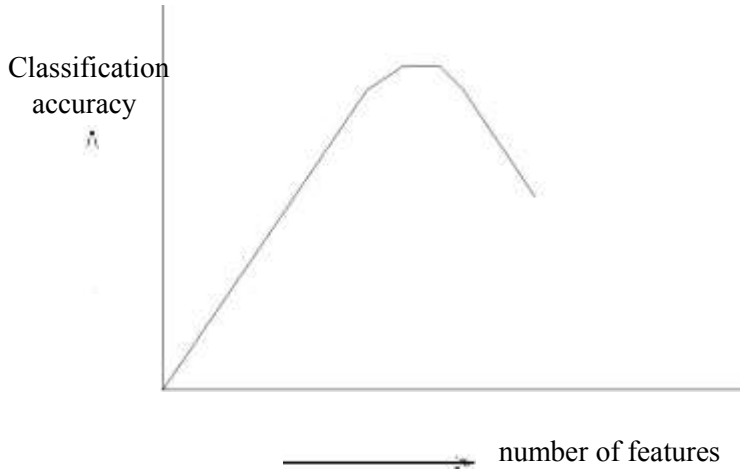

Figure 1: Illustration of curse of dimensionality

- This is illustrated in Figure 1. For a particular number of training patterns, up to a certain point, the increase in dimensionality leads to improvement in the classification accuracy. But after that, there is a reduction in the classification accuracy.
- Besides, reducing the dimensionality, reduces the time required to find the class of a test pattern when using a classifier.
- This may lead to a reduction in the classification accuracy. If it is not very much, then it might be worth doing feature selection which will lead to reduction in time and space requirement of classifiers.

- If the features removed are not discriminatory features, it ensures that the classification accuracy does not decrease.

Methods of Feature Selection

- All the methods of feature selection involve searching for the best subset of features.
- Generally after removing a feature or adding a feature, the resulting set of features is evaluated and the decision is taken as to whether to keep that feature or not.
- The evaluation function is called the criterion function J.
- The evaluation usually employed is the classification accuracy obtained on a validation set of patterns.
- This can also be the classification error. If E is the % error, then the criterion function J = 100 - E.
- It could also be based on classification accuracy and on the number of features used which are combined together in some way. If two subsets have the same classification accuracy then the subset which has a smaller number of features has a larger evaluation than the one with the larger number of features.

Exhaustive search

- In this method, all combinations of features are tried out and the cri-terion function J calculated.
- The combination of features which gives the highest value of J is the set of features selected.
- If the number of features is large, this method becomes prohibitively time consuming.
- If the number of features is d and the number of required features is k,
  then the number of different combinations to be tried out to find the
  best set of features will be $\binom{d}{k}$!.

- For every combination of features X, the criterion function J(X) has to be calculated.
- The combination of features which gives the best criterion value is chosen as the feature subset selected.
- This method is a brute force method and hence gives the optimal so-lution.
- Because of the time complexity for large feature sets, this method is impractical when the number of features goes up.

Branch and Bound Search

- The branch and bound method forms a tree where the root pertains to choosing all features.
- The children of this node pertains to the combinations of features where one feature has been removed. From each of these children, we get the nodes where another feature has been removed and so on.
- If there are d features and it is necessary to retain f features, then the height of the tree will be $d - f$.
- A leaf node represents a combination of features.
- Once a leaf node is reached, it can be evaluated and its criterion func-tion J found.
- This value is stored as the bound b.
- While evaluating the future branches, at anytime if the criterion value goes below the bound b, that branch is not further expanded.
- When another leaf node is reached, if its J value is larger than b, b is updated and that combination of features is stored as the best so far.
- The assumption made here is monotonicity. In the tree generated, it is assumed that the parent node has a higher criterion function value as compared to its children. This means that a feature set has a larger J value as compared to any of its subsets.

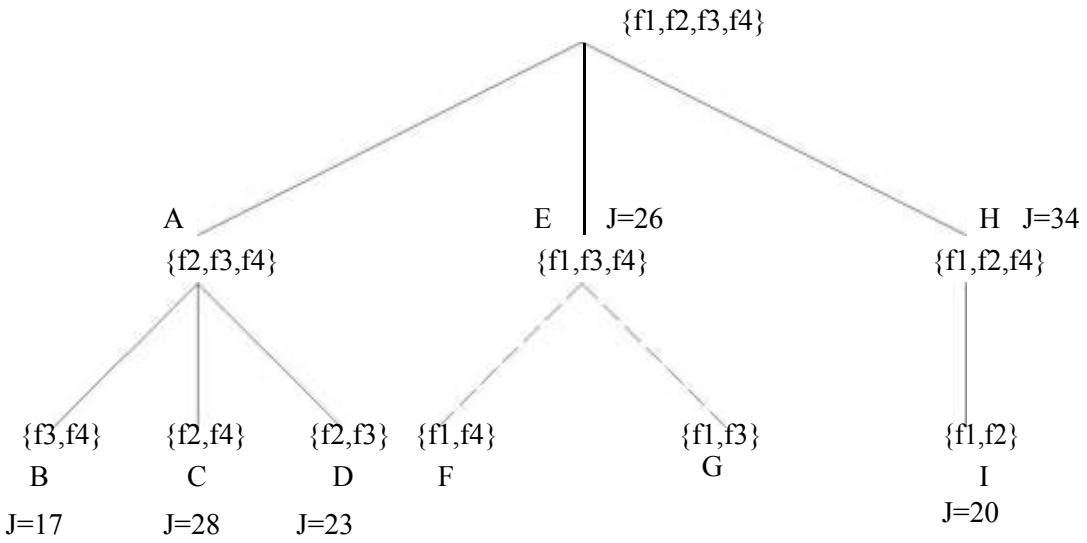

Figure 2: The solution tree for branch and bound with d=4 and f=2

- The tree generated by the branch and bound algorithm for a particular problem is shown in Figure 2.
- When the left-most node B is reached, which corresponds to the feature subset {f3,f4}, let the evaluation of the node J be 17.
- At this stage, since this is a leaf node, the bound b = 17.
- The next node generated is C having a J value of 28.
- Since this is greater than the value of b, the bound b is updated to 28 and the best subset so far is {f2,f4}.
- The next node generated is D corresponding to the feature subset {f2,f3}.
- This has a J value of 23 which is smaller than b, so b remains unchanged.
- The next node generated is E which is found to have a J value of 26.
- Since the J value of this node is less than b, this branch is not expanded any further.

- This means that the branches F and G are not generated.
- The next node to be generated is H which has a J value of 34.
- Since 34 is greater than b, this node is expanded to give I.
- The J value of I is 20 which is lower than the bound.
- Now the entire tree is completed and the node with the lower bound b as the criterion function is the best subset which is $\{f_2, f_4\}$. This is based on the fact that we are selecting 2 out of 4 features.

Relaxed Branch and Bound

- In the Relaxed Branch and Bound, the principle of monotonicity is used but is relaxed.
- A margin m is used and if the J value of any node being considered becomes less than the bound b by a value less m, it is still considered for expansion and one of its children can become the optimal subset.
- Consider Figure 2. Let the margin m = 5.
- When node E is being considered, the bound b = 28. At E, J = 26, which is lower than b. This is not expanded further, in the branch and bound method. But in the relaxed branch and bound, since the difference between its criterion 26 and the bound 28 is less than the margin, it is expanded further. Now if F has a criterion of 29, then the subset chosen would be $\{f_1, f_4\}$ instead of $\{f_2, f_4\}$.
- In this method, the principle of monotonicity is relaxed.

Selecting Best Individual Features

9. In this method, each individual feature is evaluated individually.

10. If f features are to be chosen from d features, after evaluation, the best f features are chosen.

11. This method is the simplest method for finding a subset of features and is computationally fast since different combinations of features need not be evaluated.

- However, the subset of features chosen is not likely to give good results since the dependencies between features is not taken into account.
- It is necessary to see how the combination of features perform.
- Consider the example where there are 6 features. It is necessary to choose two features out of the six features. After evaluation of individ-ual features if the evaluation is as follows :

$f_1 = 45$	$f_2 = 56$
$f_3 = 60$	$f_4 = 22$
$f_5 = 65$	$f_6 = 34$

The two features giving the highest criterion function value are features f_5 and f_3. Therefore the subset to be chosen is $\{f_3, f_5\}$,

Sequential Feature Selection

Sequential Methods

In these methods, features are either sequentially added or deleted from feature subsets, at each step evaluating the feature subset chosen.

These methods either start from an empty set and expand the feature subset at every iteration, or start from the set containing all features and contract the feature subset at each iteration.

At each step, the significant feature is determined to add to the subset or to delete from the subset.

Unlike the exhaustive search and branch and bound method, this method is not guaranteed to produce the optimal solution since all subsets are not examined.

These methods suffer from the nesting effect.

Sequential Forward Selection(SFS)

This method adds one feature at a time.

At every iteration, one feature is added. The most significant feature is selected at each iteration.

This is the bottom-up approach since the search starts with an empty set and keeps adding features.

Once a feature is chosen, it cannot be deleted. This is called the nest-ing eff ect.

Consider after the kth iteration, there will be k features in the subset.

At this point, the significance of a feature i (which is not among the k features already selected) Si is

$$Si = J(Fk \quad fi) - J(Fk)$$

where Fk is the set of k features already selected and fi is the new feature being selected.

At every stage, the most significant feature is selected to be added to the subset.

This is the feature j which has the maximum value of significance.

To illustrate this method, let us take a set of patterns with 4 features and two classes. The patterns in the training set are given below :

Training data Class 1 :

1-4,5-8,1-4,9-12 Pattern 1 : (5,7,3,12); Pattern 2 : (1,4,2,10)

Pattern 3 : (3,8,1,8); Pattern 4 : (2,4,1,11)

2 2.24 3 2.83 Class 2 :

5-8, 1-4,9-12,5-8 Pattern 5 : (6,2,10,5); Pattern 6 : (4,3,8,5)

Pattern 7 : (7,1,12,9); Pattern 8 : (8,4,11,9)

Let us also have a validation set as follows :

Validation data Class 1 :

Pattern 1 : (3,7,3,9); Pattern 2 : (4,6,1,12)

Pattern 3 : (2,5,3,10); Pattern 4 : (3,7,1,12) Class 2 :

Pattern 5 : (7,2,12,6) ; Pattern 6 : (5,1,10,8)

Pattern 7 : (6,3,11,6) ; Pattern 8 : (7,4,9,7)

Now we have a feature set which is a null set. Let F be the feature set.

Now we consider one feature at a time to find the most significant feature.

Let us classify the training set using the validation set using the first feature only. Here 6 out of 8 patterns are classified correctly.

Let us consider only feature 2. Here also 6 out of 8 patterns are classi-fied correctly.

Let us now consider only feature 3. Here 3 out of 8 patterns are clas-sified correctly.

Considering feature 4 only, 5 out of 8 patterns are classified correctly.

Since feature 3 gives the maximum number of correct classifications, feature 3 is taken as the first feature selected.

We then have F = {3}

We now have to try out the feature sets {3, 1}, {3, 2} and {3, 4} and find the feature set giving the most number of correct classifications.

Then keeping these two features, the third feature is found and so on.

Sequential Backward Selection(SBS)

- This method starts with all the features as the feature subset to be considered.
- At every iteration, one feature is discarded.
- The feature to be discarded is the least significant feature.
- This is the top-down approach since it starts with the complete set of features and keeps discarding features at each iteration.
- Once a feature is discarded, it cannot be selected again. Hence this method also suffers from the nesting effect.
- At every stage, the feature j which has the minimum value of significance is selected.
- To illustrate this, consider the example given in the previous section. We start with the entire feature set. So the feature set considered is F = {1, 2, 3, 4}. From this feature set, one feature is removed at a time to get {2,3,4},{1,3,4} and {1,2,4}. Each of these feature sets is used to classify the training patterns and the feature set which misclassifies the most number of patterns is considered and the concerned feature is removed. This feature is the least significant feature. We now have three features in the feature subset. From this the least significant feature is removed to get a feature set of 2 features. If we are interested in retaining 2 out of features, we stop at this point.

Plus l-take away r Selection

- This method was introduced to take care of the nesting effect which has been described in the previous methods.
- In this method, at every iteration, l features are added and r features are deleted.

- The l features to be added are the most significant features like in forward selection.

- The r features deleted are the least significant features like in the back-ward selection.

- Choosing the values of l and r is a difficult task and may have to be done by trial and error.

- To take care of the problem of what values to use for l and r, the floating search methods were introduced.

12. For example, if there are 10 features in the patterns, then if l is 3 and r is 1, then starting with the feature set F = φ, first 3 features are added. For each feature added, the most significant feature is chosen. At the end of this, F will have three features. The next step is to exclude one of the features chosen. The least significant feature of the three features is removed and at this stage F has two features. Again three features are added, choosing the most significant features. Then again one feature is removed and so on. Choosing the values of l and r arbitrary may not give the optimal feature subset. Choosing l and r to get the optimal subset is a difficult problem.

Sequential Floating Search

13. In this method, there is no necessity to specify the l and r values.

14. The number of forward steps and backward steps to be taken is decided dynamically so that the criterion function is maximized.

15. The size of the subset of features keeps on increasing and decreasing till the search is stopped. Hence it is called the 'floating' search method.

Sequential Floating Forward Search(SFFS)

This method starts with an empty set. To this set is added the most significant feature. At every stage, after adding a feature, the least significant feature is removed conditionally. This means that if the feature subset after removing the feature is not the best subset found so far of that size, the feature is added back again.

The algorithm is given below.

k=0
Add the most significant feature to the subset and set k=k+1

Conditionally remove the last significant feature. If the subset of size k-1 resulting from the removal is the best subset of size k-1, then let k=k-1 and go to Step 3 else return the feature removed and go to Step 2.

If the total number of features d is large and the size of the subset to be found f is small, it is good to use the SFFS method since it is started from the empty set.

As an example, consider a dataset with 10 features. First we start with F = φ. The most significant feature is found. If it is 5, then F = {5}. Then according to Step 3, we need to conditionally remove 5 and check if the criterion improves. In this case, let there be no improvement by removing 5 so it is retained. Then again the most significant feature is found. Let it be 7. Then F = {5, 7}. Let the least significant feature in 5 and 7 be 7. If there is no improvement in criterion value by removing 7, it is retained. Then the next most significant feature is found. Let it be 2. Then F = {5, 7, 2}. Now if the least significant feature in F is 7, we need to check if the criterion improves when 7 is removed. Let there be an improvement in F , then F = {5, 2}. The next most significant feature is found and added to F and so on. This process is repeated till we have the number of features we want.

Sequential Floating Backward Search

This method starts with all the features being considered. One feature is removed at each stage and the most significant feature is added. If the subset gives the best performance of all subsets of its size, the feature added is retained otherwise it is discarded.

The algorithm is given below :

k=d

Remove the least significant feature from the subset of size k. Let k=k-1.

Conditionally add the most significant feature from the features not in the current subset. If the subset generated is the best subset of size k+1 found so far, retain the feature, let k=k+1 and go to Step 3. Else remove the conditionally added feature and go to Step 2.

This method is suitable when only a few features need to be removed.

As an example, if we take a dataset having 10 features, we start with the feature set F = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}. We now have to find the least significant feature. Let it be feature 9. F then becomes F = {1, 2, 3, 4, 5, 6, 7, 8, 10}. Then we find the most significant feature in the features absent in F . Here it is only feature 9. Then we try to see if the criterion improves if 9 is added. Here let the criterion not improve and 9 is not added again. Then from F the least significant feature is found. Let it be 4. Then 4 is removed from F giving F = {1, 2, 3, 5, 6, 7, 8, 10}. Then we find the most significant feature among 4 and 9 and see if the criterion improves by including that feature. Let it not improve and we again find the least significant feature in F . Let it be feature 1. So feature 1 is removed to give F = {2, 3, 5, 6, 7, 8, 10}. If the most significant feature among 9,4 and 1 is 4 and adding 4 to F gives a better criterion function then 4 is again added to F to give= {2, 3, 4, 5, 6, 7, 8, 10}. This process is continued to get the feature subset of the size required by us.

Max-Min approach

This method considers two features at a time.

At each iteration, it includes one feature into the feature subset de-pending on the absolute difference between the criterion of using the new feature with the feature subset and the criterion of the feature subset alone.

Since only two features are considered at a time, it is not so computa-tionally intensive.

Since it does the calculations in two dimensions and not in multi- di-mensional space, the results obtained are unsatisfactory.

At a point in time, if F' is the subset of features so far selected, it is now necessary to find the feature f_j to be included.

The difference between the criterion value of using the feature f_j along with F' and the criterion value of using only F' is found. The absolute value of this measure is called $\delta J(f_j, F')$.

Therefore we get

$$\delta J(f_j, F') = | J(f_j, F') - J(F') |$$

The feature f_j chosen as the next feature is one which satisfies $\max_{\forall f_j}$

$\min_{\forall F'} \delta J(f_j, F')$

X As an example, suppose we have three features f_1 , f_2 and f_3 . Consid-ering f_1 , if the criterion function value difference between using only f_1 and other features are

$J(f_1, f_2)=20$; $J(f_1, f_3)=25$; $J(f_1, f_4)=30$;

Similarly, if the criterion function value difference between using only f_2 and other features are

$J(f_2, f_1)=5$; $J(f_2, f_3)=40$; $J(f_2, f_4)=35$

If the criterion function value difference between using only f_3 and other features are

$J(f_3, f_1)=60$; $J(f_3, f_2)=10$; $J(f_3, f_4)=25$

If the criterion function value difference between using only f_4 and other features are

$J(f_4, f_1)=20$;$J(f_4, f_2)=70$;$J(f_4, f_3)=15$

Considering f_1 and other features, the minimum value of is $J(f_2, f_1)=20$.
Considering f_2 and other features, the minimum value of is $J(f_2, f_1)=5$.
Considering f_3 and other features, the minimum value of is $J(f_3, f_2)=10$.
Considering f_4 and other features, the minimum value of is $J(f_4, f_3)=15$.

The maximum of these four values is for $J(f_2, f_1)$. Therefore the fea-ture chosen is f_1 .

Stochastic Search Techniques

5. These methods have one or more candidate solutions.

6. Each candidate solution gives the feature subset selected.

7. The candidate solution is a binary string where each element represents one feature.

- A 0 in the candidate solution represents the absence of a feature and a 1 represents the presence of a feature in the selected subset.

- Each string has an evaluation which depends on the performance of the subset of features on a training and validation sets. This evaluation is called the fitness function.
- In the Genetic algorithm, the initial population is formed at random.

- The next generation is formed by using the operators of selection, crossover and mutation on the population.

- A string with a higher fitness has a higher probability of being selected for the next generation.

- Crossover involves taking two strings, finding a crossover point and exchanging the elements of the two strings which occur to the right of the crossover point.

- Mutation involves choosing a bit and changing it from 0 to 1 or from 1 to 0.

- As the iterations increase, strings with higher average fitness will be formed.
- The string with the highest fitness will be the final solution.
- Some of the other stochastic search techniques are
    - Simulated Annealing
    - Tabu search

Nearest Neighbour Classifier

Nearest neighbour classifiers

- This is amongst the simplest of all classification algorithms in super-vised learning.

- This is a method of classifying patterns based on the class label of the closest training patterns in the feature space.

- The common algorithms used here are the nearest neighbour(NN) al-gorithm, the k-nearest neighbour(kNN) algorithm, and the modified k-nearest neighbour (mkNN) algorithm.

- These are non-parametric methods where no model is fitted using the training patterns.
- The accuracy using nearest neighbour classifiers is good. It is guaran-teed to yield an error rate no worse than twice the Bayes error rate (explained in Module 10) which is the optimal error rate.
- There is no training time required for this classifier. In other words, there is no design time for training the classifier.
- Every time a test pattern is to be classified, it has to be compared with all the training patterns, to find the closest pattern. This classifica-tion time could be large if the training patterns are large in number or if the dimensionality of the patterns is high.

Nearest neighbour algorithm

- If there are n patterns X_1, X_2, ..., X_N in the training data, X and a test pattern P,

  If X_K is the most similar pattern to P from X, then the class of P is the class of X_K .

- The similarity is usually measured by computing the distance from P to the patterns X_1, X_2, ..., X_N. If $D(P, X_I)$ is the distance from P to X_I, then P is the assigned the class label of X_K where

  $D(P, X_K) = MIN \{ D(P, X_I) \}$

  where $i = 1 \ldots n$

k-Nearest Neighbour (kNN) classification algorithm

- *An object is classified by a majority vote of the class of its neighbours.*
- *The object is assigned to the class most common amongst its* K *nearest neighbours.*
- *If k=1, this becomes the nearest neighbour algorithm.*
- *This algorithm may give a more correct classification for boundary patterns than the NN algorithm.*
- *The value of* K *has to be specified by the user and the best choice depends on the data.*
- *Larger values of* K *reduce the effect of noise on the classification. The value of* K *can be arbitrary increased when the training data set is large in size.*
- *The* K *value can be chosen by using a validation set and choosing the* K *value giving best accuracy on the validation set.*

- The main disadvantage of kNN algorithm is that it is very time con-suming especially when the training data is large.

- To overcome this problem, a number of algorithms have been proposed to access the k nearest patterns as fast as possible.

Modified k-Nearest Neighbour (mkNN) classifier

- The contribution of the neighbours to the classification is weighted according to its distance from the test pattern.

- Hence, the nearest neighbour contributes more to the classification de-cision than the neighbours further away.

- One weighting scheme would be to give each neighbour a weight of $\frac{1}{D}$, where d is the distance from P to the neighbour.

- Another weighting scheme finds the weight from the neighbour as

$$W_I = \begin{cases} \text{if } D_K = D_1 \\ \frac{D - D_I}{D - D} \quad \text{if } D_K = 6\ D_1 \end{cases}$$

  where i=1,...,k .

- The value of W_I varies from 1 for the closest pattern to 0 for the farthest pattern among the k closest patterns.

- This modification would mean that outliers will not affect the classifi-cation as much as the kNN classifier.

Example

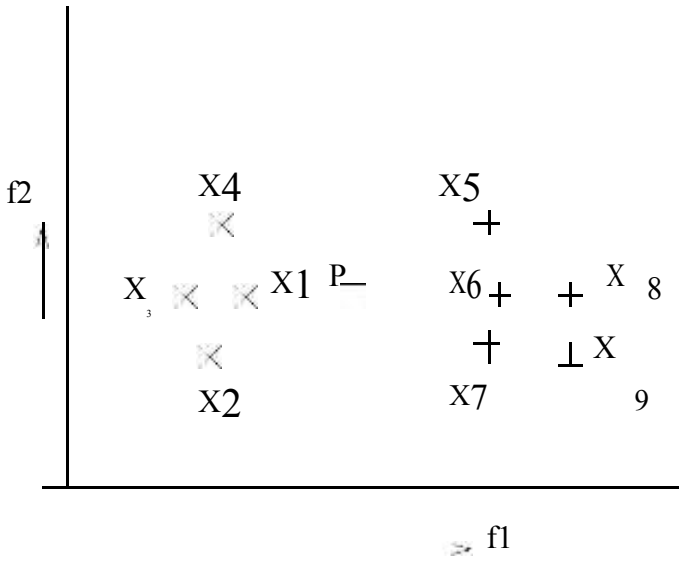

Figure 1: Two class problem

Consider the two class problem shown in Figure 1. There are four pat-terns in Class 1 marked as 'X' and there are five patterns in Class 2 marked as '+'. The test pattern is is P. Using the nearest neighbour algorithm, the

closest pattern to P is X_1 whose class is 1. Therefore P will be assigned to Class 1.

If kNN algorithm is used, after X_1, P is closest to X_6 and X_7. So, if k=3, P will be assigned to Class 2. It can be seen that the value of k is crucial to the classification. If k=1, it reduces to the NN classifier. In this case, if k=4, the next closest pattern could be X_2. If k=5 and X_3 is closer to P than X_5, then again due to majority vote, P will be assigned to Class 1. This shows how important the value of k is to the classification. If P is an outlier of one class but is closest to a pattern of another class, by taking majority vote, the misclassification of P can be prevented.

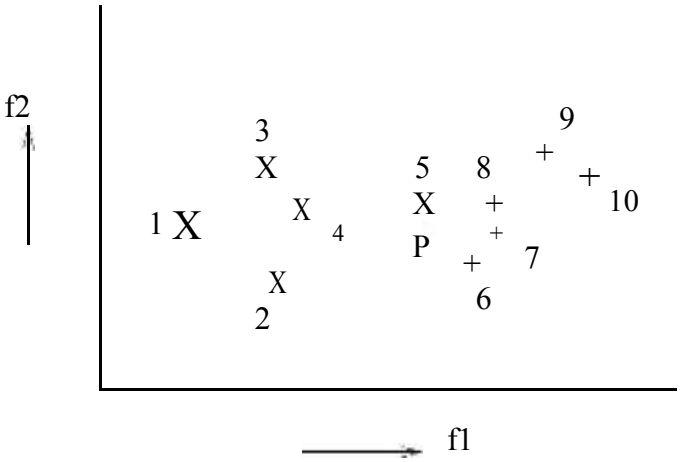

Figure 2: Another two class problem

For example, if we consider Figure 2 we can see that the test pattern P is closest to X_5 which belongs to Class 1 and therefore, it would be classified as belonging to Class 1 if NN classifier is used. X_5 is an outlier of Class 1 and it can be seen that classifying P as belonging to Class 2 would be more meaningful. If kNN algorithm is used with k=3, then P would be classified as belonging to Class 2.

Using mkNN, the classification depends on the distances of the closest patterns from the test pattern. In the kNN algorithm, all the K patterns

will have equal importance. In mkNN, the closest pattern is given more significance than the farthest pattern. The weightage given to the class of the first closest pattern is more than for the second closest pattern and so on.

For example, if the 5 nearest neighbours to P are X_1, X_2, X_3, X_4 and X_5, where $D(X_1, P) = 1$, $D(X_2, P) = 2$, $D(X_3, P) = 2.5$, $D(X_4, P) = 4$ and $D(X_5, P) = 5$, and if X_1 and X_4 belong to Class 1 and X_2, X_3 and X_5 belong to Class 2, then the weight given to Class1 by X_1 will be

$W_{11} = 1$

The weight given to Class 1 by X_4 will be

$$W_{14} = \frac{5 - 4}{5 - 1} = 0.25$$

The total weight of Class 1 will be

$w_1 = W_1 + W_2 = 1.0 + 0.25 = 1.25$

The weight given to Class 2 by X_2 will be

$$W_{22} = \frac{5 - 2}{5 - 1} = 0.75$$

The weight given to Class 2 by X_3 will be

$$W_{23} = \frac{5 - 2.5}{5 - 1} = 0.625$$

The weight given to Class 2 by X_5 which is W_{25} will be 0 since it is the farthest of the 5 neighbours.

The total weight of Class 2 will be

$w_2 = W_{22} + W_{23} + W_{25} = 0.75 + 0.625 + 0 = 1.375$

Since $w_2 > w_1$, P is classified as belonging to Class 2.

If we consider Figure 1, the closest points to P are X_1, X_6, X_7, X_2 and X_5. If the distances from P to X_1, X_6, X_7, X_2 and X_5 are 0.3, 1.0,1.1,1.5 and

1.6, then calculating the weight given to the two classes

The weight given to Class 1 by X_1 will be

$W_{11} = 1$

The weight given to Class 1 by X_2 will be

$$W_{12} = \frac{1.6 - 1.5}{1.6 - 0.3} = 0.077$$

The total weight of Class 1 will be

$w_1 = W_{11} + W_{12} = 1 + 0.077 = 1.077$

The weight given to Class 2 by X_6 will be

$$W_{26} = \frac{1.6 - 1.0}{1.6 - 0.3} = 0.462$$

The weight given to Class 2 by X_7 will be

$$W_{27} = \frac{1.6 - 1.1}{1.6 - 0.3} = 0.385$$

The weight given to Class 2 by X_5 which is W_{25} is 0, since X_5 is the farthest of the 5 neighbours. Then the total weight for Class 2 will be

$w_2 = W_{26} + W_{27} + W_{25} = 0.462 + 0.385 + 0 = 0.847$

Since $w_1 > w_2$, P is classified as belonging to Class 1.

One point to be noted here is that while kNN algorithm classifies P as belonging to Class 2, mkNN algorithm classifies P as belonging to Class 1. It can therefore be seen that the classification decision using kNN and mkNN may vary.

Soft Nearest Neighbour Classifiers

Fuzzy kNN Algorithm

- In Fuzzy kNN Algorithm, each pattern belongs to every class with a membership value in interval [0,1].
- The membership value of each pattern to each class depends on the class of its k neighbours.
- The k neighbours of a pattern are found. The membership of the pattern to the classes of the k neighbours is calculated depending on the distance of the point to the k neighbours.
- The membership value of this pattern to the classes to which the k neighbours do not belong will be zero.
- Let A =

$$\sum_{j=1}^{K} \mu_{ij} \left(\frac{1}{d(P, X_j)^{\frac{2}{M-1}}} \right)$$

Let B =

$$\sum_{j=1}^{K} \left(\frac{1}{d(P, X_j)^{\frac{2}{M-1}}} \right)$$

- Then, a new sample P has membership μ_{iP} for class i which is

given by Then $\mu_{iP} = \frac{A}{B}$

μ_{ij} gives the membership in the i^{th} class of the j^{th} vector of the training set.

- The membership value of the pattern for every class is calculated.
- The pattern is assigned to the class to which it has the highest mem-bership value.

- The value of m can be 2,3,...etc. since m=1 will lead to the value of the power being ∞. In the numerator, weightage is only given to the k neighbours of the point. The weightage to each neighbour depends on the distance of the neighbour from the point. The closest neigh-bour gets the largest weightage, then the next neighbour etc. As m is increased, the weightage given to further neighbours comes down.

Example

Consider a test point P which is at a distance of 1(unit) from the closest point. This pattern is of Class 1. P is at a distance of 2 from the next closest point which belongs to Class 2. P is at a a distance of 3 from the next closest point which is of Class 3. P is at a distance of 4 from the next closest point which is of Class 1. P is at a distance of 5 from the 5th closest point which is of Class 2. Let there be 5 classes. Let m=2.

Then the membership value of P to Class 1 will be

$$\mu_{1P} = \frac{\frac{1}{1^2} + \frac{1}{4^2}}{\frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2}} = 0.7259$$

The membership value of P to Class 2 will be

$$\mu_{2P} = \frac{\frac{1}{2^2} + \frac{1}{5^2}}{\frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2}} = 0.1981$$

The membership value of P to Class 3 will be

$$\mu_{3P} = \frac{\frac{1}{3^2}}{\frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2}} = 0.0759$$

The membership value of P to Class 4 will be

μ_{4P} = 0 since none of the closest 5 neighbours comes from Class 4.

The membership value of P to Class 5 will be

μ_{5P} = 0 since none of the closest 5 neighbours comes from Class 5.

Since the membership value of P with respect to Class 1 is the highest, P is assigned to Class 1.

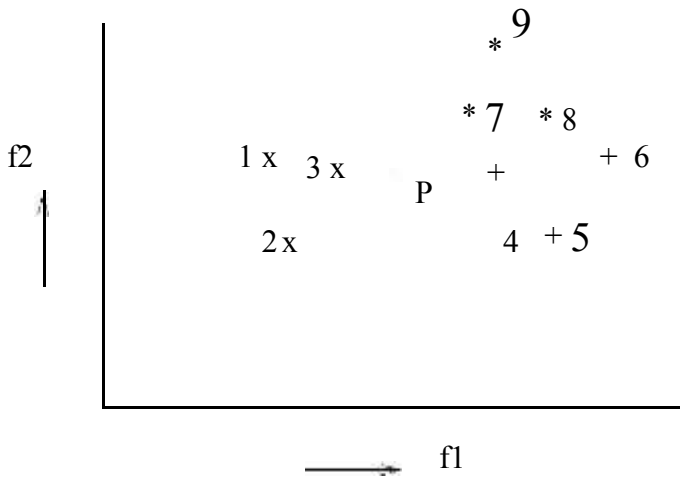

Figure 1: Three class example set

As another example, let us consider the three-class problem shown in Figure 1. The test pattern P is closest to the pattern X_3. The next closest pattern is X_4. The next closest pattern is X_7. The distances from P are are 1.0, 1.2 and 1.6 from X_3, X_4 and X_7 rsespectively. We are looking at k = 3 or the three closest patterns are being considered. Then the membership value for Class 1 will be

$$\mu_{1P} = \frac{\frac{1}{1^2}}{\frac{1}{1^2} + \frac{1}{1.2^2} + \frac{1}{1.6^2}} = 0.4796$$

The membership value for Class 2 will be

$$\mu_{2P} = \frac{\frac{1}{1.2^2}}{\frac{1}{1^2} + \frac{1}{1.2^2} + \frac{1}{1.6^2}} = 0.3331$$

The membership value for Class 3 will be

$$\mu_{3P} = \frac{\frac{1}{1.6^2}}{\frac{1}{1^2} + \frac{1}{1.2^2} + \frac{1}{1.6^2}} = 0.1873$$

Since μ_{1P} is greater than μ_2P and μ_3P , P is assigned to Class 1.

r-Near Neighbours

- Instead of looking at k nearest neighbours to the test sample P, we look

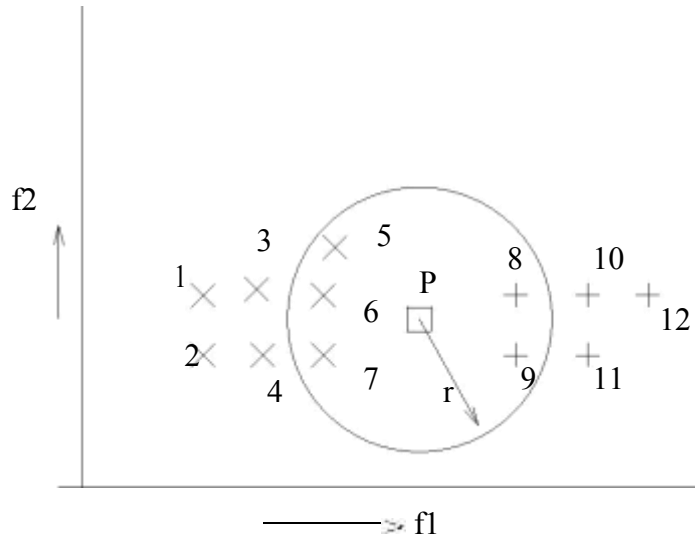

Figure 2: Two class example set

at all samples falling within a distance r from P.

- From P, consider a ball of radius r centred at P .
- All samples falling in this ball are considered and the majority class label of these samples is considered to be the class of P.
- If no samples fall within the ball of radius r, then the majority class of the entire data is considered or the possibility of the pattern being an outlier is considered.
- In this algorithm, distances of the neighbours of a point P are consid-ered and not the number of neighbours.

Example

- Consider Figure 2. If P is the test pattern then a circle of radius r is drawn from P.
- It can be seen that five points, X_5, X_6, X_7, X_8 and X_9 fall within the circle of radius r from P. The majority class of these five points is Class 1 labelled X and therefore P is assigned to Class 1.

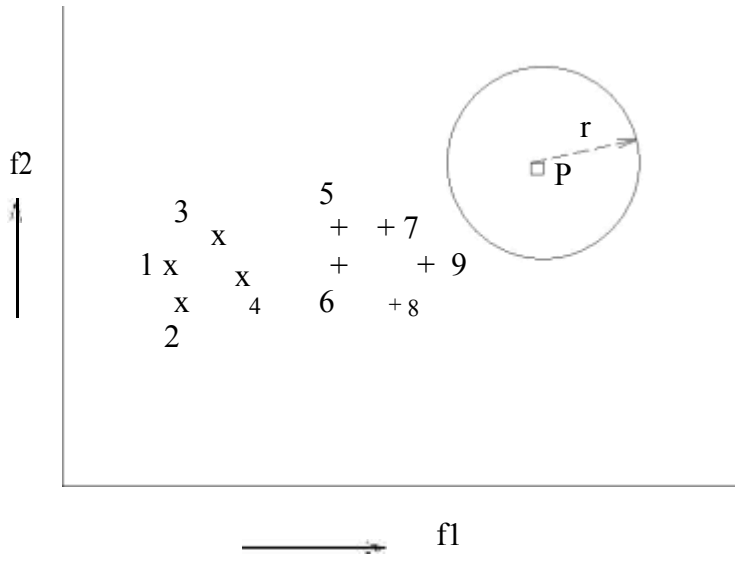

Figure 3: Outlier using r-Near Neighbours

- This algorithm is distance-based. NN, kNN and mkNN find the k nearest neighbours and use their labels to find the class of P. Here the number of patterns considered depends on the number of patterns within a fixed distance from P.
- If P is an outlier and there are no patterns within the ball of radius, then it is easy to identify that P is an outlier.
- This can be seen in Figure 3. Here, if a circle of radius r is drawn with P as the centre, no patterns fall within this circle. This will show that P is an outlier. It can be treated as a outlier and not assigned to any class. Using NN, kNN and mkNN, it would be classified as belonging to Class 2.

Drawbacks of Nearest Neighbour Classifiers

- The biggest drawback of nearest neighbour classifiers is the time taken to find the nearest neighbours of a test pattern P.
- All the training data set is used to classify each test pattern.

- If the training data is large, storing of the entire training data leads to increase in the space complexity.
- If the training data is large, the time required to find the nearest neigh-bours of a test pattern increases.
- If there are n training samples from a d-dimensional space, the space complexity will be O(nd).
- The time complexity to find the nearest training pattern to P will be also O(nd).
- Due to this, work has been done to take care of this problem by con-densation of the training data and also use of efficient algorithms to find the nearest neighbour of a test pattern.
- These aspects are studied in Module 8 and Module 9.

- Use Fuzzy kNN algorithm to classify P using m = 2 and the data given problem 1.
- Use the r-Near Neighbours classifier with r = 1.45, 5 neighbours, and data in problem 1 to classify P.

Efficient Nearest Neighbour Classifier

Efficient Algorithms for Nearest Neighbour Classifiers

- *While the nearest neighbour algorithm gives good results and is robust, it is tedious to use when the training data is very large.*
- *Finding the nearest neighbour to a test pattern requires the computa-tion of the similarity(dissimilarity) between the test pattern and every training pattern. If there are* n *patterns with dimensionality* d*, this will be* O(nd).
- *If the number of training patterns is very large or if the dimensionality of the patterns is large, this will be a very time-consuming exercise.*
- *Another disadvantage is that the entire training set should be stored and utilized to find the most similar pattern.*

- The solutions to this problem are as follows :
  - Some pre-processing is carried out on the training set so that the process of finding the nearest neighbour or finding k-neighbours is easier. These algorithms are called efficient algorithms.
  - The number of training patterns is reduced. This is done so that it does not cause the classification accuracy to come down too much. This is called prototype selection.
  - The dimensionality of the patterns is reduced. The attributes which are less significant are removed and only the attributes which are significant are retained for classification. This process is called feature selection.
- This module discusses the efficient algorithms.
- Prototype selection is discussed in the next module.

Principle of Efficient Algorithms

- All the methods require some pre-processing of the training set.
- This pre-processing involves putting some ordering in the data or ar-ranging it in some way so as to be able to access the nearest neighbour easily.
- Even though the pre-processing requires some time, it needs to be done only once.
- The pre-processed data can then be used for locating the nearest neigh-bour of a test pattern.
- Besides speeding up the process of finding the nearest neighbour(s), there maybe saving in space also depending on the way in which the data is stored.

Branch and Bound algorithm

- The patterns are clustered together so that points which are close to-gether form a cluster.
- These first level clusters are again divided into clusters and this is done recursively, till each cluster has only one point.
- For every cluster j, the center m_j and the radius r_j are calculated and stored.
- To find the closest pattern to a test pattern P , the following steps are followed :

  For every cluster calculate $a_j = (d(P, m_j) - r_j)$.

All points in the cluster j will have a distance from P which is more than a_j .

Choose the cluster k which has the smallest a_k and find the dis-tances from P to the points in the cluster. Let the smallest dis-tance be dmin.

We can say that for a cluster j, if $a_j \geq$ dmin, then that cluster need not be searched.

Only the clusters j which have a_j < dmin need to be searched.

- Since all clusters need not be searched, there is considerable improvement in computation as compared to the nearest neighbour algorithm.

Example

Consider the following set of patterns :

X_1 = (1, 1, 1, 1); X_2 = (1, 2, 1, 1); X_3 = (2, 2, 2, 1);
X_4 = (2, 2, 2, 1); X_5 = (1, 2, 2, 1); X_6 = (1, 2, 5, 2);
X_7 = (2, 1, 6, 2); X_8 = (2, 2, 5, 2); X_9 = (1, 2, 6, 2);
X_{10} = (2, 2, 6, 2); X_{11} = (4, 1, 2, 3); X_{12} = (4, 2, 1, 3);
X_{13} = (5, 2, 2, 3); X_{14} = (6, 1, 2, 3); X_{15} = (5, 1, 1, 3);

Each pattern consists of four values. The first three are the three features and the fourth value gives the class of the pattern. The first step is to cluster the points. Let the points X_1, X_2, X_3, X_4 and X_5 form the first cluster. Let the patterns X_6, X_7, X_8, X_9 and X_{10} form the second cluster and the points X_{11}, X_{12}, X_{13}, X_{14} and X_{15} form the third cluster. At the next level, each cluster has to be partitioned into subclusters. Let patterns X_1, X_2 and X_3 form cluster 1a and patterns X_4 and X_5 form cluster 1b. Similarly let patterns X_6, X_7 and X_8 form cluster 2a and patterns X_9 and X_{10} form cluster 2b. Further, let patterns X_{11}, X_{12} and X_{13} form cluster 3a and patterns X_{14} and X_{15} form cluster 2b. At the next level, each pattern is taken to belong to a subcluster. This is shown in Figure 1.

The centre of cluster 1 is (1.4,1.8,1.6) and the radius is 0.98.
The centre of cluster 2 is (1.6,1.8,5.6) and the radius is 0.98.
The centre of cluster 3 is (4.8,1.4,1.6) and the radius is 1.33.

Let us take a point P = (4,4,3) as the test point from which we are to find the closest pattern.

At the first level, we have to calculate a_j for each cluster j. For cluster 1,

$d(P, m_1) = 3.94$
$a_1 = 3.94 - 0.98 = 2.96$

For cluster 2,

$d(P, m_2) = 4.17$
$a_2 = 4.17 - 0.98 = 3.19$

For cluster 3,

$d(P, m_3) = 3.06$

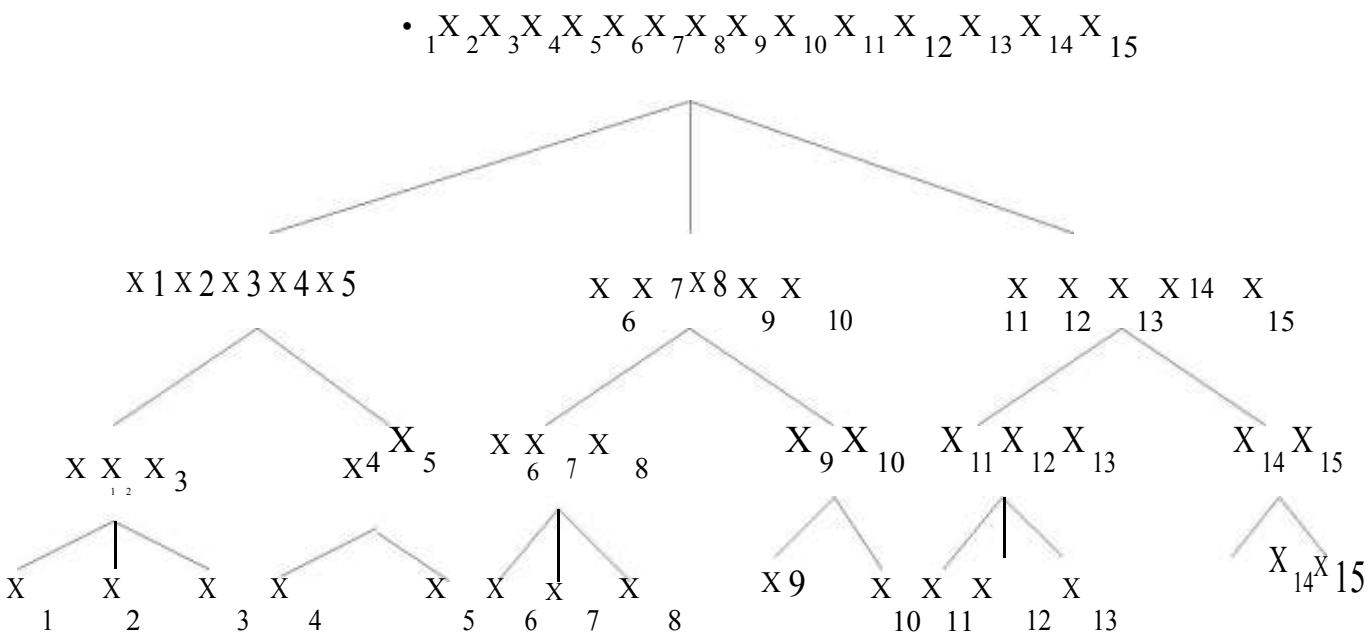

Figure 1: Clusters and subclusters for Branch and Bound technique

a_3 = 3.06 - 1.33 = 1.73

Since a_3 is smaller than a_1 and a_2, the subclusters of cluster 3 are searched.

The centre of cluster 3a is (4.33,1.66,1.66) and the radius is 0.808.
The centre of cluster 3b is (5.5,1,1.5) and the radius is 0.707.

For cluster 3a,
$d(P, m_{3a})$ = 2.70
a_{3a} = 2.70 - 0.808 = 1.89

For cluster 3b,
$d(P, m_{3b})$ = 3.67
a_{3b} = 3.67 - 0.707 = 2.96

Since a_{3a} is smaller than a_{3b} , cluster 3a is searched. Cluster 3a is searched by finding the distance from P to the points X_{11}, X_{12} and X_{13}. X_{13} is found to be closest among these points to P. This sets the bound dmin to be the

distance from X_{13} to P which is 2.44 . Since a_1 and a_2 are greater than dmin, they need not be searched and therefore X_{13} is the closest neighbour of P.

Cube Algorithm

- This algorithm takes a hypercube of side l units around the test point P.
- This implies that a distance of d = 2^l is taken on both sides of P in every dimension to form the hypercube. Let ϱ = a small value.
- In the case where there are only two dimensions, the hypercube will be a square.
- Count the number of points m in the hypercube.
- If m < k, increase l appropriately so as to include more points. From the test point P, the distance d can be increased to give d = d + ϱ on both sides.
- Repeat till there are k points in the hypercube. Only the newly included portion needs to be changed.
- If m > k, decrease l so that less points are included in the hypercube. From the test point P, the distance d can be decreased to give d = d −ϱ on both sides.
- Repeat till there are k points in the hypercube. Only the newly excluded portion of the hypercube needs to be changed.

Example

Consider the example taken for the branch and bound algorithm. We have 15 patterns as given belonging to three classes. The test point is P = (4,4,3). Let us take a hypercube with l=2. Then we need to find all patterns with the x-axis value between 3 and 5, with the y-axis value between 3 and 5 and the z-axis value between 2 and 4. None of the patterns fall in this hyper-cube. We therefore increase the size of the hypercube. Let us increase l by 1 so that l = 3. In this case, the x-axis value should be between 2.5 and 5.5, the y-axis value should be between 2.5 and 5.5 and the z-axis value should be

between 1.5 and 4.5 . Since there are no patterns falling into the hypercube we increase l= 4. Then the x-axis value should be between 2 and 6, the y-axis value should be between 2 and 6 and the z-axis value should be between 1 and 5. The points falling within this hypercube will be X3, X4, X8, X12 and X13. If k=5, the five nearest neighbours have been found. If k is less than 3, then l has to be reduced till the correct number of points are found. If k is greater than 5, then l is again increased. This process is repeated till k nearest neighbours are found.

Projection algorithm

In this algorithm, the points are ordered according to a particular axis.

If the points are ordered according to the x-axis, then the k nearest neighbours to the test point P can be determined.

The actual distance from P to these points is found. Let the minimum of these distances be d and let the point be Z.

Continue searching points according to its distance in the x-axis.

Maintain the distance d which is the minimum actual distance of the points examined from Z.

As soon as the distance of the point being considered has a distance in the x-direction which is greater than d, the search can be stopped.

This process can be modified so as to carry out each step of the algorithm in different axis.

In other words, the first step of the algorithm is carried out in the x-axis, the second step in the y-axis, etc.

The search stops as soon as search in any axis stops.

Example

Consider the same three-dimensional example considered for the exam-ples in this lesson. If we order the points according to the distance of the points from the x-axis of point P, in increasing order we get,

X_{11}, X_{12}, X_{13}, X_{15}, X_3, X_4, X_7, X_8, X_{10}, X_{14}, X_1, X_2, X_5, X_6 and X_9 with a distance of 0,0,1,1,2,2,2,2,2,2,3,3,3,3 and 3. The actual distance to P from the closest points to P are found. The actual distances from X_{11}, X_{12}, X_{13}, X_{15}, X_3, X_4, X_7, X_8, X_{10} and X_{14} from P are 3.16, 2.83, 2.44, 3.74, 4.36, 3.0, 4.69, 3.46, 4.12 and 3.74 . After the first three distances, the minimum distance is 2.44 . It remains 2.44 till X_{14} is reached. After that since the distance in the x-axis itself is larger than 2.44, those points need not be searched and X_{13} is the closest point to P.

Ordered Partitions

Ordered Partitions

- All the points are partitioned using a technique called the ordered par-titions.
- The range of values according to the first co-ordinate values is deter-mined and divided into d blocks if the patterns are d-dimensional.
- The points are partitioned according to the interval to which they be-long.
- The blocks are made so that each partition contains equal number of samples.
- Then in each block, partitions are made according to the second co-ordinate axis and so on.
- When the closest pattern to a test pattern P is to be found, the range of values which contains the first co-ordinate of P is chosen as the block to be searched. The search then continues according to the second co-ordinate and the block whose range includes the second co-ordinate of P is searched, and so on till a point is found after going through all the d dimensions.
- The distance of this point to the test pattern P is found.
- The other blocks which have a value in the first co-ordinate which is larger than the smallest distance dist found till now need not be searched.
- In this way many of the partitions need not be searched.

Example

Consider the following set of patterns :

X_1 = (1, 1, 1, 1); X_2 = (1, 2, 1, 1); X_3 = (2, 2, 2, 1);
X_4 = (2, 2, 2, 1); X_5 = (1, 2, 2, 1); X_6 = (1, 2, 5, 2);
X_7 = (2, 1, 6, 2); X_8 = (2, 2, 5, 2); X_9 = (1, 2, 6, 2);
X_{10} = (2, 2, 6, 2); X_{11} = (4, 1, 2, 3); X_{12} = (4, 2, 1, 3);
X_{13} = (5, 2, 2, 3); X_{14} = (6, 1, 2, 3); X_{15} = (5, 1, 1, 3);

Each pattern consists of four values. The first three are the three features and the fourth value gives the class of the pattern. The patterns are partitioned by giving a range for each partition to the first dimension. Each partition should have equal number of patterns. This is given in Figure 1.

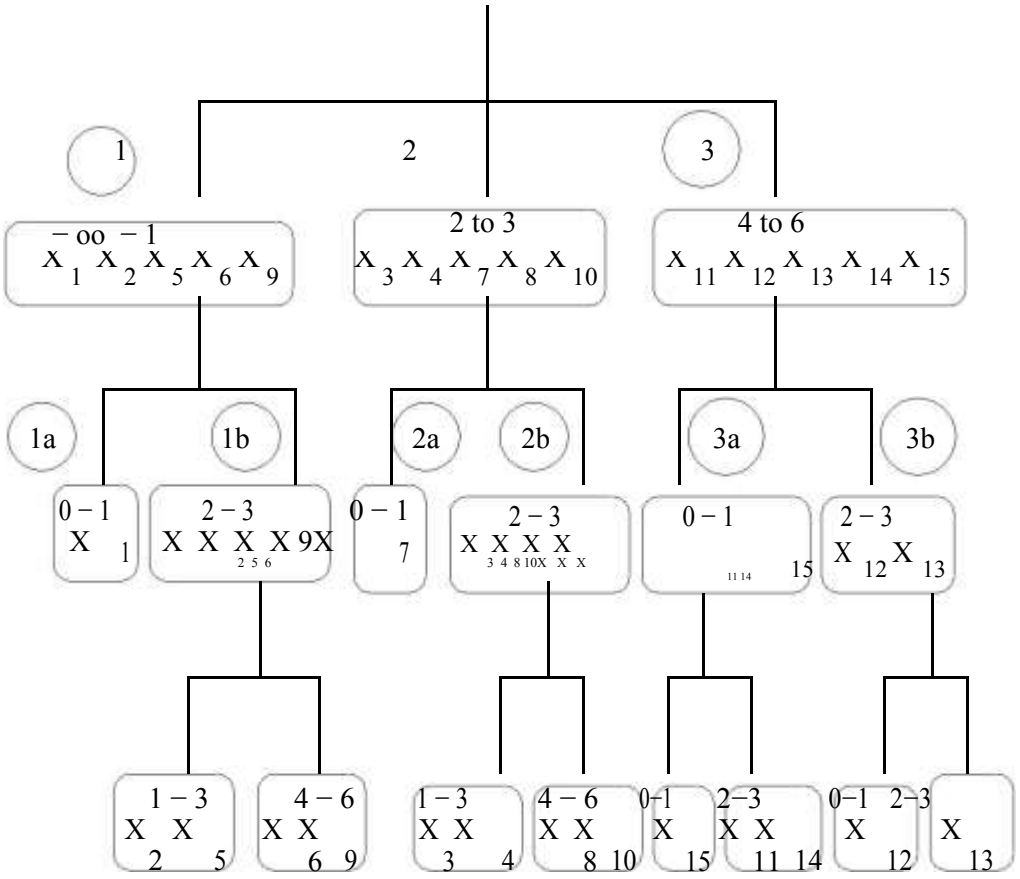

Figure 1: Partitioning using ordered partitions

Now if we have a point P = (4,4,3) and we need to find the nearest pat-tern, we first search at the first level in Figure 1. It goes along branch 3 since the first co-ordinate of the point P falls in the range 4-6 . It then searches branch 3b since the second co-ordinate is closer to that range. So the actual distance from P to X_{12} and X_{13} is found.

$d(P, X_{12}) = 2.83$

$d(P, X_{13}) = 2.45$

Since the distance between P and X_{13} is the smaller value, the present closest distance is 2.45 and X_{13} is the closest point.

The branch 3a need not be searched since the distance in the second co-ordinate from these patterns is at least 3 which is more than the minimum distance 2.45 .

Branch 2 needs to be searched but not branch 1 since the distance in the first co-ordinate itself from the patterns in this branch is 3 which is larger than the minimum distance found so far which is 2.45 . In branch 2, branch 2a need not be searched and 2b needs to be searched. After going through these patterns, it is found that X_{13} is the closest pattern to P.

KD Tree

- The kd Tree partitions the data into blocks.
- At each stage, the patterns are partitioned to form two blocks.
- The complete data is taken and divided into two blocks according to the first co-ordinate axis.
- A hypercube represents all the patterns.
- When a partition is made, a hyperplane is drawn parallel to one of the co-ordinate axis.
- This will divide the patterns into two blocks.
- Each partition is then again divided into two blocks which is represented in the kd tree by drawing a hyperplane parallel to one of the axes.
- When the entire kd tree has been drawn, each block corresponds to some of the patterns.

- To find the closest pattern to the test pattern, find the block which contains the test pattern. Search the leaf nodes to find the closest pat-tern. Let the distance from P to this point be dist. Draw a hypersphere with radius dist and P as centre.

- If this falls within a single block, then only the points falling inside this hypersphere need to be searched and the closest point found.

- If the hypersphere falls into other blocks also, then go up the tree and come down the tree to the leaf nodes in the other blocks and search there also to find the closest point to P

Example

Take the same 3-dimensional example taken for the ordered partitions.

Figure 2 gives the partitioning of the patterns. This is done by dividing the range in the first co-ordinate first, then by dividing the range in the second co-ordinate and then by dividing the range in the third co-ordinate. By using this partitioning, the kd tree can be drawn.

The kd tree for the above data is shown in Figure 3. Now if we need to find the closest neighbour to P = (4,4,2), you search down the kd tree and go to the region which contains patterns closest to P. In this case, it will be X_{13}. If we find the distance between P and X_{13}, we get 2.45 . We then need to search the other relevant points also which are within the radius of 2.45 around P.

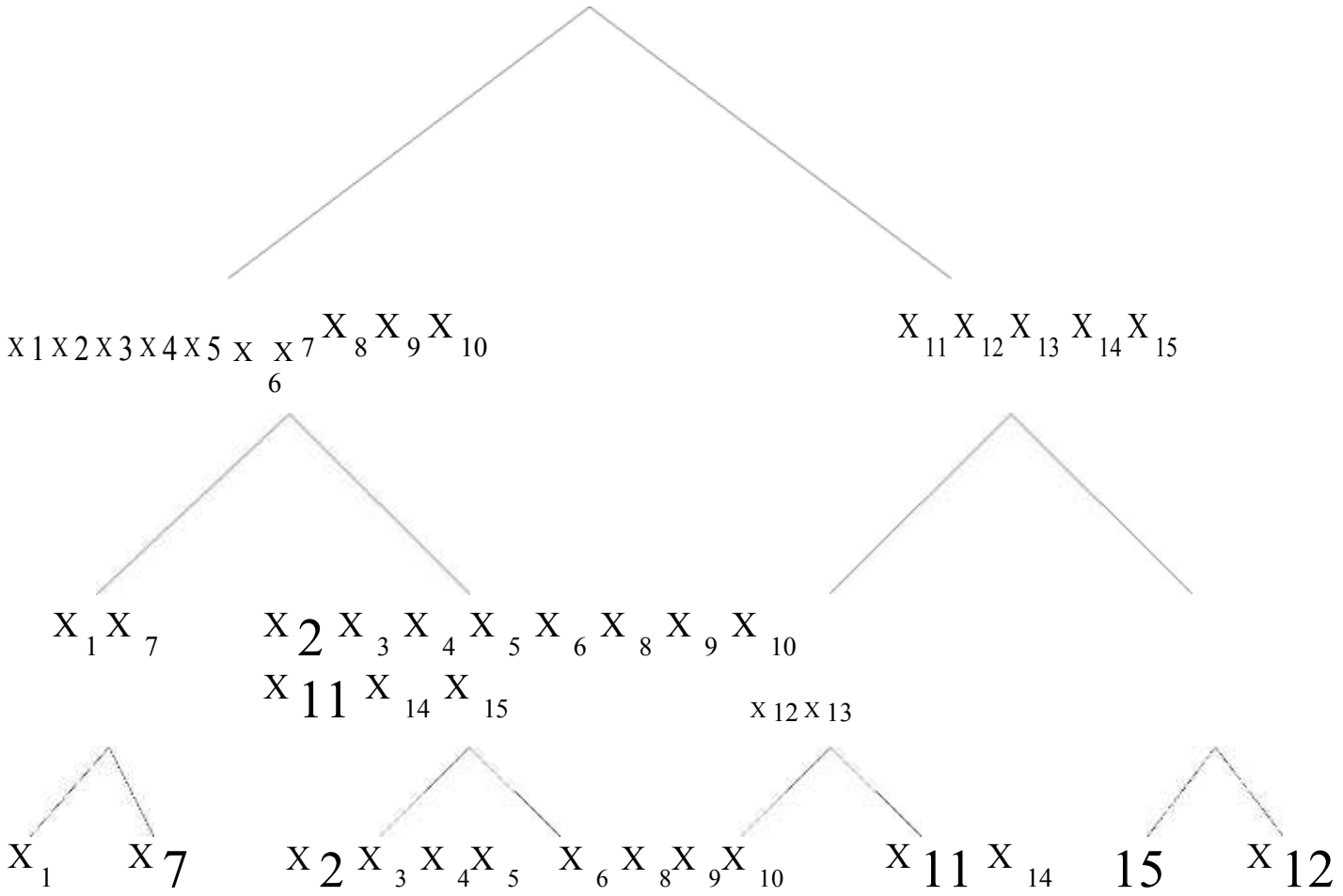

Figure 2: Binary partitioning for the kd tree

Bayes Classifier

Classification using Bayes Decision Theory

- In this approach classification is carried out using probabilities of classes.
- It is assumed that we know the a priori or the prior probability of each class. If we have two classes C_1 and C_2, then the prior probability for class C_1 is P_{C_1} and the prior probability for class C_2 is P_{C_2}.
- If the prior probability is not known, the classes are taken to be equally likely.
- If prior probability is not known and there are two classes C_1 and V_2, then it is assumed $P_{C_1} = P_{C_2} = 0.5$.
- If P_{C_1} and P_{C_2} are known, then when a new pattern x comes along, we need to calculate $P(C_1|x)$ and $P(C_2|x)$.
- The bayes theorem is used to compute $P(C_1|x)$ and $P(C_2|x)$.
- Then if $P(C_1|x) \bullet P(C_2|x)$, the pattern is assigned to Class 1 and if $P(C_1|x) < P(C_2|x)$, it is assigned to Class 2. This is called the Bayes decision rule.

Bayes Rule

- If $P(C_i)$ is the prior probability of Class i, and $P(X|C_i)$ is the conditional density of X given class C_i, then the a posteriori or posterior probability of C_i is given by

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}$$

- Bayes theorem provides a way of calculating the posterior probability $P(C_i | X)$. In other words, after observing X , the posterior probability that the class is c_i can be calculated from Bayes theorem. It is useful to convert prior probabilities to posterior probabilities.
- $P(X)$ is given by

$$P(X) = \sum_i P(X | C_i)P(C_i)$$

- Let the probability that an elephant is black be 80% and that an elephant is white be 20%. This means P(elephant is black) = 0.8 and P(elephant is white) =0.2 . With only this information, any elephant will be classified as being black. This is because the probability of error in this case is only 0.2 as opposed to classifying the elephant as white which results in a probability of error of 0.8 . When additional information is available, it can be used along with the information above.

  If we have probability that elephant belongs to region X is 0.2 . Now if the elephant belongs to region X, we need to calculate the posterior probability that the elephant is white. i.e. P(elephant is white | ele-phant belongs to region) or P (W | X). This can be calculated by using Bayes theorem. If 95% of the time when the elephant is white, it is because it belongs to the region X. Then

$$P(W \mid X) = \frac{P(X \mid W) * P(W)}{P(X)} = \frac{0.95 * 0.2}{0.2} = 0.95$$

  The probability of error is 0.05 which is the probability that the elephant is not white given that it belongs to region X.

Minimum Error Rate Classifier

- If it is required to classify a pattern X , then the minimum error rate classifier classifies the pattern X to the class C which has the maximum posterior probability P (C | X).
- If the test pattern X is classified as belonging to Class C, then the error in the classification will be (1 - P (C | X)).
- It is evident to reduce the error, X has to be classified as belonging to the class for which P (C | X) is maximum.
- The expected probability of error is given by

$$\int_{\mathbf{X}} (1 - P(C \mid X))P(X)dX$$

This is minimum when P (C | X) is maximum (for a specified value of (X).

- Let us consider an example of how to use minimum error rate classifier for a classification problem. Let us consider an example with three classes small, medium and large with prior probability

(small) = $\frac{1}{3}$
(medium) = $\frac{1}{2}$
(large) = $\frac{1}{6}$

We have a set of nails, bolts and rivets in a box and the three classes correspond to the size of these objects in the box.

Now let us consider the class-conditional probabilities of these objects :

For Class small we get

P (nail | small) = $\frac{1}{4}$
P (bolt | small) = $\frac{1}{2}$
P (rivet | small) = $\frac{1}{4}$

For Class medium we get

P (nail | medium) = $\frac{1}{2}$
P (bolt | medium) = $\frac{1}{6}$
P (rivet | medium) = $\frac{1}{3}$

For Class large we get

P (nail | large) = $\frac{1}{3}$
P (bolt | large) = $\frac{1}{3}$

P (rivet | large) = $\frac{1}{3}$

Now we can find the probability of the class labels given that it is a nail, bolt or rivet. For doing this we need to use Bayes Classifier. Once we get these probabilities, we can find the corresponding class labels of the objects.

$$P(\text{small} \mid \text{nail}) = \frac{P(\text{nail}|\text{small})P(\text{small})}{P(\text{nail}|\text{small}).P(\text{small})+P(\text{nail}|\text{medium}).P(\text{medium})+P(\text{nail}|\text{large}).P(\text{large})}$$

This will give

$$P(\text{small} \mid \text{nail}) = \frac{\frac{1}{4}.\frac{1}{3}}{\frac{1}{4}.3\frac{1}{}+2\frac{1}{}.2\frac{1}{}+\frac{1}{3}.\frac{1}{6}} = 0.2143$$

Similarly, we calculate P (medium | nail) and we get

$$P(\text{medium} \mid \text{nail}) = \frac{1}{1}\frac{1}{1}^{2}\ \frac{\frac{1}{}\textbf{.f rac}12}{\frac{1}{4}.\frac{1}{3}+\frac{1}{2}.\frac{1}{2}+\frac{1}{3}.\frac{1}{6}} = 0.6429$$

and also P (large | nail)

$$P(\text{large} \mid \text{nail}) = \frac{\frac{1}{3}.\frac{1}{6}}{\frac{1}{4}.\frac{1}{3}+\frac{1}{2}.\frac{1}{2}+\frac{1}{3}.6\frac{1}{}} = 0.1429$$

Since P (medium | nail) > P (small | nail) and
P (medium | nail) > P (large | nail)

we classify nail as belonging to the class medium. The probability of error P (error | nail) = 1 − 0.6429 = 0.3571

In a similar way, we can find the posterior probability for bolt

$$P(\text{small} \mid \text{bolt}) = \frac{\frac{1}{2}.\frac{1}{3}}{\frac{1}{2}.3\frac{1}{}+\frac{1}{6}.2\frac{1}{}+\frac{1}{3}.\frac{1}{6}} = 0.5455$$

$$P(\text{medium} \mid \text{bolt}) = \frac{\frac{1}{6}.\frac{1}{2}}{\frac{1}{2}.3\frac{1}{}+\frac{1}{6}.\frac{1}{2}+\frac{1}{3}.\frac{1}{6}} = 0.2727$$

$$P(\text{large} \mid \text{bolt}) = \frac{\frac{1}{3} \cdot \frac{1}{6}}{\frac{1}{2} \cdot \frac{1}{3} + \frac{1}{6} \cdot \frac{1}{2} + \frac{1}{3} \cdot \frac{1}{6}} = 0.1818$$

Since P (small | bolt) > P (medium | bolt)
and P (small | bolt) > P (large | bolt)

we classify bolt as belonging to the class small and the probability of error P (error | bolt) = 1 − 0.5455 = 0.4545

In a similar way, we can find the posterior probability for rivet

$$P(\text{small} \mid \text{rivet}) = \frac{\frac{1}{4} \cdot \frac{1}{3}}{\frac{1}{4} \cdot 3\frac{1}{3} + \frac{1}{3} \cdot \frac{1}{2} + \frac{1}{3} \cdot \frac{1}{6}} = 0.2727$$

$$P(\text{medium} \mid \text{rivet}) = \frac{\frac{1}{3} \cdot \frac{1}{2}}{\frac{1}{4} \cdot \frac{1}{3} + \frac{1}{3} \cdot \frac{1}{2} + \frac{1}{3} \cdot 6\frac{1}{6}} = 0.5455$$

$$P(\text{large} \mid \text{rivet}) = \frac{\frac{1}{3} \cdot \frac{1}{6}}{\frac{1}{4} \cdot \frac{1}{3} + \frac{1}{3} \cdot 2\frac{1}{2} + 3\frac{1}{6} \cdot 6\frac{1}{6}} = 0.1818$$

Since P (medium | rivet) > P (small | rivet) and
P (medium | rivet) > P (large | rivet)

we classify bolt as belonging to the class medium and the probability of error P (error | rivet) = 1 − 0.5455 = 0.4545

Naive Bayes Classifier

Naive Bayes Classifier

- A naive bayes classifier is based on applying Bayes theorem to find the class of a pattern.
- The assumption made here is that every feature is class conditionally independent.
- Due to this assumption, the probabilistic classifier is simple.
- In other words, it is assumed that the effect of each feature on a given class is independent of the value of other features.
- Since this simplifies the computation, though it may not be always true, it is considered to be a naive classifier.

- Even though this assumption is made, the Naive Bayes Classifier is found to give results comparable in performance to other classifiers like neural network classifiers and classification trees.

- Since the calculations are simple, this classifier can be used for large databases where the results are obtained fast with reasonable accuracy.

- Using the minimum error rate classifier, we classify the pattern X to the class with the maximum posterior probability $P(c \mid X)$. In the naive bayes classifier, this can be written as

  $P(C \mid f_1, \ldots, f_d)$.
  where $f_1, \ldots, f_d$ are the features.

- Using Bayes theorem, this can be written as

  $$P(C \mid f_1, \ldots, f_d) = \frac{P(C)\ P(f_1,\ldots,f_d) \mid C}{p(f_1,\ldots,f_d)}$$

  Since every feature f_i is independent of every other feature f_j, for $j \neq i$, given the class
  $P(f_i, f_j \mid C) = P(f_i \mid C) P(f_j \mid C)$

  So we get,

  $$P(C, f_1, \ldots, f_d) = P(C)\, P(f_1 \mid C)\, P(f_2 \mid C) \cdots p(f_d \mid C)$$

  $$= p(C) \prod_{i=1}^{d} p(f_i|C).$$

  The conditional distribution over the class variable C is

  $$p(C \mid f_1, \ldots, f_n) = \frac{1}{Z} p(C) \prod_{i=1}^{n} p(f_i|C)$$

  where Z is a scaling factor.

- The Naive Bayes classification uses only the prior probabilities of classes P(C) and the independent probability distributions $p(f_i \mid C)$.

Parameter Estimation

- In supervised learning, a training set is given. Using the training set, all the parameters of the bayes model can be computed.
- If n_C of the training examples out of n belong to Class C , then the prior probability of Class C will be

$$P(C) = \frac{n_C}{n}$$

- In a class C , if n_1 samples take a range of values (or a single value) out of a total of n_C samples in the class, then the prior probability of the feature being in this range in this class will be

$$P(f_1 \text{ is in range } (a,b)) = \frac{n_1}{n_C}$$

In case of the feature taking a small number of integer values, this can be calculated for each of these values. For example, it would be

$$P(f_1 \text{ is } 6) = \frac{n_2}{n_C}$$

if n_2 of the n_C patterns of Class c take on the value 6.

- If some class and feature never occur together, then that probability will be zero. When this is multiplied by other probabilities, it may make some probabilities zero. To prevent this, it is necessary to give a small value of probability to every probability estimate.
- Let us estimate the parameters for a training set which has 100 patterns of Class 1, 90 patterns of Class 2, 140 patterns of Class 3 and 100 patterns of Class 4. The prior probability of each class can be calcu-lated.
  The prior probability of Class 1 is

$$P(C_1) = \frac{100}{100+90+140+100} = 0.233$$

The prior probability of Class 2 is

$P(C_2) = \frac{90}{100+90+140+100} = 0.210$

The prior probability of Class 3 is

$P(C_2) = \frac{140}{100+90+140+100} = 0.326$

The prior probability of Class 4 is

$P(C_2) = \frac{100}{100+90+140+100} = 0.233$

Out of the 100 examples of Class 1, if we consider a particular feature f_1 and if 30 patterns take on the value 0, 45 take on the value 1 and 25 take on the value 2, then the prior probability that in Class 1 the feature f_1 is 0 is

$P(f_1 \text{ is } 0) = \frac{30}{100} = 0.03$

The prior probability that in Class 1 the feature f_1 is 1 is

$P(f_1 \text{ is } 1) = \frac{45}{100} = 0.45$

The prior probability that in Class 1 the feature f_1 is 2 is

$P(f_1 \text{ is } 2) = \frac{25}{100} = 0.25$

Example for Naive Bayes Classifier

Let us take an example dataset.

Consider the example given in decision trees given in the Table 1. We have a new pattern

money = 90, has-exams=yes, and weather=fine

We need to classify this pattern as either belonging to goes-to-movie=yes or goes-to-movie=no.
There are four examples out of 11 belonging to goes-to-movie=yes.
The prior probability of P(goes-to-movie=yes)= $\frac{4}{11} = 0.364$

The prior probability of P(goes-to-movie=no) = $\frac{7}{11} = 0.636$

There are 4 examples with money50 − 150 and goes-to-movie=no and 1 examples with money < 50 and goes-to-movie=yes. Therefore,

Money	Has-exams	weather	Goes-to-movie
25	no	fine	no
200	no	hot	yes
100	no	rainy	no
125	yes	rainy	no
30	yes	rainy	no
300	yes	fine	yes
55	yes	hot	no
140	no	hot	no
20	yes	fine	no
175	yes	fine	yes
110	no	fine	yes

Table 1: Example training data set

P (money50 − 150 | goes − to − movie = yes) = $\frac{1}{4}$ = 0.25 and

P (money50 − 150 | goes − to − movie = no) = $\frac{4}{7}$ = 0.429

There are 4 examples with has-exams=yes and goes-to-movie=no and 2 examples with has-exams=yes and goes-to-movie=yes. Therefore,

P (has − exams | goes − to − movie = yes) = $\frac{2}{4}$ = 0.5 P

(has − exams | goes − to − movie = no) = $\frac{4}{7}$ = 0.429

There are 2 examples with weather=fine and goes-to-movie=no and 2 examples with weather=fine and goes-to-movie=yes. Therefore,

P (weather = f ine | goes − to − movie = yes) = $\frac{2}{4}$ = 0.5 P

(weather = f ine | goes − to − movie = no) = $\frac{2}{7}$ = 0.286

Therefore

P (goes − to − movie = yes | X) = 0.364 * 0.25 * 0.5 * 0.5 = 0.023

P (goes − to − movie = no | X) = 0.636 * 0.429 * 0.429 * 0.286 = 0.033

Since P (goes−to−movie = no | X) is larger, the new pattern is classified as belonging to the class goes-to-movie=no.

Introduction to Decision Trees

Decision Trees

- A decision tree is a tree structure which is used to make decisions at the nodes and reach some outcome at the leaf nodes.
- Decision trees are used for classification where each path is a set of decisions leading to one class.
- Decision trees are used in decision analysis, where the decision tree visually represents decision making.
- In pattern classification, the variable to be used for splitting at each node is to be determined.
- A path from the root to a leaf node represents a set of rules which leads to a classification.

Example of Decision Tree for Problem Solving

- The decision tree can be used for problem solving.
- At every node in the decision tree, a decision is taken depending on the outcome. It takes one of the branches from the node depending on the outcome.
- Each leaf node of the decision tree pertains to one of the decisions possible.
- As an example, we consider having a salt whose basic radical has to be found. The decision tree is shown in Figure 1.
- By doing a number of tests and looking at the outcome of the tests, it is possible to check if the basic radical of the salt belongs to Group I, Group II, Group III, Group IV, Group V or Group VI.
- The decision tree gives a clear description of the procedure to be fol-lowed to find the solution.
- It can been that there are six leaf nodes corresponding to the identifi-cation of the salt.

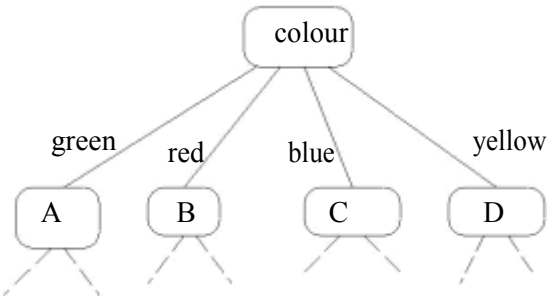

Figure 2: Four way decision

- Each path from the root to a leaf node defines one path in the decision tree. This corresponds to a rule. For example one rule would be :
  If ((salt+water+ dil HCl gives precipitate=n) and (salt+water+dil H_2S gives precipitate=n) and (salt+water+$NH_4Cl + NH_4OH$ gives white precipitate=y)) then (salt ϙ Group III basic radical)

- The different leaf nodes have different path lengths. The leaf node belonging to Group I basic radical requires just one test or has only one decision node. Group II basic radical requires two tests and so on.

- In this way, by forming the decision tree, problem solving can be carried out.

Features of Decision Trees

- The following are some of the features of the decision tree.

  The nodes of the tree represent decisions leading to two or more branches or outcomes. If the nodes have only two decisions at every node, it is a binary decision tree. However, a multiway decision can be always split up into a number of two-way (or binary) decisions. Therefore a binary decision tree is sufficient to represent any situation. This is illustrated in Figures 2 and 3. Figure 2 shows a situation in a decision tree where there is a four-way decision depending on colour. This has been replaced by three binary decisions as shown in Figure 3.

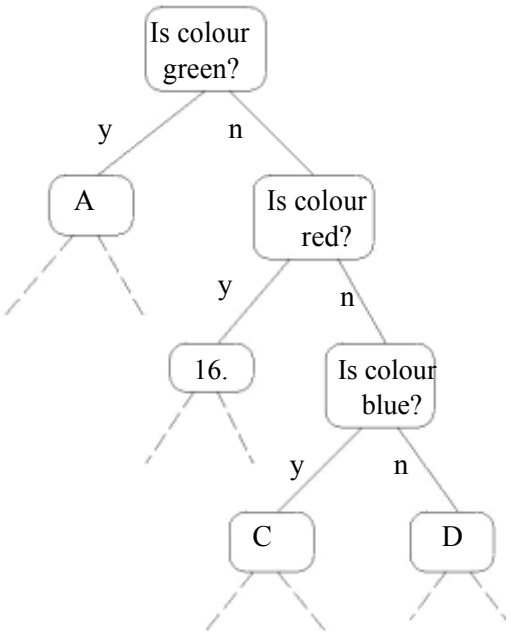

Figure 3: Binary decision

- X The decision at each node is done using attributes which may take values which are numerical or categorical. The numerical values may be integers or real numbers. By categorical attributes, we mean attributes which are described by using words. An example would be, if we have colour as an attribute and it can take values such as brown, red, green, etc.
- X The decision taken at a node may involve just one attribute or more attributes. If one attribute is involved, it is called a univari-ate decision. This results in a axis-parallel split. If more than one attribute is involved, it is called a multivariate split.

Decision Tree for Pattern Classification

\endash Decision trees can be used for pattern classification where by making decisions on the attribute values at each node, we can identify the class of the pattern.

\endash The nodes represent the decisions and each leaf node represents a class.

\endash The decision at each node may involve more than one feature and is called multivariate or may involve just one feature and is called uni-variate.

\endash The outcome at each node may be two or more. If every node has only two outcomes, it is a binary tree.

\endash The features or attributes can be either numerical or categorical.

\endash The number of classes involved can be two or more.

\endash Use of decision trees for pattern classification is shown in Figure 4. Here it is a two class problem, where the class is true or false, i.e. whether Ajay goes to the movies or not. The features of this problem are

The amount of money Ajay has in his pocket

Whether Ajay has exams or not

Is the weather hot, fine or rainy

\endash The first feature is a numerical feature, the second feature is a boolean feature and the third feature is a categorical feature. Hence, it can be seen that decision trees for pattern classification handle numerical, boolean and categorical features.

\endash It is to be noted that the first decision node has a three-way split. The decision node based on whether Ajay has exams or not is a binary split and the node on the weather has a three-way split.

\endash Every leaf node of the tree is associated with a class and every path from the root to the leaf gives a classification rule. For example, If (Has money = 50-150) and (Has exams = true) then (Goes to a movie
false).

Computer Programs for Learning Decision Trees

\endash Several programs are available for decision tree learning.

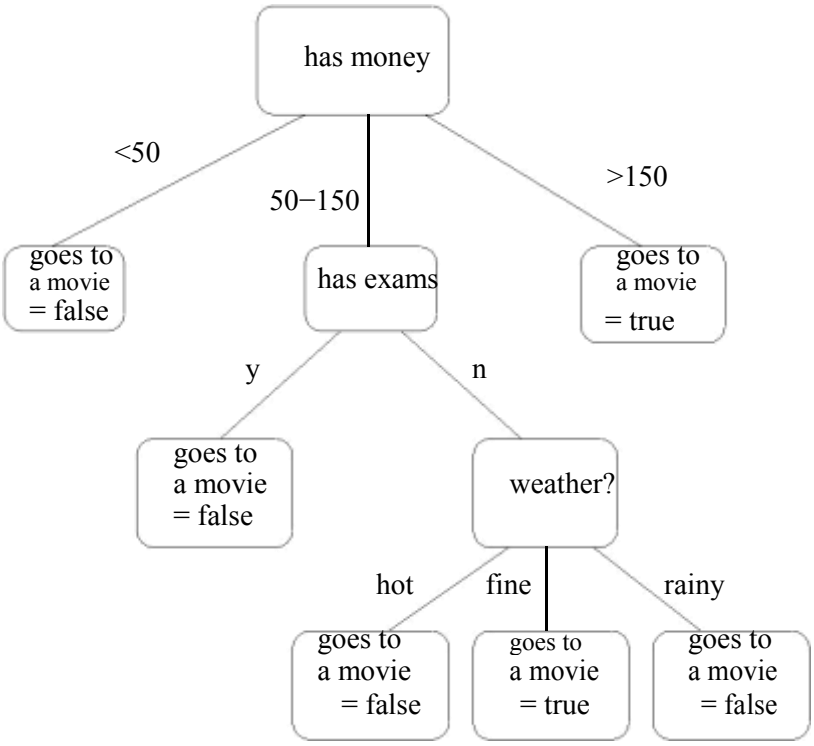

Figure 4: Decision tree for pattern classification

X Some of them are

- ID3
- C4.5 . This program is developed by Quinlan and is an improve-ment of ID3.
- CART

Advantages of Decision Trees

8. The rules are simple and easy to understand. It consists of a set of decisions followed by an outcome.
9. Able to handle both categorical and numerical data.
10. Easy to explain the outcome by observing the decision tree.
11. It is robust and performs quite well even if it deviates from the true model.
12. Can be used for classification of large amounts of data in a short time.

Limitations of Decision Trees

X Learning an optimal decision tree is NP-complete in some cases. For example, the number of oblique splits possible could be exponential in the number of training patterns. At each split point, the complexity of selecting an optimal oblique hyperplane is exponential. In some problems, axis- parallel decision trees do not represent non-rectangular regions well.

X Complex trees can be created due to overfitting. Pruning is required to avoid this problem.

X Some problems are hard to solve using decision trees like the XOR problem, parity problem and the multiplexer problem. In these cases, the decision tree becomes prohibitively large.

LINEAR DISCRIMINANT FUNCTIONS

INTRODUCTION TO DISCRIMINANT FUNCTIONS

Linear Discriminant Functions

- One popular way of separating patterns belonging to two classes is by using a simple decision boundary.
- We illustrate this using the collection of two-dimensional patterns shown in Figure 1.

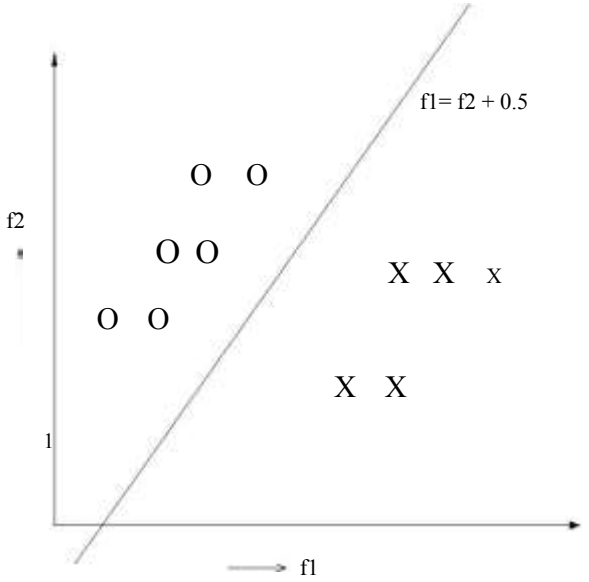

Figure 1: Classification using a Linear Discriminant Function

- Two-dimensional example data

Example 1

There are five patterns from one class (labelled 'X') and six patterns from the second class (labelled 'O'). This set of labelled patterns may be described using Table 1.

- Linear separability
  The first six patterns are from class 'O' and the remaining patterns

Pattern No.	feature$_1$	feature$_2$	Class
1	0.5	1.5	'O'
2	1.0	1.5	'O'
3	1.0	2.0	'O'
4	1.5	2.0	'O'
5	1.5	2.5	'O'
6	2.0	2.5	'O'
7	3.0	1.0	'X'
8	3.5	1.0	'X'
9	3.5	2.0	'X'
10	4.0	2.0	'X'
11	4.5	2.0	'X'

Table 1: Description of the patterns

are from class 'X'. Consider the line represented by $f_1 = f_2 + 0.5$ shown in Figure 1. All the patterns labelled 'O' are to the left of the line and patterns labelled 'X' are on the right side. In other words, patterns corresponding to these two classes are linearly separable in this example as patterns belonging to each of the classes fall on only one side of the line.

- Positive and negative half spaces

  One more way of separating the patterns of the two classes is by noting that all the patterns of class 'O' satisfy the property that $f_1 < f_2 + 0.5$ or equivalently $f_1 - f_2$ is less than 0.5. For example, pattern 1 in Table 1 has a value of 1.5 for feature$_2$ and 0.5 for feature$_1$ and so this property is satisfied (because 0.5 - 1.5 = -1.0 (< 0.5)). Similarly, the sixth pattern satisfies this property because $f_1 - f_2$ is $-$ 0.5. In a symmetric manner, all the patterns belonging to class 'X' have the value of f_1 to be larger than that of $f_2 + 0.5$; equivalently, $f_1 - f_2 > 0.5$. For example, for the seventh pattern in the table, $f_1 - f_2$ is 2.0 (3.0 - 1.0) which is greater than 0.5. Based on the location of the patterns with respect to the line, we say that patterns of class 'X' are on the positive side of the line (because $f_1 - f_2 > 0.5$ for these patterns) and patterns of class 'O' are on the negative side (equivalently $f_1 - f_2 < 0.5$ for patterns labelled

'O'). Here, the line separates the two-dimensional space into two parts which can be called as positive half space (where patterns from class 'X' are located) and negative half space (which has patterns from class 'O').

- Variety of separating lines

  It is easy to observe that there are possibly infinite ways of realizing the decision boundary, a line in this two-dimensional case, which can separate patterns belonging to the two classes. For example, the line $f_1 = f_2$ also separates the first six points in Table 1 from the remaining 5 points.

- Functional form of the linear decision boundary

  It is convenient to abstract such lines using the following functional form:

$$f(X) = w_1 f_1 + w_2 f_2 + b = 0 \quad (1)$$

  Correspondingly, the line $f_1 - f_2 = 0$ has $w_1 = 1$; $w_2 = -1$; and $b = 0$. Similarly, $w_1 = 1$; $w_2 = -1$; and $b = -0.5$ for $f_1 - f_2 = 0.5$. This representation permits us to deal with patterns and decision boundaries in the multi-dimensional space. For example, in a d-dimensional space, the decision boundary is a hyperplane and it can be represented by

$$f(X) = w^T X + b = 0 \quad (2)$$

  where w and X are d-dimensional vectors.

- Separation of patterns based on w and b

  We can use this representation to characterize linear separability. We say that two classes, say classes labelled 'X' and 'O' are linearly sepa-rable, if we can find a weight vector w and a scalar b such that

  $w_T X + b > 0$ for all patterns X belonging to one class (say class 'X') and

  $w_T X + b < 0$ for all the patterns belonging to the other class (that is class 'O').

- Non-linear decision boundary

  Another possibility is to use a non-linear decision boundary to characterize the separation between the two classes. For example, the non-linear decision boundary

$$f_1 = -f_2^2 + 5f_2 - 3$$

  is depicted in Figure 2.

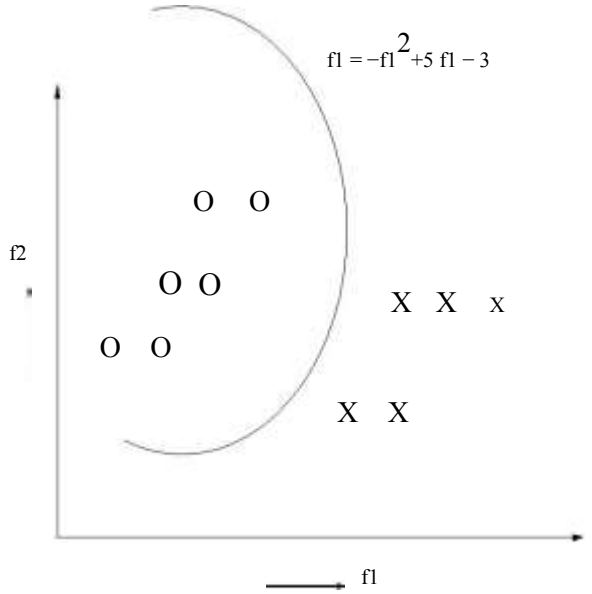

Figure 2: Classification using a Nonlinear Discriminant Function

- Location of the decision boundary

  Here, the value of b plays a role in deciding the location of the decision boundary. Note that the decision boundary between the two classes is characterized by $w_T X + b = 0$. Based on the location of the origin (the zero vector), we can make the following observations with respect to the value of b.

  Consider the situation where $b = 0$. In such a case, the origin lies on the decision boundary. This is because $w_T X + b = 0$ for $X = 0$ (origin) and $b = 0$, for any value of w.

When the value of b is negative, that is when $b < 0$, the origin lies in the negative side; this can be observed from the fact that $w_T X + b < 0$ when $X = 0$ and $b < 0$, for any value of w.

The above observations hold in general in a d-dimensional space. We examine them with respect to the two-dimensional data set ($d = 2$) shown in the example below.

Example 2

Consider the data shown in Figure 1 and Table 1. Let us consider the line $f_1 = f_2$; here, $w^T = (1,-1)$ and $b = 0$. The origin ((0,0)) is on the line $f_1 = f_2$ which is the decision boundary.

Now consider the line $f_1 = f_2 + 0.5$; this can also be written as $2f_1 - 2f_2 - 1 = 0$. This line is obtained by shifting $f_1 - f_2 = 0$ down appropriately. Note that the points (0.5,0) and (0,-0.5) are on this line. Also, all the points labelled 'X' are on one side (positive side) and those labelled 'O' are located on the negative side of the line. Note that the origin is in the negative part.

- Solving for the vector w

Now consider the decision boundary charcaterized by

$$f(X) = w_1 f_1 + w_2 f_2 + b = 0$$

As noted earlier, (0.5,0) and (0,-0.5) are on the decision boundary. We can get the values of w_1 and w_2 by solving the equations obtained by substituting these points in the equation of the decision boundary.

By substituting (0.5,0), we get

$$0.5w_1 + 0.0w_2 + b = 0 \Rightarrow 0.5w_1 + b = 0 \quad (3)$$

Similarly, by substituting (0,-0.5), we get

$$0.0w_1 - 0.5w_2 + b = 0 \Rightarrow -0.5w_2 + b = 0 \quad (4)$$

By subtracting (4) from (3), we get

$$0.5w_1 + 0.5w_2 = 0 \Rightarrow w_1 + w_2 = 0 \Rightarrow w_1 = -w_2 \quad (5)$$

- The role of w

  One possible instantiation is $w_1 = 1$; $w_2 = -1$; so, the value of b is -0.5 from both (3) and (4). The other possibility is $w_1 = -1$; $w_2 = 1$; so, the value of b is 0.5. Out of these two possibilities, only the first is acceptable; the second is not. We can verify this by considering a positive example from Table 1. Let us consider the example $(3, 1)^T$. In this case, we need $w^T x + b > 0$; that is by using the first set of values, we have

$$1 * f_1 - 1 * f_2 - 0.5 = 3 - 1 - 0.5 = 1.5 > 0$$

  Now, by using the second set of values, we have an inconsistency as

$$-1 * x_1 + 1 * x_2 + 0.5 = -3 + 1 + 0.5 = -1.5 < 0$$

- w is orthogonal to the decision boundary

  So, we accept the first set of values; this means that the vector w is orthogonal to the decision boundary as exemplified by equations (3),

  (4), and (5). In general, if p and q are two points on the decision boundary, then we have

$$w^T p + b = 0 = w^T q + b \Rightarrow w^T (p - q) = 0$$

  So, the vectors w and $(p - q)$ are orthogonal to each other; observe that $(p - q)$ characterizes the direction of the decision boundary.

- w points towards the positive half space

  Further w points towards the positive half space. This can be verified by noting that the angle, θ, between w and any positive pattern vector is such that $-90 < \theta < 90$. As a consequence $\cos(\theta)$ is positive.

Learning the Discriminant Function

Learning the Linear Discriminant Functions

- **Learning the Discriminant Function**

  It is convenient to use the normalized data for automatically learning the decision boundary when the classes are linearly separable. In gen-eral, it is possible to obtain the decision boundary in the form of a hyperplane if we learn the vector w. It is simpler to learn the weight vector w when the classes are linearly separable. We describe an al-gorithm, called **Perceptron learning algorithm**, for learning the weight vector when the classes are linearly separable.

- **Learning the weight vector**

  This algorithm considers patterns sequentially and updates the weight vector w if a pattern is misclassified by the weight vector. It iterates through the set of patterns till there is no updation, or equivalently no pattern is misclassified during an entire iteration. It is convenient to consider patterns in the normalized form while applying this algo-rithm. This would mean that a pattern X is misclassified if $w^tX \leq 0$. In order to use the homogeneous form, we need to consider patterns after transformation and normalization. For example, the set of 3-dimensional vectors corresponding to the patterns given in lesson 26 is given in Table 1. The problem is to find a 3-dimensional weight vector w which classifies all the patterns correctly. Equivalently, w should be such that w^tX is greater than 0 for all the patterns; for example for all the patterns in Table 1.

Pattern No.	$feature_1$	$feature_2$	$feature_3$
1	- 1.0	- 1.5	- 1.0
2	- 1.5	- 2.0	- 1.0
3	- 1.5	- 2.5	- 1.0
4	- 2.0	- 2.5	-1.0
5	3.0	1.0	1.0
6	3.5	2.0	1.0
7	4.0	2.0	1.0

Table 1: Description of the normalized patterns

- **Perceptron Learning Algorithm**

  Now we describe an algorithm, called perceptron learning algorithm, for learning w. Let the normalized patterns in the d + 1-dimensional space be $X_1, X_2, \cdot \cdot \cdot, X_n$.

  \endash Initialize w by choosing a value for w_1 (for simplicity it is adequate to choose $w_1 = 0$).

  \endash For j = 1 to n do
  if $w_i^t X_j \leq 0$ (that is when the normalized pattern X_j is misclassified) then i = i+1 and $w_i = w_{i-1} + X_j$ (update the current w vector to classify X_j better).

  \endash Repeat step 2 till the value of i has not changed during the entire iteration (that is the current w vector classifies all the training patterns correctly).

- **Updation of Weight Vector**

  It is possible to explain the intuition behind the proposed scheme for updating the weight vector as follows. Let w_i be the current weight vector which has misclassified the pattern X_j ; that is $w_i^t X_j < 0$. So, using the above algorithm, we have

  $$w_{i+1} = w_i + X_j$$

  Note that

  $$w_{i+1}^t X_j = w_i^t X_j + || X_j ||^2$$

  As a consequence, $w_{i+1}^t X_j$ is larger than $w_i^t X_j$ by $|| X_j ||^2$ because $\bullet$ $X_j ||^2$ is positive. This would mean that the updated vector w_{i+1} is better suited, than w_i, to classify X_j correctly because $w_{i+1} X_j$ is larger than $w_i X_j$ and so can be positive even if $w_i X_j$ were not.

  Example 1

  We can illustrate the working of the algorithm using the data in Table 1 with the help of the following steps.

  - $w_1 = (0, 0, 0)^t$ and $X_1 = (-1.0, -1.5, -1.0)^t$. Here, $w_1^t X_1 = 0$ and so $w_2 = w_1 + X_1$ which is represented by

    $$w_2 = (-1.0, -1.5, -1.0)^t.$$

17. Next we consider pattern X_2 and $w_2^t X_2$ is 5.0 (> 0). Similarly, X_3, and X_4 also are properly classified.

Note that $w_2^t X_3$ = 6.25, and $w_2^t X_4$ = 6.75. So, these patterns do not affect the weight vector.

18. However, $w_2^t X_5$ = -5.5 (< 0). So, $w_3 = w_2 + X_5$, that is

$$w_3 = (2.0, -0.5, 0.0)^t.$$

Note that w_3 classifies patterns X_6, and X_7 correctly.

X However x_1 is misclassified by w_3. Note that $w_3^t X_1$ is $-$ 1.25(< 0). So, $w_4 = w_3 + X_1$ is obtained. It is given by

$$w_4 = (1.0, -2.0, -1.0)^t.$$

w_4 correctly classifies X_2, X_3, and X_4.

5. However, w_4 misclassifies X_5 as $w_4^t X_5$ = 0.0 (≤ 0.0). So, w_5 = w_4 + X_5, that is

$$w_5 = (4.0, -1.0, 0.0)^t.$$

w_5 classifies X_6 and X_7 correctly.

6. Note that w_5 misclassifies X_1 as $w_5^t X_1$ = $-$ 2.5 < 0. So, w_6 = w_5 + X_1, that is

$$w_6 = (3, -2.5, -1)^t.$$

w_6 classifies X_2, X_3, X_4, X_5, X_6, X_7, and X_1. Specifically, $w_6^t X_2$ = 3, $w_6^t X_3$ = 2.75, $w_6^t X_4$ = 1.25, $w_6^t X_5$ = 5.5, $w_6^t X_6$ = 4.5, $w_6^t X_7$ = 6, and $w_6^t X_1$ = 1.75.

So, the algorithm converges to w_6 which is the desired vector. In other words, $3X_1 - 2.5X_2 - 1 = 0$ is the equation of the decision boundary; equivalently, the line separating the two classes is $6X_1 - 5X_2 - 2 = 0$

\endash **Convergence of the Perceptron Algorithm**

In general, it is possible to show that the perceptron learning algorithm will converge to a correct weight vector in a finite number of iterations when the classes are linearly separable. The number of iterations may increase based on the location of the training patterns.

\endash **Updation of the w vector**

Let us consider the following. Let the training set of patterns, from two classes, after transformation and normalization be $\{\chi_1, \chi_2, \cdots, \chi_n\}$ and let $w_1 = 0$. Let us say that the first pattern, if any, misclas-sified by w_1 be χ^1 which means that $w_1{}^t\chi^1 \leq 0$; further $\chi^1 \in \{\chi_1, \chi_2, \cdots, \chi_n\}$. Now we get $w_2 = w_1 + \chi^1$. Let χ_k be the pattern misclassified by w_2; update w_2. In this manner let χ^2 be mis-classified by w_k. Note that $w_{k+1} = w_k + \chi^k$.

\endash **Linear Separability**

If the classes are linearly separable, then there exists a weight vector w such that $w^t\chi > 0$ for every training pattern χ, irrespective of its class label, because of normalization. Now consider w_{k+1} which is given by

$$w_{k+1} = w_1 + \chi^1 + \chi^2 + \cdots + \chi^k \tag{1}$$

Now

$$w^t w_{k+1} = w^t(w_1 + \chi^1 + \chi^2 + \cdots + \chi^k) > k\delta \tag{2}$$

because $w_1 = 0$ and where $\delta = min\{w^t\chi_i\}$ over all χ_i in the training set. Further,

$$w_k{}^t{}_{+1}w_{k+1} = (w_k + \chi^k)^t(w_k + \chi^k) = ||\, w_k\,||^2 + ||\, \chi^k\,||^2 + 2w_k{}^t\chi^k$$

Note that $w_k{}^t\chi^k \leq 0$ and if $\gamma = max\{||\, \chi^k\,||^2\}$, then

$$||\, w_{k+1}\,||^2 < k\gamma \tag{3}$$

This is obtained by expanding recursively w_k in terms of w_{k-1} and χ^{k-1}.

X **Cosine of the angle between w and w_{k+1}**

Let ϑ be the angle between w and w_{k+1}. So, using (3) and (4), we have

$$\cos\vartheta = \frac{w^t w_{k+1}}{||\, w_{k+1}\,||} > \frac{k\delta}{\sqrt{k\gamma}}$$

But we know that cos ϑ < 1. So, we have

$$\frac{\sqrt{k}\delta}{\sqrt{\gamma}} < 1 \Rightarrow k < \frac{\gamma}{\delta^2} \quad (4)$$

Note that k, the number of iterations is bounded because γ is finite for finite size training vectors. Also, δ cannot be zero as the classes are linearly separable and $w^t x_i > 0$. So, in a finite number of iterations the algorithm converges. It is possible that k is very large if δ is close to zero; this can happen if one of the x_is is almost orthogonal to w.

• Multi-class Problems

A linear discriminant function is ideally suited to separate patterns belonging to two classes that are linearly separable. However, in real life applications, there could be a need for classifying patterns from **three** or more classes. It is possible to extend a binary classifier, in different ways, to classify patterns belonging to C classes where $C > 2$, We list two of the popularly used schemes below:

1. Consider a pair of classes at a time. Note that there are $\frac{C(C-1)}{2}$ such pairs when there are C classes in the given collection. Learn a linear discriminant function for each pair of classes. Combine these decisions to arrive at the class label.

X Consider two-class problems of the following type. For each class C_i, create the class C_i which consists of patterns from all the remaining classes. So, $C_i = \bigcup_{j=1;j\neq i}^{C} C_j$. Learn a linear discriminant function to classify each of these two-class problems. Note that there are C such two-class problems. These C linear discriminants give us the overall decision.

Support Vector Machines

Introduction to Support Vector Machines (SVMs)

- **SVM as a Linear Discriminant**

  In a simplistic sense, SVM is based on linear discriminant functions. So, the classification is based on a function of the form $w_T X + b$; w and b are learnt from the training data. In a two-class problem with a positive class and a negative class, for any pattern X from the positive class $w_T X + b > 0$ and $w_T X + b < 0$ for any pattern X from the negative class when the patterns from the two classes are linearly separable.

- **SVM as a Maximum Margin Classifier**

  Consider a two-class problem where the classes are linearly separable. Let the positive class be characterized by a hyperplane $w_T X + b = 1$ and the negative class by a parallel plane $w_T X + b = -1$ so that training patterns, near the margin, from the respective classes fall on these planes. These planes are called the support planes. The decision boundary or the separating plane is characterized by $w_T X + b = 0$. It may be illustrated using the two-dimensional data shown in Figure 1.

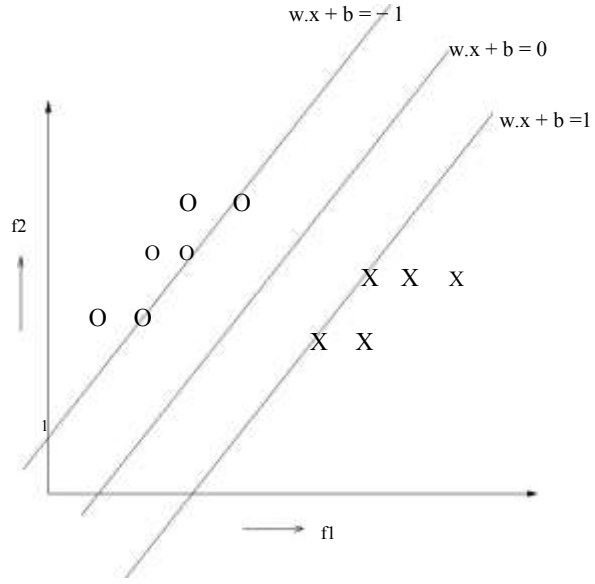

Figure 1: Support Lines Characterizing the Margin

- **Margin Between the two Lines**

  In the two-dimensional case, we have support lines, instead of planes, and the decision boundary also is a line as shown in the figure. Patterns labelled 'O' are from the negative class and those labelled 'X' are from the positive class. As discussed in lesson 26 (refer to equation (7)), the distance from any point X on the line $w_T X + b = 1$ to the decision boundary is given by

$$\frac{f(X)}{\| X \|} = \frac{1}{\| w \|}$$

  In the above, note that $f(X) = 1$ for any X on the line $w_T X + b = 1$.

  Similarly, the distance from a point on the line $w_T X + b = -1$ to the decision boundary is given by $\| \frac{-1}{w} \|$. So, the distance between the two supporting lines, or the margin, is $\frac{2}{\|w\|}$. Observe that this expression for margin holds good even in the d- dimensional ($d > 2$) case.

- **Maximizing the Margin**

  Maximizing the margin is achieved by $\frac{\|w\|}{2}$ maximizing $\|w^2\|$ ~~or equivalently~~

  by finding a w that minimizes or for the sake of simplicity in calculus, $\frac{\|w\|^2}{2}$. Optimization is carried out by using constraints of the form $w_T X + b \leq -1$ for all X in the negative class and $w_T X + b \geq 1$ for all X in the positive class.

- **SVM and Data Reduction**

  SVM selects the training patterns falling on the support planes. Such vectors are called support vectors. These vectors are adequate to learn both w and b. This amounts to data reduction or compression.

- **Non-linear decision boundaries**

  It is possible that the two classes are not linearly separable. In such a case, the decision boundary can be non-linear. In a generalized sense, a non-linear function can be represented by a linear function. We discuss this issue next.

- **Generality of Linear Discriminants**

  The notion of linear discriminant functions is very general. This idea may be extended to handle even non-linear discriminants using the

homogeneous representation. For example, consider the non-linear decision boundary

$$f_1 = -f_2^2 + 5f_2 - 3 = 0 \text{ as}$$

shown in Figure 2.

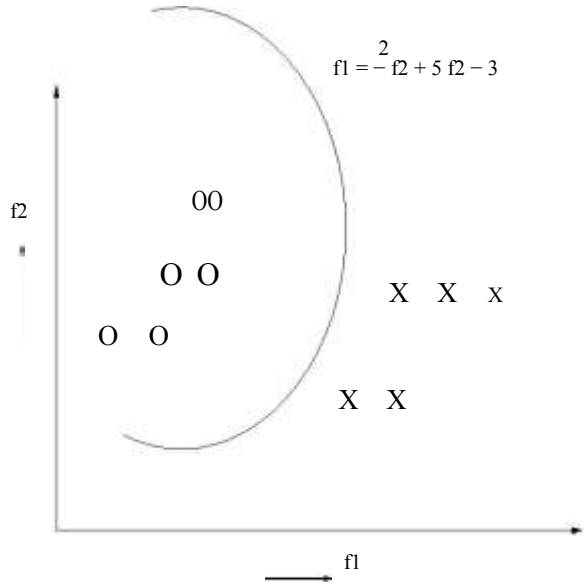

Figure 2: Classification using a Nonlinear Discriminant Function

- **Representing a Non-Linear Function Using a Linear Form**

  Such a non-linear function may be represented using the homogeneous form

$$f(X') = w^T X' = 0 \qquad (1)$$

where

$$w = \begin{matrix} -1 \\ 5 \\ -3 \end{matrix}$$

$$X' = \begin{matrix} f_2^2 \\ f_2 \\ 1 \end{matrix}$$

It may be illustrated using the following example.

- **Learning the generalized linear discriminant function**

  Consider a binary classifier which assigns x to class 'O' (negative class) if $f(x) < 0$ and to class 'X' (positive class) if $f(X) > 0$, where

$$f(x) = a + bx + cx^2 \tag{2}$$

  Observe, based on the discussion above, that equivalently we can assign

  • to 'O' if $w_T X' < 0$ and to class 'X' if $w_T X' > 0$, where

$$w = \begin{matrix} a \\ b \\ c \end{matrix}$$

$$X' = \begin{matrix} 1 \\ x \\ x^2 \end{matrix}$$

Let us consider a set of labelled patterns that are not linearly separa-ble. Specifically, let us consider the one-dimensional data set shown in Table 1. Observe that the data is not linearly separable. Further, the

Pattern No.	x	class
1	-1	'O'
2	-2	'O'
3	0	'O'
4	1	'X'
5	3	'X'
6	5	'O'

Table 1: Non-Linearly Separable Classes

decision boundary is characterized by $f(x) = 0$. Now by appropriate transformation and normalization, where the components in X' are 1, value of x, and value of x^2, we get the data shown in Table 2. We can use the perceptron learning algorithm to obtain the weight vector w.

Pattern No.	1	x	x^2
1	-1	1	-1
2	-1	2	-4
3	-1	0	0
4	1	1	1
5	1	3	9
6	-1	-5	-25

Table 2: Normalized non-linearly separable data

1. Here, we start with

$$w_1 = \begin{matrix} 0 \\ 0 \end{matrix}$$

and

$$x_1' = \begin{matrix} -1 \\ 1 \\ -1 \end{matrix}$$

w_1 misclassifies X_1'. So, gets updated to get $w_2 = w_1 + X_1'$.

19. Continuing with the algorithm, we end up with w_{70} which classifies all the 6 patterns in Table 2 correctly. It is given by

$$w_{70} = \begin{matrix} -1 \\ 35 \\ -11 \end{matrix}$$

So, the decision boundary is given by

$$f(x) = -1 + 35x - 11x^2 = 0 \qquad (3)$$

This example illustrates the generality of the linear discriminants. It is possible to deal with non-linearly separable classes using a linear discriminant. Further, it is possible to extend this idea to vector-valued X also.

X SVM for Classification

Support vector machine (SVM) generates an abstraction in the form

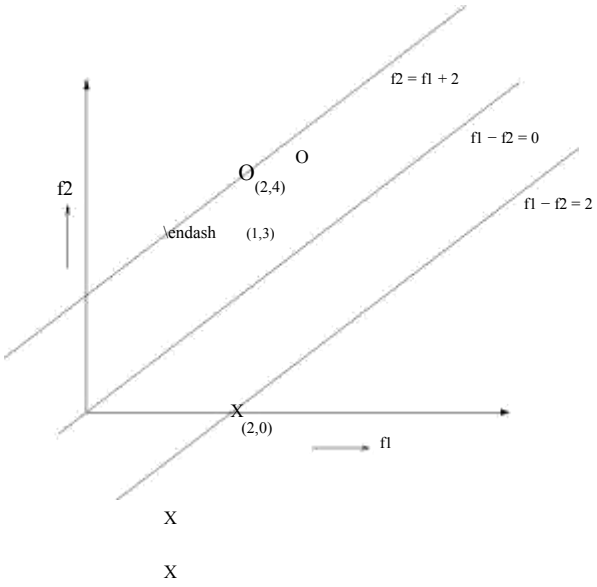

Figure 3: Illustration of Support Vector Machines

of a linear discriminant. It also selects a set of vectors called support vectors which are margin patterns and are members of the training set. We illustrate this using Figure 3.

\endash **Support Vectors**

Consider three of the points shown in Figure 3. These are from two classes ('X' and 'O'). Here, $(1, 3)^T$ and $(2, 4)^T$ are from class 'O' and $(2, 0)^T$ is from class 'X'. The lines $f_2 - f_1 = 2$ and $f_1 - f_2 =$

2 characterize the boundaries of the classes 'O' and 'X' respectively. These lines are the **support lines** and the points are the **support vectors**. These three support vectors are adequate to characterize the classifier.

\endash **Adequacy of support vectors**

Now consider adding points $(1, 4)^T$, $(1, 5)^T$ and $(2, 5)^T$ from class 'O' and $(2, -1)^T$, $(1, -2)^T$, and $(1, -3)^T$ from class 'X'. They are properly classified using the support lines (in turn using the support vectors). The region between the two lines is the margin and because the support lines correspond to maximum value of the margin, they are as far away

from each other as possible.

X Decision Boundary

The line $f_1 - f_2 = 0$ is equidistant from the two decision lines and it forms the right choice for the decision boundary between the two classes. Here, points satisfying the property that $f_1 - f_2 < 0$ are classified as members of class 'O' and those satisfying the condition $f_1 - f_2 > 0$ are of class 'X'.

X Properties of the SVM Classifier

From the above discussion, we make the following observations.

SVM may be viewed as a binary (two class) classifier. It abstracts a linear decision boundary from the data and uses it to classify patterns belonging to the two classes.

Support vectors are some of the vectors falling on the support planes in a d-dimensional space.

SVMs learn, from the data, linear discriminants of the form $w_T + b$ that corresponds to the maximum margin.

When the two classes are linearly separable, it is easy to compute the classifier characterizing the maximum margin.

X Linearly Separable Case

Consider again the points $(1, 3)_T$ and $(2, 4)_T$ from class 'O' and $(2, 0)_T$ from class 'X' shown in Figure 3. They are linearly separable. In fact, we can draw several (possibly infinite) lines separating correctly the two classes represented by these three points. The corresponding normalized data is shown in Table 3.

Pattern No.	$feature_1$	$feature_2$	bias
1	-1	-3	-1
2	-2	-4	-1
3	2	0	1

Table 3: Normalized set of patterns in three dimensions

13. Which Linear Discriminant?

Using Perceptron learning algorithm, we get the decision boundary characterized by $f_1 - 3f_2 = 0$. The initial weight vector, $w_1 = 0$ misclassifies $(-1, -3, -1)^T$. So, $w_2 = (-1, -3, -1)^T$. It misclassifies $(2, 0, 1)^T$. So, $w_3 = (1, -3, 0)^T$. Note that w_3 classifies all the three patterns properly and it corresponds to the decision boundary $f_1 - 3f_2 = 0$.

14. Decision Boundary of the SVM

In the case of the SVM, we choose that line as decision boundary which provides maximum margin. When the patterns are linearly separable, we can get the linear decision boundary corresponding to the maximum margin solution. For example, consider the three patterns in Figure 3. In this two-dimensional example, two points of a class are adequate to

characterize the support line. So, for the two points $(1, 3)^T$ and $(2, 4)^T$ from class 'O', the support line is $f_2 - f_1 = 2$. Now consider a line

parallel to this support line and passing through the point $(2, 0)^T$ from class 'X'; this is $f_1 - f_2 = 2$ and it is the support line for class 'X'. These two lines, namely $f_2 - f_1 = 2$ and $f_1 - f_2 = 2$ characterize

the margin. So, the decision boundary which is equidistant from these two lines is characterized by $f_1 - f_2 = 0$.

15. Learning the SVM

Consider two points $(2, 2)^T$ and $(1, 1)^T$ on the line $f_1 - f_2 = 0$. Because these points are on the decision boundary, we need $w = (w_1, w_2)^T$ to

satisfy

$2w_1 + 2w_2 + b = 0$ (corresponding to $(2, 2)^T$) and

$w_1 + w_2 + b = 0$ (for the point $(1, 1)^T$). From these two equations, we get $w_1 + w_2 = 0$; so, w is of the form $(\alpha, -\alpha)^T$ and correspondingly, b

X 0. In general, α is any constant. However, it is good to choose α in a normalized manner. For example, it is convenient to choose α such that

$w^T X + b = 0$, or equivalently, $w^T X = 0$ for all the points on the decision

boundary;

$w^T X = 1$ for all the points on the support line $f_1 - f_2 = 2$; and $w^T X = -1$ for all the points on the support line $f_2 - f_1 = 2$

This can be achieved by choosing a value of $\frac{1}{2}$ for α. So, correspondingly

$w = (\frac{1}{2}, -\frac{1}{2})^T$ and $b = 0$.

Clustering

What is Clustering?

Clustering

- Clustering is the Task of Grouping Data based on Similarity It is the process of partitioning a set of patterns.

  Example 1

  Let us consider the following collection of nine characters:

  bA . B a , C ′c

  After partitioning, we get the following three clusters of characters:

  . , ′ b a c A B C

  Here, clustering is done based on size. The first cluster has the smallest sized characters; the second one has lower-case characters and the third cluster has three upper-case letters. Thus, clustering means grouping of data or dividing a large dataset into smaller data sets of some similarity.

- Clustering is Data Organization We illustrate this idea with an example.

  Example 2

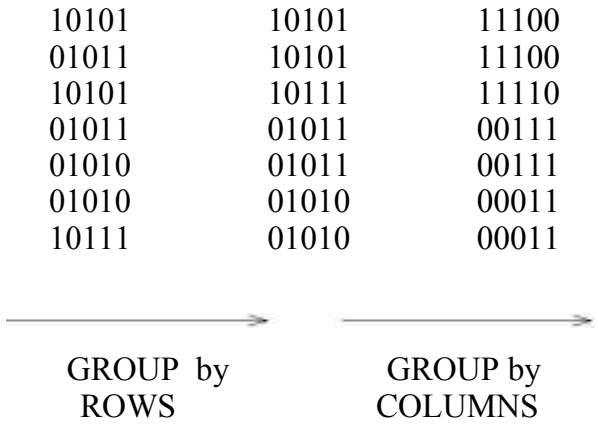

Figure 1: Clustering is Data Organization

Data Organization

Consider Figure 1. There are 7 patterns; each pattern is characterized by 5 binary features. We use the Hamming distance between patterns to organize the data. Hamming distance is the number of mismatch-ing bits between two patterns. For example, the Hamming distance between ‘10101’ and ‘01011’ is 4 as the first four values do not match whereas the distance between ‘10101’ and ‘10111’ is 1 as there is a mismatch in the fourth bit.

– Clustering rows

Initially, we group the seven patterns (rows of the pattern matrix) which is called group by rows in the figure. In fact we rearrange the patterns such that similar patterns are placed next to each other; similarity between a pair of patterns is larger if the Hamming distance between them is smaller. The patterns are given in the second column of the figure after rearranging.

– Clustering columns

The columns in these patterns are further rearranged based on similarity between columns. Here again we used the Hamming distance between columns to rearrange; each column is a binary vector of length 7. For example, distance between the first column ‘1 1 1 0 0 0 0’ and the fifth column ‘1 1 1 1 1 0 0’ is 2 as there is a mismatch in the fourth and fifth bit positions.

– Structure in the data

The data resulting after rearranging based on columns is shown in the third column of the figure. Observe that such a rearrange-ment shows the structure in the data better than the original data (shown in the first column of the figure). Note the closeness of 1s in the first three rows in the left side columns and also in the last 4 rows.

- Clustering is Partitioning a Dataset

We illustrate this notion using a set of patterns shown in the following example. We consider a collection of nine 3-dimensional patterns here.

Example 3

Consider the set of points, X *, given by*

- $= \{(1, 1, 1), (1, 1, 2), (1, 2, 1), (2, 1, 1), (6, 3, 1), (6, 4, 1)\ (6, 6, 6), (6, 6, 7), (7, 7, 7)\}$

A partitioning of X *into* C_1, C_2, *and* C_3 *is*

$$C_1 = \{(1, 1, 1), (1, 1, 2), (1, 2, 1), (2, 1, 1)\}$$

$$C_2 = \{(6, 3, 1), (6, 4, 1)\} \text{ and}$$

$$C_3 = \{(6, 6, 6), (6, 6, 7), (7, 7, 7)\}$$

– Inter-Cluster and Intra-Cluster Similarity

Here, each cluster is compact in the sense that the distance be-tween any two points in a cluster is smaller than that between

points in different clusters. So, intra-cluster distances are smaller and inter-cluster distances are larger. For example, using the city-block (Manhattan) distance, between $X = (x_1, x_2, \cdots, x_d)$ *and*

- $= (y_1, y_2, \cdots, y_d)$, *which is defined by*

$$d_1(X, Y) = \sum_{i=1}^{i=d} | x_i - y_i |$$

the maximum intra-cluster distance in each cluster and the min-imum inter-cluster distance between a pair of clusters is shown in Table 1. Note that inter-cluster distance between C_i *and* C_j *is the same as that between* C_j *and* C_i. *Also, the minimum of the minimum inter-cluster distances between all possible clusters (which is 6 between* C_1 *and* C_2*) is greater than the maximum of the maximum intra-cluster distances (which is 3 in* C_3*) in this example. In this example, the clusters are* well separated.

– Clustering Based on Similarity

Such an ideal partition may not be available in practice; in such a case, we need to characterize the intra-cluster and inter-cluster

distances appropriately. There could be a variety of ways to char-acterize it; one other characterization, for example, is that a pat-tern is in the same cluster as its nearest neighbor. Such a charac-terization may help in partitioning patterns into clusters that are elongated. In general, we partition such that intra-cluster similar-ity is high and inter-cluster similarity is low.

Cluster	Maximum Intra Cluster Distance	Minimum Inter Cluster Distance
C_1	2	with C_2 is 6
C_2	1	with C_3 is 7
C_3	3	with C_1 is 14

Table 1: Intra-cluster and Inter-cluster Distances

– K-Partition of a Dataset

Here, we have obtained a hard partition of the data set X . If there are K clusters in the resulting partition, then we call it a K-partition. We define a K-partition of X as follows.

$$Partition(X, K) = \pi_X^K = \{C_1, C_2, \cdots, C_K\},$$

where

$$\bigcup_{i=1}^{K} C_i = X \; C_i \cap C_j = \phi, i \neq j \text{ and } C_i \neq \phi \text{ for any } i.$$

So, the collection of K clusters is exhaustive which means every element of X is assigned to exactly one of the clusters and no two clusters have a common element. It is possible to show that the number of K-partitions of an n element set is

$$\frac{1}{2o.!} \sum_{i=1}^{K} K\,(i)n.$$

– Exhaustive Enumeration

There are approximately 11,259,666,000 4-partitions (K=4) of a set having 19 (n=19) elements. So, exhaustive enumeration of all

the partitions to get the best partition is expensive for large n. As a consequence, it is practical to consider only a subset of all possible partitions. This is the motivation behind the design of a variety of clustering algorithms; each of the algorithms considers an appropriate subset of partitions to pick one.

\endash Clustering is Data Compression

– It is possible to view clustering as a data compression tool.

Let a set, S_n, of n patterns be partitioned into K clusters. Let each of the K clusters be represented by a point. Then the collection of representatives is a set, S_K , of size K.

– Data Compression

Typically K < n. So, if we use S_K instead of S_n in decision making, then there is a data compression by a factor of $\frac{n}{K}$. We illustrate this idea using the two-dimensional data set shown in Figure 2. Here, there are patterns corresponding to two classes with labels: X and O. There are 32 patterns labelled X and 24 patterns labelled O. There are three clusters; of these one cluster has only one pattern which is an outlier. We ignore it. The remaining two clusters are represented by their respective centroids, where the centroid of a cluster C is given by

$$\text{C entroid(C)} = \frac{1}{|C|} \sum_{X \in C} X$$

– Representing Clusters

It is also possible to represent a cluster using a medoid where medoid is a member of the cluster which is most centrally located. That is sum of the distances from the medoid to the remaining points in the cluster is minimum. There could be several other ways of representing clusters.

– Classification of the Compressed Data

For example, if we use S_K , instead of S_n, as a training set in NNC, then the number of distances to be computed, for each test pattern, will reduce from n to K. This can lead to a significant

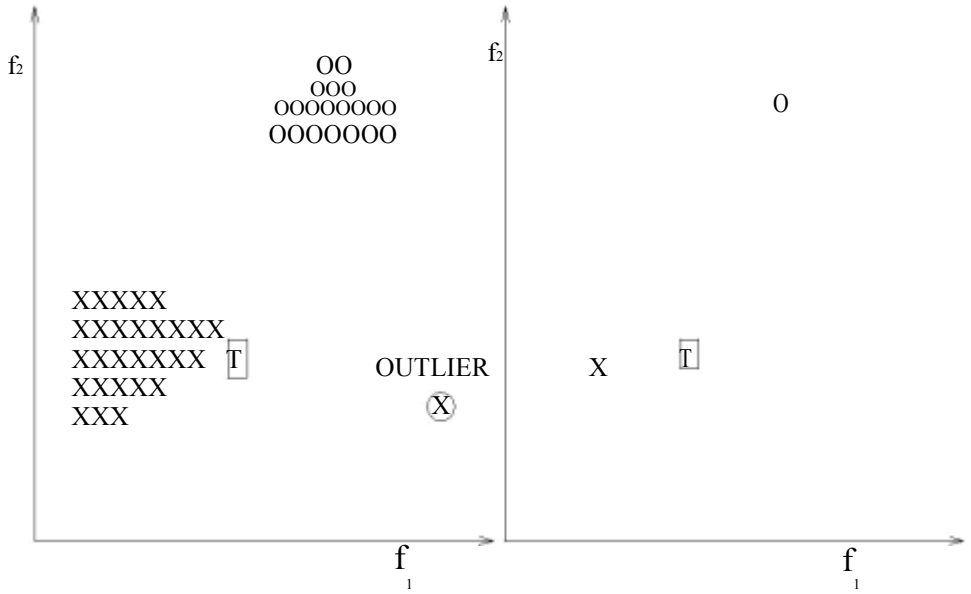

Figure 2: Representing Clusters by their Representatives

reduction in the classification time. For the data shown in Fig-ure 2, using the NNC the test pattern marked T is assigned to class X whether we use the entire data set shown on the left or the compressed data shown on the right. In fact, not only in the neighborhood classifiers, clustering has been successfully used in a variety of classifiers including those based on Support Vector Ma-chines, Artificial Neural Networks, and Hidden Markov Models. In all these cases, some kind of data compression is achieved by using the cluster representatives instead of the entire data consid-ered originally.

\endash Applications of Clustering

There are several important application areas where clustering plays an effective role. These include:

Document Analysis and Recognition: Here, we are interested in analysis and recognition of printed and handwritten text. Specific examples include recognition of zip code/pin code, analysis and recognition of signatures of customers on bank transaction slips, and classification of multi-lingual text documents.

Data Mining: Clustering generates abstraction of the data set in the form of clusters and their representatives. In data mining, clustering is popularly used to group large data sets and the clustering output is used in further decision making activities like classification and regression.

Information Retrieval from WWW: Text mining is at the heart of information retrieval (IR). In IR both the content of each web page and linkage (hypertext) information between pages is used. Typically, content is a collection of multimedia documents, mostly text and also images. Clustering is used in grouping both the content and linkage information.

Biometrics: Face images, iris of the eye, speech/speaker data, and fingerprint images are some of the typically encountered data here. Clustering is popularly used in grouping data in all these cases.

Bioinformatics: Clustering is used in both DNA and protein sequence analysis in bioinformatics.

Clustering a set of Physical Objects

In all the above cases, data is typically available in a form amenable for direct processing by the computer. For example, text may be viewed as a sequence of ASCII characters and image and speech data is available in digital format. Similarly, sequence data encountered in bioinformat-ics and transaction data collected in data mining are naturally stored on a digital computer. However, there are applications in pattern clus-tering where the data is obtained from physical objects. For example, in order to discriminate humans from the chairs in a lecture hall, we need to represent humans and chairs on a computer. Note that chairs and humans are physical objects.

Representation of Patterns and Clusters

Clustering Process

- The Process of Clustering

The process of clustering consists of the following steps.

1. Representation of patterns:

Representation is the most important and difficult part in pattern clustering in particular and pattern recognition in general. We consider various issues associated with representation.

– Features may not be Relevant:

We illustrate it using the following example.

Example 1

For example, consider the simple two-class data set shown in Table 1.

Number of Terms	Class Label
100	Politics
110	Politics
120	Politics
130	Politics
105	Sports
115	Sports
135	Sports

Table 1: Classification into Politics and Sports Classes

– Representation of documents

Here, 7 news documents are represented using their size in terms of number of terms. Also, the class label in the form whether the news item is on Politics or Sports is also pro-vided in each case. If a test pattern (news item) has 132 terms, then its class label is Politics based on NNC and and another news item with 133 terms is classified as a Sports item. Even though we can make such decisions using the NNC, these decisions are not appealing intuitively because

document size and class labels politics or sports are not necessarily correlated. Suppose we cluster these points into two groups:{100, 105, 110, 115, 120}, {130, 135}. By considering the centroids of the two classes Politics(P) and Sports(S) in each cluster, we get 110 - S, 110 - P, 130 -P, 135 - S as representatives. Note that we get the same decisions using NNC and these representatives as training patterns on the two doc-uments considered earlier with sizes 132 and 133.

– Inappropriate representation

In this example, there is a problem in the representation which is easier to perceive. The feature Size used to repre-sent news items is not correlated with the class labels Politics and Sports. In general, in an application area, it may not be easy to obtain features that are correlated with the class labels; further, the complexity of the problem increases with the dimensionality of data set.

– Representation of Physical Objects:

There are several applications where the patterns to be clus-

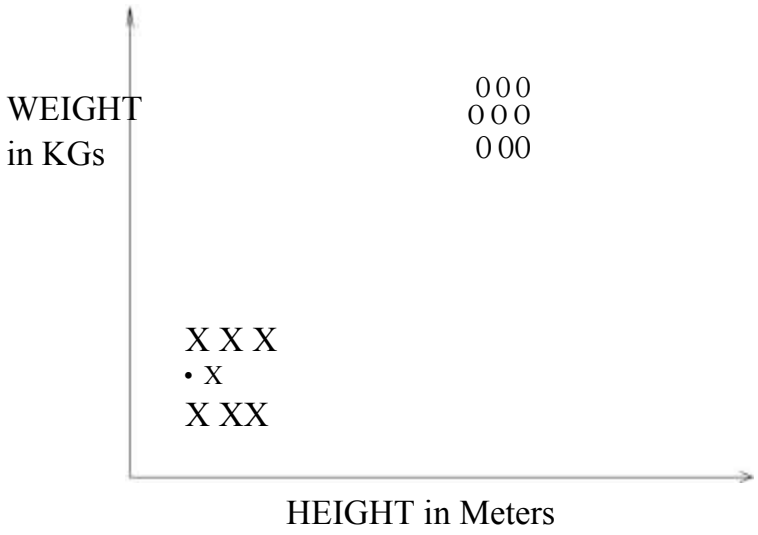

Figure 1: Chairs and Humans: Chair by X and Human by O

tered are physical objects. For example, discrimination be-

tween a cricket ball and a soccer ball in sports; between a laptop and a desktop on an executive's desk; between a two-wheeler and a four-wheeler on roads.

– Chairs and humans

Consider the problem of clustering a collection of humans and chairs in a lecture hall. We can represent the collection using height and weight of the objects. Consider the representations in the two-dimensional space shown in Figure 1. Note that the two classes are well-separated. In general, it is possible to use many more features including cost, academic qualification, and material used to make. Some of these features may not have meaningful values for either of two classes of objects. We may not know which subset of features are ideal to be used.

– Feature Selection/Extraction:

* Feature selection is the process of selecting a sub-set of the given collection of features.

  We illustrate this idea using a simple example data shown in Table 2.

Transaction No	a	b	c	d	e
t_1	1	0	1	0	1
t_2	0	1	0	1	1
t_3	1	0	1	0	1
t_4	0	1	0	1	1
t_5	0	1	0	1	0
t_6	0	1	0	1	0
t_7	1	0	1	1	1

Table 2: Feature Selection

* Clustering transactions

  There are seven transactions and each transaction is based on buying one or more of the five items, a, b, c, d, and e. A '1' against an item in the transaction indicates that the item is bought and a '0' indicates that the item is not

bought. For example, in transaction t_1, items a, c, e are bought. Consider a partitioning of the seven transactions into two clusters C_1 and C_2 where

$C_1 = \{t_1, t_3, t_7\}$
$C_2 = \{t_2, t_4, t_5, t_6\}$

* Hamming distance based clustering

Note that the maximum Hamming distance between any two 5-bit binary patterns in either C_1 or C_2 is 1. By ig-noring item 'e', we get the following clusters:

$C_1 = \{1010, 1010, 1011\}$
$C_2 = \{0101, 0101, 0101, 0101\}$

* Distance between clusters

Now the maximum distance between any two patterns in C_1 is still 1, whereas the maximum distance between any pair of patterns in C_2 is 0. Observe that by ignoring item 'd' also, the items selected in representing the seven

transactions are 'a', 'b', and 'c'. If only these three items are used, then all the three patterns in C_1 are identical. Similarly, all the four patterns in C_2 are also identical.

* Feature selection

This amounts to reducing the number of features from 5 to 3, such that patterns in each cluster are identical. Of course, ideally it is adequate to select feature b from the five features as patterns in a cluster have the same value of b and in different clusters the values are different.

* Feature extraction

Feature extraction is a process of selecting a smaller set of features that are linear or non-linear combinations of the original features. Stated another way, feature extraction is selection of a subset of features from a transformed set. One of the most popular schemes for feature extraction is based on Principal Components.

* Principal components (PCs) are linear combina-

tions of the input features.

– Representation of Clusters:

* Cluster descriptions

  In addition to representing patterns, it is important to generate representations or descriptions of the clusters. These are used in decision making; classification is one of the popular decision making paradigms used here. Clus-ters and their descriptions are also used in other prediction tasks like regression.

* Inductive learning

  Inductive Learning is typically used to generate such gen-eralized descriptions of clusters. One can use either a formal language like mathematical logic or descriptions based on statistics to represent a partition of a data set. We explain a scheme for describing clusters using the fol-lowing example.

  Example 2

  We illustrate the notion of cluster description using the data in Figure 2.

* Logical description

  There are four points in a cluster and they are described by

$$(f_1 = a) \wedge (f_2 = d)$$
$$(f_1 = c) \wedge (f_2 = f)$$
$$(f_1 = a) \wedge (f_2 = f)$$
$$(f_1 = c) \wedge (f_2 = d)$$

The cluster as a whole is described by

$$(f_1 = a \cdot\cdot c) \wedge (f_2 = d \cdot\cdot f)$$

Note that the description of the cluster is generated such that some of the unseen patterns are captured by it. For

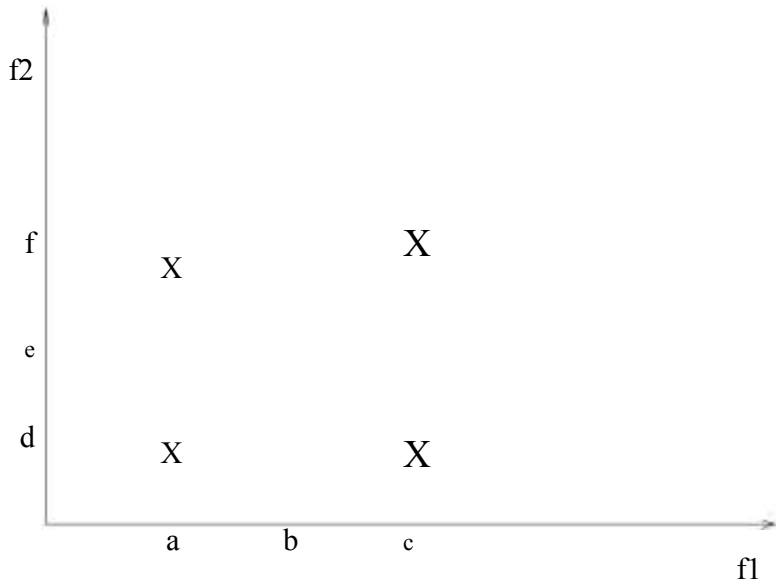

Figure 2: Description of Clusters

example, a pattern of the form (f_1 = b) ∧ (f_2 = e) is also captured by the cluster description even though the pattern is not explicitly present in the collection.

Example 3

Let us consider again the data given in Table 2. We can represent it using a tree structure as follows. Initially the root node is added. Each of the transactions is added to the tree incrementally.

* Adding the first transaction

  First t_1 is added; here, items a, c, and e are present. So, a child node is added to the root and item 'a' is stored in it with a count of 1; similarly a path is set up by adding nodes appropriately to add items c and e in that order with counts of 1 each. The corresponding tree is shown in Figure 3.

* Tree corresponding to the first two transactions Now transaction t_2 is considered; the items present in the transaction are b, d, and e. Because item 'a' is absent here, we start a new branch at the root and include b

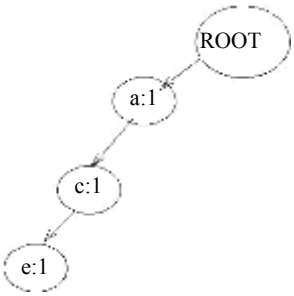

Figure 3: Representation of one Transaction

with a count 1 and add the remaining part to set a path b, d, e with counts of 1 each. This is depicted in Figure 4.

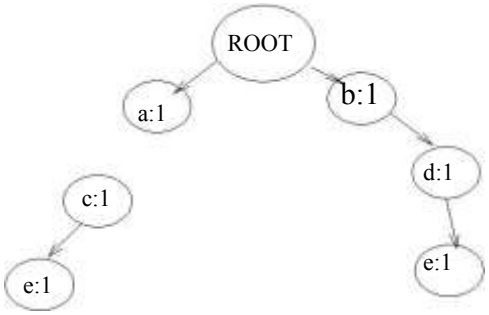

Figure 4: Representation of Two Transactions

* Incremental updation

  Now addition of t_3 requires no additional nodes, but the counts of nodes a, c, and e are increased by 1 each in the existing path. Thus we increment the counts of items if a new transaction has its corresponding path already present from the root node. The intermediate tree, so obtained, is shown in Figure 5.

* The final tree structure

  However, when the path corresponding to the transaction is not present in the tree, then we need to insert additional nodes appropriately. For example, when t_7 is added, there is a path from the root for a, c; but d is not present in the

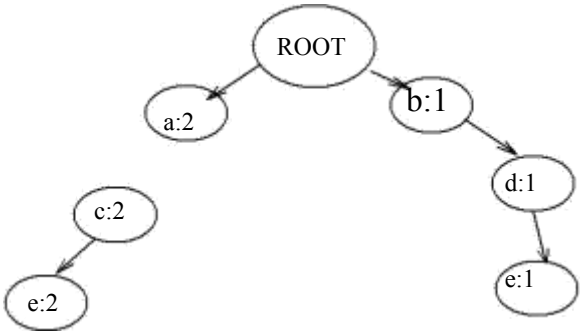

Figure 5: Representation of Three Transactions

path next of the current tree. So, we add a new node to accommodate d with a count 1 and add one more node to store item e with count 1. The final tree after inserting all the transactions is shown in Figure 6.

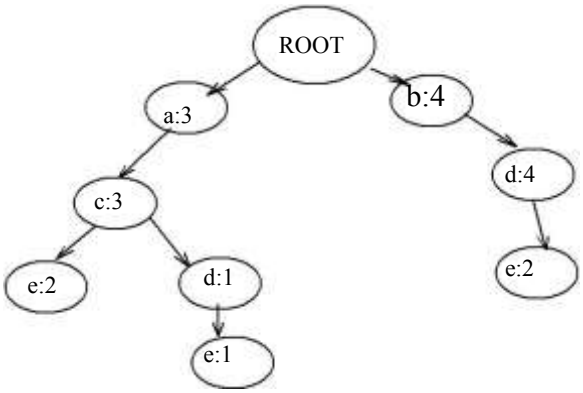

Figure 6: Representation of Transactions

– Statistical descriptions

We illustrate the notion of inductive learning further using another example.

Example 4

Consider the three-cluster data described in Example 3 of lesson 30. We can represent each cluster by one or more representative patterns. One popular cluster representative is its

centroid. The centroids of the three clusters C_1, C_2, and C_3 respectively are

$$\mu_1 = (1.25, 1.25, 1.25)$$
$$\mu_2 = (6.00, 3.50, 1.00)$$
$$\mu_3 = (6.33, 6.33, 6.66)$$

Note that even these representatives can be used to describe unseen patterns. For example, pattern (2, 2, 2) is not a member of C_1 but μ_1 is closer to the pattern. Similarly, the pattern (6, 4, 2) which is not in C_2 is better explained by μ_2 than by the other two centroids. So, even this representation can be viewed as an inductive description of the clusters.

X Computation of Similarity between Patterns:

There are a variety of similarity and distance (dissimilarity) func-tions that are used to characterize the proximity between a pair of patterns. We describe it further in the next lesson.

X Algorithms for grouping or Clustering:

Popularly clustering algorithms are categorized as either parti-tional or hierarchical. In the case of partitional algorithms, a single partition of the data set is generated and a hierarchy of partitions is generated using hierarchical algorithms.

Clustering Process

Clustering Process

- Representation:

– Representation of patterns and clusters is an important part of clustering.

Specifically, feature selection and extraction are useful in repre-senting patterns. Feature selection was briefly discussed in the previous lesson. Here, we consider feature extraction. In feature extraction, we extract a set of features from a given collection of input features.

– Typically, the number of extracted features is less than the number of input features.

It is possible that each extracted feature is either a linear or a non-linear combination of the input features. A popular scheme for linear feature extraction is based on principal components.

- Principal Components
  Let
  $$X_1, X_2, \cdots, X_n$$
  be a collection of patterns in a D-dimensional space of reals. So,
  $$X_i = \{X_{i1}, X_{i2}, \cdots, X_{iD}\} \text{ for } 1 \leq i \leq n$$

- Eigenvectors corresponding to the largest eigenvalues
  The eigenvectors $P_1, \cdots, P_d$ corresponding to the d largest eigenvalues of the covariance matrix Ω are the d principal components. This means that the eigenvalues of Ω are such that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \geq \lambda_{d+1} \geq \cdots, \geq \lambda_D$.

Example 1

We illustrate this with the help of a two-dimensional example. Consider the set of 6 two-dimensional patterns shown in Figure 1. They are represented by the following vectors in the two-dimensional space.
$$(1, 2)^t, (2, 1)^t, (2, 2)^t, (6, 7)^t, (7, 6)^t, (6, 6)^t.$$

- Computation of principal components
  In order to compute the principal components, we need to compute the covariance matrix, Ω, which is given by
  $$= E[(X - \mu)(X - \mu)^t]$$

- Sample mean
  In this example, μ is computed by finding the sample mean (centroid) of the data which is given by
  $$\mu = \frac{1}{6} \sum_{i=1}^{6} X_i = \frac{1}{6}(24, 24)^t = (4, 4)^t \quad (1)$$

- Sample covariance matrix
  Using the value of μ from (1), we compute Ω based on the sample covariance matrix given by
  $$\Omega = \frac{1}{6} \sum_{i=1}^{6} (X_i - \mu)(X_i - \mu)^t \quad (2)$$
  Now computing the sample covariance matrix for the data, we get
  $$\begin{matrix} \frac{1}{7} & \frac{1}{6} \end{matrix}$$

$$\Omega = \begin{matrix} 3 & 3 \\ \frac{16}{3} & \frac{17}{3} \end{matrix} \qquad (3)$$

- Characteristic equation

  The characteristic equation is given by $\mathrm{Det}(\Omega - \lambda I) = 0$, where I is the identity matrix; so, the characteristic equation is given by

$$\begin{vmatrix} 3 - \lambda & 3 \\ \frac{16}{3} & \frac{17}{3} - \lambda \end{vmatrix} = 0$$

So, we get the characteristic equation $(11 - \lambda)(\frac{1}{3} - \lambda) = 0$. As a consequence, the eigenvalues, in decreasing order of magnitude, are

$\lambda_1 = 11$; $\lambda_2 =$ [illegible]

- Principal components

  The corresponding eigenvectors are

  $P_1 = (1, 1)^t$; $P_2 = (1, -1)^t$,

  where P_1 and P_2 are the principal components which are depicted as PC1 and PC2 respectively in Figure 1. Note that the two eigenvectors are orthogonal to each other and PC1 is in the direction of maximum variance of the data.

- Popular linear feature extraction scheme

  Principal component analysis is the most popular linear feature extrac-tion scheme. There are a variety of other linear schemes and non-linear schemes for feature extraction. We do not discuss them here. Inter-ested readers may refer to the important references provided at the end of the module.

- Similarity Computation

  Computation of similarity between patterns is an important part of any clustering algorithm. Dissimilarity between patterns is captured by a distance function; larger the similarity between a pair of patterns, the distance between them is smaller or vice versa.

  – Distance functions

    Metrics are popularly used to characterize the distance; a metric, d, satisfies the following properties for all patterns X, Y, and Z:

    * Reflexivity: $d(X, X) = 0$; $d(X, Y) \geq 0$
    * Symmetry: $d(X, Y) = d(Y, X)$
    * Triangular Inequality: $d(X, Y) + d(Y, Z) \geq d(X, Z)$ which is equivalent to stating that the sum of the lengths of any two sides of a triangle is greater than the length of the third side.

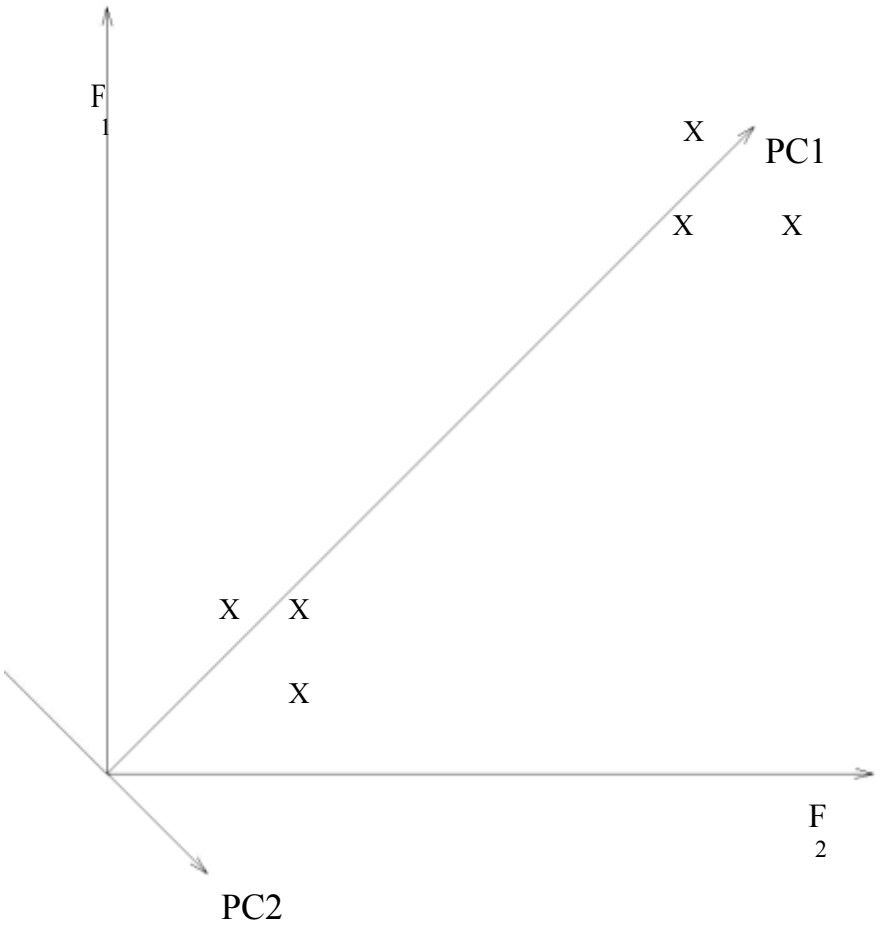

Figure 1: Principal Components for the two-dimensional data

We use distance functions to characterize clusters. Distance be-tween patterns in the same cluster is less than that between pat-terns in different clusters.

– Euclidean distance

Euclidean distance, also called the L_2 norm, is the most popular distance measure in clustering. It is given, for two d-dimensional vectors X and Y by

$$L_2(X, Y) = d_2(X, Y) = \left[\sum_{i=1}^{d} (X_i - Y_i)^2\right]^{\frac{1}{2}}$$

– Translation and rotation invariance

The advantage with Euclidean distance is that it is invariant to translations (shifting all the patterns by the same amount) or rotations (points are rotated in the plane around a point). Further, it is easy for human perception as it corresponds to the crow-flying distance. However, it varies with scaling; that is when the patterns scaled differently in different feature directions.

Theorem of the Ugly Duckling:

This theorem states that the number of predicates shared by any two patterns is constant when all logically possible predicates are used to represent patterns.

There is no unsupervised classification

Based on the above discussion we can conclude that there is no unsuper-vised classification; some kind of knowledge is essential for clustering. Such knowledge is used in one or more of the following:

Representation: Domain knowledge may be used to represent patterns appropriately.

Representation of documents

For example, consider the data given in Example 4 in the previous lesson. Instead of characterizing each document by the number of terms, if we use the importance or weight of each term in the documents, then we get a better representation.

TF-IDF

For example, in text classification, the weight of each term is char-acterized by the term frequency - inverse document frequency. In this case, domain knowledge is used to select the term frequency and inverse document frequency. If a term does not occur in a document, then the corresponding weight should be zero; simi-larly, if a term occurs more frequently in a document, then the weight should be large. Term frequency captures the importance of a term based on its frequency of occurrence in a document. Fur-ther, if a term, like is, occurs frequently in all documents, then its weight is low; note that frequent terms like is, the, of cannot play a discriminative role.

If there are N documents in a collection, then the weight, popu-larly called as tf-idf of a term i in a document j is

$$weight_{ij} = \log(tf_{ij}) * \log\left(\frac{N}{N_i}\right)$$

where,
tf_{ij} = Number of times term i occurs in document j, and
N_i = Number of documents, out of N, in which term i occurs

\endash Similarity Computation: Knowledge may be used to characterize the distance/dissimilarity measure.

Different distance measures

For example, using Euclidean distance instead of the city-block distance may lead to a different clustering. We illustrate it with the following example.

Example 2

Consider three two-dimensional points A, B, and C given by

$= (0, 0)^t$ B $= (2, 2)^t$ C $= (5, 2)^t$. N ow,

$L_1(A, B) =$ $| - ^2 | ^+ | - ^2 | = 4; L_1(B, C) = 3$

$L_2(A, B) = (4 + 4)$ $_2 = \sqrt{}$ $8; L_2(B, C) = 3$

Resulting partitions could be different

Note that, based on L_1 distance, B and C are clustered in prefer-ence to A and B. However, in terms of L_2 distance, A and B are closer than B and C; so A and B clustered together before B and C.

Clustering Algorithm: Knowledge may be used in selecting the kind of clustering algorithm or in fixing the values of the parameters used by the algorithm. We discuss some of the relevant issues in the next lesson.

Clustering labelled data

Patterns to be clustered are either labelled or unlabelled. Clustering algorithms are typically used to group sets of unlabelled patterns. This paradigm is popular and so clustering is viewed as grouping of unla-belled patterns.

– Classification based on clustering

  However, it is possible that clustering of labelled patterns can also be useful in several engineering applications. For example, consider classification of handwritten digits 0 to 9. There are 10 classes; further it is possible that each class has one or more subclasses. For example, in a collection of handwritten 1s, there could be several subclasses because of the way different people write the character. Clustering a class of 1s might reveal the subclasses existing in the class; the same idea might be applicable to obtain subclasses corresponding to other digits as well.

– Labelled and unlabelled clustering

  It is important to note that clustering the labelled patterns might give different set of representatives compared to clustering the data without taking the labels into account. We illustrate this with the help of Figure 2. We obtained the data here by adding some more chairs (heavier chairs) and humans (kids) to the data shown in Figure 3 shown in the previous lesson.

– Difference between labelled and unlabelled clustering

  If we generate four clusters by grouping patterns in each class into two clusters, we get the clusters shown by rectangular boundaries. On the contrary, by grouping the entire collection of patterns into 4 clusters, we end up with the clusters depicted by using rectangles with rounded corners; these clusters are not convincing. A cluster is formed using subsets of chairs and humans. Further, another subset of chairs is partitioned into two clusters. So, in this case, it is good to use the class labels in clustering.

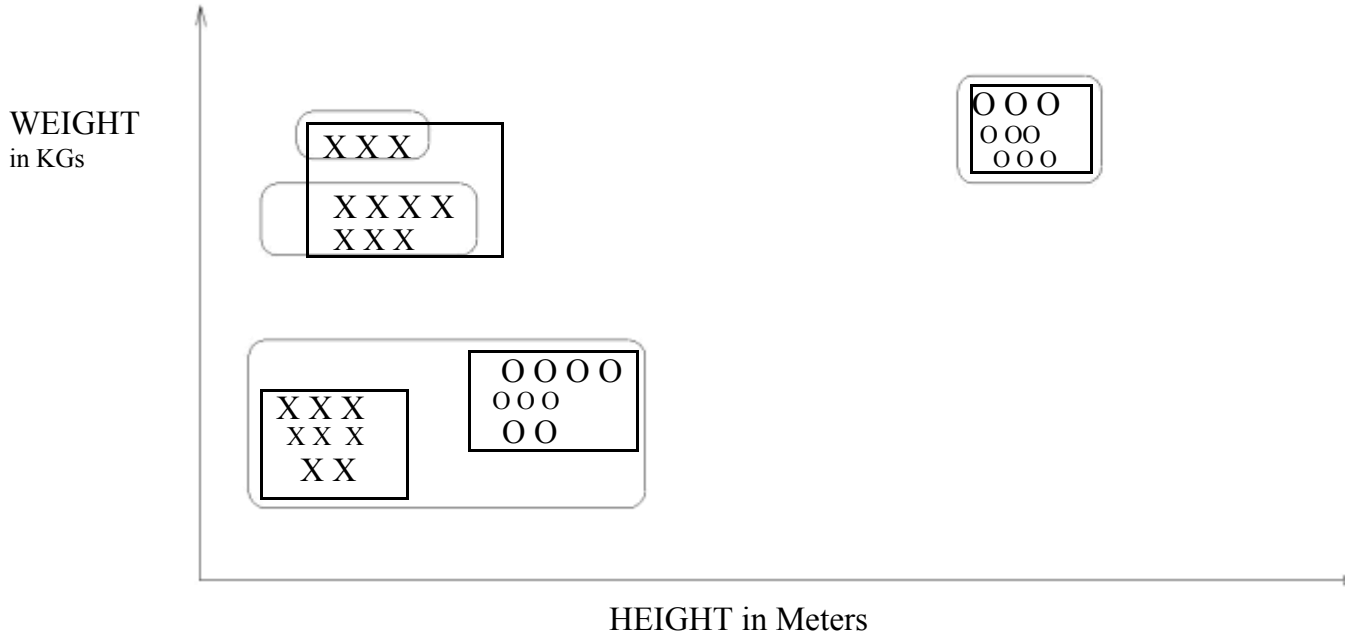

Figure 2: Labelled Clustering

Clustering Algorithms

Algorithms for Clustering

- In this lesson, we describe several clustering algorithms with an em-phasis on the kind of knowledge used.
- There is a wide variety of clustering algorithms.
- Different approaches can be described with the help of the hierarchy shown in Figure 1.

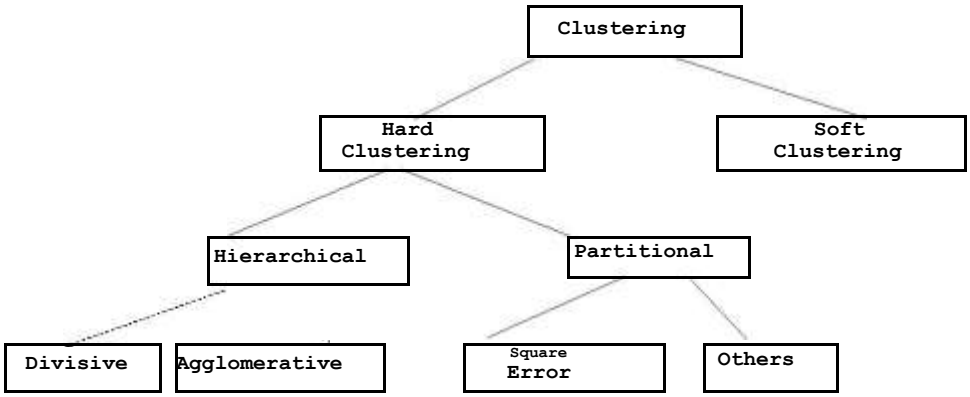

Figure 1: Taxonomy of Clustering Algorithms

- At the top level, there is a distinction between the hierarchical, par-titional (hard) and soft clustering paradigms based on whether the clusterings generated are partitions or overlapping.
- The clustering algorithms are either hierarchical, where a nested se-quence of partitions is generated, or partitional where a partition of the given data set is generated.
- The soft clustering algorithms are based on fuzzy sets, rough sets, artifi-cial neural nets (ANNs), or evolutionary algorithms, specifically genetic algorithms (GAs).

Hierarchical Algorithms

- Hierarchical algorithms produce a nested sequence of partitions of the data which can be depicted by using a tree structure that is popularly called as dendrogram.
- Hierarchical algorithms are either divisive or agglomerative.

- In the former, starting with a single cluster having all the patterns, at each successive step a cluster is split; this process continues till we end up with each pattern in a cluster or a collection of singleton clusters.

- Divisive algorithms use a top-down strategy for generating partitions of the data. The effort involved here is in identifying which cluster to split and how to split. Typically, a cluster with a large variance (or average squared deviation from the centroid) is selected to split.

  Further, exhaustive enumeration of all possible splits (2-partitions) of a set of size m is (2^m).

- Agglomerative algorithms, on the other hand, use a bottom-up strategy. They start with n singleton clusters when the input data set is of size n, where each input pattern is in a different cluster. At successive levels, the most similar pair of clusters is merged to reduce the size of the partition by 1.

- An important property of the agglomerative algorithms is that once two patterns are placed in the same cluster at a level, then they remain in the same cluster at all subsequent levels.

- Similarly, in the divisive algorithms, once two patterns are placed in two different clusters at a level, then they remain in different clusters at all subsequent levels.

Agglomerative Clustering

Typically, an agglomerative clustering algorithm goes through the follow-ing steps:

- Initialize each cluster with a distinct pattern. Compute the proximity (similarity/dissimilarity matrix) between all pairs of patterns.
- Find closest pair of clusters and merge them. Update the proximity matrix to reflect the merge.
- If all the patterns are in one cluster, stop. Else, go to step 2.
- Note that step 1 in the above algorithm requires $O(n^2)$ time to compute pairwise similarities and $O(n^2)$ space to store the values, when there are n patterns in the given collection.
- In the single-link algorithm, the distance between two clusters C_1 and C_2 is the minimum of the distances d(X, Y), where $X \in C_1$ and $Y \in C_2$.
- In the complete-link algorithm, the distance between two clusters C_1 and C_2 is the maximum of the distances d(X, Y),where $X \in C_1$ and $Y \in C_2$.
- Note that the single-link algorithm constructs a minimal spanning tree with nodes located at the data points. On the other hand, the complete-link algorithm characterizes strongly connected components as clusters.
- Hierarchical clustering algorithms are computationally expensive.
- The agglomerative algorithms require computation and storage of a similarity or dissimilarity matrix of values that has $O(n^2)$ time and space requirement.
- Initially, they were used in applications where hundreds of patterns were to be clustered.
- However, when the data sets are larger in size, these algorithms are not feasible because of the non-linear time and space demands.
- It may not be easy to visualize a dendrogram corresponding to 1000 patterns.
- Similarly, divisive algorithms require exponential time in the number of patterns or the number of features. So, they too do not scale up well in the context of large-scale problems involving millions of patterns.

Partitional Clustering

- Partitional clustering algorithms generate a hard or a soft partition of the data.
- The most popular of this category of algorithms is the K-Means algo-rithm.

K-Means Algorithm

A simple description of the K-Means algorithm is given below:

Select K initial cluster centers. Assign each of the n patterns to one of the K clusters; a pattern is assigned to its closest center/cluster.

Compute the cluster centers based on the current assignment of pat-terns.

Assign each of the n patterns to its closest center/cluster.

If there is no change in the assignment of patterns to clusters during two successive iterations, then stop; else, go to step 2.

The above algorithm is called convergent K-Means algorithm.

Selecting the initial cluster centers is a very important issue. There are a variety of schemes for selecting the initial cluster centers; these include selecting the first K of the given n patterns, selection K random patterns out of the given n patterns, and viewing the initial cluster seed selection as an optimization problem and using a robust tool to search for the globally optimal initial seed selection.

Example 1

K-means algorithm may be illustrated with the help of the three-dimensional data set of 10 points given below.

$$(1, 1, 1)^t \ (1, 1, 2)^t \ (1, 3, 2)^t \ (2, 1, 1)^t \ (6, 3, 1)^t$$
$$(6, 4, 1)^t \ (6, 6, 6)^t \ (6, 6, 7)^t \ (6, 7, 6)^t \ (7, 7, 7)^t$$

$Cluster_1$	$Cluster_2$	$Cluster_3$
$(1, 1, 1)^t$	$(1, 1, 2)^t$	$(1, 3, 2)^t$
$(2, 1, 1)^t$		$(6, 3, 1)^t$
		$(6, 4, 1)^t$
		$(6, 6, 6)^t$
		$(6, 6, 7)^t$
		$(6, 7, 6)^t$
		$(7, 7, 7)^t$

Table 1: The first iteration of the K-Means algorithm

Considering the first three points as the initial seed points of a 3-partition, The three cluster representatives and assignment of the remaining 7 points to the nearest centroids is shown in Table 1 using L_1 metric as the distance measure

The centroids of the clusters are

$$(1.5, 1, 1)^t, (1, 1, 2)^t, (5.4, 5.1, 4.3)^t$$

Assignment of the 10 patterns to the nearest updated centroids is shown in Table 2. The updated cluster centroids are

$Cluster_1$	$Cluster_2$	$Cluster_3$
$(1.5, 1, 1)^t$	$(1, 1, 2)^t$	$(5.4, 5.1, 4.3)^t$
$(1, 1, 1)^t$	$(1, 1, 2)^t$	$(6, 3, 1)^t$
$(2, 1, 1)^t$	$(1, 3, 2)^t$	$(6, 4, 1)^t$
		$(6, 6, 6)^t$
		$(6, 6, 7)^t$
		$(6, 7, 6)^t$
		$(7, 7, 7)^t$

Table 2: The second iteration of the K-Means algorithm

$$(1.5, 1, 1)^t, (1, 2, 2)^t \ (6.1, 5.5, 4.66)^t$$

Subsequent iterations do not lead to any changes in the assignment of points or in the centroids obtained. K-Means algorithm minimizes the sum of the squared deviations of patterns in a cluster from the center. If m_i is the centroid of the i^{th} cluster, then the function minimized by the K-Means algorithm is

$$\endash \quad x\ i\text{th cluster}\ (x - m_i)^t(x - m_i).\ i=1 \in$$

Note that the value of the function is 55.5 for the 3-partition. However, if we consider the initial seeds to be

$$(1, 1, 1)^t, (6, 3, 1)^t, (6, 6, 6)^t,$$

then we have the assignment shown in Table 3. Further iterations do not

$Cluster_1$	$Cluster_2$	$Cluster_3$
$(1, 1, 1)^t$	$(6, 3, 1)^t$	$(6, 6, 6)^t$
$(2, 1, 1)^t$	$(6, 4, 1)^t$	$(6, 6, 7)^t$
$(1, 1, 2)^t$	$(1, 3, 2)^t$	$(6, 7, 6)^t$
$(1, 3, 2)^t$		$(7, 7, 7)^t$

Table 3: The optimal 3-partition of the K-Means algorithm

affect any changes in the assignment of patterns. The value of squared error criterion for this partition is 8. This shows that the K-Means algorithm is not guaranteed to find the globally optimal partition.

- The time complexity of the algorithm is O(nK dl), where l is the number of iterations and d is the dimensionality.
- The space requirement is O(K d).
- These features make the algorithm very attractive.
- It is one of the most frequently used algorithms in a variety of applications; some of these applications involve large volumes of data, for example, the satellite image data.

- It is best to use the K-Means algorithm when the clusters are hyper-spherical.
- It does not generate the intended partition, if the partition has non-spherical clusters; in such a case, a pattern may not belong to the same cluster as its nearest centroid.
- Even when the clusters are spherical, it may not find the globally op-timal partitions as discussed in Example 1.
- Several schemes have been used to find the globally optimal partition corresponding to the minimum value of the squared-error criterion.

References

1. **Devi, V.S; Murty M. N.** (2011) Pattern Recognition: An Introduction, Universities Press, Hyderabad.
2. **Jain, A.K; Murty M. N; Flynn, P. J.** (1999) Data Clustering: A Review, ACM Computing Surveys, Vol. 31, No. 3, pp 264-323.
3. **Xu, R; Wunsch,D. C.** (2009) Clustering, IEEE Press.
4. **Nagy, G.**, (1968) State of the Art in Pattern Recognition, Proceedings of IEEE, Vol. 56, No. 5, pp 836-862.
5. **Jain, A.K.; Dubes, R.C.** (1988) Algorithms for Clustering Data, Prentice-Hall, New Jersey.
6. **Anderberg, M.R.** (1973) Cluster Analysis for Applications, Academic Press, New york.
7. **Duda R.O; Hart, P.E; Stork, D.J.** (2000) Pattern Classification, Wiley-interscience, New York.