MODULE 1

CHAPTER 1

Introduction to Machine Learning

Syllabus

Machine Learning, Types of Machine Learning, Issues in Machine Learning, Application of Machine Learning, Steps in developing a Machine Learning Application.

Training Error, Generalization error, Overfitting, Underfitting, Bias-Variance trade-off.

1.1 Machine Learning 1-2

UQ. What is Machine learning? How it is different from data mining ? **(MU(Comp.) - May 17, 5 Marks, May 19, 5 Marks)** 1-2

UQ. Define Machine learning and explain with example importance of Machine Learning. **(MU(Comp.) – Dec. 19, 5 Marks)** 1-2

1.2 Key Terminology 1-3

UQ. What are the key tasks of Machine Learning? **(MU(Comp.) - May 16, 5 Marks)** 1-3

1.3 Types of Machine Learning 1-5

UQ. Explain how supervised learning is different from unsupervised learning. **(MU(Comp.) - May 17, 5 Marks)** 1-5

1.4 Issues in Machine Learning 1-6

UQ. What are the issues in Machine learning? **(MU(Comp.) - May 15, 5 Marks)** 1-6

1.5 How to choose the right algorithm ? 1-7

UQ. Explain the steps required for selecting the right machine learning algorithm. **(MU(Comp.) - May 16, 8 Marks)** 1-7

1.6 Steps in developing a machine learning application 1-7

UQ. Explain the steps of developing Machine Learning applications. **(MU(Comp.) - May 19, 10 Marks)** 1-7

1.7 Applications of Machine Learning 1-9

UQ. Write short note on: Machine learning applications. **(MU(Comp.) - May 16, May 17, 10 Marks)** 1-9

1.8 Training error and generalization error 1-10

1.9 Underfitting, overfitting, bias and variance trade off 1-11

- **Chapter Ends** 1-13

1.1 MACHINE LEARNING

UQ. What is Machine learning? How it is different from data mining ? **(MU(Comp.) - May 17, 5 Marks, May 19, 5 Marks)**

UQ. Define Machine learning and explain with example importance of Machine Learning. **(MU(Comp.) – Dec. 19, 5 Marks)**

- A machine that is intellectually capable as much as humans, have always attracted writers and early computer scientist who were excited about artificial intelligence and machine learning.
- The first machine learning system was developed in the 1950s. In 1952, Samuel has developed a program to play checkers. The program was able to observe positions at game and learn the model that gives better moves for machine player.
- In 1957, Frank Rosenblatt designed the Perceptron, which is a simple classifier but when it is combined in large numbers, in a network, it became a powerful tool.
- Minsky in 1960, came up with limitation of perceptron. He showed that the X-OR problem could not be represented by perceptron and such inseparable data distribution cannot be handled and following this Minsky's work neural network research went to dormant until 1980s.
- Machine learning became very famous in 1990s,due to the introduction of statistics. Computer science and statistics combination lead to probabilistic approaches in Artificial intelligence. This area is further shifted to data driven techniques. As Huge amount of data is available, scientists started to design intelligent systems that are able to analyze and learn from data.
- Machine learning is a category of Artificial Intelligence. In machine learning computers has the ability to learn themselves, explicit programming is not required.
- Machine focuses on the study and development of algorithms that can learn from data and also make predictions on data.
- Machine learning is defined by Tom Mitchell as "A program learns from experience 'E' with respect to some class of tasks 'T' and performance measure 'P', if its performance on tasks in 'T' as measured by 'P' improves with 'E'." Here 'E' represents the past experienced data and 'T' represents the tasks such as prediction, classification, etc. Example of 'P', we might want to increase accuracy in prediction.
- Machine learning mainly focuses on the design and development of computer programs that can teach themselves to grow and change when exposed to new data.
- Using machine learning we can collect information from a dataset by asking the computer to make some sense from data. Machine learning is turning data into information.

Fig. 1.1.1: Machine Learning

- The Fig. 1.1.1 is the schematic representation of the ML system. ML system takes the training data and background knowledge as the input. Background knowledge and data helps the Learner program to provide a solution for a particular task or problem. Performance corresponding to the solution can be also measured. ML system comprises of mainly two components, Learner and a Reasoner. Learner use the training data and background knowledge to build the model and this can be used by reasoner to provide the solution for a task.
- Machine learning can be applied to many applications such as politics to geosciences. It is a tool that can be applied to many problems. Any application which needs to extract some information from data and also takes some action on data, can benefit from machine learning methods.
- Some of the applications are spam filtering in email, face recognition, product recommendations from Amazon.com and handwriting digit recognition.

- In detecting spam email, if you check for the occurrence of single word it will not be very helpful. But checking the occurrences of certain words used together and combined this with the length of the email and other parameters, you could get a much clearer idea of whether the email is spam or not.
- Machine learning is used by most of the companies to increase productivity, forecast weather, to improve business decisions, detect disease and do many more things.
- Machine learning uses statistics. There are many problems where the solution is not deterministic. There are certain problems for which we don't have that much information and also don't have that much computing power to properly model the problem. For these problems we need statistics, example of such type of problem is prediction of motivation and behavior of humans. The behavior and motivation of humans is a problem that is currently very difficult to model.

Fig. 1.1.2 : Schematic diagram of Machine Learning

Machine learning = Take data + understand it + process it + extract value from it + visualize it + communicate it

1.2 KEY TERMINOLOGY

UQ. What are the key tasks of Machine Learning? **(MU(Comp.) - May 16, 5 Marks)**

Expert System

- Expert system is a system which is developed using some training set, testing set, and knowledge representation, features, algorithm and classification terminology.
  - *(i)* ***Training Set :*** A training set comprises of training examples which will be used to train machine learning algorithms.
  - *(ii)* ***Testing Set :*** To test machine learning algorithms what's usually done is to have a training set of data and a separate dataset, called a test set.
  - *(iii)* ***Knowledge Representation :*** Knowledge representation may be stored in the form of a set of rules. It may be an example from the training set or a probability distribution.
  - *(iv)* ***Features :*** Important properties or attributes.
  - *(v)* ***Classification :*** We classify the data based on features.
- **Process:** Suppose we want to use a machine learning algorithm for classification. The next step is to train the algorithm, or allows it to learn. To train the algorithm we give as a input a quality data called as training set.
- Each training example has some features and one target variable. The target variable is what we will be trying to predict with our machine learning algorithms. In a training dataset the target variable is known. The machine learns by finding some relationship between the target variable and the features. In the classification tasks the target variables are known as classes. It is assumed that there will be a limited number of classes.
- The class or target variable that the training example belongs to is then compared to the predicted value, and we can get a idea about the accuracy of the algorithm.
- **Example:** First we will see some terminologies that are frequently used in machine learning methods. Let's take an example that we want to design a classification system that will classify the instances in to either Acceptable or Unacceptable. This kind of system is a fascinating topic often related with machine learning called *expert systems*.

- Four features of the various cars are stored in Table 1.2.1. The features or the attributes selected are Buying_Price, Maintenance_Price, Lug_Boot and Safety. Examples belong to Table 1.2.1 represents a record comprises of features.
- In Table 1.2.1 all the features are categorical in nature and takes limited disjoint values. The first two features represent the buying price and maintenance price of a car such as high, medium and low. Third feature shows the luggage capacity of a car as small, medium or big. Fourth feature represents whether the car has safety measures or not, which takes the value as low, medium or high.
- Classification is one of the important task in machine learning. In this application we want to evaluate the car out of a group of other cars. Suppose we have all information about car's Buying_Price, Maintenance_Price, Lug_Boot and Safety. Classification method is used to evaluate a given car as Acceptable or Unacceptable. Many machine learning algorithms are there that can be used for classification. The target or the response variable in this example is the evaluation of a car.
- Suppose we have selected a machine learning algorithm to use for classification. The main task in the classification is to train the algorithm, or allow it to learn. We give the experienced data as the input to train the algorithm which is called as training data.
- Let's assume training dataset contains 14 training records in Table 1.2.1. Suppose each training record has four features and one target or the response variable as shown in Fig. 1.2.1.The machine learning algorithm is used to predict the target variable.
- In classification task the target variable takes a discrete value, and in the task of regression its value could be continuous.
- In a training dataset we have the value of target variable. The relationship that exists between the features and the target variable is used by machine for learning. The target variable is the evaluation of the car. Classes are the target variables in the classification task. In classification systems it is assumed that classes are to be of limited number.
- Attributes or features are the individual values that, when combined with other features, make up a training example. This is usually columns in a training or test set.
- A training dataset and a testing dataset, is used to test machine learning algorithms. First the training dataset is given as input to the program. Program uses this data to learn. Next, the test set is given to the program. The program decides which instance of test data belongs to which class. The predicted output is compared with the actual output of the program, and we can get a idea about the accuracy of the algorithm. There are best ways to use all the information in the training dataset and test dataset.
- Assume in car evaluation classification system, we have tested the program and it meets the desired level of accuracy. Knowledge representation is used to check what the machine has learned. There are many ways in which knowledge can be represented.
- We can use set of rules or a probability distribution to represent the knowledge.
- Many algorithms represent the knowledge which is more interpretable to humans than others. In some situations we may not want to build an expert system but we are interested only in the knowledge representation that's acquired from training a machine learning algorithm.

Table 1.2.1 : Car evaluation classification based on four features

Buying_Price	Maintenance_ Price	Lug_Boot	Safety	Evaluation?
High	High	Small	High	Unacceptable
High	High	Small	Low	Unacceptable
Medium	High	Small	High	Acceptable
Low	Medium	Small	High	Acceptable
Low	Low	Big	High	Acceptable
Low	Low	Big	Low	Unacceptable

Buying_Price	Maintenance_ Price	Lug_Boot	Safety	Evaluation?
Medium	Low	Big	Low	Acceptable
High	Medium	Small	High	Unacceptable
High	Low	Big	High	Acceptable
Low	Medium	Big	High	Acceptable
High	Medium	Big	Low	Acceptable
Medium	Medium	Small	Low	Acceptable
Medium	High	Big	High	Acceptable
Low	Medium	Small	Low	Unacceptable

Fig. 1.2.1: Features and target variable identified

1.3 TYPES OF MACHINE LEARNING

UQ. Explain how supervised learning is different from unsupervised learning. **(MU(Comp.) - May 17, 5 Marks)**

Some of the main types of machine learning are :

1. Supervised Learning

- In this type of learning we use data which is comprises of input and corresponding output. For every instance of data we can have input 'X' and corresponding output 'Y'. From this ML system will build model so that given an observation 'X', for new observation 'X' it will try to find out what is corresponding 'Y'.
- In supervised learning training data is labelled with the correct answers, e.g. "spam" or "ham." Two most important types of supervised learning are **classification** (where the outputs are discrete labels, as in spam filtering) and **regression** (where the outputs are real-valued).

Fig. 1.3.1 : Supervised Learning

2. Unsupervised learning

- In unsupervised learning you are only given input 'X', there is no label to the data and given the data or different data points, you may want to form clusters or want to find some pattern.
- Two important unsupervised learning tasks are **dimension reduction** and **clustering**.

Fig. 1.3.2 : Unsupervised Learning

3. Reinforcement learning

- In reinforcement learning you have an agent who is acting in an environment and you want to findout what action the agent must take based on the reward or penalty that the agent gets it.
- In this an agent (e.g., a robot or controller) seeks to learn the optimal actions to take based the outcomes of past actions.

Fig. 1.3.3 : Reinforcement Learning

4. Semi-supervised learning

- It is a combination of supervised and unsupervised learning. In this there is some amount of labeled training data and also you have large amount of unlabeled data and you try to come up with some learning algorithm that convert even when training data is not labeled.
- In classification task, the aim is to predict class of an instance of data. Another method in machine learning is regression.
- Regression is the prediction of a numeric value. Regression's example is to draw a best fit line which passes through some data points in order to generalize the data points.
- Classification and regression are examples of supervised learning. These types of problems are called as supervised because we are asking the algorithm what to predict.
- The exact opposite of supervised is a task called as unsupervised learning. In unsupervised learning, target value or label is not given for the data. A problem in which similar items are grouped together is called as clustering. In unsupervised learning, we may also want to find statistical values that describe the data. This is called as density estimation. Another task of unsupervised learning may be reducing the huge amount of data from many attributes to a small number so that we can properly visualize it in two or three dimensions.

Table 1.3.1 : Supervised learning tasks

K-Nearest Neighbours	Linear
Naïve Bayes	Locally weighted linear
Support Vector Machines	Ridge
Decision Trees	Lasso

Table 1.3.2 : Unsupervised learning tasks

K-Means	Expectation Maximization
DBSCAN	Parzen Window

1.4 ISSUES IN MACHINE LEARNING

UQ. What are the issues in Machine learning? **(MU(Comp.) - May 15, 5 Marks)**

1. Which algorithm we have to select to learn general target functions from specific training dataset? What should be the settings for particular algorithms, so as to converge to the desired function, given sufficient training data? Which algorithms perform best for which type of problems and representations?

2. How much training data is sufficient? What should be the general amount of data that can be found to relate the confidence in learned hypotheses to the amount training experience and the character of the learner's hypothesis space?
3. Prior knowledge held by the learner is used at which time and manner to guide the process of generalizing from examples? If we have approximately correct knowledge, will it helpful even when it is only approximately correct?
4. What is the best strategy for choosing a useful next training experience, and how does the choice of this strategy after the complexity of the learning problem?
5. To reduce the task of learning to one or more function approximation problems, what will be the best approach? What specific functions should the system attempt to learn? Can this process itself be automated?
6. To improve the knowledge representation and to learn the target function, how can the learner automatically alter its representation?

1.5 HOW TO CHOOSE THE RIGHT ALGORITHM ?

UQ. Explain the steps required for selecting the right machine learning algorithm. **(MU(Comp.) - May 16, 8 Marks)**

With all the different algorithms available in machine learning, how can you select which one to use ? First you need to focus on your goal. What are you trying to get out of this? What data do you have or can you collect ? Secondly you have to consider the data.

1. **Goal :** If you are trying to predict or forecast a target value, then you need to look into supervised learning. Otherwise, you have to use unsupervised learning.
   (a) If you have chosen **supervised** learning, then next you need to focus on what's your **target** value?
   If **target value** is **discrete** (e.g. Yes/ No, 1 /2/3, A/B/C), then use **Classification**.
   If **target value** is continuous i.e. Number of values (e.g. 0 – 100, – 99 to 99), then use **Regression.**
   (b) If you have chosen **unsupervised** learning, then next you need to focus on what is your **aim?**
   If you want to **fit your data** into some **discrete groups,** then use **Clustering**
   If you want to **find numerical estimate** of how strong the fit into each group, then use **density estimation algorithm**
2. **Data :** Are the features continuous or nominal ? Are there missing values in features? If yes, what is a reason for missing values? Are there outliers in the data? To narrow the algorithm selection process, all of these features of your data can help you.

Table 1.5.1 : Selection of Algorithm

	Supervised Learning	Unsupervised Learning
Discrete	Classification	Clustering
Continuous	Regression	Density Estimation

1.6 STEPS IN DEVELOPING A MACHINE LEARNING APPLICATION

UQ. Explain the steps of developing Machine Learning applications. **(MU(Comp.) - May 19, 10 Marks)**

1. Collection of Data

You could collect the samples from a website and extracting data.

- From RSS feed or an API
- From device to collect wind speed measurement
- Publicly available data.

2. Preparation of the input data

- Once you have the input data, you need to check whether it's in a useable format or not.
- Some algorithm can accept target variables and features as string; some need them to be integers.
- Some algorithm accepts features in a special format.

3. Analyse the input data

- Looking at the data you have passed in a text editor to check collection and preparation of input data steps are properly working and you don't have a bunch of empty values.
- You can also check at the data to find out if you can see any patterns or if there is anything obvious, such as a few data points greatly differ from remaining set of the data.
- Plotting data in 1, 2 or 3 dimensions can also help.
- Distil multiple dimensions down to 2/3 so that you can visualize the data.

4. The importance of this step is that it makes you understand that you don't have any garbage value coming in.

5. Train the algorithm

- Good clean data from the first two steps is given as input to the algorithm. The algorithm extracts information or knowledge. This knowledge is mostly stored in a format that is readily useable by machine for next 2 steps.
- In case of unsupervised learning, training step is not there because target value is not present. Complete data is used in the next step.

6. Test the algorithm

- In this step the information learned in the previous step is used. When you are checking an algorithm, you will test it to find out whether it works properly or not. In supervised case, you have some known values that can be used to evaluate the algorithm.
- In case of unsupervised, you may have to use some other matrices to evaluate the success. In either case, if you are not satisfied, you can again go back to step 4, change some things and test again.
- Mostly problem occurs in collection or preparation of data and you will have to go back to step 1.

7. Use it

In this step a real program is developed to do some task, and once again it is checked if all the previous steps worked as you expected. You might encounter some new data and have to revisit step 1-5.

Fig. 1.6.1 : Typical example of Machine Learning Application

1.7 APPLICATIONS OF MACHINE LEARNING

UQ. Write short note on: Machine learning applications. **(MU(Comp.) - May 16, May 17, 10 Marks)**

1. Learning Associations

- A supermarket chain-one an example of retail application of machine learning is basket analysis, which is finding associations between products bought by customers:
- If people who buy P typically also buy Q and if there is a customer who buys Q and does not buy P, he or she is a potential P customer. Once we identify such customers, we can target them for cross-selling.
- In finding an association rule, we are interested in learning a conditional probability of the form P (Q/P) where Q is the product we would like to condition on P, which are the product / products which we know that customer has already purchased.

$$P(\text{Milk} / \text{Bread}) = 0.7$$

- It implies that 70% of customers who buy bread also buy milk

2. Classification

- A credit is an amount of money loaned by a financial institution. It is important for the bank to be able to predict in advance the risk associated with a loan. Which is the probability that the customer will default and not pay the whole amount back?
- In credit scoring, the bank calculates the risk given the amount of credit and the information about the customer. (Income, savings, collaterals, profession, age, past financial history). The aim is to infer a general rule from this data, coding the association between a customer's attributes and his risk.
- Machine Learning system fits a model to the past data to be able to calculate the risk for a new application and then decides to accept or refuse it accordingly.

  If income > Q_1 and savings > Q_2

  Then low – risk ELSE high – risk
- Other classification examples are Optical character recognition, face recognition, medical diagnosis, speech recognition and biometric.

Fig. 1.7.1 : Classification for credit scoring

3. Regression

- Suppose we want to design a system that can predict the price of a flat. Let's take the inputs as the area of the flat, location and purchase year and other information that affects the rate of flat. The output is the price of the flat. The applications where output is numeric are regression problems.
- Let X represents flat features and Y is the price of flat. We can collect training data by surveying past purchased transactions and the Machine Learning algorithm fits a function to this data to learn Y as a function of X for the suitable values of W and W_0.

$$Y = w^*x + w_0$$

Fig. 1.7.2 : Regression for prediction of price of flat

4. Unsupervised Learning

- One of the important unsupervised learning problem is clustering. In clustering dataset is partitioned in to meaningful sub classes known as clusters. For example, suppose you want to decorate your home using given items.
- Now you will classify them using unsupervised learning (no prior knowledge) and this classification can be on the basis of color of items, shape of items, material used for items, type of items or whatever way you would like.

5. Reinforcement Learning

- There are some of the applications where output of system is a sequence of actions. In such applications the sequence of correct actions instead of single action is important in order to reach goal. An action is said to be good if it is part of good policy. Machine learning program generates a policy by learning previous good action sequences. Such methods are called reinforcement methods
- A good example of reinforcement learning is chess playing. In artificial intelligence and machine learning, one of the most important research area is game playing. Games can be are easily described but at the same time, they are quite difficult to play well. Let's take a example of chess that has limited number of rules, but the game is very difficult because for each state there can be large number of possible moves.
- Another application of reinforcement learning is robot navigation. The robot can move in all possible directions at any point of time. The algorithm should reach goal state from an initial state by learning the correct sequence of actions after conducting number of trial runs.
- When the system has unreliable and partial sensory information, it makes reinforcement learning complex. Let's take an example of robot with incomplete camera information. Here robot does not know its exact location.

1.8 TRAINING ERROR AND GENERALIZATION ERROR

Training Error

In **machine learning, training** a predictive model means finding a function which maps a set of values x to a value y. If we apply the model to the data it was trained on, we are calculating the **training error**. If we calculate the **error** on data which was unknown in the **training** phase, we are calculating the test **error**. Training error is calculated as follows:

$$E_{train} = \frac{1}{n}\sum_{i=1}^{n} \text{error}\,(f_D\,(X_i), Y_i)$$

In the above equation n represents the number of training examples. $f_D\,(X_i)$ represents the predicted value and Y_i represents the true or actual values, error $(f_D\,(X_i), Y_i)$ is used to represent that these two values are same or not and if not then these values differs by how much.

Generalization Error

For supervised learning applications in machine learning and statistical learning theory, **generalization error** is a measure of how accurately an algorithm is able to predict outcome values for previously unseen data. Generalization error is calculated as follows:

$$E_{gen} = \int \text{error}\,(f_D\,(X_i), Y_i)\, P\,(Y, X)\, dX$$

In the above equation error is calculated over all possible values of X and Y. $error\,(f_D\,(X_i), Y_i)$ is used to represent that these two values are same or not and if not then these values differes by how much. P(X, Y) represents how often we expect to see such X and Y.

Training Error versus Generalization Error

- The training error is the error of our model as calculated on the training dataset, while generalization error is the expectation of our model's error were we need to apply it to an infinite stream of additional data examples drawn from the same underlying data distribution as our original sample.

- Problematically, we can never calculate the generalization error exactly. That is because the stream of infinite data is an imaginary object. In practice, we must estimate the generalization error by applying our model to an independent test set constituted of a random selection of data examples that were withheld from our training set.
- Let's see an example. Consider a college student trying to prepare for his final exam. A diligent student will strive to practice well and test his abilities using exams from previous years. Nonetheless, doing well on past exams is no guarantee that he will excel when it matters. For instance, the student might try to prepare by rote learning the answers to the exam questions. This requires the student to memorize many things. She might even remember the answers for past exams perfectly. Another student might prepare by trying to understand the reasons for giving certain answers. In most cases, the latter student will do much better.
- Let's see one more example, consider the problem of trying to classify the outcomes of coin tosses (class 0: heads, class 1: tails) based on some contextual features that might be available. Suppose that the coin is fair. No matter what algorithm we come up with, the generalization error will always be 1/2. However, for most algorithms, we should expect our training error to be considerably lower, depending on the luck of the draw, even if we did not have any features! Consider the dataset {0, 1, 1, 1, 0, 1}. Our feature-less algorithm would have to fall back on always predicting the majority class, which appears from our limited sample to be 1. In this case, the model that always predicts class 1 will incur an error of 1/3, considerably better than our generalization error. As we increase the amount of data, the probability that the fraction of heads will deviate significantly from 1/2 diminishes, and our training error would come to match the generalization error.
- When we train our models, we attempt to search for a function that fits the training data as well as possible. If the function is so flexible that it can catch on to spurious patterns just as easily as to true associations, then it might perform *too well* without producing a model that generalizes well to unseen data. This is precisely what we want to avoid or at least control. Many of the techniques in deep learning are heuristics and tricks aimed at guarding against overfitting.
- When we have simple models and abundant data, we expect the generalization error to resemble the training error. When we work with more complex models and fewer examples, we expect the training error to go down but the generalization gap to grow.

Fig. 1.8.1 : Training Error and Generalization Error

1.9 UNDERFITTING, OVERFITTING, BIAS AND VARIANCE TRADE OFF

- Let us consider that we are designing a machine learning model. A model is said to be a good machine learning model if it generalizes any new input data from the problem domain in a proper way. This helps us to make predictions in the future data, that data model has never seen.
- Now, suppose we want to check how well our machine learning model learns and generalizes to the new data. For that we have overfitting and underfitting, which are majorly responsible for the poor performances of the machine learning algorithms.

- Before diving further let's understands two important terms:
- Bias – Assumptions made by a model to make a function easier to learn. (The algorithms error rate on the training set is algorithms bias.)Variance – If you train your data on training data and obtain a very low error, upon changing the data and then training the same previous model you experience high error, this is variance. (How much worse the algorithm does on the test set than the training set is known as the algorithms variance.)

Underfitting

- A statistical model or a machine learning algorithm is said to have underfitting when it cannot capture the underlying trend of the data.
- Underfitting destroys the accuracy of our machine learning model. Its occurrence simply means that our model or the algorithm does not fit the data well enough.
- It usually happens when we have less data to build an accurate model and also when we try to build a linear model with a non-linear data. In such cases the rules of the machine learning model are too easy and flexible to be applied on such minimal data and therefore the model will probably make a lot of wrong predictions.
- Underfitting can be avoided by using more data and also reducing the features by feature selection.

In a nutshell, **Underfitting – High bias and low variance**

Techniques to reduce underfitting:

1. Increase model complexity
2. Increase number of features, performing feature engineering
3. Remove noise from the data.
4. Increase the number of epochs or increase the duration of training to get better results.

Overfitting

- A statistical model is said to be overfitted, when we train it with a lot of data . When a model gets trained with so much of data, it starts learning from the noise and inaccurate data entries in our data set. Then the model does not categorize the data correctly, because of too many details and noise.
- The causes of overfitting are the non-parametric and non-linear methods because these types of machine learning algorithms have more freedom in building the model based on the dataset and therefore they can really build unrealistic models.
- A solution to avoid overfitting is using a linear algorithm if we have linear data or using the parameters like the maximal depth if we are using decision trees.

In a nutshell, **Overfitting – High variance and low bias**

Techniques to reduce overfitting:

1. Increase training data.
2. Reduce model complexity.
3. Early stopping during the training phase (have an eye over the loss over the training period as soon as loss begins to increase stop training).
4. Ridge Regularization and Lasso Regularization
5. Use dropout for neural networks to tackle overfitting.

- Ideally, the case when the model makes the predictions with 0 error, is said to have a *good fit* on the data. This situation is achievable at a spot between overfitting and underfitting. In order to understand it we will have to look at the performance of our model with the passage of time, while it is learning from training dataset.
- With the passage of time, our model will keep on learning and thus the error for the model on the training and testing

data will keep on decreasing. If it will learn for too long, the model will become more prone to overfitting due to the presence of noise and less useful details. Hence the performance of our model will decrease. In order to get a good fit, we will stop at a point just before where the error starts increasing. At this point the model is said to have good skills on training datasets as well as our unseen testing dataset.

Bias-variance trade-off

So what is the right measure? Depending on the model at hand, a performance that lies between overfitting and underfitting is more desirable. This trade-off is the most integral aspect of Machine Learning model training. As we discussed, Machine Learning models fulfil their purpose when they generalize well. Generalization is bound by the two undesirable outcomes — high bias and high variance. Detecting whether the model suffers from either one is the sole responsibility of the model developer.

Fig. 1.9.1 : Underfitting and Overfitting

Bias-variance trade-off

So what is the right measure? Depending on the model at hand, a performance that lies between overfitting and underfitting is more desirable. This trade-off is the most integral aspect of Machine Learning model training. As we discussed, Machine Learning models fulfil their purpose when they generalize well. Generalization is bound by the two undesirable outcomes — high bias and high variance. Detecting whether the model suffers from either one is the sole responsibility of the model developer.

Fig. 1.9.2 : Bias variance Tradeoff as a function of model capacity

Chapter Ends...

❑❑❑

MODULE 2

CHAPTER 2

Learning with Regression and Trees

Syllabus

Learning with Regression : Linear Regression, Multivariate Linear Regression, Logistic Regression.

Learning with Trees: Decision Trees, Constructing Decision Trees using Gini Index (Regression), Classification and Regression Trees (CART).

Performance Metrics: Confusion Matrix, [Kappa Statistics], Sensitivity, Specificity, Precision, Recall, F-measure, ROC curve.

2.1 Linear Regression 2-2
UQ. Explain Regression line, Scatter plot, Error in prediction and Best fitting line. **(MU(Comp) – May 15, 4 Marks)** 2-2
UQ. Explain in brief Linear Regression Technique. **(MU(Comp) – May 16, 5 Marks)** 2-2
2.1.1 Simple Linear Regression 2-2
2.2 Examples of Linear Regression 2-4
UEx. 2.2.3 **(MU - May 17, 10 Marks)** 2-5
UEx. 2.2.4 **(MU - Dec. 19, 10 Marks)** 2-6
2.3 Multiple Linear Regression 2-7
2.4 Multivariate Linear Regression 2-9
2.5 Multivariate Multiple Linear Regression 2-10
2.6 Logistic Regression 2-11
UQ. Write short note on : Logistic Regression. **(MU(Comp) – May 17, 10 Marks)** 2-11
UQ. Explain how regression problem can be solved using Steepest descent method. Write down the steps. **(MU(Comp) – May 19, 5 Marks)** 2-11
2.7 Decision Trees 2-14
UQ. Write short note on Issues in Decision Tree. **(MU(Comp) - May 15, 10 Marks)** 2-14
2.8 Constructing Decision Trees 2-17
2.9 Example of Classification Tree Using ID3 2-18
2.10 Example of Decision Tree Using Gini Index 2-29
UEx. 2.10.4 **(MU - May 15, 12 Marks)** 2-35
UEx. 2.10.5 **(MU - May 17, 10 Marks)** 2-36
UEx. 2.10.6 **(MU - May 19, 10 Marks)** 2-38
2.11 Classification and Regression Tree (CART) 2-40
2.12 Example of Regression Tree 2-40
2.13 Performance Metrics 2-48
2.13.1 Performance Metrics for Regression 2-48
2.13.2 Performance Metrics for Classification 2-49
2.13.3 Kappa Statistics 2-51
• **Chapter Ends** 2-51

2.1 LINEAR REGRESSION

UQ. Explain Regression line, Scatter plot, Error in prediction and Best fitting line. **(MU(Comp) – May 15, 4 Marks)**

UQ. Explain in brief Linear Regression Technique. **(MU(Comp) – May 16, 5 Marks)**

- One of the most important supervised learning tasks is regression. In regression set of records are present with X and Y values and this values are used to learn a function, so that if you want to predict Y from an unknown X this learn function can be used. In regression we have to find value of Y, So, a function is required which predicts Y given X. Y is continuous in case of regression.
- Here Y is called as criterion variable and X is called as predictor variable. There are many types of functions or modules which can be used for regression. Linear function is the simplest type of function. Here, X may be a single feature or multiple features representing the problem.

Fig. 2.1.1 : Types of Regression Models

2.1.1 Simple Linear Regression

- Let's see simple regression first, in this X contains a single feature. In multiple regressions, X contains more than one feature. In simple regression training records are plotted as value of X vs. value of Y. Next task is to find a function, so that if a random unknown X value is given we can predict Y. There are different types of functions that can be used. In linear regression, we assume that the function is linear as shown in Fig. 2.1.2(a).

Fig. 2.1.2 : Types of regression functions

- In Linear regression a best fitted straight line is drawn which passes through the points called as the regression line. The line shown in Fig. 2.1.2 is the regression line. Regression line shows the calculated values of Y for each possible value of X. Errors in the prediction is shown by the vertical lines drawn from the points to the regression line. Error in prediction is less when the point is very near to the regression line as shown by first and second point in Fig. 2.1.3. When the point is far from the regression line then the error in prediction is high, as shown by the third and fourth point in Fig. 2.1.3.
- The difference between the value of point and the predicted value is called as error of prediction. Predicted value is the value of point on the line.

Fig. 2.1.3 : Prediction and the Error in the Prediction

- Let's take an example, the predicted values (Y') and the errors of prediction ($Y-Y'$) are shown in Table 2.1.1. From the table we can say that, the second point has Y value as 4 and a predicted Y value as 3.2. The error of prediction is 0.8.

Table 2.1.1 : Regression data

Sr. No.	X	Y	Y'	$Y-Y'$	$(Y-Y')^2$
1	2	2	2.2	– 0.2	0.04
2	4	4	3.2	0.8	0.64
3	6	3	4.3	– 1.3	1.69
4	8	6	5.4	0.6	0.36

- The best-fit line is called as the line that will minimize the sum of the squared errors of prediction. This criterion is used to draw the line in Fig. 2.1.3. The squared errors of prediction are shown in the last column of Table 2.1.1.
- The regression line is represented using the following equation,

$$Y' = aX + b + e$$

- In the above equation Y' represents the predicted value, a represents the slope of the line, b shows the Y intercept and e is the random error. Here we assume that mean value of random error is 0, so the equation becomes,

$$Y' = aX + b$$

Table 2.1.2 : Calculation of Regression Parameters

Sr. No.	X	Y	XY	X^2
1	2	2	4	4
2	4	4	16	16
3	6	3	18	36
4	8	6	48	64
Total	**20**	**15**	**86**	**120**

- The regression line equation is,

$$Y' = aX + b$$

$$a = \frac{n\sum XY - \sum X \sum Y}{n\sum X^2 - (\sum X)^2} = \frac{4 \times 86 - 20 \times 15}{4 \times 120 - 400} = 0.55$$

$$b = \frac{1}{n}(\sum Y - a \times \sum X) = \frac{1}{4}(15 - 0.55 \times 20) = 1$$

- Now the equation for the line becomes,

$$Y' = 0.55X + 2.2$$

For X = 2, $Y' = (0.55)(2) + 1 = 2.2$

For X = 4, $Y' = (0.55)(4) + 1 = 3.2$

For X = 6, $Y' = (0.55)(6) + 1 = 4.3$

For X = 8, $Y' = (0.55)(8) + 1 = 5.4$

2.2 EXAMPLES OF LINEAR REGRESSION

Ex. 2.2.1 : The expenditure of an organization (in thousand) for every month is shown in table below :

X (Month)	1	2	3	4	5
Y (Expenditure)	12	19	29	37	45

Find regression line, Y = a X + b using least square method.

Estimate the expenditure of company in 6th month using line as a model.

Soln. :

Sr. No.	X	Y	XY	X^2
1	1	12	12	1
2	2	19	38	4
3	3	29	87	9
4	4	37	148	16
5	5	45	225	25
Total	**15**	**142**	**510**	**55**

(a) The equation for the regression line is,

$$Y' = a X + b$$

$$a = \frac{n\sum XY - \sum X \sum Y}{n\sum X^2 - (\sum X)^2} = \frac{5 \times 510 - 15 \times 142}{5 \times 55 - 225} = 8.4$$

$$b = \frac{1}{n}(\sum Y - a \times \sum X) = \frac{1}{5}(142 - 8.4 \times 15) = 3.2$$

Now the equation for the line becomes,

$$Y' = 8.4 X + 3.2$$

(b) In 6th month

$$Y' = 8.4 \times 6 + 3.2$$

$$Y' = 53.6 \text{ Thousand}$$

Ex. 2.2.2 : Consider the set of data as {(– 1, – 1), (2, 2), (3, 2)} (a) Find the equation of regression line. (b) Draw the scatter plot of data and regression line.

Soln. :

Sr. No.	X	Y	XY	X^2
1	– 1	– 1	1	1
2	2	2	4	4
3	3	2	6	9
Total	4	3	11	14

(a) The equation for the regression line is,

$$Y' = a X + b$$

$$a = \frac{n\sum XY - \sum X \sum Y}{n\sum X^2 - (\sum X)^2} = \frac{3 \times 11 - 4 \times 3}{3 \times 14 - 16} = 0.807$$

$$b = \frac{1}{n}(\sum Y - a \times \sum X) = \frac{1}{3}(3 - 0.807 \times 4) = -0.076$$

Now the equation for the line becomes,

$$Y' = 0.807\,X - 0.076$$

(b) Now we can plot the regression line given by equation $Y' = 0.807\,X - 0.076$ and the given data points.

Sr. No.	X	Y'
1	– 1	– 0.883
2	2	1.54
3	3	2.34

Fig. Ex. 2.2.2 : Scatter plot of data

UEx. 2.2.3 **MU - May 17, 10 Marks**

Following table shows the midterm and final exam grades obtained for students in a database course. Use the method of least squares using regression to predict the final exam grade of a student who received 86 in the mid term exam.

Midterm exam(X)	72	50	81	74	94	86	59	83	86	33	88	81
Final exam(Y)	84	53	77	78	90	75	49	79	77	52	74	90

☑ **Soln. :**

Sr. No.	X	Y	XY	X^2
1	72	84	6048	5184
2	50	53	2650	2500
3	81	77	6237	6561
4	74	78	5772	5476
5	94	90	8460	8836
6	86	75	6450	7396
7	59	49	2891	3481

Sr. No.	X	Y	XY	X^2
8	83	79	6557	6889
9	86	77	6622	7396
10	33	52	1716	1089
11	88	74	6512	7744
12	81	90	7290	6561
Total	**887**	**878**	**67205**	**69113**

The equation for the regression line is,

$$Y' = aX + b$$

$$a = \frac{n\sum XY - \sum X \sum Y}{n\sum X^2 - (\sum X)^2} = 0.65$$

$$b = \frac{1}{n}(\sum Y - a \times \sum X) = 25.12$$

Now the equation for the line becomes,

$$Y' = 0.65X + 25.12$$

The final exam grade of a student who received 86 in the mid term exam,

$$Y' = 0.65 \times 86 + 25.12$$

$$Y' = 81.02$$

UEx. 2.2.4 **MU - Dec. 19, 10 Marks**

Given the following data for the sales of car of an automobile company for six consecutive years. Predict the sales for next two consecutive years.

Years (x)	2013	2014	2015	2016	2017	2018
Sales (y)	110	100	250	275	230	300

☑ **Soln :**

We will take t = x – 2013

Sr. No.	t	Y	tY	t^2
0	0	110	0	0
1	1	100	100	1
2	2	250	500	4
3	3	275	825	9
4	4	230	920	16
5	5	300	1500	25
Total	**15**	**1265**	**3845**	**55**

The equation for the regression line is,

$$Y' = at + b; \qquad a = 39; \qquad b = 113.33$$

Now the equation for the line becomes,

$$Y' = 39t + 113.33$$

The sale of company for next two years,

$$X = 2019, t = 6; \qquad Y' = 39 \times 6 + 113.33 = 347.33$$

$$X = 2020, t = 7; \qquad Y' = 39 \times 7 + 113.33 = 386.33$$

2.3 MULTIPLE LINEAR REGRESSION

- In Multiple linear regressions there are two or more number of features. We can also say it is an extension to simple linear regression.
- The regression line is represented using the following equation,

$$Y' = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \text{----------} + \beta_n X_n + e$$

- In the above equation Y' is the predicted value, X_1 X_2 ..... X_n are the predictors, e is random error and $\beta_0, \beta_1, \beta_2, \ldots\ldots \beta_n$ are regression parameters.
- **Computing parameters**

Now let us talk in terms of matrices as it is easier that way. As discussed before, if we have n independent variables in our training data, our matrix X has n + 1 rows, where the first row is the 0th term added to each vector of independent variables which has a value of 1 (this is the coefficient of the constant term α). So, X is as follows,

$$X = [X_1..X_m]^t$$

X_i contains n entries corresponding to each feature in training data of ith entry. So, matrix **X** has m rows and n+1 columns (0th column is all 1s and rest for one independent variable each).

$$Y = [Y_1 Y_2..Y_m]^t$$

and coefficient matrix C,

$$C = [\alpha, \beta_1..\beta_n]$$

and our final equation for our hypothesis is,

$$Y = XC$$

To calculate the coefficients, we need **n + 1** equations and we get them from the minimizing condition of the error function. Equating partial derivative of $E(\alpha, \beta_1, \beta_2,\ldots,\beta_n)$ with each of the coefficients to 0 gives a system of n + 1 equations. Solving these is a complicated step and gives the following nice result for matrix C,

$$C = (X^T X)^{-1} X^T y$$

where y is the matrix of the observed values of dependent variable. This method seems to work well when the n value is considerably small (approximately for 3-digit values of n). As n grows big the above computation of matrix inverse and multiplication take large amount of time. In future tutorials lets discuss a different method that can be used for data with large no. of features.

Example : Multiple Linear Regression

Ex. 2.3.1 : The data in following table relate grams plant dry weight ,y, to percent soil organic matter, x_1, and kilograms of supplemental soil nitrogen added per 1000 square meters, x_2. Obtain the multiple regression equation.

Y	X_1	X_2
78.5	7	2.6
74.3	1	2.9
104.3	11	5.6
87.6	11	3.1
95.9	7	5.2
109.2	11	5.5
102.7	3	7.1

☑ **Soln. :** We want to find the estimated coefficients β_0, β_1 and β_2 for the regression equation.

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \text{ using matrices}$$

First, we have to write the data in matrix form :

$$Y = \begin{bmatrix} 78.5 \\ 74.3 \\ 104.3 \\ 87.6 \\ 95.9 \\ 109.2 \\ 102.7 \end{bmatrix} \qquad X = \begin{bmatrix} 1 & 7 & 2.6 \\ 1 & 1 & 2.9 \\ 1 & 11 & 5.6 \\ 1 & 11 & 3.1 \\ 1 & 7 & 5.2 \\ 1 & 11 & 5.5 \\ 1 & 3 & 7.1 \end{bmatrix}$$

Now, we find the $(3x_3)$ X^TX matrix

$$X^T X = \begin{bmatrix} 7 & 51 & 32 \\ 51 & 471 & 235 \\ 32 & 235 & 163.84 \end{bmatrix}$$

Find the inverse of the matrix $(X^TX)^{-1}$

$$= \begin{bmatrix} 1.7996 & -0.0685 & -0.2532 \\ -0.0685 & 0.0101 & -0.0011 \\ -0.2532 & -0.0011 & 0.0571 \end{bmatrix}$$

The estimated regression coefficients is given by: $\beta = (X^TX)^{-1} X^T Y$

$$= \begin{bmatrix} 51.6 \\ 1.5 \\ 6.72 \end{bmatrix}$$

We get the multiple regression equation:

$$Y = 51.6 + 1.5 X_1 + 6.72 X_2$$

2.4 MULTIVARIATE LINEAR REGRESSION

- In this section we present multivariate regression model, in which we consider the relationship between more than one dependent variable and one independent variable. This model is similar to the multiple regression model in solving the normal equations and estimate the regression parameters. These parameters are easily estimated using matrix form.
- Suppose that the number of response variables is m, so we have n observations for each y_i, i = 1,2,...,m. The general formula for the multivariate regression model is given by :

$$y_i = \beta_{0i} + \beta_{1i} x_1 + \varepsilon_i$$

$$\hat{y}_i = \beta_{0i} + \beta_{1i} x_1 , i = 1,2,\ldots, m$$

- There are two parameters for each response to be estimated when the linear model includes the intercept β_0.
- Four matrices are needed to express the linear model in matrix notation:
- Y : the n × m matrix of observations on the dependent variable y .
- X : the n × 2 matrix consisting of a column of ones, which is labeled 1, followed by the column vector of the observations on the independent variable.
- B : the 2 × m matrix of parameters to be estimated.

Example : Multivariate Linear Regression

Ex. 2.4.1 : Given the following data in table , find the multivariate regression equations.

X_0	Y_1	Y_2
0	1	-1
1	4	-1
2	3	2
3	8	3
4	9	2

☑ **Soln. :** The regression equation takes the form,

$$\dot{y}_1 = \dot{\beta}_{01} + \dot{\beta}_{11} x_1 ; \qquad \dot{y}_2 = \dot{\beta}_{02} + \dot{\beta}_{12} x_2$$

$$Y = \begin{bmatrix} 1 & -1 \\ 4 & -1 \\ 3 & 2 \\ 8 & 3 \\ 9 & 2 \end{bmatrix} \qquad X = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix}$$

Now, we find the $X^T X$ matrix

$$X^T X = \begin{bmatrix} 5 & 10 \\ 10 & 30 \end{bmatrix}$$

Find the inverse of the matrix $(X^T X)^{-1}$

$$= \begin{bmatrix} 0.6 & -0.25 \\ -0.2 & 0.1 \end{bmatrix}$$

The estimated regression coefficients is given by: $\hat{\beta}_{01} = (X^T X)^{-1} X^T Y$

$$= \begin{bmatrix} 1 & -1 \\ 2 & 1 \end{bmatrix}$$

We get the equation :

$$\hat{Y}_1 = 1 + 2x_1 \quad , \quad \hat{Y}_2 = -1 + x_1$$

2.5 MULTIVARIATE MULTIPLE LINEAR REGRESSION

In multivariate multiple linear regression model we consider the relationship between more than one dependent variable and more than one independent variable. That is, we extend the regression model to the situation where we have measured m responses $y_1, y_2, \ldots, y_m$ and the same set of k predictors $x_1, x_2, \ldots, x_k$ is used in a sample of size n, then each response variable follows its own regression model:

$$y_1 = \beta_{01} + \beta_{11} x_1 + \beta_{21} x_2 + \ldots + \beta_{k1} x_k + \varepsilon_1$$
$$y_2 = \beta_{02} + \beta_{12} x_1 + \beta_{22} x_2 + \ldots + \beta_{k2} x_k + \varepsilon_2$$
$$\vdots$$
$$y_m = \beta_{0m} + \beta_{1m} x_1 + \beta_{2m} x_2 + \ldots + \beta_{km} x_k + \varepsilon_m$$

The error term $\varepsilon^T = [\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_m]$ has $E(\varepsilon) = 0$ and $\text{var}(\varepsilon) = \Sigma$. Thus the error terms associated with different responses may be correlated .

β_{ij}, i = 1,2,...,k and j = 1,2,...,m, is the estimated regression coefficient of the jth response in the effect of the ith predictor. β_{0j} is the intercept parameter of the jth response.

To represent the model in matrix form, we need four types of matrices:

Y : the $n \times m$ matrix consisting of m column vectors of the observations on each of the dependent variables;

X : the $n \times (k+1)$ matrix consisting of a column of ones followed by the k column vectors of the observations on the independent variables;

B : the $(k+1) \times m$ matrix consisting of column vectors of parameters to be estimated.

E : the $n \times m$ matrix consisting of column vectors of random errors.

Example : Multivariate Multiple Linear Regression

Suppose we had the following six sample observations, as shown in Table 3.2, on two independent variables (palatability and texture) and two dependent variables (purchase intent and overall quality) for some product. Use the data to estimate the multivariate multiple linear regression model.

Palatability(X_1)	Texture(X_2)	Overall quality(Y_1)	Purchase intent(Y_2)
65	71	63	67
72	77	70	70
77	73	72	70
68	78	75	72
81	76	89	88
73	87	76	77

$$\hat{y}_1 = \hat{\beta}_{01} + \hat{\beta}_{11} x_1 + \hat{\beta}_{21} x_2$$

$$\hat{y}_2 = \hat{\beta}_{02} + \hat{\beta}_{12} x_1 + \hat{\beta}_{22} x_2$$

First, we have to write the data in matrix form:

$$Y = \begin{bmatrix} 63 & 67 \\ 70 & 70 \\ 72 & 70 \\ 75 & 72 \\ 89 & 88 \\ 76 & 77 \end{bmatrix} \qquad X = \begin{bmatrix} 1 & 63 & 67 \\ 1 & 70 & 70 \\ 1 & 72 & 70 \\ 1 & 75 & 72 \\ 1 & 89 & 88 \\ 1 & 76 & 77 \end{bmatrix}$$

Now, we find the (3x3) $X^T X$ matrix

$$X^T X = \begin{bmatrix} 6 & 436 & 462 \\ 436 & 31852 & 33591 \\ 462 & 33591 & 35728 \end{bmatrix}$$

Find the inverse of the matrix $(X^T X)^{-1}$

$$= \begin{bmatrix} 65.5606 & -0.3783 & -0.4553 \\ -0.3783 & 0.060 & -0.0007 \\ 0.4533 & -0.0007 & 0.0066 \end{bmatrix}$$

The estimated regression coefficients is given by : $\hat{\beta} = (X^T X)^{-1} X^T Y$

$$= \begin{bmatrix} -37.5012 & -21.4323 \\ 1.1346 & 0.9409 \\ 0.3795 & 0.3514 \end{bmatrix}$$

So now, we get the multivariate multiple regression equations:

$\hat{y} = -37.5012 + 1.1346\, x_1 + 0.3795\, x_2$ and $\hat{y}_2 = -21.4323 + 0.9409\, x_1 + 0.3514\, x_2$

Using this estimated regression model, we can find the predicted values of $\hat{y}_1$ and $\hat{y}_2$ for specific values of x_1 and x_2. For example, suppose we randomly select a product which has a palatability 68 and a texture of 78, that is, $x_1 = 68$ and $x_2 = 78$. Then the predicted value of the overall quality for this product is:

$\hat{y} = -37.5012 + 1.1346\,(68) + 0.3795\,(78) = 69.2526$ and

the predicted value of the purchase intent is : $\hat{y}_2 = -21.4323 + 0.9409\,(68) + 0.3514\,(78) = 69.9581$

2.6 LOGISTIC REGRESSION

UQ. Write short note on : Logistic Regression. **(MU(Comp) – May 17, 10 Marks)**

UQ. Explain how regression problem can be solved using Steepest descent method. Write down the steps. **(MU(Comp) – May 19, 5 Marks)**

- Suppose we have different training data which belongs to two different classes, and we have to design a system that will identify which data is from which class. The output of this function will be a real value and it is not suitable for classification method. We can use another function on this linear function so that we can use the result for classification. In logistic regression logistic function or the sigmoid can be used.

- Let's first see what is the meaning of Logistic or Sigmoid function.
- In Logistic regression classifier we will take features as the input. These features are multiplied with the logistic coefficients and we add the product. This is called as the net input that will be given to the sigmoid function. The output will be between 0 and 1. Anything above 0.5 is classified as 1and anything below 0.5 is classified as 0.
- The net input to the sigmoid function is,

$$Z = b_0 + b_1 x_1 + b_2 x_2 + \text{.....} + b_n x_n$$

Fig. 2.6.1 : Logistic or Sigmoid Function

- In the net input equation x represents the input data or the features. When we use optimized coefficient b, Classifier will be successful. Optimized b can be calculated using the optimization concept.

Gradient Ascent Method

- In gradient ascent method we move in the direction of the gradient to find the maximum point on a function.

$$\nabla f(x, y) = \begin{bmatrix} \dfrac{\partial f(x, y)}{\partial x} \\ \dfrac{\partial f(x, y)}{\partial y} \end{bmatrix}$$

- In the above equation $\partial f(x, y)/\partial x$ represents the amount by which updation is applied in x direction and $\partial f(x, y)/\partial y$ represents the amount by which updation applied in y direction
- The gradient operator will always point towards the direction of gradient increase.

$$b = b + \alpha \times \nabla f(b)$$

- This step is repeated until we reach stopping criterion.

Pseudo code

```
Start using the logistic coefficient, all set to 0
Repeat no. of times
Find the gradient of complete dataset
Update, logistic coefficient = logistic coefficient + alpha × gradient
Return the logistic coefficient
```

Gradient Descent Method

- In gradient descent method we move in the opposite direction of the gradient to find the minimum point on a function
- The gradient operator will always point opposite to the direction of gradient increase.

$$b = b - \alpha \times \nabla f(b)$$

- This step is repeated until we reach stopping criterion.
- Using above mentioned methods optimized b is calculated, net input is calculated and then the prediction is calculated by giving the net input to the function,

$$\text{Prediction} = \frac{1}{1 + e^{-x}}$$

- Finally the data points are classified as,

$$\text{Class} = 1 \text{ if Prediction} \geq 0.5$$
$$= 0 \text{ else}$$

Fig. 2.6.2 : Sample of output of Logistic regression

- The classified data using the decision boundary is as shown in Fig. 2.6.2.
- Logistic regression is used to model a relationship between input variables and a target variable. Let's take an example of a house management system , we can use logistic regression to model the relationship between the parameters such as the total monthly income of the family, various liabilities, monthly expenditure of a family to predict if monthly savings (investment) can be done or not.
- There are three types of logistic regression as below,

Binary Logistic Regression

- Binary logistic regression is used if the target variable is binary.
- The application of this method can be above mentioned house management example.

Nominal Logistic Regression

- Nominal Logistic Regression is used if there are three or more classes without any specific orders.
- The examples of this type of regression could be different departments of the engineering college (Computer, IT, Mechanical, Civil, Electronics etc.).

Ordinal Logistic Regression

- Ordinal Logistic Regression is used if there are three or more classes with specific orders.
- The examples of this type of regression could be how customers rate the taste of food using a scale of 1-3 (Bad, Good, and Excellent).

Ex. 2.6.1 : A Bank has to decide whether to sanction loan or not based on two attributes as person's income and his savings. The data is given in the following table where 1 represents loan is sanctioned and 0 represents loan is not sanctioned. Predict whether a person 3 will get a loan or not having annual income as 12.5 lakhs and savings as 10 lakhs.

Person	Annual income in lakhs (x_1)	Savings in lakhs (x_2)	Loan sanctioned? (y)
1	14.5	12.5	1
2	8.5	4.5	0

☑ **Soln. :** Initially assume logistic regression coefficients $b_0 = b_1 = b_2 = 0$

For 1st row, $x_1 = 14.5$, $x_2 = 12.5$ and $y = 1$

Now we will calculate prediction for the first row,

$$\text{Prediction} = 1 / (1 + \hat{e}\ (-(b_0 + b_1 \times x_1 + b_2 \times x_2)))$$

$$\text{Prediction} = 1 / (1 + \hat{e}\ (-(0 + 0 \times 14.5 + 0 \times 12.5)))$$

$$\text{Prediction} = 0.5$$

Now we will calculate the new coefficient values using a simple update equation. Ideal values for alpha are from 0.1 to 0.3. Let's take alpha as 0.3. For b_0 by default input is 1.

$$b_{new} = b_{old} + \text{alpha} \times (y - \text{prediction}) \times \text{prediction} \times (1 - \text{prediction}) \times \text{input}$$

$$b_{0new} = 0 + 0.3 \times (1 - 0.5) \times 0.5 \times (1 - 0.5) \times 1.0 = 0.0375$$

$$b_{1new} = 0 + 0.3 \times (1 - 0.5) \times 0.5 \times (1 - 0.5) \times 14.5 = 0.54375$$

$$b_{2new} = 0 + 0.3 \times (1 - 0.5) \times 0.5 \times (1 - 0.5) \times 12.5 = 0.46875$$

Now we will calculate prediction for the second row,

$$\text{Prediction} = 1 / (1 + \hat{e}\ (-(b_0 + b_1 \times x_1 + b_2 \times x_2)))$$

$$\text{Prediction} = 1 / (1 + \hat{e}(-(0.0375 + 0.54735 \times 8.5 + 0.46875 \times 4.5)))$$

$$\text{Prediction} = 0.99$$

Now we will calculate the new coefficient values

$$b_{new} = b_{old} + \text{alpha} \times (y - \text{prediction}) \times \text{prediction} \times (1 - \text{prediction}) \times \text{input}$$

$$b_{0new} = 0.0375 + 0.3 \times (0 - 0.99) \times 0.99 \times (1 - 0.99) \times 1.0 = 0.034$$

$$b_{1new} = 0.54735 + 0.3 \times (0 - 0.99) \times 0.99 \times (1 - 0.99) \times 8.5 = 0.523$$

$$b_{2new} = 0.46875 + 0.3 \times (0 - 0.99) \times 0.99 \times (1 - 0.99) \times 4.5 = 0.456$$

Now we will use these values for prediction

$$\text{Prediction} = 1 / (1 + \hat{e}(-(b_0 + b_1 \times x_1 + b_2 \times x_2)))$$

$$\text{Prediction} = 1 / (1 + \hat{e}(-(0.034 + 0.523 \times 12.5 + 0.456 \times 10)))$$

$$\text{Prediction} = 0.99$$

$$\text{Since prediction} \geq 0.5$$

Prediction for person 3 is, his loan will be sanctioned.

Note: Generally huge amount of data is used for training and number of iterations are also applied to get accuracy. Here for example purpose only two records for training are taken and a single iteration is shown.

2.7 DECISION TREES

UQ. Write short note on Issues in Decision Tree. **(MU(Comp) - May 15, 10 Marks)**

- Decision trees are very strong and most suitable tools for classification and prediction. The attractiveness of decision trees is due to the fact that, in contrast to neural network, decision trees represent rules.
- Rules are represented using linguistic variables so that user interpretability may be achieved. By comparing the records with the rules one can easily find a particular category to which the record belongs to.
- In some applications, the accuracy of a classification or prediction is the only thing that matters in such situations we do not necessarily care how or why the model works. In other situations, the ability to explain the reason for a decision is crucial, in marketing one has described the customer segments to marketing professionals, so that they can use this knowledge to start a victorious marketing campaign.
- This domain expert must acknowledge and approve this discovered knowledge and for this we need good descriptions. There are a variety of algorithms for building decision trees that share the desirable quality of interpretability (ID3).

1. Where Decision Tree is applicable?

Decision tree method is mainly used for the tasks that possess the following properties.

- The tasks or the problems in which the records are represented by attribute-value pairs.
  - Records are represented by a fixed set of attribute and their value Example: For 'temperature' attribute the value is 'hot'.
  - When there are small numbers of disjoint possible values for each attribute, then decision tree learning becomes very simple.
  - Example: Temperature attribute takes three values as hot, mild and cold.
  - Basic decision tree algorithm may be extended to allow real valued attributes as well.
  - Example: we can define floating point temperature.
- An application where the target function takes discrete output values.
  - In Decision tree methods an easiest situation exists, if there are only two possible classes.

- Example: Yes or No
- When there are more than two possible output classes then decision tree methods can also be easily extended.
- A more significant extension allows learning target functions with real valued outputs, although the application of decision trees in this area is not frequent.

- The tasks or the problems where the basic requirement is the disjunctive descriptors.

  Decision trees naturally represent disjunctive expressions.
- In certain cases where the training data may contain errors.

  Decision tree learning methods are tolerant to errors that can be a classification error of training records or attribute-value representation error.
- The training data may be incomplete as there are missing attribute values

  Although some training records have unknown values, decision tree methods can be used.

2. Decision Tree Representation

- Decision tree is a classifier which is represented in the form of a tree structure where each node is either a leaf node or a decision node.
  - Leaf node represents the value of the target or response attribute (class) of examples.
  - Decision node represents some test to be carried out on a single attribute-value, with one branch and sub tree for each possible outcome of the test.
- Decision tree generates regression or classification models in the form of a tree structure. Decision tree divides a dataset into smaller subsets with increase in depth of tree.
- The final decision tree is a tree with decision nodes and leaf nodes. A decision node (e.g., Buying_Price) has two or more branches (e.g., High, Medium and Low). Leaf node (e.g., Evaluation) shows a classification or decision. The topmost decision node in a tree which represents the best predictor is called root node. Decision trees can be used to represent categorical as well as numerical data.
  - **Root Node :** It represents entire set of records or dataset and this is again divided into two or more similar sets.
  - **Splitting :** Splitting procedure is used to divide a node into two or more sub-nodes depending on the criteria.
  - **Decision Node :** A decision node is a sub-node which is divided into more sub-nodes.
  - **Leaf/ Terminal Node :** Leaf node is a node which is not further divided or a node with no children.

  - **Parent and** Child **Node :** Parent node is a node, which is split into sub-nodes and sub-nodes are called as child of parent node.
  - **Branch / Sub-Tree :** A branch or sub-tree is a sub part of decision tree.
  - **Pruning :** Pruning method is used to reduce the size of decision trees by removing nodes.

3. Attribute Selection Measure

1. Gini Index

- All attributes are assumed to be continuous valued.
- It is assumed that there exist several possible split values for each attribute.

- Gini index method can be modified for categorical attributes.
- Gini is used in Classification and Regression Tree (CART).

If a data set T contains example from n classes, gini index, gini (T) is defined as,

$$\text{Gini}(T) = 1 - \sum_{j=1}^{n} (P_j)^2 \quad \text{...(2.7.1)}$$

In the above equation P_j represents the relative frequency of class j in T.

After splitting T into two subsets T_1 and T_2 with sizes N_1 and N_2, gini index of split data is,

$$\text{gini}_{\text{split}}(T) = \frac{N_1}{N}\text{gini}(T_1) + \frac{N_2}{N}\text{gini}(T_2) \quad \text{... (2.7.2)}$$

The attribute with smallest $\text{gini}_{\text{split}}(T)$ is selected to split the node.

2. Information Gain (ID3)

In this method all attributes are assumed to be categorical. The method can be modified for continuous valued attributes. Here we select the attribute with highest information gain.

Assume there are 2 classes P and N. Let the set of records S contain p records of class P and n records of class N.

The amount of information required to decide if a random record in S belongs to P or N is defined as,

$$I(p, n) = -\left(\frac{p}{p+n}\right)\log_2\left(\frac{p}{p+n}\right) - \left(\frac{n}{p+n}\right)\log_2\left(\frac{n}{p+n}\right) \quad \text{... (2.7.3)}$$

Assume that using attribute A, a set S will be partitioned in to sets $\{s_1, s_2, ...s_k\}$

If S_i has p_i records of P and n_i records of N, the entropy or the expected information required to classify objects in all subtrees S_i is,

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} I(p_i, n_i) \quad \text{... (2.7.4)}$$

- **Entropy (E) :** Expected amount of information (in bits) needed to assign a class to a randomly drawn object in S under the optimal shortest length code.
- **Gain (A) :** Measures reduction in entropy achieved because of split. Choose split that achieves most reduction (maximum Gain).

$$\text{Gain}(A) = I(p, n) - E(A) \quad \text{...(2.7.5)}$$

4. Avoid Overfitting in classification (Tree pruning)

The generated tree may overfit the training data.

- If there are too many branches then some may reflect anomalies due to noise or outliers.
- Overfitting result in poor accuracy for unseen samples.

There are two approaches to avoid overfitting, prune the tree so that it is not too specific.

Prepruning (prune while building tree)

Stop tree construction early do not divide a node if this would result in the goodness measure falling below threshold.

Postpruning (prune after building tree)

Fully constructed tree get a sequence of progressively pruned trees.

5. Strengths of Decision Tree Method

- Able to generate understandable rules
- Performs classification without requiring much computation.

- Able to handle both continuous and categorical variables.
- Decision tree clearly indicates which fields are most important for prediction or classification.

6. Weakness of Decision Tree Method

- Not suitable for prediction of continuous attribute
- Perform poorly with many class and small data
- Computationally expensive to train.

2.8 CONSTRUCTING DECISION TREES

- The ID3 algorithm starts with the original set S as the root node. On each iteration of the algorithm, it iterates through every unused attribute of the set S and calculates the entropy (or information gain) of that attribute. Algorithm next selects the attribute which has the smallest entropy (or largest information gain) value.
- The set S is then divided by the chosen attribute (e.g. Income is less than 20 K , Income is between 20 K and 40 K, Income is greater than 40 K) to produce subsets of the data. The algorithm is recursively called for each subset, considering the attributes which are not selected before.
- The stopping criteria for recursion can be one of these situations :
  - When all records in the subset belongs to the same class (+ or –), then the node is converted into a leaf node and labelled with the class of the records.
  - When we have selected all the attributes, but the records still do not belong to the same class (some are + and some are –), then the node is converted into a leaf node and labelled with the most frequent class of the records in the subset
  - When there are no records in the subset, this is due to the non coverage of a specific attribute value for the record in the parent set, for example if there was no record with income = 40 K. Then a leaf node is generated, and labelled with the most frequent class of the record in the parent set.
- Decision tree is generated with each non-terminal node representing the selected attribute on which the data was split, and terminal nodes representing the class label of the final subset of this branch.

Summary

Entropy of each and every attribute is calculated using the data set

- Divide the set S into subsets using the attribute for which the resulting entropy (after splitting) is minimum (or, equivalently, information gain is maximum)
- Make a decision tree node containing that attribute
- Recurse on subsets using remaining attributes.

Pseudocode

```
ID3 (Records, Target_Attribute, Attributes)
Generate a root node for the tree
If all records are positive, Return the single-node tree Root, with '+ 'label.
If all records are negative, Return the single-node tree Root, with '-' label.
If number of predicting attributes is empty, then return the single node tree Root, and label with most frequent value
of the target attribute in the records.
Otherwise Begin
 A ← The Attribute that best classifies records.
 Decision Tree attribute for Root 'A'.
```

For each possible value, vi, of A,

Add a new tree branch below Root, corresponding to the test A = vi.

Let Record (vi) be the subset of records that have the value vi for A

If Record (vi) is empty

Then below this new branch add a leaf node and label with most frequent target value in the records

Else below this new branch add the sub tree ID3 (Records (vi), Target_Attribute,

Attributes – {A})

End

Return Root

2.9 EXAMPLE OF CLASSIFICATION TREE USING ID3

Ex. 2.9.1 : Suppose we want ID3 to evaluate car database as whether the car is acceptable or not. The target classification is "Should we accept car?" which can be acceptable or unacceptable.

Buying_Price	Maintenance_ Price	Lug_Boot	Safety	Evaluation?
High	High	Small	High	Unacceptable
High	High	Small	Low	Unacceptable
Medium	High	Small	High	Acceptable
Low	Medium	Small	High	Acceptable
Low	Low	Big	High	Acceptable
Low	Low	Big	Low	Unacceptable
Medium	Low	Big	Low	Acceptable
High	Medium	Small	High	Unacceptable
High	Low	Big	High	Acceptable
Low	Medium	Big	High	Acceptable
High	Medium	Big	Low	Acceptable
Medium	Medium	Small	Low	Acceptable
Medium	High	Big	High	Acceptable
Low	Medium	Small	Low	Unacceptable

Class P: Evaluation = "Acceptable"

Class N: Evaluation = "Unacceptable"

Total records = 14

No. of records with Acceptable = 9 and Unacceptable = 5

$$I(p, n) = -\left(\frac{p}{p+n}\right)\log_2\left(\frac{p}{p+n}\right) - \left(\frac{n}{p+n}\right)\log_2\left(\frac{n}{p+n}\right)$$

$$I(9, 5) = -\left(\frac{9}{14}\right)\log_2\left(\frac{9}{14}\right) - \left(\frac{5}{14}\right)\log_2 = 0.940$$

Step 1

1. Compute entropy for Buying_Price

For Buying_Price = High

$$p_i = 2 \text{ and } n_i = 3$$

$$I(p_i, n_i) = I(2, 3) = -\left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right) - \left(\frac{3}{5}\right)\log_2\left(\frac{3}{5}\right) = 0.971$$

Similarly we will calculate $I(p_i, n_i)$ for Medium and Low.

Buying_Price	p_i	n_i	$I(p_i, n_i)$
High	2	3	0.971
Medium	4	0	0
Low	3	2	0.971

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

$$E(\text{Buying_Price}) = \left(\frac{5}{14}\right) I(2,3) + \left(\frac{4}{14}\right) I(4,0) + \left(\frac{5}{14}\right) I(3,2) = 0.694$$

$$\text{Gain}(S, \text{Buying_Price}) = I(p, n) - E(\text{Buying_Price}) = 0.940 - 0.694 = 0.246$$

Similarly, Gain (S, Maintenance_ Price) = 0.029, Gain (S, Lug_Boot) = 0.151, Gain (S, Safety) = 0.048

Since Buying_Price is the highest we select Buying_Price as the root node.

Step 2

As attribute Buying_Price at root, we have to decide on remaining tree attribute for High branch.

Buying_Price	Maintenance_ Price	Lug_Boot	Safety	Evaluation?
High	High	Small	High	Unacceptable
High	High	Small	Low	Unacceptable
High	Medium	Small	High	Unacceptable
High	Low	Big	High	Acceptable
High	Medium	Big	Low	Acceptable

No. of records with Acceptable = 2 and Unacceptable = 3

$$I(p, n) = -\left(\frac{p}{p+n}\right)\log_2\left(\frac{p}{p+n}\right) - \left(\frac{n}{p+n}\right)\log_2\left(\frac{n}{p+n}\right)$$

$$I(2, 3) = -\left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right) - \left(\frac{3}{5}\right)\log_2\left(\frac{3}{5}\right) = 0.971$$

1. Compute entropy for Maintenance_ Price

Maintenance_ Price	p_i	n_i	$I(P_i, n_i)$
High	0	2	0
Medium	1	1	1
Low	1	0	0

$$E(\text{Maintenance_ Price}) = \left(\frac{2}{5}\right) I(0,2) + \left(\frac{2}{5}\right) I(1,1) + \left(\frac{1}{5}\right) I(1,0) = 0.4$$

$$\text{Gain}(S_{High}, \text{Maintenance_ Price}) = I(p, n) - E(\text{Maintenance_ Price}) = 0.971 - 0.4 = 0.571$$

2. Compute entropy for Lug_Boot

$P_i = 0$ and $n_i = 3$

$I(P_i, n_i) = I(0, 3) = 0$

Lug_Boot	p_i	n_i	$I(P_i, n_i)$
Small	0	3	0
Big	2	0	0

$$E(\text{Lug_Boot}) = \left(\frac{3}{5}\right) I(0, 3) + \left(\frac{2}{5}\right) I(2, 0) = 0$$

$$\text{Gain}(S_{High}, \text{Lug_Boot}) = I(p, n) - E(\text{Lug_Boot}) = 0.971 - 0 = 0.971$$

3. Compute entropy for Safety

$p_i = 1$ and $n_i = 2$

$I(p_i, n_i) = I(1, 2) = 0.918$

Safety	p_i	n_i	$I(P_i, n_i)$
High	1	2	0.918
Low	1	1	1

$$E(\text{Safety}) = \left(\frac{3}{5}\right) I(1, 2) + \left(\frac{2}{5}\right) I(1, 1) = 0.951$$

$$\text{Gain}(S_{High}, \text{Safety}) = I(p, n) - E(\text{Safety}) = 0.971 - 0.951 = 0.02$$

Since Lug_Boot is the highest we select Lug_Boot as a next node below High branch.

▶ **Step 3 :** Consider now only Maintenance_ Price and Safety for Buying_Price = Medium

Buying_Price	Maintenance_ Price	Lug_Boot	Safety	Evaluation?
Medium	High	Small	High	Acceptable
Medium	Low	Big	Low	Acceptable
Medium	Medium	Small	Low	Acceptable
Medium	High	Big	High	Acceptable

Since for any combination of values of aintenance_ Price and Safety, Evaluation? value is Acceptable, so we can directly write down the answer as Acceptable.

▶ **Step 4 :** Consider now only Maintenance_ Price and Safety for Buying_Price = Low

Buying_Price	Maintenance_ Price	Lug_Boot	Safety	Evaluation?
Low	Medium	Small	High	Acceptable
Low	Low	Big	High	Acceptable
Low	Low	Big	Low	Unacceptable
Low	Medium	Big	High	Acceptable
Low	Medium	Small	Low	Unacceptable

$$P_i = 3 \text{ and } n_i = 2$$

$$I(P_i, n_i) = I(3,2) = 0.970$$

1. Compute entropy for Safety

Safety	p_i	n_i	$I(P_i, n_i)$
High	3	0	0
Low	0	2	0

$$E(\text{Safety}) = \left(\frac{3}{5}\right) I(3, 0) + \left(\frac{2}{5}\right) I(0, 2) = 0$$

$$\text{Gain}(S_{Low}, \text{Safety}) = I(p, n) - E(\text{Safety}) = 0.970 - 0 = 0.970$$

2. Compute entropy for Maintenance_ Price

Maintenance_ Price	p_i	n_i	$I(P_i, n_i)$
High	0	0	0
Medium	2	1	0.918
Low	1	1	1

$$E(\text{Maintenance_ Price}) = \left(\frac{0}{5}\right) I(0, 0) + \left(\frac{3}{5}\right) I(2, 1) + \left(\frac{2}{5}\right) I(1, 1) = 0.951$$

$$\text{Gain}(S_{Low}, \text{Maintenance_ Price}) = I(p, n) - E(\text{Maintenance_ Price}) = 0.970 - 0.951 = 0.019$$

Since, Safety is the highest we select Safety below Low branch.

- Now we will check the value of 'Evaluation?' from the database, for all branches,

  Buying_Price = High and Lug_Boot = Small –> Evaluation? = Unacceptable

  Buying_Price = High and Lug_Boot = Big –> Evaluation? = Acceptable

  Buying_Price = Low and Safety = Low –> Evaluation? = Unacceptable

  Buying_Price = Low and Safety = High –> Evaluation? = Acceptable

- Final Decision Tree is

Ex. 2.9.2 : Suppose we want ID3 to decide whether the loan is to be sanctioned or not. The target classification is "Should we sanction loan?" which can be yes or no.

Customer no	Spending_ Habit	Collateral	Income	Credit_Score	Sanction?
1	High	None	Low	Bad	No
2	High	None	Medium	Unknown	No
3	Low	None	Medium	Unknown	No
4	Low	None	Low	Unknown	No
5	Low	None	High	Unknown	Yes
6	Low	Sufficient	High	Unknown	Yes
7	Low	None	Medium	Bad	No
8	Low	Sufficient	High	Bad	No
9	Low	None	High	Good	Yes
10	High	Sufficient	High	Good	Yes
11	High	None	Low	Good	No
12	High	None	Medium	Good	No

Class P : Sanction = "Yes"

Class N : Sanction = "No"

Total records = 12

No. of records with Yes = 4 and No = 8

$$I(p, n) = -\left(\frac{p}{p+n}\right)\log_2\left(\frac{p}{p+n}\right) - \left(\frac{n}{p+n}\right)\log_2\left(\frac{n}{p+n}\right)$$

$$I(4, 8) = -\left(\frac{4}{12}\right)\log_2\left(\frac{4}{12}\right) - \left(\frac{8}{12}\right) - \log_2\left(\frac{8}{12}\right) = 0.922$$

Step 1

1. Compute entropy for Spending_Habit

For Spending_Habit = High

$p_i = 1$ and $n_i = 4$

$$I(p_i, n_i) = I(1, 4) = -\left(\frac{1}{5}\right)\log_2\left(\frac{1}{5}\right) - \left(\frac{4}{5}\right)\log_2\left(\frac{4}{5}\right) = 0.721$$

Similarly, we will calculate $I(p_i, n_i)$ for Low.

Spending_Habit	p_i	n_i	$I(p_i, n_i)$
High	1	4	0.721
Low	3	4	0.985

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

$$E(\text{Spending_Habit}) = \left(\frac{5}{12}\right) \times 0721 + \left(\frac{7}{12}\right) \times 0.985 = 0.874$$

$$\text{Gain}(S, \text{Spending_Habit}) = I(p, n) - E(\text{Spending_Habit}) = 0.922 - 0.874 = 0.048$$

2. Compute entropy for Collateral

Collateral	p_i	n_i	$I(p_i, n_i)$
None	2	7	0.77
Sufficient	2	1	0.918

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

$$E(\text{Collateral}) = \left(\frac{9}{12}\right) \times 0.77 + \left(\frac{3}{12}\right) \times 0.918 = 0.806$$

$$\text{Gain}(S, \text{Collateral}) = I(p, n) - E(\text{Collateral}) = 0.922 - 0.806 = 0.116$$

3. Compute entropy for Income

Income	p_i	n_i	$I(p_i, n_i)$
Low	0	3	0
Medium	0	4	0
High	4	1	0.721

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

$$E(\text{Income}) = \left(\frac{9}{12}\right) \times 0 + \left(\frac{4}{12}\right) \times 0 + \left(\frac{5}{12}\right) \times 0.721 = 0.3$$

$$\text{Gain}(S, \text{Income}) = I(p, n) - E(\text{Income}) = 0.922 - 0.3 = 0.622$$

4. Compute entropy for Credit_Score

Credit_Score	p_i	n_i	$I(p_i, n_i)$
Bad	0	3	0
Unknown	2	3	0.97
Good	2	2	1

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

$$E\,(\text{Credit_Score}) = \left(\frac{3}{12}\right)\times 0 + \left(\frac{5}{12}\right)\times 0.97 + \left(\frac{4}{12}\right)\times 1 = 0.734$$

$$\text{Gain}(S, \text{Credit_Score}) = I\,(p, n) - E\,(\text{Credit_Score})$$

$$= 0.922 - 0.734 = 0.188$$

Since Income is the highest we select Income as the root node.

Step 2

As attribute Income at root, we have to decide on remaining tree attribute for Low branch.

Consider only Spending_Habit, Collateral and Credit_Score for Income = Low

Customer no	Spending_ Habit	Collateral	Income	Credit_Score	Sanction?
1	High	None	Low	Bad	No
4	Low	None	Low	Unknown	No
11	High	None	Low	Good	No

Since for any combination of values of Spending_Habit, Collateral and Credit_Score, Sanction? value is No for Income = Low, so we can directly write down the answer as No

Step 3

Consider only Spending_Habit, Collateral and Credit_Score Income = Medium

Customer no	Spending_ Habit	Collateral	Income	Credit_Score	Sanction?
2	High	None	Medium	Unknown	No
3	Low	None	Medium	Unknown	No
7	Low	None	Medium	Bad	No
12	High	None	Medium	Good	No

Since for any combination of values of Spending_Habit, Collateral and Credit_Score, Sanction? value is No for Income = Medium, so we can directly write down the answer as No

Step 4

Consider only Spending_Habit, Collateral and Credit_Score Income = High

Customer no	Spending_ Habit	Collateral	Income	Credit_Score	Sanction?
5	Low	None	High	Unknown	Yes
6	Low	Sufficient	High	Unknown	Yes
8	Low	Sufficient	High	Bad	No
9	Low	None	High	Good	Yes
10	High	Sufficient	High	Good	Yes

No. of records with Yes = 4 and No = 1

$$I\,(p, n) = -\left(\frac{p}{p+n}\right)\log_2\left(\frac{p}{p+n}\right) - \left(\frac{n}{p+n}\right)\log_2\left(\frac{n}{p+n}\right)$$

$$I\,(2, 3) = -\left(\frac{4}{5}\right)\log_2\left(\frac{4}{5}\right) - \left(\frac{1}{5}\right)\log_2\left(\frac{1}{5}\right) = 0.721$$

1. Compute entropy for Spending_Habit

Spending_Habit	p_i	n_i	$I(p_i, n_i)$
Low	3	1	0.811
High	1	0	0

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

$$E(\text{Spending_Habit}) = \left(\frac{4}{5}\right) \times 0811 + \left(\frac{1}{5}\right) \times 0 = 0.648$$

Gain (S_{High}, Spending_Habit) = I (p, n) – E (Spending_Habit) = 0.721 – 0.648 = 0.073

2. Compute entropy for Collateral

Collateral	p_i	n_i	$I(p_i, n_i)$
None	2	0	0
Sufficient	2	1	0.918

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

$$E(\text{Collateral}) = \left(\frac{2}{5}\right) \times 0 + \left(\frac{3}{5}\right) \times 0.918 = 0.55$$

Gain (S_{High}, Collateral) = I (p, n) – E (Collateral) = 0.721 – 0.55 = 0.171

3. Compute entropy for Credit_Score

Credit_Score	p_i	n_i	$I(p_i, n_i)$
Unknown	1	0	0
Bad	0	1	0
Good	2	0	0

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

$$E(\text{Credit_Score}) = \left(\frac{1}{5}\right) \times 0 + \left(\frac{1}{5}\right) \times 0 + \left(\frac{2}{5}\right) \times 0 = 0$$

Gain (S_{High}, Credit_Score) = I (p, n) – E (Credit_Score)

= 0.721 – 0 = 0.721

- Since Credit_Score is the highest we select Credit_Score as a next node below High branch.
- Now we will check the value of 'Sanction?' from the database, for all branches,

  Income = High and Credit_Score = Bad -> Sanction? = No

  Income = High and Credit_Score = Unknown -> Sanction? = Yes

  Income = High and Credit_Score = Good -> Sanction? = Yes

- Final Decision Tree is

Ex. 2.9.3 : Suppose we want ID3 to decide whether the car will be stolen or not. The target classification is "car is stolen?" which can be Yes or No.

Car no	Colour	Type	Origin	Stolen?
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports	Imported	Yes
6	Yellow	SUV	Imported	No
7	Yellow	SUV	Imported	Yes
8	Yellow	SUV	Domestic	No
9	Red	SUV	Imported	No
10	Red	Sports	Imported	Yes

Class P: Stolen = "Yes"

Class N: Stolen = "No"

Total records = 10

No. of records with Yes = 5 and No = 5

$$I(p, n) = \left(\frac{p}{p+n}\right)\log_2\left(\frac{p}{p+n}\right) - \left(\frac{n}{p+n}\right)\log_2\left(\frac{n}{p+n}\right)$$

$$I(5, 5) = -\left(\frac{5}{10}\right)\log_2\left(\frac{5}{10}\right) - \left(\frac{5}{10}\right)\log_2\left(\frac{5}{10}\right) = 1$$

Soln. :

Step 1

1. Compute entropy for Colour

Colour	p_i	n_i	$I(p_i, n_i)$
Red	3	2	0.971
Yellow	2	3	0.971

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} I(p_i, n_i)$$

$$E(\text{Colour}) = \left(\frac{5}{10}\right) \times 0.971 + \left(\frac{5}{10}\right) \times 0.971 = 0.971$$

$$\text{Gain}(S, \text{Colour}) = I(p, n) - E(\text{Colour}) = 1 - 0.971 = 0.029$$

2. Compute entropy for Type

Type	p_i	n_i	$I(p_i, n_i)$
Sports	4	2	0.923
SUV	1	3	0.811

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

$$E(\text{Type}) = \left(\frac{6}{10}\right) \times 0.923 + \left(\frac{4}{10}\right) \times 0.811 = 0.878$$

$$\text{Gain}(S, \text{Type}) = I(p, n) - E(\text{Type}) = 1 - 0.878 = 0.1218$$

3. Compute entropy for Origin

Origin	p_i	n_i	$I(p_i, n_i)$
Domestic	2	3	0.971
Imported	3	2	0.971

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

$$E(\text{Origin}) = \left(\frac{5}{10}\right) \times 0.971 + \left(\frac{5}{10}\right) \times 0.971 = 0.971$$

$$\text{Gain}(S, \text{Origin}) = I(p, n) - E(\text{Origin}) = 1 - 0.971 = 0.029$$

Since Type is the highest we select Type as the root node.

Step 2

As attribute Type at root, we have to decide on remaining tree attribute for Sports branch.

Consider only Colour and Origin for Type = Sports

Car no	Colour	Type	Origin	Stolen?
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports	Imported	Yes
10	Red	Sports	Imported	Yes

No. Of records with Yes = 4 and No = 2

$$I(p, n) = \left(\frac{p}{p+n}\right) \log_2\left(\frac{p}{p+n}\right) - \left(\frac{n}{p+n}\right) \log_2\left(\frac{n}{p+n}\right)$$

$$I(4, 2) = -\left(\frac{4}{6}\right) \log_2\left(\frac{4}{6}\right) - \left(\frac{2}{6}\right) \log_2\left(\frac{2}{6}\right) = 0.923$$

1. Compute entropy for Colour

Colour	p_i	n_i	$I(p_i, n_i)$
Red	3	1	0.811
Yellow	1	1	1

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

$$E(\text{Colour}) = \left(\frac{4}{6}\right) \times 0.811 + \left(\frac{2}{6}\right) \times 1 = 0.873$$

$$\text{Gain}(S_{\text{Sports}}, \text{Colour}) = I(p, n) - E(\text{Colour}) = 0.923 - 0.873 = 0.05$$

2. Compute entropy for Origin

Origin	p_i	n_i	$I(p_i, n_i)$
Domestic	2	2	1
Imported	2	0	0

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

$$E(\text{Origin}) = \left(\frac{4}{6}\right) \times 1 + \left(\frac{2}{6}\right) \times 0 = 0.666$$

$$\text{Gain}(S_{\text{Sports}}, \text{Origin}) = I(p, n) - E(\text{Origin}) = 0.923 - 0.666 = 0.257$$

Since Origin is the highest we select as a next node below Sports branch.

Step 3

- As attribute Type and Origin is already chosen, we have to decide on only remaining Colour attribute for SUV branch.
- Now we will check the value of 'Stolen?' from the database, for all branches,
  - For, Type = Sports and Origin = Domestic, Stolen? = Yes as well as No
- So for this type of case we have to select the most common class. In this example there are 2 instances for Yes as well as No, so we can select any one. Let's we select No.
  - For, Type = Sports and Origin = Imported, Stolen? = Yes
  - For, Type = SUV and Colour = Red, Stolen? = No

    For, Type = SUV and Colour = Yellow, Stolen? = Yes as well as No
- So for this type of case we have to select the most common class. In this example there are 2 instances for No and 1 instance of Yes, so we will select No.

- Final Decision Tree is

2.10 EXAMPLE OF DECISION TREE USING GINI INDEX

Ex. 2.10.1 : Create a decision tree using Gini Index to classify following dataset.

Sr. No.	Income	Age	Own Car
1	Very High	Young	Yes
2	High	Medium	Yes
3	Low	Young	No
4	High	Medium	Yes
5	Very High	Medium	Yes
6	Medium	Young	Yes
7	High	Old	Yes
8	Medium	Medium	No
9	Low	Medium	No
10	Low	Old	No
11	High	Young	Yes
12	Medium	Old	No

Soln. :

- In this example there are two classes Yes and No.

  No. of records for Yes = 7

  No. of records for No = 5

  Total No. of records = 12

- Now we will calculate Gini of the complete database as,

$$\text{Gini (T)} = 1-\left(\left(\frac{7}{12}\right)^2+\left(\frac{5}{12}\right)^2\right)=0.48$$

- Next we will calculate Split for all attributes, i.e. Income and Age.

Income

$$\text{Split} = \frac{2}{12}\text{gini (Very High)} + \frac{4}{12}\text{gini (High)} + \frac{3}{12}\text{gini (Low)} + \frac{3}{12}\text{gini (Medium)}$$

$$= \frac{2}{12}\left[1-\left(\left(\frac{2}{2}\right)^2+\left(\frac{0}{2}\right)^2\right)\right]+\frac{4}{12}\left[1-\left(\left(\frac{4}{4}\right)^2+\left(\frac{0}{4}\right)^2\right)\right]+\frac{3}{12}\left[1-\left(\left(\frac{0}{3}\right)^2+\left(\frac{3}{3}\right)^2\right)\right]$$

$$+\frac{3}{12}\left[1-\left(\left(\frac{1}{3}\right)^2+\left(\frac{2}{3}\right)^2\right)\right]$$

$$= 0.1125$$

$$\text{Age Split} = \frac{4}{12}\text{ gini (Yong)} + \frac{5}{12}\text{ gini (Medium)} + \frac{3}{12}\text{ gini (Old)}$$

$$= \frac{4}{12}\left[1-\left(\left(\frac{3}{4}\right)^2+\left(\frac{1}{4}\right)^2\right)\right]+\frac{5}{12}\left[1-\left(\left(\frac{3}{5}\right)^2+\left(\frac{2}{5}\right)^2\right)\right]+\frac{3}{12}\left[1-\left(\left(\frac{1}{3}\right)^2+\left(\frac{2}{3}\right)^2\right)\right]$$

$$= 0.4375$$

- Split value of Income is smallest, so we will select Income as root node.

- From the database we can see that,

  Own Car = Yes for Income = Very High, so we can directly write down 'Yes' for Very High branch.

  Own Car = Yes for Income = High, so we can directly write down 'Yes' for High branch.

  Own Car = No for Income = Low, so we can directly write down 'No' for Low branch.

- Since Income is taken as root node, now we have to decide on the Age attribute, so we will take Age as next node below Medium branch.
- From the database we can see that,

  Own Car = Yes for Income = Medium and Age = Young, so we can directly write down 'Yes' for Young branch.

  Own Car = No for Income = Medium and Age = Medium, so we can directly write down 'No' for medium branch.

  Own Car = No for Income = Medium and Age = Old, so we can directly write down 'No' for Old branch.
- Final Decision Tree is,

Ex. 2.10.2 : Stock market involving only Discrete ranges has profit as categorical value {up, down}.Use Gini index method to draw classification tree.

Age	Competition	Type	Profit
old	Yes	software	down
old	No	software	down
old	No	hardware	down

Age	Competition	Type	Profit
mid	Yes	software	down
mid	Yes	hardware	down
mid	No	hardware	up
mid	No	software	up
new	Yes	software	up
new	No	hardware	up
new	no	software	up

☑ **Soln. :**

- In this example there are two classes down and up.

  No. of records for down = 5

  No. of records for up = 5

  Total No. of records = 10

- Now we will calculate Gini of the complete database as,

$$\text{Gini (T)} = \left[1-\left(\left(\frac{5}{10}\right)^2+\left(\frac{5}{10}\right)^2\right)\right]=0.5$$

- Next we will calculate Split for all attributes, i.e. Age, Competition and Type.

Age

$$\text{Split} = \frac{3}{10}\text{gini (old)}+\frac{4}{10}\text{gini (mid)}\frac{3}{10}\text{ gini (new)}$$

$$= \frac{3}{10}\left[1-\left(\left(\frac{0}{3}\right)^2+\left(\frac{3}{3}\right)^2\right)\right]+\frac{4}{10}\left[1-\left(\left(\frac{2}{4}\right)^2+\left(\frac{2}{4}\right)^2\right)\right]+\frac{3}{10}\left[1-\left(\left(\frac{3}{3}\right)^2+\left(\frac{0}{3}\right)^2\right)\right]=0.2$$

Competition

$$\text{Split} = \frac{4}{10}\text{gini (yes)}+\frac{6}{10}\text{gini (no)}$$

$$= \frac{4}{10}\left[1-\left(\left(\frac{1}{4}\right)^2+\left(\frac{3}{4}\right)^2\right)\right]+\frac{6}{10}\left[1-\left(\left(\frac{4}{6}\right)^2+\left(\frac{2}{6}\right)^2\right)\right]=0.42$$

Type

$$\text{Split} = \frac{6}{10}\text{gini (software)}+\frac{4}{10}\text{gini (hardware)}$$

$$= \frac{6}{10}\left[1-\left(\left(\frac{3}{6}\right)^2+\left(\frac{3}{6}\right)^2\right)\right]+\frac{4}{10}\left[1-\left(\left(\frac{2}{4}\right)^2+\left(\frac{2}{4}\right)^2\right)\right]=0.5$$

- Split value of Age is smallest, so we will select Age as root node.
- From the database we can see that, Profit = Down for Age = Old, so we can directly write down 'Down' for Old branch node.
- Next we will check for Age = mid

Age	Competition	Type	Profit
mid	Yes	Software	Down
mid	Yes	Hardware	Down
mid	No	Hardware	Up
mid	No	Software	Up

- Now we will calculate Split for only Competition and Type attributes.

Competition

$$\text{Split} = \frac{2}{4}\text{gini (yes)} + \frac{2}{4}\text{gini (no)}$$

$$= \frac{2}{4}\left[1-\left(\left(\frac{0}{2}\right)^2+\left(\frac{2}{2}\right)^2\right)\right]+\frac{2}{4}\left[1-\left(\left(\frac{2}{2}\right)^2+\left(\frac{0}{2}\right)^2\right)\right]=0$$

Type

$$\text{Split} = \frac{2}{4}\text{gini (software)} + \frac{2}{4}\text{gini (hardware)}$$

$$= \frac{2}{4}\left[1-\left(\left(\frac{1}{2}\right)^2+\left(\frac{1}{2}\right)^2\right)\right]+\frac{2}{4}\left[1-\left(\left(\frac{1}{2}\right)^2+\left(\frac{1}{2}\right)^2\right)\right]=0.5$$

- Split value of Competition is smallest, so we will select Competition as next node below mid branch.
- From the database we can see that, Profit = Up for Age = New, so we can directly write down 'Up' for New branch node.
- Now we will check the value of 'Profit' from the database, for all branches,
  - Age = Mid and Competition = No -> Profit = Up
  - Age = Mid and Competition = Yes -> Profit = Down
- Final Decision Tree is,

Ex. 2.10.3 : Suppose we want Gini index to decide whether the car will be stolen or not. The target classification is "car is stolen?" which can be Yes or No.

Car no	Colour	Type	Origin	Stolen?
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No

Car no	Colour	Type	Origin	Stolen?
5	Yellow	Sports	Imported	Yes
6	Yellow	SUV	Imported	No
7	Yellow	SUV	Imported	Yes
8	Yellow	SUV	Domestic	No
9	Red	SUV	Imported	No
10	Red	Sports	Imported	Yes

☑ **Soln. :**

- In this example there are two classes Yes and No.

  No. of records for Yes = 5

  No. of records for No = 5

  Total No. of records = 10

- Now we will calculate Gini of the complete database as,

$$\textbf{Gini (T)} = 1-\left(1-\left(\left(\frac{5}{10}\right)^2+\left(\frac{5}{10}\right)^2\right)\right)=\mathbf{0.5}$$

- Next we will calculate Split for all attributes, i.e. Colour, Type and Origin.

Colour

$$\text{Split} = \frac{2}{4}=\frac{5}{10}\left[1-\left(\left(\frac{3}{5}\right)^2+\left(\frac{2}{5}\right)^2\right)\right]+\frac{5}{10}\left[1-\left(\left(\frac{2}{5}\right)^2+\left(\frac{3}{5}\right)^2\right)\right]=\mathbf{0.48}$$

Type

$$\text{Split} = \frac{6}{10}\text{gini (Sports)}+\frac{4}{10}\text{ gini (SUV)}=\frac{6}{10}\left[1-\left(\left(\frac{4}{6}\right)^2+\left(\frac{2}{6}\right)^2\right)\right]+\frac{4}{10}\left[1-\left(\left(\frac{1}{4}\right)^2+\left(\frac{3}{4}\right)^2\right)\right]=\mathbf{0.42}$$

Origin

$$\text{Split} = \frac{5}{10}\text{ gini (Domestic)}+\frac{5}{10}\text{gini (imported)}$$

$$=\frac{5}{10}\left[1-\left(\left(\frac{2}{5}\right)^2+\left(\frac{3}{5}\right)^2\right)\right]+\frac{5}{10}\left[1-\left(\left(\frac{3}{5}\right)^2+\left(\frac{2}{5}\right)^2\right)\right]=\mathbf{0.48}$$

- Split value of Type is smallest, so we will select Type as root node.
- Next we will check for Type = Sports
- As attribute Type at root, we have to decide on remaining tree attribute for Sports branch.
- Consider only Colour and Origin for Type = Sports

Car no	Colour	Type	Origin	Stolen?
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports	Imported	Yes
10	Red	Sports	Imported	Yes

Colour

$$\text{Split} = \frac{4}{6}\text{gini (Red)} + \frac{2}{6}\text{gini (Yellow)}$$

$$= \frac{4}{6}\left[1-\left(\left(\frac{3}{4}\right)^2+\left(\frac{1}{4}\right)^2\right)\right]+\frac{2}{6}\left[1-\left(\left(\frac{1}{2}\right)^2+\left(\frac{1}{2}\right)^2\right)\right]=0.417$$

Origin>

$$\text{Split} = \frac{4}{6}\text{gini (Domestic)} + \frac{4}{6}\text{gini (Imported)}$$

$$= \frac{4}{6}\left[1-\left(\left(\frac{2}{4}\right)^2+\left(\frac{2}{4}\right)^2\right)\right]+\frac{2}{6}\left[1-\left(\left(\frac{2}{2}\right)^2+\left(\frac{0}{2}\right)^2\right)\right]=0.33$$

- Split value of Origin is smallest, so we will select Origin as next node.
- Next we will check for Type = SUV
- As attribute Type and Origin is already chosen, we have to decide on only remaining Colour attribute for SUV branch.
- Now we will check the value of 'Stolen?' from the database, for all branches,

  For, Type = Sports and Origin = Domestic, Stolen? = Yes as well as No
- So for this type of case we have to select the most common class. In this example there are 2 instances for Yes as well as No, so we can select any one. Let's we select No.
  - For, Type = Sports and Origin = Imported, Stolen? = Yes
  - For, Type = SUV and Colour = Red, Stolen? = No
  - For, Type = SUV and Colour = Yellow, Stolen? = Yes as well as No

- So for this type of case we have to select the most common class (Since all attributes are already considered). In this example there are 2 instances for No and 1 instance of Yes, so we will select No.
- Final Decision Tree is

UEx. 2.10.4 **MU - May 15. 12 Marks**

Create a decision tree for the attribute "class" using the respective values :

Eyecolour	Married	Sex	Hairlength	Class
Brown	yes	male	Long	Football
Blue	yes	male	Short	Football
Brown	yes	male	Long	Football
Brown	no	female	Long	Netball
Brown	no	female	Long	Netball
Blue	no	male	Long	Football
Brown	no	female	Long	Netball
Brown	no	male	Short	Football
Brown	yes	female	Short	Netball
Brown	no	female	Long	Netball
Blue	no	male	Long	Football
Blue	no	male	Short	Football

☑ **Soln. :** In this example there are two classes Football and Netball.

No. of records for Football = 7

No. of records for Netball = 5

Total No. Of records = 12

Now we will calculate Gini of the complete database as,

$$\text{Gini (T)} = 1-\left(\left(\frac{7}{12}\right)^2+\left(\frac{5}{12}\right)^2\right)=0.48$$

Next we will calculate Split for all attributes, i.e. Eyecolor, Married, Sex and Hairlength.

Eyecolor->

$$\text{Split} = \frac{8}{12}\text{gini (Brown)}+\frac{4}{12}\text{gini (Blue)}=\frac{8}{12}\left[1-\left(\left(\frac{3}{8}\right)^2+\left(\frac{5}{8}\right)^2\right)\right]+\frac{4}{12}\left[1-\left(\left(\frac{4}{4}\right)^2+\left(\frac{0}{4}\right)^2\right)\right]=0.31$$

Married->

$$\text{Split} = \frac{4}{12}\text{gini (yes)}+\frac{8}{12}\text{gini (no)}=\frac{4}{12}\left[1-\left(\left(\frac{3}{4}\right)^2+\left(\frac{1}{4}\right)^2\right)\right]+\frac{8}{12}\left[1-\left(\left(\frac{4}{8}\right)^2+\left(\frac{4}{8}\right)^2\right)\right]=0.458$$

Sex->

$$\text{Split} = \frac{7}{12}\text{gini (male)}+\frac{5}{12}\text{gini (female)}=\frac{7}{12}\left[1-\left(\left(\frac{7}{7}\right)^2+\left(\frac{0}{7}\right)^2\right)\right]+\frac{5}{12}\left[1-\left(\left(\frac{0}{5}\right)^2+\left(\frac{5}{5}\right)^2\right)\right]=0$$

Hairlength->

$$\text{Split} = \frac{8}{12}\text{gini (long)}+\frac{4}{12}\text{gini (short)}=\frac{8}{12}\left[1-\left(\left(\frac{4}{8}\right)^2+\left(\frac{4}{8}\right)^2\right)\right]+\frac{4}{12}\left[1-\left(\left(\frac{3}{4}\right)^2+\left(\frac{1}{4}\right)^2\right)\right]=0.458$$

Split value of Sex is smallest, so we will select Sex as root node.

From the database we can see that,

class = Football for Sex = male, so we can directly write down 'Football' for male branch.

class = Netball for Sex = female, so we can directly write down 'Netball' for female branch.

Final decision tree is,

UEx. 2.10.5 **MU - May 17, 10 Marks**

For a Sunburn dataset given below, construct a decision tree

Name	Hair	Height	Weight	Location	Class
Swati	Blonde	Average	Light	No	Yes
Sunita	Blonde	Tall	Average	Yes	No
Anita	Brown	Short	Average	Yes	No
Lata	Blonde	Short	Average	No	Yes
Radha	Red	Average	Heavy	No	Yes
Maya	Brown	Tall	Heavy	No	No
Leena	Brown	Average	Heavy	No	No
Rina	Blonde	Short	Light	Yes	No

☑ **Soln. :**

We will calculate Split for all attributes, i.e. Hair, Height, Weight and Location.

Hair->

$$\text{Split} = \frac{4}{8}\text{gini (Blonde)} + \frac{3}{8}\text{gini (Brown)} + \frac{1}{8}\text{gini (Red)}$$

$$= \frac{4}{8}\left[1-\left(\left(\frac{2}{4}\right)^2+\left(\frac{2}{4}\right)^2\right)\right]+\frac{3}{8}\left[1-\left(\left(\frac{0}{3}\right)^2+\left(\frac{3}{3}\right)^2\right)\right]+\frac{1}{8}\left[1-\left(\left(\frac{1}{1}\right)^2+\left(\frac{0}{1}\right)^2\right)\right]=0.25$$

Height->

$$\text{Split} = \frac{3}{8}\text{gini (Average)} + \frac{2}{8}\text{gini (Tall)} + \frac{3}{8}\text{gini (Short)}$$

$$= \frac{3}{8}\left[1-\left(\left(\frac{2}{3}\right)^2+\left(\frac{1}{3}\right)^2\right)\right]+\frac{2}{8}\left[1-\left(\left(\frac{0}{2}\right)^2+\left(\frac{2}{2}\right)^2\right)\right]+\frac{3}{12}\left[1-\left(\left(\frac{1}{3}\right)^2+\left(\frac{2}{3}\right)^2\right)\right]=0.40$$

Weight->

$$\text{Split} = \frac{2}{8}\text{gini (Light)} + \frac{3}{8}\text{gini (Average)} + \frac{3}{8}\text{gini (Heavy)}$$

$$= \frac{2}{8}\left[1-\left(\left(\frac{1}{2}\right)^2+\left(\frac{1}{2}\right)^2\right)\right]+\frac{3}{8}\left[1-\left(\left(\frac{1}{3}\right)^2+\left(\frac{2}{3}\right)^2\right)\right]+\frac{3}{8}\left[1-\left(\left(\frac{1}{3}\right)^2+\left(\frac{2}{3}\right)^2\right)\right]=0.525$$

Location->

$$\text{Split} = \frac{5}{8}\text{gini (No)} + \frac{3}{8}\text{gini (Yes)}$$

$$= \frac{5}{8}\left[1-\left(\left(\frac{3}{5}\right)^2+\left(\frac{2}{5}\right)^2\right)\right]+\frac{3}{8}\left[1-\left(\left(\frac{0}{3}\right)^2+\left(\frac{3}{3}\right)^2\right)\right]=0.3$$

Split value of Hair is smallest, so we will select Hair as root node.

Now we will split the remaining attributes considering Blonde data

Height->

$$\text{Split} = \frac{1}{4}\text{gini (Average)} + \frac{1}{4}\text{gini (Tall)} + \frac{2}{4}\text{gini (Short)}$$

$$= \frac{1}{4}\left[1-\left(\left(\frac{1}{1}\right)^2+\left(\frac{0}{1}\right)^2\right)\right]+\frac{1}{4}\left[1-\left(\left(\frac{0}{1}\right)^2+\left(\frac{1}{1}\right)^2\right)\right]+\frac{2}{4}\left[1-\left(\left(\frac{1}{2}\right)^2+\left(\frac{1}{2}\right)^2\right)\right]=0.25$$

Weight->

$$\text{Split} = \frac{2}{4}\text{gini (Light)} + \frac{2}{4}\text{gini (Average)}$$

$$= \frac{2}{4}\left[1-\left(\left(\frac{1}{2}\right)^2+\left(\frac{1}{2}\right)^2\right)\right]+\frac{2}{4}\left[1-\left(\left(\frac{1}{2}\right)^2+\left(\frac{1}{2}\right)^2\right)\right]=0.5$$

Location->

$$\text{Split} = \frac{2}{4}\text{gini (No)} + \frac{2}{4}\text{gini (Yes)}$$

$$= \frac{2}{4}\left[1-\left(\left(\frac{2}{2}\right)^2+\left(\frac{0}{2}\right)^2\right)\right]+\frac{2}{4}\left[1-\left(\left(\frac{0}{2}\right)^2+\left(\frac{2}{2}\right)^2\right)\right]=0$$

Split value of Location is smallest, so we will select Location node below Blonde branch.

From the database we can see that,

class = Yes for Hair = Blonde and Location = No , so we can directly write down 'Yes' for No branch.

class = No for Hair = Blonde and Location= Yes, so we can directly write down 'No' for Yes branch.

class = Yes for Hair = Red , so we can directly write down 'Yes' for Red branch.

class = No for Hair = Brown, so we can directly write down 'No' for Hair branch.

Final decision tree is,

UEx. 2.10.6 **MU - May 19, 10 Marks**

For a dataset given below, construct a decision tree For the following data, Calculate Gini indexes and determines which attribute is root attribute and generate two level deep decision tree.

Sr. No.	Income	Defaulting	Credit score	Location	Give Loan?
1	Low	High	High	bad	No
2	Low	High	High	good	No
3	High	High	High	bad	Yes
4	Medium	Medium	High	bad	Yes
5	Medium	Low	Low	bad	No
6	Medium	Low	Low	good	Yes
7	High	Low	Low	good	Yes
8	Low	Medium	High	bad	No
9	Low	Low	Low	bad	No
10	Medium	Medium	Low	bad	No
11	Low	Medium	Low	good	Yes
12	High	Medium	High	good	Yes
13	High	High	Low	bad	No
14	Medium	Medium	High	good	Yes

☑ **Soln. :** We will calculate Split for all attributes, i.e. Income, Defaulting, Creditscore and Location.

Income->

$$\text{Split} = \frac{5}{14}\text{gini (Low)} + \frac{4}{14}\text{gini (High)} + \frac{5}{14}\text{gini (Medium)}$$

$$= \frac{5}{14}\left[1-\left(\left(\frac{1}{5}\right)^2+\left(\frac{4}{5}\right)^2\right)\right]+\frac{4}{14}\left[1-\left(\left(\frac{3}{4}\right)^2+\left(\frac{1}{4}\right)^2\right)\right]+\frac{5}{14}\left[1-\left(\left(\frac{3}{5}\right)^2+\left(\frac{2}{5}\right)^2\right)\right]=0.392$$

Defaulting->

$$\text{Split} = \frac{4}{14}\text{gini (High)} + \frac{6}{14}\text{gini (Medium)} + \frac{4}{14}\text{gini (Low)} = 0.438$$

Creditscore->

$$\text{Split} = \frac{7}{14}\text{gini (High)} + \frac{7}{14}\text{gini (Low)} = 0.493$$

Location->

$$\text{Split} = \frac{8}{14}\text{gini (bad)} + \frac{6}{14}\text{gini (good)} = \frac{5}{8}\left[1-\left(\left(\frac{3}{5}\right)^2+\left(\frac{2}{5}\right)^2\right)\right]+\frac{3}{8}\left[1-\left(\left(\frac{0}{3}\right)^2+\left(\frac{3}{3}\right)^2\right)\right]=0.336$$

Split value of Location is smallest, so we will select Location as root node.

Now we will split the bad branch considering remaining attributes

Income->

$$\text{Split} = \frac{3}{8}\text{gini (Low)} + \frac{2}{8}\text{gini (High)} + \frac{3}{8}\text{gini (Medium)} = 0.295$$

Defaulting->

$$\text{Split} = \frac{3}{8}\text{gini (High)} + \frac{3}{8}\text{gini (Medium)} + \frac{2}{8}\text{gini (Low)} = 0.34$$

Creditscore->

$$\text{Split} = \frac{4}{8}\text{gini (High)} + \frac{4}{8}\text{gini (Low)} = 0.25$$

Split value of Creditscore is smallest, so we will select Creditscore node below bad branch.

Now we will split the good branch considering remaining attributes

Income->

$$\text{Split} = \frac{2}{6}\text{gini (Low)} + \frac{2}{6}\text{gini (High)} + \frac{2}{6}\text{gini (Medium)} = 0.295$$

Defaulting->

$$\text{Split} = \frac{1}{6}\text{gini (High)} + \frac{2}{6}\text{gini (Medium)} + \frac{3}{6}\text{gini (Low)} = 0$$

Split value of Defaulting is smallest, so we will select Defaulting node below good branch

Since only one attribute is remaining, we can directly select Income below creditscore = High branch

For Location = bad and creditscore = High and Income = Low,

Giveloan = No

For Location = bad and creditscore = High and Income = Medium,

Giveloan = Yes

For Location = bad and creditscore = High and Income = High,

Giveloan = Yes

For Location = bad and creditscore = Low, Giveloan = No

For Location = good and Defaulting = High , Giveloan = No

For Location = good and Defaulting = Low , Giveloan = Yes

For Location = good and Defaulting = Medium ,

Giveloan = yes

2.11 CLASSIFICATION AND REGRESSION TREE (CART)

- Classification trees are used to divide the dataset into classes belonging to the target variable. Mainly the target variable has two classes that can be yes or no. When the target variable type is categorical classification trees are used.
- In certain applications the target variable is numeric or continuous in that case regression trees are used. Let's take an example of prediction of price of a flat. Hence regression trees are used for problems or tasks where we want to predict some data instead of classifying the data.
- Based on the similarity of the data the records are classified in a standard classification tree. Let's take an example of an Income tax evades. In this example we have two variables, Income and marital status that predict if a person is going to evade the income tax or not. In our training data if it showed that 85% of people who are married does not evade the income tax, we split the data here and Marital status becomes a root node in tree. Entropy or Ginny index is used in classification trees.
- The main basic working of regression tree is to fit a model. The target or response variable does not have classes so a regression model is fit using each independent variable to the target variable. Then the data is split at various split points for each independent variable. At each split point sum of squared errors (SSE) is calculated by taking the square of the difference between predicted and actual value. The criteria for root node is to select the node which is having minimum SSE among all split point errors. The further tree is built using the recursive procedure.

2.12 EXAMPLEA OF REGRESSION TREE

Example 1

Buying_Price	Lug_Boot	Safety	Maintenance_ Price? (in thousand)
Low	Small	High	25
Low	Small	Low	30
Medium	Small	High	46
High	Small	High	45
High	Big	High	52
High	Big	Low	23
Medium	Big	Low	43
Low	Small	High	35
Low	Big	High	38
High	Big	High	46
Low	Big	Low	48
Medium	Small	Low	52
Medium	Big	High	44
High	Small	Low	30

Standard deviation

A decision tree is built up top down from root node and involved partitioning the data into subsets that contain instances with similar values. We use SD to calculate homogeneity of a numerical sample.

$$SD, S = \sqrt{\frac{\sum (x-\mu)^2}{n}} = 9.32$$

SD Reduction

It is based on the decrease in SD after a dataset is split on an attribute. Constructing a tree is all about finding attribute that returns highest SDR.

- **Step 1 :** SD (Maintenance_Price?) = 9.32
- **Step 2 :** The dataset is then split on the different attribute.SD for each branch is calculated. The resulting SD is subtracted from SD before split.

		Maintenance_ Price(SD)
Buying_Price	Low	7.78
	Medium	3.49
	High	10.87

SD (Maintenance_Price, Buying_Price) = P (Low) SD (Low) + P (Medium) SD (Medium) + P (High) SD (High)

$= \frac{5}{14} \times 7.78 + \frac{4}{14} \times 3.49 + \frac{5}{14} \times 10.87 = 7.66$

SDR = SD (Maintenance_ Price) – SD (Maintenance_ Price, Buying_Price) = 9.32 – 7.66 = 1.66

		Maintenance_ Price (SD)
Lug_Boot	Small	9.36
	Big	8.37

SD (Maintenance_ Price, Lug_Boot) = P (Small) SD (Small) + P (Big) SD (Big) $= \frac{7}{14} \times 9.36 + \frac{7}{14} \times 8.37 = 8.86$

SDR = SD (Maintenance_ Price) – SD (Maintenance_ Price, Lug_Boot) = 9.32 – 8.86 = 0.46

		Maintenance_ Price (SD)
Safety	High	7.87
	Low	10.59

SD (Maintenance_ Price, Safety) = P (High) SD (High) + P (Low) SD (Low) $= \frac{8}{14} \times 7.87 + \frac{6}{14} \times 10.59 = 9.02$

SDR = SD (Maintenance_ Price) – SD (Maintenance_ Price, Safety) = 9.32 – 9.02 = 0.3

SDR of Buying_Price is highest so we select Buying_Price as our root node.

To avoid over fitting we should terminate unnecessary building branches. For example if there are less than five instances in the sub data set or standard deviation can be less than 5% of the entire data set. I prefer to apply the first one. I will terminate the branch if there are less than 5 instances in current sub data set. If this termination condition is satisfied then i will calculate average of sub data set.

- **Step 2(a) :** Now we will consider the records of 'High'.

Buying_Price	Lug_Boot	Safety	Maintenance_ Price? (in thousand)
High	Small	High	45
High	Big	High	52

Buying_Price	Lug_Boot	Safety	Maintenance_ Price? (in thousand)
High	Big	Low	23
High	Big	High	46
High	Small	Low	30

For Buying_Price = High, SD = 10.87

We will calculate SDR of only Lug_Boot and Safety

		Maintenance_ Price (SD)
Lug_Boot	Small	7.5
	Big	12.49

SD(High, Lug_Boot) = P (Small) SD (Small) + P (Big) SD (Big) = $\frac{2}{5} \times 7.5 + \frac{3}{5} \times 12.49 = 10.49$

SDR = SD(High) – SD (Maintenance_ Price, Lug_Boot) = 10.87 – 10.49 = 0.38

		Maintenance_ Price (SD)
Safety	High	3.09
	Low	3.50

SD (High, Safety) = P (High) SD (High) + P (Low) SD (Low) = $\frac{3}{5} \times 3.09 + \frac{2}{6} \times 3.5 = 3.25$

SDR = SD (High) – SD (Maintenance_ Price, Safety) = 10.87 – 3.25 = 7.62

- SDR of Safety is highest so we select Safety as next node below High branch.

  For Buying_Price = High and Safety

  = High, we can directly write down the answer.

  For Buying_Price = High and Safety

  = Low, we can directly write down the answer.

- To write down the answer we take average of values of following records,

  For Buying_Price = High and Safety = High, Maintenance _Price

  = (45 + 52 + 46) /3 = 47.7

  For Buying_Price = High and Safety = Low, Maintenance _Price

  = (23 + 30) /2 = 26.5

▶ **Step 3 :** Now we will consider the records of 'Medium'

Buying_Price	Lug_Boot	Safety	Maintenance_ Price? (in thousand)
Medium	Small	High	46
Medium	Big	Low	43
Medium	Small	Low	52
Medium	Big	High	44

For Buying_Price = Medium, we can directly write down the answer as 46.3. The answer is calculated by taking the average of values of Maintenance_ Price for Medium records (average of 46, 43, 52, and 44).

▶ **Step 4 :** Now we will consider the records of 'Low'.

Buying_Price	Lug_Boot	Safety	Maintenance_ Price? (In thousand)
Low	Small	High	25
Low	Small	Low	30
Low	Small	High	35
Low	Big	High	38
Low	Big	Low	48

For Buying_Price = Low, SD = 7.78

- Now only Lug_Boot attribute is remaining, so we can directly take Lug_Boot as a next node below Low branch.
- To write down the answer we take average of values of following records,

  For Buying_Price = Low and Lug_Boot = Small, Maintenance_Price

  = (25 + 30 + 35) /3 = 30

  For Buying_Price = Low and Lug_Boot

  = Big, values of Maintenance_Price

  = (38 + 48) /2 = 43.

- Final Regression Tree is,

Example 2

Day	Outlook	Temp	Humidity	Windy	Hours Play?
1	Rainy	Hot	High	False	25
2	Rainy	Hot	High	True	30
3	Overcast	Hot	High	False	46
4	Sunny	Mild	High	False	45
5	Sunny	Cool	Normal	False	52
6	Sunny	Cool	Normal	True	23
7	Overcast	Cool	Normal	True	43
8	Rainy	Mild	High	False	35

Day	Outlook	Temp	Humidity	Windy	Hours Play?
9	Rainy	Cool	Normal	False	38
10	Sunny	Mild	Normal	False	46
11	Rainy	Mild	Normal	True	48
12	Overcast	Mild	High	True	52
13	Overcast	Hot	Normal	False	44
14	Sunny	Mild	High	True	30

Standard deviation

A decision tree is built up top down from root node and involved partitioning the data into subsets that contain instances with similar values. We use SD to calculate homogeneity of a numerical sample.

$$SD, S = \sqrt{\frac{\sum (x-\mu)^2}{n}} = 9.32$$

SD Reduction

It is based on the decrease in SD after a dataset is split on an attribute. Constructing a tree is all about finding attribute that returns highest SDR.

- **Step 1 :** SD (Hours Play?) = 9.32
- **Step 2 :**

The dataset is then split on the different attribute.SD for each branch is calculated. The resulting SD is subtracted from SD before split.

		Hours Play(SD)
Outlook	Rainy	7.78
	Overcast	3.49
	sunny	10.87

SD (HoursPlay, Outlook) = P (Rainy) SD (Rainy) + P (Overcast) SD (Overcast) + P (Sunny) SD (Sunny)

$$= \frac{5}{14} \times 7.78 + \frac{4}{14} \times 3.49 + \frac{5}{14} \times 10.87 = 7.66$$

SDR = SD (Hours Play) – SD (Hours Play, Outlook) = 9.32 – 7.66 = 1.66

		Hours Play(SD)
Temp	Cool	10.51
	Hot	8.95
	Mild	7.65

SD(Hours Play, Temp) = P (Cool) SD (Cool) + P (Hot) SD (Hot) + P (Mild) SD (Mild)

$$= \frac{4}{14} \times 10.51 + \frac{4}{14} \times 8.95 + \frac{6}{14} \times 7.65 = 8.84$$

SDR = SD (Hours Play) – SD(Hours Play, Temp) = 9.32 – 8.84 = 0.48

		Hours Play (SD)
Humidity	High	9.36
	Normal	8.37

SD (Hours Play, Humidity) = P (High) SD (High) + P (Normal) SD (Normal)

$= \frac{7}{14} \times 9.36 + \frac{7}{14} \times 8.37 = 8.86$

SDR = SD (Hours Play) – SD (Hours Play, Humidity) = 9.32 – 8.86 = 0.46

		Hours Play(SD)
Windy	False	7.87
	True	10.59

SD (HoursPlay, Windy) = P (False) SD (False) + P (True) SD (True) = $\frac{8}{14} \times 7.87 + \frac{6}{14} \times 10.59 = 9.02$

SDR = SD (Hours Play) – SD (Hours Play, Windy) = 9.32 – 9.02 = 0.3

SDR of outlook is highest so we select outlook as our root node.

▶ **Step 2(a) :** Now we will consider the records of 'sunny'.

Day	Outlook	Temp	Humidity	Windy	Hours Play?
4	Sunny	Mild	High	False	45
5	Sunny	Cool	Normal	False	52
6	Sunny	Cool	Normal	True	23
10	Sunny	Mild	Normal	False	46
14	Sunny	Mild	High	True	30

For Outlook = Sunny, SD = 10.87

We will calculate SDR of only Temp, Humidity and Windy

		Hours Play (SD)
Temp	Cool	14.5
	Mild	7.31

SD (Sunny, Temp) = P (Cool) SD (Cool) + P (Mild) SD (Mild) = $\frac{2}{5} \times 14.5 + \frac{3}{5} \times 7.31 = 10.18$

SDR = SD(Sunny) – SD (Hours Play, Temp) = 10.87 – 10.18 = 0.69

		Hours Play (SD)
Humidity	High	7.5
	Normal	12.49

SD (Sunny, Humidity) = P (High) SD (High) + P (Normal) SD (Normal) = $\frac{2}{5} \times 7.5 + \frac{3}{5} \times 12.49 = 10.49$

SDR = SD (Sunny) – SD (Hours Play, Humidity) = 10.87 – 10.49 = 0.38

		Hours Play (SD)
Windy	False	3.09
	True	3.50

SD (Sunny, Windy) = P (False) SD (False) + P (True) SD (True) = $\frac{3}{5} \times 3.09 + \frac{2}{5} \times 3.5 = 3.25$

SDR = SD (Sunny) – SD (Hours Play, Windy) = 10.87 – 3.25 = 7.62

- SDR of windy is highest so we select windy as next node below sunny branch.

  For Outlook = sunny and Windy = false, we can directly write down the answer.

  For Outlook = sunny and Windy = true, we can directly write down the answer

- To write down the answer we take average of values of following records,

  For Outlook = sunny and Windy = false, Hours Play

  = (45 + 52 + 46) /3 = 47.7

  For Outlook = sunny and Windy = true, Hours Play

  = (23 + 30) /2 = 26.5

Step 3 : Now we will consider the records of 'overcast'. For Overcast we will directly write down the answer.

Day	Outlook	Temp	Humidity	Windy	Hours Play?
3	Overcast	Hot	High	False	46
7	Overcast	Cool	Normal	True	43
12	Overcast	Mild	High	True	52
13	Overcast	Hot	Normal	False	44

The answer is calculated by taking the average of values of Hours Play for overcast records (average of 46, 43, 52, and 44).

▶ **Step 4:** Now we will consider the records of 'Rainy'.

Day	Outlook	Temp	Humidity	Windy	Hours Play?
1	Rainy	Hot	High	False	25
2	Rainy	Hot	High	True	30
8	Rainy	Mild	High	False	35
9	Rainy	Cool	Normal	False	38
11	Rainy	Mild	Normal	True	48

For Outlook = Rainy, SD = 7.78

Now we will calculate SDR of only Humidity and Temp

		Hours Play (SD)
Temp	Hot	2.5
	Cool	0
	Mild	6.5

SD (Rainy, Temp) = P (Hot) SD (Hot) + P (Cool) SD (Cool) + P (Mild) SD (Mild) = $\frac{2}{5} \times 2.5 + \frac{1}{5} \times 0 + \frac{2}{5} \times 6.5 = 3.6$

SDR = SD (Rainy) – SD (Rainy Play, Temp) = 7.78 – 3.6 = 4.18

		Hours Play(SD)
Humidity	High	5
	Normal	5

SD (Sunny, Humidity) = P (High) SD (High) + P (Normal) SD (Normal) = $\frac{3}{5} \times 5 + \frac{2}{5} \times 5 = 5$

SDR = SD (Rainy) – SD (Hours Play, Humidity) = 7.78 – 5 = 2.28

- SDR of Temp is highest so we select Temp as next node below rainy branch.

  For Outlook = Rainy and Temp = Cool, we can directly write down the answer.

  For Outlook = Rainy and Temp = Hot, we can directly write down the answer.

  For Outlook = Rainy and Temp = Mild, we can directly write down the answer.

- To write down the answer we take average of values of following records,

  For Outlook = Rainy and Temp = Hot, Hours Play = (25 + 30) /2 = 27.5

  For Outlook = Rainy and Temp = Mild, Hours Play = (35 + 48) /2 = 41.5

  For Outlook = Rainy, Temp = Cool, Hours Play = 38

- Final Regression Tree is.

2.13 PERFORMANCE METRICS

2.13.1 Performance Metrics for Regression

Regression analysis is a subfield of supervised machine learning. It aims to model the relationship between a certain number of features and a continuous target variable. Following are the performance metrics used for evaluating a regression model :

1. Mean Absolute Error (MAE)

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

Where y_i is the actual expected output and $\hat{y}_i$ is the model's prediction. It is the simplest evaluation metric for a regression scenario and is not much popular compared to the following metrics.

Say, y_i = [5,10,15,20] and $\hat{y}_i$ = [4.8,10.6,14.3,20.1]

Thus, $MAE = \frac{1}{4} * (|5 - 4.8| + |10 - 10.6| + |15 - 14.3| + |20 - 20.1|) = 0.4$

2. Mean Squared Error (MSE)

$$MAE = \frac{1}{N}\sum_{i=1}^{N} \left(y_i - \hat{y}_i\right)^2$$

Here, the error term is squared and thus more sensitive to outliers as compared to Mean Absolute Error (MAE).

Thus, $MSE = \frac{1}{4} * (|5 - 4.8|\hat{2} + |10 - 10.6|\hat{2} + |15 - 14.3|\hat{2} + |20 - 20.1|\hat{2}) = 0.225$

3. Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n} \left(y_j - \hat{y}_j\right)^2}$$

Since MSE includes squared error terms, we take the square root of the MSE, which gives rise to Root Mean Squared Error (RMSE). Thus, $RMSE = (0.225)\hat{0.5} = 0.474$

4. R-Squared

$$R_2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\Sigma_i \left(y_i - \hat{y}_i\right)^2}{\Sigma_i \left(y_i - \bar{y}\right)^2}$$

- R-squared is calculated by dividing the sum of squares of residuals (SS_{res}) from the regression model by the total sum of squares (SS_{tot}) of errors from the average model and then subtract it from 1. R-squared is also known as the Coefficient of Determination. It *explains the degree to which the input variables explain the variation of the output / predicted variable.*
- A R-squared value of 0.81, tells that the input variables explains 81 % of the variation in the output variable. The higher the R squared, the more variation is explained by the input variables and better is the model. Although, there exists a limitation in this metric, which is solved by the Adjusted R-squared.

5. Adjusted R-squared

$$\text{Adjusted } R^2 = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1}$$

- Here, N - total sample size (number of rows) and p - number of predictors (number of columns). The limitation of R-squared is that it will *either stay the same or increases with the addition of more variables, even if they do not have any relationship with the output variables.
- To overcome this limitation, Adjusted R-square comes into the picture as it penalizes you for adding the variables which do not improve your existing model. Hence, if you are building Linear regression on multiple variables, it is always suggested that you use Adjusted R-squared to judge the goodness of the model. If there exists only one input variable, R-square and Adjusted R squared are same.

2.13.2 Performance Metrics for Classification

Classification is the problem of identifying to which of a set of categories/classes a new observation belongs, based on the training set of data containing records whose class label is known. Following are the performance metrics used for evaluating a classification model :

- To understand different metrics, we must understand the Confusion matrix. A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.

	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

TN- True negatives (actual 0 predicted 0) & TP- True positives (actual 1 predicted 1)
FP- False positives (actual 0 predicted 1) & FN- False Negatives (actual 1 predicted 0)

- Consider the following values for the confusion matrix-

True negatives (TN) = 300

True positives (TP) = 500

False negatives (FN) = 150

False positives (FP) = 50

1. Accuracy

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

- Accuracy is defined as the ratio of the number of correct predictions and the total number of predictions. It lies between [0,1]. In general, higher accuracy means a better model (TP and TN must be high).

- However, accuracy is not a useful metric in case of an imbalanced dataset (datasets with uneven distribution of classes). Say we have a data of 1000 patients out of which 50 are having cancer and 950 not, a dumb model which always predicts as no cancer will have the accuracy of 95%, but it is of no practical use since in this case, we want the number of False Negatives as a minimum. Thus, we have different metrics like recall, precision, F1-score etc.
- Thus, Accuracy using above values will be (500+300)/(500+50+150+300) = 800/1000 = 80%

2. Precision and Recall

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN}$$

- Recall is a useful metric in case of cancer detection, where we want to minimize the number of False negatives for any practical use since we don't want our model to mark a patient suffering from cancer as safe. On the other hand, predicting a healthy patient as cancerous is not a big issue since, in further diagnosis, it will be cleared that he does not have cancer. Recall is also known as Sensitivity.
- Thus, Recall using above values will be 500/(500+150) = 500/650 = 76.92%
- Precision is useful when we want to reduce the number of False Positives. Consider a system that predicts whether the e-mail received is spam or not. Taking spam as a positive class, we do not want our system to predict non-spam e-mails (important e-mails) as spam, i.e., the aim is to reduce the number of False Positives.
- Thus, Precision using above values will be 500/(500+50) = 500/550 = 90.90%

3. Specificity

- Specificity is defined as the ratio of True negatives and True negatives + False positives. We want the value of specificity to be high. Its value lies between [0,1].

$$\text{Specificity} = \frac{\text{True Negatives}}{\text{True Negatives + False Positives}}$$

- Thus, Specificity using above values will be 300/(300+50) = 300/350 = 85.71%

4. F1-score

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision + recall}}$$

- F_1-score is a metric that combines both Precision and Recall and equals to the harmonic mean of precision and recall. Its value lies between [0,1] (more the value better the F1-score).
- Using values of precision=0.9090 and recall=0.7692, F1-score = 0.8333 = 83.33%

5. AUC-ROC

- AUC (Area Under The Curve)- ROC (Receiver Operating Characteristics) curve is one of the most important evaluation metrics for checking any classification model's performance.
- It is plotted between FPR (X-axis) and TPR (Y-axis). If the value is less than 0.5 than the model is even worse than a random guessing model.

$$\text{True Positive Rate (TPR)} = \frac{TP}{FP + FN}$$

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP + TN}$$

2.13.3 Kappa Statistics

- It is a statistic which measures inter-rater agreement for qualitative (categorical) items. It is generally thought to be a more robust measure than simple percent agreement calculation, since k takes into account the agreement occurring by chance. Cohen's kappa measures the agreement between two raters who each classify N items into C mutually exclusive categories.
- Cohen's kappa coefficient is defined and given by the following function :

$$K = \frac{p_0 - p_e}{1 - p_e}$$

Where :

p_0 = relative observed agreement among raters.

p_e = the hypothetical probability of chance agreement.

p_0 and p_e are computed using the observed data to calculate the probabilities of each observer randomly saying each category. If the raters are in complete agreement then k = 1. If there is no agreement among the raters other than what would be expected by chance (as given by p_e), $k \leq 0$.

Example

Problem Statement

Ex. 2.13.1 : Suppose that you were analyzing data related to a group of 50 people applying for a grant. Each grant proposal was read by two readers and each reader either said "Yes" or "No" to the proposal. Suppose the disagreement count data were as follows, where A and B are readers, data on the diagonal slanting left shows the count of agreements and the data on the diagonal slanting right, disagreements :

		B	
		Yes	No
A	Yes	20	5
	No	10	15

Calculate Cohen's kappa coefficient.

Soln. :

Note that there were 20 proposals that were granted by both reader A and reader B and 15 proposals that were rejected by both readers. Thus, the observed proportionate agreement is

$$p_0 = \frac{20 + 15}{50} = 0.70$$

To calculate p_e (the probability of random agreement) we note that –

- Reader A said "Yes" to 25 applicants and "No" to 25 applicants. Thus reader A said "Yes" 50% of the time.
- Reader B said "Yes" to 30 applicants and "No" to 20 applicants. Thus reader B said "Yes" 60% of the time.

Using formula P (A and B) = P (A) × P (B) where P is probability of event occuring.

The probability that both of them would say "Yes" randomly is 0.50 × 0.60 = 0.30 and the probability that both of them would say "No" is 0.50 × 0.40 = 0.20. Thus the overall probability of random agreement is p_e p_e = 0.3 + 0.2 = 0.5.

So now applying our formula for Cohen's Kappa we get:

$$k = 0.70 - \frac{0.50}{1 - 0.50} = 0.40$$

Chapter Ends...

MODULE 3

CHAPTER

3

Ensemble Learning

Syllabus

Understanding Ensembles, K-fold cross validation, Boosting, Stumping, XGBoost
Bagging, Subagging, Random Forest, Comparison with Boosting, Different ways to combine classifiers.

3.1 Ensemble Learning 3-2
GQ. Explain the concept of Ensemble learning. **(5 Marks)** 3-2
3.2 Cross-validation in machine learning 3-2
3.2.1 Methods used for Cross-Validation 3-2
3.2.2 K-Fold Cross-Validation 3-2
3.2.3 Life Cycle of K-fold Cross-Validation 3-2
3.2.4 Thumb Rules Associated with K-Fold 3-3
3.2.5 Some Remarks 3-4
3.3 Stumping ensembles 3-5
3.3.1 For Continuous Features 3-5
3.3.2 Remarks 3-6
3.4 Boosting in machine learning 3-6
3.5 XG-Boost 3-6
3.5.1 Decision Tree 3-6
3.5.2 Table of Contents 3-7
GQ. Write a short note on : Bagging and Boosting. **(10 Marks)** 3-7
3.6 Random Forest 3-7
3.6.1 Random Forest 3-9
3.6.2 Random Forest Algorithm 3-9
3.6.3 Bagging 3-9
3.6.4 Row Sampling 3-10
3.6.5 Important Features of Random Forest 3-10
3.6.6 Difference Between Decision Tree and Random Forest 3-11
3.6.7 Advantages and Disadvantages of Random Forest Algorithm 3-11
3.7 Comparison Between Boosting Random Forest And Boosting 3-12
3.8 Different Ways To Combine Classifiers 3-12
• **Chapter Ends** 3-13

3.1 ENSEMBLE LEARNING

GQ. Explain the concept of Ensemble learning. **(5 Marks)**

- An ensemble is a **machine learning model that combines** the predictions from two or more models.
- The models that contribute to the ensemble are called as ensemble members.
- They may be of the same type or of different types.
- They may or may not be trained on the same training data.

Remarks

1. An ensemble method is a technique that uses multiple independent similar or different models to derive an output or make some predictions.

   For example, a random forest is an ensemble of multiple decision trees.

2. **Ensemble Learning :** This uses a single machine learning algorithm. It is an unpruned decision tree. It trains each model on a different sample of the same training dataset.

   The predictions made by the ensemble members are combined using simple statistics, such as voting or averaging.

3.2 CROSS-VALIDATION IN MACHINE LEARNING

- Cross-validation is a technique for validating the model efficiency by training it on the subset of input data and testing on previously unseen subset of the input data. **It is a technique how a statistical model generalises to an independent dataset**.
- In **machine learning**, we need to test the stability of the model. It means based only on the training dataset, we cannot fit our model on the training dataset.
- For this purpose, we test our model on the sample which is not part of the training dataset. And then we deploy the model on that sample.
- This complete process comes under cross-validation.
- The basic steps of cross-validation are :
  - (i) Reserve a subset of the dataset as a validation set.
  - (ii) Provide the training to the model using the training dataset.
  - (iii) Evaluate model performance using the validation set.
- If the model performs well with the validation set, perform the further step, else check for the issues.

3.2.1 Methods used for Cross-Validation

The common methods used for cross-validation are :

(1) Validation set approach (2) Leave-P-out cross-validation

(3) Leave one out cross- validation (4) K-fold cross-validation

(5) Stratified K-fold cross-validation

Among these, K-fold cross-validation is easy to understand, and the output is less biased than other methods.

3.2.2 K-Fold Cross-Validation

- In each set (fold) training and the test would be performed precisely once during this entire process. It helps us to avoid overfitting. When a model is trained using all of the data in a single short and give the best performance accuracy.
- This K-fold cross-validation helps us to build the model which is a generalized one.

- To achieve this K-fold cross validation, we have to split the data set into three sets, training, testing and validation, with the challenge of the volume of the data.
- Here test and train data set will support building model and hyper parameter assessment.
- The model is validated multiple times based on the value assigned as a parameter and which is called K and it should be an **INTEGER**.
- The dataset X is divided randomly into K equal-sized parts, X_i, i = 1, 2, ..., K.
- To generate each pair, we keep one of the K parts out as validation set and combine the remaining (K – 1) parts to form the training set.
- Doing this K times, each time leaving out another one of the K parts out, we get K pairs :

$$V_1 = X_1, T_1 = X_2 \cup X_3 \cup \ldots \cup X_K$$
$$V_2 = X_2, T_2 = X_1 \cup X_3 \cup \ldots \cup X_K$$
$$\vdots$$
$$V_K = X_K, T_K = X_1 \cup X_2 \cup \ldots \cup X_{K-1}$$

- We come across two problems with this. First, to keep the training set large, we allow validation sets that are small.
- Second, the training sets overlap considerably namely, any two training sets share (K – 2) parts.
- K is typically 10 or 30. As K increases, the percentage of training instances increases and we get more robust estimators, but the validation set becomes smaller.
- Also, there is the cost of training the classifier K times, which increases as K is increased.
- As N increases, K can be smaller, if N is small, K should be large to allow large large enough training sets. (N is the number of instances).
- One extreme case of K-fold cross-validation is **leave-one-out** where given a dataset of N instances, only one instance is left out as the validation set (instance) and training uses (N – 1) instances.
- We then get N separate parts by leaving out a different instance at each iteration. This is typically used in applications such as medical diagnosis.

3.2.3 Life Cycle of K-fold Cross-Validation

- Let us have a generalised K-value. If K = 5, it means, we are splitting the given dataset into 5 folds and running the Train and Test.

Fig. 3.2.1

- During each run, one fold is considered for testing and the rest will be for training and moving on with iterations, the below pictorial representation gives an idea of the flow of the fold-defined size

Fig. 3.2.2

- Here, each data-points is used, once in the hold-out set and K – 1 in Training. So during the full iteration at least once, one fold will be used for testing and the rest for training. In the above set, 5-Testing 20 Training.
- In each iteration, we will get an accuracy score and have to sum them and find the mean.
- Here we can understand how the data is spread in a way of consistency and will make a conclusion whether to go for the production with this model (or) NOT.

Fig. 3.2.3

3.2.4 Thumb Rules Associated with K-Fold

We discuss a few thumb-rules while dealing with K-fold.

(i) K should be always ≥ 2 and equal to number of records.

If 2, then just 2 iterations

If K = No of records in the dataset, then 1 for testing and n-for training.

(ii) The optimised value for the K is 10 and used with the data of good size, (i.e. commonly used).

(iii) If the K-value is large, then this will lead to less variance across the training set and limit the model currency, difference across the iterations.

(iv) The number of folds is generally inversely proportional to the size of the data set, which means, if the dataset size is too small, the number of folds can increase.

(v) Larger values of K eventually increase the running time of the cross-validation process.

3.2.5 Some Remarks

(1) Short note on K-cross Validation

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.

The procedure has a single parameter called K that refers to the number of groups that a given data sample is to be split into. Hence, the procedure is often called K-fold cross-validation.

(2) Purpose of K-fold Cross-Validation

K-fold cross-validation is one method that attempts to maximise the use of the available data for training and then testing a model. It is particularly useful for assessing model performance, as it provides a range of accuracy scores across (somewhat) different data sets.

(3) Folds in K-fold Cross-Validation

The key configuration parameter for K-fold cross-validation is K that defines the number of folds in which to split a given dataset.

Common values are K = 3, K= 5 and K = 10, and the most popular value used in applied machine learning to evaluate models is K = 10.

(4) Difference between K-fold and cross-validation

Cross-validation score is a function which evaluates a data and returns the score. On the other hand, K-fold is a class, which lets to split data to K-folds.

(5) Does K-fold cross-validation prevent over fitting ?

K-fold cross-validation won't reduce over fitting on its own, but using it will generally gives a better insight on the model, which can help to avoid or reduce over fitting.

3.3 STUMPING ENSEMBLES

A decision stump is a machine learning model consisting of a one-level decision tree. That is, it is a decision tree with one internal node (the root) which is immediately connected to the terminal nodes (its levels).

A decision stump makes a prediction based on the value of just a single input feature. Sometimes they are also called as **1-rules**.

[An example of a decision stump that discriminates between two of three classes of **iris** flower data set : **iris versicolor** and **iris verginica.** The petal width is in centimetres. This particular stump achieves 94% accuracy on the iris dataset for these two classes.]

Fig. 3.3.1

Depending on the type of the input features, several variations are possible. For normal features, one may build a stump which contains a leaf for each possible feature value, or a stump with the two leaves, one of which corresponds to some chosen category, and the other leaf to all the other categories.

For binary features these two schemes are identical. A missing value may be treated as a yet another category.

3.3.1 For Continuous Features

Usually, some threshold feature value is selected, and the stump contains two leaves-for values below and above the threshold. However, rarely, multiple thresholds may be chosen and the stump therefore contains three or move leaves.

Decision stumps are often used as components (called "weak learners" or "base learner") in **machine learning ensemble** techniques such as **bagging** and **boosting**.

3.3.2 Remarks

(1) Meaning of stump in decision tree

- A decision stump is a decision tree, which uses only a single attribute for splitting.
- For discrete attributes, this means that the tree consists only of a single interior node (i.e., the root has only leaves as successor nodes).
- If the attribute is numerical, the tree may be more complex.

(2) Are decision stumps linear ?

A decision stump is not a linear model. The decision boundary can be a line, even if the model is not linear.

3.4 BOOSTING IN MACHINE LEARNING

- Boosting is an ensemble modelling technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using a weak models in series.
- Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model.
- This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.
- AdaBoost was the first really successful boosting algorithm developed for the purpose of binary classification.
- AdaBoost is short for **Adaptive Boosting** and is a very popular boosting technique that combines multiple "weak classifiers" into a single "Strong Classifiers".
- AdaBoost is the best starting point for understanding boosting algorithms. It is adaptive in the sense that subsequent classifiers built are tweaked in favour of those instances misclassified by previous classifiers.
- It is sensitive to noisy data and outliers.
- AdaBoost uses multiple iterations to generate a single composite strong learner. It creates a strong learner by iteratively adding weak learners. During each phase of training, a new weak learner is added to the ensemble, and a weighting vector is adjusted to focus on examples that were misclassified in previous rounds. The result is a classifier that has higher accuracy than the weak learner classifiers.

3.5 XG-BOOST

- XG-Boost stands for Extreme Gradient Boosting. It is written in C++ which optimises the training for Gradient boosting.
- To understand the XG-boost, we first need to understand the trees especially the decision tree :

3.5.1 Decision Tree

- A decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.
- A tree can be " learned" by splitting the source set into subsets based on an attribute value test.
- This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursive is completed when the subset at a node all has the same value of the target variable, or when splitting no larger adds value to the predictions.

3.5.2 Table of Contents

We mention below the contents of XG-Boost.

(1) The power of XG-Boost

(2) Why Ensemble Learning ?

(i) Bagging (ii) Boosting

(3) Splitting Algorithms

(4) System Features.

(5) Unique features of XG-Boost, and Goals of XG-Boost.

(1) The power of XG-Boost

- The beauty of this powerful algorithm lies in its scalability, which drives fast learning through parallel and distributed computing and offers efficient memory usage.
- XG-Boost is the most useful, straight forward and robust solution.

(2) Why Ensemble Learning ?

GQ. Write a short note on : Bagging and Boosting. **(10 Marks)**

- XG-Boost is an **ensemble learning** method. It may not be sufficient to rely upon the results of just one machine learning model.
- Ensemble learning offers a systematic solution to combine the predictive power of multiple learners. The resultant is a single model which gives the aggregated output from several models.
- The models that form the ensemble, could be either from the same learning algorithm or different learning algorithms. Bagging and Boosting are two widely used ensemble learners. First we discuss Bagging

(i) Bagging

- While decision trees are one of the most easily interpretable models, they exhibit highly variable behaviour.
- Consider a single training dataset that we randomly split into two parts. Now, we use each part to train a decision tree in order to obtain two models.
- When we fit both these models, they would yield different results. Decision trees are said to be associated with high variance due to this behaviour.
- Bagging or boosting aggregation helps to reduce the variance in any learner. Data sampled with replacement is fed to these learners for training. The final prediction is the averaged output from all the learners.

(ii) Boosting

- In boosting, the trees are built sequentially such that each subsequent tree aims to reduce the errors of the previous tree. Each tree learns from its predecessors and updates the residual errors.

- Hence, the tree that grows next in the sequence will learn from an updated version of the residuals.
- The base learners in boosting are weak learners in which the bias is high, and the predictive power is just better than random guessing.
- Each of these weak learners contributes some vital information for prediction. It enables the boosting technique to produce a strong learner by effectively combining these weak learners.
- The final strong learner brings down both the bias and the variance.

(3) Splitting algorithms

(i) **Exact greedy algorithm :** The main problem in tree learning is to find the best split. This algorithm gives all the possible splits on all the features. It is computationally demanding to enumerate all the possible splits for continuous features.

(ii) **Approximate algorithm :** The exact greedy algorithm is very powerful since it gives all possible splitting points greedily. But it is not possible to efficiently carry it out when the data does not fit entirely into memory. Approximate algorithm proposes candidate splitting points according to percentiles of feature distribution. The algorithm then maps the continuous features into buckets split by these candidate points, aggregates the statistics and finds the best solution among proposals based on the aggregated statistics.

(iii) **Weighted quantile sketch :** One important step in the approximate algorithm is to propose candidate split points. XG-Boost has a distributed weighted quantile sketch algorithm to effectively handle weighted data.

(iv) **Sparsity-aware split finding :** In many real-world problems, it is quite common for the input X to be sparse. There are multiple possible causes for sparsity :

(a) Presence of missing values in the data

(b) Frequent zero entries in the statistics.

(c) Artifacts of features engineering such as one-hot encoding.

It is important to make the algorithm aware of the sparsity pattern in the data. XG boost handles all sparsity patterns in a unified way.

(4) System features

(i) Collecting statistics for each column can be parallelised giving us a parallel algorithm for split-finding.

(ii) **Cache-aware access :** XG boost has been designed to make optimal use of hardware. This is done by allocating internal buffers in each thread, where the gradient statistics can be stored.

(iii) **Blocks for out of core** computation for very large datasets that don't fit into memory.

(iv) **Distributed computing** for training very large models using a cluster of machines.

(v) **Column block for parallel learning :** The most time-consuming part of tree learning is to get the data into sorted order. In order to reduce the cost of sorting, the data is stored in the column blocks in sorted order in compressed format.

(5) XG-Boost features

(i) **Regularised learning :** Regularisation term helps to smooth the final learnt weights to avoid overfitting. The regularised objective will tend to select a model employing simple and predictive functions.

(ii) **Gradient tree boosting :** The tree ensemble model cannot be optimised using traditional optimization methods in Euclidean space. Instead, the model is trained in an additive manner.

(iii) **Shrinkage and column subsampling :**

- Besides the regularised objective, two additional techniques are used to further prevent overfitting.
- The first technique is shrinkage. Shrinkage scales newly added weights by a factor η after each step of tree boosting.

- Similar to a learning rate in stochastic optimisation, shrinkage reduces the influence of each tree and leaves space for future trees to improve the model.
- The second technique is the column (feature) subsampling. This technique is used in random forest. Column sub-sampling prevents over-fiting even more so than the traditional row sub-sampling. The usage of column sub-samples also speeds up computations of the parallel algorithm.

(6) Goals of XG Boost

(i) **Execution speed :** XG boost was almost always faster than the other benchmarked. Implementation from python is really factor when compared to the other algorithms.

(ii) **Model performance :** XG boost dominates structured or tabular datasets on classification and regression predictive modelling problems.

(7) Conclusion

- XG boost is a faster algorithm when compared to other algorithms because of its parallel and distributed computing.
- XG Boost is developed with both deep considerations in terms of systems optimisation and principles in machine learning.
- The goal of this library is to push the extreme of the computation limits of machines to provide a scalable, portable and accurate library.

3.6 RANDOM FOREST

3.6.1 Random Forest

- Random forest is a **supervised machine learning algorithm** that is used widely in **classification** and **Regression problems.**
- It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

Fig. 3.6.1

3.6.2 Random Forest Algorithm

- Random forest Algorithm can handle the data set containing **continuous variables** in case of regression and **categorical variables** in case of classification.
- It performs better results for classification problems.

Working of Random Forest Algorithm

Ensemble Technique

- Ensemble means combining multiple models. Here a collection of models make predictions rather than an individual model.
- Ensemble uses two types of methods

1. **Bagging :** It creates a different training subset from sample training data with replacement. The final output is based on majority voting. For example, Random forest.
2. **Boosting :** It creates sequential models such that the final model has the highest accuracy. Then it combines weak learners into strong learners. For example, ADA BOOST, XG BOOST.

Fig. 3.6.2

Remark

Random forest works on the Bagging principle.

3.6.3 Bagging

- Bagging is also known as Bootstrap Aggregation. It is an ensemble technique used by random forest.
- Bagging chooses a random sample from the data set.
- Hence each model is generated from the samples (Bootstrap samples) provided by the original data with replacement. It is known as Row sampling.

3.6.4 Row Sampling

- This step of row sampling with replacement is called bootstrap.
- Each model is trained independently and it generates results. The final output is based on majority voting after combining the results of all models.
- The output that is based on majority voting is known as **aggregation.**
- We observe that the bootstrap sample is taken from the actual data (Bootstrap sample 01, Bootstrap sample 02 and Bootstrap sample 03) with a replacement.
- Clearly each sample will not contain unique data.
- Now, the model, Model 01, Model 02, Model 03 obtained from this bootstrap sample are trained independently.
- Each model generates results. And based on majority voting final output is obtained

Fig. 3.6.3

Steps involved in random forest algorithm

- **Step 1 :** In Random forest n number of random records are taken from the data set having k number of records.
- **Step 2 :** Individual decision trees are constructed for each sample.

Fig. 3.6.4

- **Step 3 :** Each decision tree will generate an output.
- **Step 4 :** Final output is based on **majority voting or averaging** for classification and regression respectively.

3.6.5 Important Features of Random Forest

1. **Diversity :** Not all attributes / Variables / features are considered while making an individual tree, each tree is different.
2. **Immune to the curse of dimensionality :**
   Since each tree does not consider all the features, the feature-space is reduced.
3. **Parallelisation :** Each tree is created independently out of different data and attributes. This implies that we can make full use of CPU to build random forests.
4. **Train-Test split :** In a random forest we don't have to segregate the data for training and testing. It is because there will always be 30% of the data which is not seen by the decision-tree.
5. **Stability :** Stability arises because the result is based on majority voting / averaging.

3.6.6 Difference Between Decision Tree and Random Forest

Actually, Random forest is a collection of decision trees, still there are differences in their behaviour.

Sr. No.	Decision Trees	Random Forest
1.	Decision trees normally suffer from the problem of overfitting. Since it is allowed to grow without any control.	Random forests are created from subsets of data and the final output is based on average or majority ranking and hence the problem of overfitting is taken care of.
2.	A single decision tree is faster in computation.	It is comparatively slower.
3.	When a dataset with features is taken as input by a decision tree, it will formulate some set of rules to do prediction.	Random forest randomly selects observations, builds a decision tree and the average result is taken. It does not use any set of formulas.

We conclude that if the trees are diverse and acceptable then random forests are much more successful than decision trees.

☞ Important Hyper parameters

Hyper parameters are used in random forests to

(i) Increase the performance power and

(ii) Predictive power of models or

(iii) To make the model faster.

☞ (I) Following hyper parameters increase the predictive power

1. **n-estimators :** Number of trees the algorithm builds before averaging the predictions.
2. **Max-features :** Maximum number of features-random forest split a node.
3. **Mini-sample-leaf :** It determines the minimum number of leaves to split on internal node.

☞ (II) Following hyper parameters increases the speed

1. **n-jobs :** It tells the engine how many processors it is allowed to use. If the value is 1, it can use only one processor but if the value is – 1, there is no limit.

2. **Random-state :** It controls the randomness of the sample. The model will always produce the same results if it has a definite value of random state and if it has been given the same hyper parameters and the same training data.
3. **OOB-score :** OOB implies out of the bag. It is a random forest cross-validation method. In this method, one-third of the sample is not used to train the data instead it is used to evaluate the performance.

   These samples are called as out of bag samples.

3.6.7 Advantages and Disadvantages of Random Forest Algorithm

Advantages

1. It can be used in classification and regression problems.
2. If solves the problem of overfitting as output is based on majority voting or averaging.
3. Even if data contains missing values or null values, it performs well.
4. Each decision tree created is independent of the other. It means that is shows the property of parallelisation.
5. It is highly stable as the average answers given by a large number of trees are taken.
6. If maintains diversity as all the attributes are not considered while making each decision tree even through it is not true in all cases.
7. It is immune to the curse of dimensionality. Since each tree does not consider all the attributes, feature-space is reduced.
8. We need not segregate data to train and test, as there will always be 30% of the data which is not seen by the decision tree which is made out of **bootstrap**.

Disadvantages

1. Random forest is highly complex when compared to decision trees where decisions can be made by following the path of the tree.
2. Training time is more compared to other models due to its complexity.

   Whenever it has to make a prediction, each decision tree has to generate output for the given input data.

Remark

1. Random forest can handle binary, continuous and categorical data.
2. Random forest can handle missing values.
3. Random forest is a fast, simple, flexible and robust model with some limitations.

3.7 COMPARISON BETWEEN BOOSTING RANDOM FOREST AND BOOSTING

Sr. No.	Random Forest	Boosting
1.	Random forest is a bagging technique	The decision trees are built additively, i.e. each decision tree is built one after another
2.	It is not a boosting technique.	In boosting, one is learning from other, which in turn boosts the learning.
3.	The trees in random forest run parallel.	Boosting is an approach to increases the complexity of models that suffer from high bias, that is, models that underfit the training data.

Sr. No.	Random Forest	Boosting
4.	There is no interaction between these trees while building the trees.	Boosting reduces error mainly by reducing bias, and also to some extent variance, by aggregating the output from many models.
5.	Random forest and bagging are "bagging" algorithms that aim to reduce the complexity of models that overfit the training data.	XG boost always gives more importance to functional space when reducing the cost of a model.
6.	Random forest uses fully grown decision trees (low bias, high variance). It tackles the error reduction in the opposite way : by reducing variance.	Gradient boosting may not be a good choice it there is lot of noise, as it can result in overfitting.
7.	Random forest tries to give more preferences to hyper parameters to optimise the model.	They also tend to be harder to tune than random forest.
8.	Random forest may not be give better performance compared to gradient boosting.	Gradient boosting trees can be more accurate than random forests.
9.	The basic idea in random forest is : although a single tree may be inaccurate, the collective decisions of a bunch of trees are likely to be right most of the time.	We train gradient boosting trees to correct each other's errors, they can capture complex patterns in the data.
10.	Forests are more robust and more accurate than a single tree. But, they are harder to interpret since each classification decision or regression output is not one but multiple decision paths.	If the data is noisy, the boosted trees may overfit and start modelling the noise.

3.8 DIFFERENT WAYS TO COMBINE CLASSIFIERS

(1) The simplest way of combining classifier output is to allow each classifier to make its our prediction and then choose the plurality prediction as the "final" output. The simple voting scheme easy to implement and easy to understand, but it does not always produce the best possible results.

(2) To form different ways to combine classifier : they can be divided into two big groups.

- **Ensemble methods :** Refers to sets of systems that combine to create a new system using the same learning technique. Bagging and boosting are the most extended ones.
- **Hybrid methods :** Take a set of different learners and combine them using new learning techniques.
- **Model ensembling** is also used for combining different classifiers.

(3) **Combining classifiers better than selecting the best one :** We mention two extensions of this method : (i) using an extended set of meta-level features and (ii) the other using multiresponse model trees to learn at the meta-level. The latter extension performs better than existing stacking approaches and better than selecting the best classifier by cross validation.

(4) **Ensemble classifiers always better than single classifier :** There is a possibility that an ensemble model performs better than an individual model. If we build many of those and the individual classifier is weak, then the overall performance should be better than an individual model.

(5) **To use stacking classifier :** A simple way to achieve this is to split training set in half, as shown below :

Fig. 3.8.1

(6) **Classes of classifiers**

There are three classes of classifiers :

(i) **Descriptive classifier :** Descriptive classifier are used to describe shape, size, texture or a pattern.

(ii) **Instrument classifier :** The handshapes of instrument classifiers describe how an object is handled.

(iii) **Element classifiers :** It takes a set of different learners and combines them using new learning techniques.

Chapter Ends...

❑❑❑

MODULE 4

CHAPTER 4

Learning with Classification

Syllabus

Support Vector Machine, Constrained Optimization, Optimal decision boundary, Margins and support vectors, SVM as constrained optimization problem, Quadratic Programming, SVM for linear and nonlinear classification, Basics of Kernel trick. Support Vector Regression, Multiclass Classification.

4.1 Introduction 4-3

4.1.1 Relation to Constraint Satisfaction Problems 4-3

4.1.2 General Form 4-3

4.1.3 Solution Methods 4-3

4.2 Supervised Learning : Support vector machine 4-5

UQ. Define Support Vector Machine. Explain how margin is computed and optimal hyper-plane is decided ? (MU - Dec. 19, 10 Marks) 4-5

4.2.1 Optimal Decision Boundary 4-5

GQ. What is decision boundary in decision tree ? 4-6

GQ. What is decision boundary ? How do you draw it and what is the advantage of it ? 4-6

UQ. What is SVM ? Explain the following terms: separating hyperplane, margin and support vectors with suitable example. (MU - May 15, 4 Marks) 4-6

UQ. What are the key terminologies of Support Vector Machine ? (MU - May 16, 5 Marks) 4-6

UQ. What is Support Vector Machine ? (MU - May 17, Dec. 19, 4 Marks) 4-6

UQ. Illustrate Support Vector machine with neat labeled sketch. (MU - May 19, 4 Marks 4-6

4.2.2 Quadratic Programming Solution to Find Maximum Margin Separator 4-7

UQ. Explain the term : Hyperplane with suitable example. (MU - May 15, 1 Marks) 4-7

UQ. Write detail notes on : Quadratic Programming solution for finding maximum margin separation in support vector machine. (MU - May 16, 10 Mark) 4-7

UQ. How to compute the margin ? (MU - May 17, 6 Marks) 4-7

UQ. Explain how margin is computed and optimal hyper-plane is decided ? (MU - Dec. 19, 6 Marks) 4-7

UQ. Show how to derive optimal hyper-Plane? (MU - May 19, 6 Marks) 4-7

4.2.3 Kernels for Learning Non-Linear Functions 4-9
4.2.4 Rules for the Kernel Function.....4-10
4.3 Different Types of SVM Kernels.....4-10
4.4 SVM as constrained optimisation problem.....4-11
4.4.1 Global Optimisation.....4-11
4.4.2 Examples 4-12
4.4.3 Local Optimisation.....4-12
4.4.4 Solving Optimisation Problems 4-12
4.4.5 Decision Tree 4-13
4.4.6 Constrained Optimization.....4-13
GQ. What is constrained optimization technique ?.....4-13
GQ. The difference between constrained and unconstrained optimization. 4-13
GQ. What are the three common elements in a constrained optimization problem ? 4-13
GQ. Types of optimization. 4-13
GQ. Top optimization methods in machine learning. 4-13
GQ. Requirement of optimization. 4-14
4.5 Quadratic programming 4-14
4.5.1 Problem Formulation.....4-14
4.5.2 Equality Constraints 4-14
GQ. How are quadratics used in programming ? 4-15
GQ. Is quadratic programming convex ?.....4-15
4.6 Types of support vector machines : (SVM).....4-15
4.6.1 Working of Linear SVM 4-16
4.6.2 Non-linear SVM.....4-16
4.6.3 Linear Classifier 4-17
4.7 Definition of Linear Classification.....4-17
4.7.1 Classification V/S Regression.....4-17
4.7.2 Classification 4-17
4.8 Non-linear functions 4-18
4.8.1 Support Vector Machines : (SVM).....4-19
4.9 SVM kernel trick.....4-19
4.9.1 Linearity and Non-linearity 4-20
4.9.2 Dimensions 4-20
4.9.3 Planes and Hyperplane 4-21
4.9.4 Kernel 4-21
4.9.5 Kernel Trick.....4-21
4.9.6 Applications of Kernel Trick 4-21
4.9.7 The Kernel Trick (Mathematically).....4-22
4.9.8 Few Popular Kernels 4-22
4.10 Support vector regression.....4-23
4.10.1 Support Vector Regression.....4-23
4.10.2 Important Terminologies in Support Vector Regression.....4-24
4.10.3 Support Vector Regression.....4-24
4.11 Multiclass Classification.....4-25
4.11.1 Popular Algorithms for Multiclass Classification.....4-25
4.11.2 Method of Multi-class Classification Approach ... 4-26
4.11.3 The Multiclass Data 4-26
4.11.4 Applying Decision Tree to the Multiclass Data 4-26
4.11.5 KNN for Multi-class Classification.....4-26
4.11.6 KNN and SVM Comparison.....4-26
4.11.7 KNN and Decision Tree.....4-26
4.11.8 KNN and Random Forest 4-27
4.12 Categorisation of Multi-class.....4-27
4.12.1 Transformation to Binary 4-27
4.12.2 One-vs-rest.....4-27
4.12.3 One-vs-One 4-27
4.13 Extension from binary.....4-28
4.13.1 Neural Networks 4-28
4.13.2 Extreme Learning Machines.....4-28
4.13.3 Naive Bayes.....4-28
4.14 Hierarchical classification 4-28
• **Chapter Ends** 4-28

4.1 INTRODUCTION

- Constraint optimisation also called as constrained optimisation is the process of optimising an objective function with respect to some variables. And these variables are constraint variables.
- The objective function is either a cost function or energy function, which is to be minimized OR a reward function, or utility function, which is to be maximised.
- Constraints can be either **hard constraints**, which set conditions for the variable that are required to be satisfied, or **soft constraints**, which have some variable values that are penalized in the objective function, if the conditions on the variables are not satisfied.

4.1.1 Relation to Constraint Satisfaction Problems

The constrained optimisation problem (COP) is a significant generalisation of the classis

Constraint-satisfaction problem (CSP) model

COP is a CSP that includes an **objective function** to be optimised. Many algorithms are used to handle the optimisation part.

4.1.2 General Form

A general constrained minimisation problem may be written as :

min f(x),

subject to : $g_i(x) = C_i$, for i = 1, 2, ... n equality constraints

$h_j(x) \geq d_j$ for j = 1, 2, ..., m

Where $g_i(x) = C_i$ and $h_j(x) \geq d_j$

for i = 1, 2, ..., n, and j = 1, 2,..., m are constraints that are to be satisfied and f(x) is the objective function that needs to be optimised subject to the constraints.

Note that, in some problems, the objective function is the sum of cost functions.

4.1.3 Solution Methods

(I) Equality constraints

- For equality constraints, substitution method is used.
- For a function of two variables subject of a single equality constraint, we use the method of substitution.

For example, assume the objective is to maximise : $f(x, y) = x \cdot y$ subject to $x + y = 10$

The constraint implies $y = 10 - x$, which can be substituted into the objective function, and we have

$p(x) = x(10 - x) = 10x - x^2$

The first order condition gives :

$$\frac{dp}{dx} = 10 - 2x = 0$$

$\therefore$ x = 5 and y = 10 – 5 = 5

Remark Lagrange Multiplier

For equality constraints, the method of Lagrange's multiplier can be used.

If the constraints are all linear, they can be solved for same of the variables in terms of the others, and the former can be substituted out of the objective function, and solution can be obtained.

(II) Inequality constraints

With inequality constraints, the problem can be characterised in terms of the **geometric optimality conditions.**

- **(A) Linear programming**

  If the objective function and all of the hard constraints are linear and some hard constraints are inequalities, then the problem is a **linear** programming problem. This can be solved by the **simplex method.**

- **(B) Non-linear programming**

  If the objective function or some of the constraints are nonlinear, and some constraints are inequalities, then the problem is a **nonlinear programming problem.**

- **(C) Quadratic programming**

  If all the hard constraints are linear and some are inequalities, but the objective function is quadratic, the problem is a quadratic programming problem. It is one type of nonlinear programming. It can be solved by ellipsoid method or by NP hard method.

- **(D) KKT conditions**

  Allowing inequality constraints, the KKT approach to nonlinear programming generalises the method of Lagrange multipliers. It can be applied under differentiability, and convexity.

(III) Constrained optimisation technique

Constrained optimisation is a **set of methods designed to identify efficiently** and systematically the best solution (the optimal solution) to a problem. It is characterised by a number of potential solutions in the presence of identical constraints.

(IV) Difference between constrained and unconstrained optimisation

In optimisation problem : Unconstrained simply means that the choice variable can take any value there are no restrictions.

Constrained means that the choice variable can only take on certain values within a larger range.

(V) Branch and Bound

- Constraint optimization can be solved by branch-and-bound algorithms. These are back tracking algorithms storing the cost of the best solution found during execution and using it to avoid part of the search.
- Precisely speaking, whenever the algorithm encounters a partial solution that cannot be extended to form a solution of letter cost than the stored best cost, the algorithm backtracks, instead of trying to extend this solution.
- Assuming that cost is to be minimized, the efficiency of these algorithms depends on how the cost can be obtained from the evaluated partial solution.
- If the algorithm can backtrack from a partial solution, part of the search is skipped. The lower the estimated cost, the better the algorithm, as a lower estimated cost is more likely to be lower than the best cost of solution found so far.
- On the other hand, this estimated cost cannot be lower than the effective cost that can be obtained by extending solution. Otherwise the algorithm can bracktrack while a solution better than the best found so far exists.
- Hence, the algorithm requires an upper bound on the cost that can be obtained from extending a partial solution, and this upper bound should be as small as possible.
- A variation of this approach is called Hansen's method. It uses **interval methods**. It implements rectangular constraints.

(VI) First-choice Bounding Functions

- One way to evaluate this upper bound for a partial solution is to consider each soft constraint separately.
- For each of constraint the maximal possible value for any assignment to the unassigned variables is assumed. The sum of these values is an upper bound. It is because the soft constraints cannot assume a higher valve.

- It is exact because the maximal values of soft constraints may derive from different equations : a soft constraint may be maximal for x = a while another constraint is maximal for x = b.

(VII) Russian Doll Search

- This method runs a branch-and-bound algorithm an n-problems, where n is the number of variables.
- Each such problem is the sub problem obtained by dropping a sequence of variables $x_1, x_2, \ldots, x_i$ from the original problem, along with the constraints containing them.
- After the problem on variables $x_{i+1}, \ldots, x_n$ is solved, its optimal cost can be used as an upper bound while solving the other problems.
- In particular, the cost estimate of a solution having $x_{i+1}, \ldots, x_n$ as unassigned variables is added to the cost that derives from the evaluated variables.
- This corresponds on ignoring the evaluated variables and solving the problem on the unassigned ones.
- More precisely, the cost of soft constraints containing both assigned and unassigned variables is estimated as above.
- The cost of soft constraints containing only unassigned variables is estimated using the optimal solution of the corresponding problem, which is already known at this point.
- There is similarity between the Russian Doll search method and **dynamic programming**.
- Like dynamic programming, Russian Doll search solved sub-problems in order to solve the whole problem. But, whereas dynamic programming directly combines the results the results obtained on sub-problems to get the result of the whole problem. Russian Doll search only uses them as bounds during its search.

(VIII) Bucket elimination

- **The bucket elimination algorithm** can be adapted for constraint optimization. A given variables can be removed from the problem by replacing all soft constraints containing it with a new soft constraint. The cost of this new constraint is computed assuming a maximal value for every value of the removed variable.
- If x is the variable to be removed, $C_1, C_2, \ldots, C_n$ are the soft constraints containing it, and $Y_1, Y_2, \ldots, Y_m$ are their variables except x, the new soft constraint is defined by,

$$C(Y_1 = a, \ldots, Y_n = a_n) = \max_a \sum_i C_i(x = a, y_1 = a_1, \ldots, Y_n = a_n)$$

- Bucket elimination with an ordering of the variables. Every variable is associated a bucket of constraints.
- Bucket elimination proceeds from the last variable to the first. For each variable, all constraints of the bucket are replaced as above to remove the variable. The resulting constraint is then placed in the appropriate bucket.

4.2 SUPERVISED LEARNING : SUPPORT VECTOR MACHINE

UQ. Define Support Vector Machine. Explain how margin is computed and optimal hyper-plane is decided ? **(MU - Dec. 19, 10 Marks)**

4.2.1 Optimal Decision Boundary

- A solution to the classification problem is a rule that partitions the features and assigns each and all the features and partition to the same class.
- The **'boundary'** of this partitioning is the decision boundary of the rule.
- **The boundary that this rule produces is** the optimal decision boundary.
- A decision boundary is the region of a problem space in which the output label of a **classifier** is **ambiguous**.
- If the decision surface is a hyperplane, then the classification problem is linear, and the classes are **linearly separable**.
- Decision boundaries are not always clear-cut. That is, the transition from one class in the **feature space** to another is not discontinuous, but gradual.

GQ. What is decision boundary in decision tree ?

- The first node of the tree called the "root node" contains the number of instances of all the classes respectively.
- Basically, we have to draw a line called "decision boundary" that separates the instances of different classes into different regions called "decision regions".

GQ. What is decision boundary ? How do you draw it and what is the advantage of it ?

A decision boundary is a line (in the case of two features), where all (or most) samples of one class are on one side of that line, and all samples of the other class are on one side of that line, and all samples of the other class are on the opposite side of the line. The line separates the two classes.

☞ Maximum Margin Linear Separators

UQ. What is SVM ? Explain the following terms: separating hyperplane, margin and support vectors with suitable example. **(MU - May 15, 4 Marks)**

UQ. What are the key terminologies of Support Vector Machine ? **(MU - May 16, 5 Marks)**

UQ. What is Support Vector Machine ? **(MU - May 17, Dec. 19, 4 Marks)**

UQ. Illustrate Support Vector machine with neat labeled sketch. **(MU - May 19, 4 Marks)**

(1) Support Vector Machine is a type of supervised learning that can be used for classification or regression. Even if the data points are unseen (not from the training dataset), support vector machine classifies the data properly.

(2) Let's take an example of dataset that belongs to two different categories, and the distribution of data is proper means the data is separated from each other properly.

Fig. 4.2.1 : Different types of data

(3) In this case we can draw a straight line (decision boundary) on the graph in such a way that the input space is divided in to two regions.

(4) Data points that belongs to one category lies on one side of the decision boundary and the data points of other category lies on the opposite side. Such type of data is called as linearly separable data.

(5) **Separating hyperplane** is the line which is used to separate the dataset. If we are using simple 2-dimensional plots then it's just a line. We require a plane to separate the data if data is 3 dimensional. So we can say that if data is N dimensional, we require N-1 dimensional hyperplane.

(6) We want that our classifier should be designed in a manner that if a data point is far away from the decision boundary then we will be more confident about the prediction we have made.

(7) We would like to find the data point near to the separating hyperplane and also make sure that this point should be far away from the separating line as possible. This is called as **margin**. We would like to find the greatest possible margin, because if we trained our classifier on limited data or made a mistake, we would want it to be as robust as possible.

Fig. 4.2.2 : Support vectors and Margin

(8) **Support vectors** are the points which are nearest to the separating hyperplane. We have to maximize the distance between the support vectors and the separating line.

Distance between a hyperplane (w, b) and a point x is calculated as,

$$\text{Distance} = \frac{|w^t x + b|}{\|w\|}$$

4.2.2 Quadratic Programming Solution to Find Maximum Margin Separator

UQ. Explain the term : Hyperplane with suitable example. (MU - May 15, 1 Marks)

UQ. Write detail notes on : Quadratic Programming solution for finding maximum margin separation in support vector machine (MU - May 16, 10 Mark)

UQ. How to compute the margin ? (MU - May 17, 6Marks)

UQ. Explain how margin is computed and optimal hyper-plane is decided ? (MU - Dec. 19, 6 Marks)

UQ. Show how to derive optimal hyper-Plane? (MU - May 19, 6 Marks)

(1) A **hyperplane** is formally defined by the following notation as,

$$F(x) = w^t x + b$$

In above equation, w represents the weight vector and b represents the bias.

(2) By scaling the values of w and b we can represent the optimal hyperplane in many ways. As a matter of convention among all the possible notations of the hyperplane the one selected is

$$|w^t x + b| = 1$$

(3) Here x represents the training records closest to the hyperplane. In general the training records that are closest to the hyperplane are called as support vectors. This notation is called as the canonical hyperplane.

(4) The distance between a point x and a hyperplane (w, b) is given by the result of geometry as follows,

$$\text{Distance} = \frac{|w^t x + b|}{\|w\|}$$

(5) In general, the numerator is equal to one for the canonical hyperplane and distance to the support vector is given as,

$$\text{Distance}_{sv} = \frac{|w^t x + b|}{\|w\|} = \frac{1}{\|w\|}$$

(6) Margin is twice the distance to nearest samples

$$M = \frac{2}{\|w\|}$$

(7) Ultimately, the task of maximizing M is same as compared to the task of minimizing a function L (w) subject to some conditions. The conditions used to model the requirements for correct classification of all training samples x_i by the hyperplane are formally stated as,

$$\min_{w,\,b} L(w) = \frac{1}{2}\|w\|^2 \text{ subject to } y_i\,(w^t x + b) \geq 1 \text{ for all i.}$$

Where y_i represents the labels of training.

Fig. 4.2.3 : Solution to find maximum margin

(8) This is a problem of Lagrangian optimization that can be solved using Lagrange multiplier to calculate weight vector 'w' and the bias 'b' of the optimal hyperplane.

(9) Let's assume that we have 2 classes of 2 dimensional data to separate. Let's also assume that each class consist of only one point.

These points are,

$$X_1 = A_1 = (3, 3) \; ; \qquad X_2 = B_1 = (6, 6)$$

Find the hyper plane that separates theses 2 classes

$$f(w) = \frac{1}{2}\|w\|^2$$

(10) The constraints are,

$$c_i(w, b) = y_i \mid wx_i + b \mid - 1 \geq 0$$
$$c_1(w, b) = 1 \mid wx_1 + b \mid - 1 \geq 0$$
$$c_2(w, b) = -1 \mid wx_2 + b \mid - 1 \geq 0$$

(11) Next, we put equation into form of Lagrangian

$$L(w, b, m) = f(w) - m_1 c_1(w, b) - m_2 c_2(w, b)$$
$$= \frac{1}{2}\|w\|^2 - m_1((wx_1 + b) - 1) - m_2(-(wx_2 + b) - 1)$$
$$= \frac{1}{2}\|w\|^2 - m_1((wx_1 + b) - 1) + m_2((wx_2 + b) + 1)$$

(12) We solve for the gradient of Lagrangian

$$\nabla L(w, b, m) = \nabla f(w) - m_1 \nabla c_1(w, b) + m_2 \nabla c_2(w, b) = 0$$

$$\frac{\partial}{\partial w} L(w, b, m) = w - m_1 x_1 + m_2 x_2 = 0 \quad \text{...(4.2.1)}$$

$$\frac{\partial}{\partial b} L(w, b, m) = -m_1 + m_2 = 0 \quad \text{...(4.2.2)}$$

$$\frac{\partial}{\partial \lambda_1} L(w, b, m) = (wx_1 + b) - 1 = 0 \quad \text{...(4.2.3)}$$

$$\frac{\partial}{\partial \lambda_2} L(w, b, m) = (wx_2 + b) + 1 = 0 \quad \text{...(4.2.4)}$$

(13) Equating Equation (4.2.3) and (4.2.4), we get

$$(wx_1 + b) - 1 = (wx_2 + b) + 1 \qquad \therefore (wx_1) - 1 = (wx_2) + 1$$
$$(wx_1) - (wx_2) = 2 \qquad \therefore w(x_1 - x_2) = 2$$

w is divided into parts as,

$$w = (w_1, w_2) \qquad \therefore W(x_1 - x_2) = 2$$
$$(w_1, w_2)[(3, 3) - (6, 6)] = 2 \qquad \therefore (w_1, w_2)[(-3, -3)] = 2$$
$$-3w_1 - 3w_2 = 2 \qquad \therefore w_1 = -(0.67 + w_2) \quad \text{...(4.2.5)}$$

(14) Adding values to Equation (4.2.1) and combining with Equation (4.5.2)

$$(w_1, w_2) - m_1(1, 1) + m_2(2, 2) = 0$$

From Equation (4.2.2)

$$m_1 = m_2; \qquad (w_1, w_2) - m_1(3, 3) + m\lambda_1(6, 6) = 0$$
$$(w_1, w_2) + m_1(3, 3) = 0; \qquad w_1 + 3m_1 = 0 \quad \text{...(4.2.6)}$$
$$w_2 + 3m_1 = 0 \quad \text{...(4.2.7)}$$

(15) Equating these we get, $w_1 = w_2$

Putting this in Equation (4.2.5)

$$w_1 = w_2 = -0.34$$

Putting this in either Equation (4.2.6) or Equation (4.2.7) will give

$$m_1 = m_2 = 0.11$$

(16) And finally, using this in Equation (4.2.3) and Equation (4.2.4)

$$b = 1 - (wx_1) \text{ or } = -1 - (w\,x_2) = 1 - ((-0.34, -0.34), (3, 3)) \text{ or}$$
$$= 1 - ((-0.34, -0.34), (6, 6)) = 3.04$$

4.2.3 Kernels for Learning Non-Linear Functions

- Linear classifiers are able to separate only linearly separable data. Support vector machine provides the solution to this problem by transforming an input space into a feature space that contains non linear features.
- A hyperplane is constructed in the feature space so that other equations remain the same. This is also known as non-linear support vector machine. Here we separate the data linearly using a high dimensional space.

Fig. 4.2.4 : Mapping of Input space to Feature space

- Kernel functions with its own set of variables are used for this purpose. The result is going to be non linear if we convert this back to the original feature space.

Particularly, the data is preprocessed with

$$X \rightarrow \Phi(x)$$

And then $\Phi(x)$ is mapped to y $\quad F(x) = w\,\Phi(x) + b$

- Kernel function transforms the data in to an easily understandable form. This is done via mapping input space to another feature space.
- In support vector machine inner product is calculated of two vectors and the result of this is always a single number. When we replace this inner product by a kernel it is called as kernel trick.
- There are different algorithms that use different kinds of kernel functions.
- Among the different types of kernel functions such as nonlinear, linear, radial basis function (RBF), polynomial, and sigmoid, RBF is mostly used. The reason for this is along the X-axis radial basis function gives the localized and finite response.
- The number of support vectors will be determined based on the different criteria's such as what is the complexity of the model, how much slack is allowed.
- One or more than one support vectors need to be defined for every complications in the final model from the input space.

Fig. 4.2.5 : Mapping of one feature space to another feature space

- Support vector machines output compromises of support vectors and alpha. This is used to specify the effect of support vectors on the final decision.
- If we select the model with high complexity it will result in to over fitting. For better generalization if large margin is selected then it may lead to incorrect classification.
- And accuracy depends on the trade-off between these two selections criteria. If we over fit the data then the range of support vectors may vary from very less to each single point. This tradeoff is controlled through the selection of kernel and its parameters.
- In support vector machine the data points are tested by taking the dot product of each support vector with the test point.
- Hence the computational complexity increases, if we increase the number of support vectors. Classification of test points will be faster if we have less number of support vectors.

4.2.4 Rules for the Kernel Function

Kernel function or a window is defined as follows:

If $\|\bar{x}\| \leq 1$ then $K(\bar{x}) = 1$ else 0.

This kernel function is shown by the Fig. 4.2.6,

$$K((z - x_i)/h) = 1$$

Fig. 4.2.6

For a fixed value of x_i, the function takes the value as 1 as shown in Fig. 4.2.7.

By selecting the argument of K (·), window can be moved to be centerd at the point x_i and to be of radius h.

Fig. 4.2.7

4.3 DIFFERENT TYPES OF SVM KERNELS

1. **Polynomial kernel**

Polynomial kernel is mostly used in image processing methods.

Polynomial Kernel is represented as,

$$K(x_j, x_k) = (x_j \cdot x_k + 1)^p$$

Here p represents the degree of the polynomial.

2. **Gaussian kernel**

There are some applications where prior knowledge is not available. For this type of applications Gaussian kernel is used.

Gaussian kernel is defined as,

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

3. **Gaussian radial basis function (RBF)**

This is also used for the applications where prior knowledge is not available.

Gaussian radial basis function is defined as,

$$K(x_i, x_j) = \exp(-\gamma \|x - y\|^2) \text{ for } \gamma > 0$$

Sometimes it is parametrized using the value of γ as $1/2\sigma^2$

4. **Laplace RBF kernel**

Laplace RBF kernel is defined as,

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

5. **Hyperbolic tangent kernel**

Hyperbolic tangent kernel is used in neural networks.

It is defined as,

$$K(x_i, x_j) = \tanh(kx_i * x_j + c), \text{ for some (not every) } k > 0 \text{ and } c < 0.$$

6. **Sigmoid kernel**

Sigmoid kernel can be used as a proxy for neural networks. It is defined as,

$$K(x, y) = \tanh(\alpha x^d y + c)$$

7. **Bessel function of the first kind Kernel**

Cross terms in mathematical functions can be removed by using this type of kernel function.

It is defined as,

$$K(x, y) = \frac{J_{v+1}(\sigma \| x - y \|)}{\| x - y \|^{-n(v+1)}}$$

Here j represents the Bessel function of first type.

8. **ANOVA radial basis kernel**

In regression problems this kernel can be used.

It is defined as,

$$K(x, y) = \sum_{k=1}^{n} \exp(-\sigma (x^k - y^k)^2)^d$$

4.4 SVM as CONSTRAINED OPTIMISATION PROBLEM

- Constrained optimisation problem are problems for which a function f(x) is to be minimized or maximized subject to the given constraints $\phi(x)$.
- Here $f : R^n \rightarrow R$ is called the objective function and $\phi(x)$ is a Boolean valued formula.
- The constraints $\phi(x)$ can be an arbitrary Boolean combination of equations $g(x) = 0$, weak inequalities $g(x) \geq 0$, strict inequalities $g(x) > 0$, and $x \in Z$ statements.
- The following notations are used :

  Min · f(x)
  s.t $\phi(x)$

- Stands for "minimize f(x) subject to constraints $\phi(x)$", and $\begin{array}{l}\max \cdot f(x)\\ \text{s.t} \quad \phi(x)\end{array}$ stands for "maximize f(x) subject to constraints $\phi(x)$".
- We say a point $u \in R^n$ satisfies the constraints ϕ it $\phi(u)$ is true.

4.4.1 Global Optimisation

- A point $u \in R^n$ is said to be a global minimum of f subject to constraints ϕ if u satisfies the constraints and for any point v that satisfies the constraints,

$$f(u) \leq f(v)$$

- A value $a \in R$ is said to be the global minimum valve of f subject to constraints and for any point v that satisfies the constraints, $a \leq f(v)$.
- The global minimum valve a exists for any f and ϕ. The global minimum value a is attained if there exists a point u such that $\phi(u)$ is true and $f(u) = a$.
- Such a point u is necessarily a global minimum.
- If f is a continuous function and the set of points satisfying the constraints ϕ is compact (i.e. closed and bounded) and non empty, then a global minimum exists. Otherwise a global minimum may or may not exist.

4.4.2 Examples

1. Minimise $[\{x, x^2 + y^2 < 1\}, (x, y)]$

   Here minimum value is not attained. The set of points satisfying the constraints is not closed :

   $\because \{-1, \{x \to -1, y \to 0\}\}$

2. Minimise $[\{x^2 + (y-1)^2 y > x^2\}, (x, y)]$

   $\{0, (x \to 0, y \to 1)\}$

4.4.3 Local Optimisation

- A point $u \in R^n$ is said to be a local minimum of f subject to constraints ϕ if u satisfies the constraints and for some $r > 0$, if v satisfies $|v - u| < r \wedge \phi(v)$, then $f(u) \leq f(v)$.
- **Remark** : A local minimum may not be a global minimum.
- A global minimum is always a local minimum.

Examples

Find a local minimum that is not a global minimum :

(i) $\{3x^4 - 28x^3 + 84x^2 - 96x + 42, (x, 1\text{-}1)\}$

Ans. : $\{5, (x \to 1)\}$

(ii) Minimise $[3x^4 - 28x^3 + 84x^2 - 96x + 42, \{x\}]$ $\{-22, (x \to 4)\}$

Fig. 4.4.1

4.4.4 Solving Optimisation Problems

- The methods used to solve local and global optimization problems depend on specific problem type.
- Optimization problems can be categorized according to several criterion.
- Depending on the type of functions involved there are linear and nonlinear (Polynomial, algebraic, transcendental,...) optimization problems.
- If the constraints involve $x \in Z$, we have integer and mixed integer-real optimization problems. Additionally optimization algorithms can be divided into numeric and symbolic (exact) algorithms.
- Functions for constrained optimization include minimize, maximize, N minimize and N maximize, for global constrained optimization, **Find minimum** for local constrained optimization and **Linear Optimisation** for efficient and direct access to linear optimization methods.
- We prepare table to briefly summarise each of the functions :

4.4.5 Decision Tree

Here, we draw a decision tree which will help us in deciding which optimisation function to be used :

Fig. 4.4.2

4.4.6 Constrained Optimization

GQ. What is constrained optimization technique ?

Constrained optimisation is a set of methods to identify efficiently and systematically the best solution (the optimal solution) to a problem.

The problem is characterised by a number of potential solutions in the presence of identified constraints.

GQ. The difference between constrained and unconstrained optimization.

(i) Unconstrained means that the choice variable can take on any value there are no restrictions.

(ii) Constrained means that the choice variables can only take on certain values within a large range.

GQ. What are the three common elements in a constrained optimization problem ?

Optimisation problems are classified according to the mathematical characteristics of the objective.

They are : **an objective function**, decision **variable**, and **constraints**.

When one wants to formulate an optimisation problem, it implies translating a 'actual-world' problem into the mathematical equations and variables which contain these three components.

Types of optimization

GQ. Types of optimization.

(i) Linear and quadratic programming problems

(ii) Quadratic constraints and conic optimisation problems

(iii) Integer and constraint programming problems

(iv) Smooth nonlinear optimisation problems.

(v) Non-smooth optimisation problems.

GQ. Top optimization methods in machine learning.

(i) Gradient descent. The gradient descent method is the most popular optimisation method.

(ii) Stochastic Gradient descent
(iii) Adaptive learning rate method
(iv) Conjugate gradient method
(v) Derivative-free optimisation
(vi) Zerothe order optimisation
(vii) For meta learning

Requirement of optimization

GQ. Requirement of optimization.

The goal of optimisation is to achieve the "best" design relative to a set of prioritised criteria or constraints. These include maximising factors such as productivity, strength, etc.

4.5 QUADRATIC PROGRAMMING

- Quadratic programming (QP) is the process of solving certain **mathematical optimization problems** involving **quadratic functions**.
- Here, we seek to optimize (minimize or maximize) a multivariate quadratic function subject to linear constraints on the variables.
- Quadratic programming is a type of **non-linear programming**. "Programming" in this context refers to a formal procedure for solving mathematical problems.
- Quadratic programming is also referred to as "Quadratic optimization"

4.5.1 Problem Formulation

The quadratic programming problem with n variables and m constraints can be formulated as follows :

Given

(i) a real-valued, n-dimensional vector C,

(ii) an $n \times n$ dimensional real symmetric matrix Q,

(iii) an $m \times n$ dimensional real vector b, the objective of quadratic programming is to find an n-dimensional vector X, that will : minimise $\frac{1}{2} X^T QX + C^T X$

Subject to $AX \leq b$,

where X^T denotes the **vector transpose** of X, and the notation $AX \leq b$ means that every entry of the vector AX is less than or equal to the corresponding entry of the vector b (i.e., component-wise inequality).

4.5.2 Equality Constraints

Quadratic programming is particularly simple when Q is positive definite and there are only equality constraints, and then the solution process is linear.

Using Lagrange multipliers and seeking the extremum of the Lagrangian, it can be shown that the solution to the equality constraint problem :

Minimise : $\frac{1}{2} X^T QX + C^T X$

Subject to : $EX = d$

is given by the linear system

$$\begin{bmatrix} Q & E^T \\ E & O \end{bmatrix} \begin{bmatrix} X \\ \lambda \end{bmatrix} = \begin{bmatrix} -c \\ d \end{bmatrix}$$

Where λ is a set of language multipliers which come out of the solution with X.

The problem can be solved by using LU factorisation.

GQ. How are quadratics used in programming ?

- QP is widely used in image and signal processing, to optimise financial portfolios, to perform the least squares method of regression, to control scheduling in chemical plants, and in sequential quadratic programming, a technique for solving more complex non-linear programming problems.
- A quadratic programming (QP) problem has a quadratic cost function and linear constraints. Such problems are encountered in many real-world applications.
- In addition, many general nonlinear programming algorithms require solution of a quadratic programming sub problem at each iteration.

GQ. Is quadratic programming convex ?

The quadratic objective function may be convex – which makes the problem easy to solve-or non-convex, which makes it very difficult to solve.

The "best" QPs are positive definite (in a minimisation problem) or negative definite (in a maximisation problem).

4.6 TYPES OF SUPPORT VECTOR MACHINES : (SVM)

SVM are of two types

(I) Linear SVM

Linear SVM is used for linearly separable data. If a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data. The classifier that is used is called **Linear SVM classifier**.

(II) Non-linear SVM

Non-linear SVM is used for non-linearly separated data. If a dataset cannot be classified by using a straight line, then such data is termed as non-linear data. The classifier that is used is called **Non-linear SVM Classifier**.

Hyperplane and support vector in the SVM algorithm

Hyperplane : There can be multiple lines or decision boundaries to segregate the classes in n-dimensional space. Our aim is to locate the best decision boundary that helps to classify the data points. This best decision boundary is known as the hyperplane of SVM. The dimensions of the hyperplane depend on the features present in the dataset.

If there are 2 features, then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimensional plane. A hyperplane is created which has a maximum margin, and it means that there is maximum distance between the data points.

Support vectors

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as support vector. Since these vectors support the hyporplane, hence called a support vector.

4.6.1 Working of Linear SVM

We consider example to understand how SVM algorithm works. Suppose we have a dataset having two tgs red and blue, and the dataset has two features X_1 and X_2. We want a classifier that can classify the pair (X_1, X_2) of coordinates.

Since it is a 2-d space so by just using a straight line, we can separate the two classes. But there can be many lines which can separate these classes.

Fig. 4.6.1

Consider the following image

We want to find best line or decision boundary. This best boundary or region is called a hyperplane. SVM algorithm finds the closet point of the lines from both the classes. These points are called as support vectors.

The distance between the vectors and the hyperplane is called as **margin**. The goal of SVM is to maximise this margin. The hyperplane with maximum margin is called the **optimal hyperplane**.

Fig. 4.6.2

Fig. 4.6.3

4.6.2 Non-linear SVM

If data is linear, we can separate it by using a straight line, but if data is non-linear, then we cannot draw a single straight line.

So to separate these data points, we need one more dimension.

For linear data, we have used two dimensions x and y, so for non-linear data, we add a third dimension Z. It can be calculated as (say), $Z = x^2 + y^2$.

After adding the third dimenension, the sample space becomes :

Fig. 4.6.4

Fig. 4.6.5

Now, SVM will divide the datasets into classes in the following way. Refer Fig. 4.6.6

Fig. 4.6.6

Fig. 4.6.7

If we put Z = 1, then it will become as : Refer Fig. 4.6.7

$x^2 + y^2 = 1$

We get a circumference of radius 1 as hyperplane-surface in case of non-linear data.

4.6.3 Linear Classifier

- In the field of machine learning, the goal of **statistical classification** is to use an object's characteristics to identify which class it belongs to. A linear classifier makes a classification decision. And it is based on the value of a **linear** combination of the characteristics. An object's characteristics are known as **feature values**.
- They are presented to the machine in a vector called a **feature-vector**. Such classifiers work for problems with many variables (features) and reach accuracy levels comparable to non-linear classifiers and take less time to train and use.

4.7 DEFINITION OF LINEAR CLASSIFICATION

(1) If the input feature vector to the classifier is a real vector $\bar{x}$, then the output score is

$$y = f(\bar{w} \cdot \bar{x}) = f\left(\sum_j w_j x_j\right),$$

Where $\bar{w}$ is a real vector of weights and function f converts the dot product of the two vectors into the desired output. The weight vector $\bar{w}$ is decided from a set of labelled training samples. Generally, f is **a threshold function** : e.g.

$$f(x) = \begin{cases} 1, & \text{if } w^T \cdot x > \theta \\ 0, & \text{otherwise} \end{cases}$$

(2) The superscript T indicates the transpose and θ is a scalar threshold. For a two-class classification problem, the operation of a linear classifier as splitting a **high-dimensional** input space with a hyperplane : all points on one side of the hyperplane are classified as "yes" while the others are classified as "no".

(3) A linear classifier is the fastest classifier, when $\bar{x}$ is sparse. Linear classifiers also work well when the number of dimensions in $\bar{x}$ is large, as in document classification.

4.7.1 Classification V/S Regression

(1) Classification is about predicting a label and regression is about predicting a quantity; i.e. continuous value such as price, salary, age etc.

(2) Classification algorithms are used to predict the discrete values such as Male, Female, True or False Spam or Not Spam, etc.

4.7.2 Classification

(1) Classification is a process of finding a function which helps in dividing the dataset into classes based on different parameters. In classification, a computer program is trained on the training dataset. Based on that training it categorises the data into different classes.

(2) The task of the classification algorithm is to find the mapping function. If maps the input (x) to the discrete output (y).

4.8 NON-LINEAR FUNCTIONS

A nonlinear function is a function whose graph is not a straight line. Its graph can be any curve other than a straight line.

(1) **For example,** if there are 100 fishes in a pond initially. They become double every week, then the function of this situation will be

$f(x) = 100(2)^x$, where x is the number of weeks and f(x) is the number of fishes.

We prepare a table.

x	y
0	100
1	200
2	400
3	800

Fig. 4.8.1

(2) **For example, Which of the following functions are non-linear?**

(i) $f(x) = 4$, (ii) $f(x) = 3^x - 4$, (iii) $f(x) = \cos x$

Soln.

(i) $f(x) = 4 = 0.x + 4$; it is of the form $f(x) = ax + b$; it is linear.

(ii) $f(x) = 3^x - 4$ is an exponential function and is liners.

(iii) $f(x) = \cos x$ is cosine curve, it is periodic with period 2π.

It is non-linear function.

(3) **Triangulation of GPS signals**

A device like cellphone receives signals from GPS satellites, which have known orbital positions around the earth, a signal from a single satellite allows a cellphone to know that it is somewhere on a circle.

Additional signals are additional circles that intersect each other, and the cellphone's actual position is at the intersection.

Three or more signals reduce the solution of the system to a single coordinate point.

Fig. 4.8.2

(4) Non-linear Separable

Here the function is

$$f\left(\bar{x}, \bar{w}, \bar{b}\right) = \sum_j w_j x_j + b$$

x : input vector,

w : weight matrix

b : bias vector

Fig. 4.8.3

4.8.1 Support Vector Machines : (SVM)

Support vector machine (SVM) is a supervised machine learning algorithm. It is capable of performing classification, regression and even outlier detection.

Types of Support Vector Machine

1. Linear SVM
2. Non- linear SVM
3. Use of Dot Product in SVM
4. Polynomial kernel
5. Sigmoid kernel
6. RBF kernel
7. Bessel function kernel
8. Anova kernel

Support Vector Regression

Support vector regression (SVR) uses the same principle as SVM, but for regression problems. The problem of regression is to find a function that approximates mapping from an input domain to real numbers on the basis of a training sample.

We consider the two **red lines** as the decision boundary and the blue line as the hyperplane. When we move on with SVR, our objective is to consider the points that are within the decision boundary line. The best fit line is the hyperplane that has a maximum number of points.

Fig. 4.8.4

We first define decision boundary. Consider these red lines at a distance 'a' from the hyperplane; i.e. these are the lines at distance '+a' and '–a' from the hyperplane [Generally, we take 'a' as epsilon].

Let the equation of the hyperplane be

$y = mn + b$; then the equation of decision boundary become

$$wx + b = +a$$

and $$wx + b = -a$$

Thus, any hyperplane that satisfies our SVR is

$$-a < y = mx + b < +a.$$

Here, we are going to consider only those points that are within the decision boundary and they have least error rate, or they are within the margin of tolerance. This gives a better fitting model.

4.9 SVM KERNEL TRICK

Before we proceed to SVM kernel trick, let us first go over few basic concepts :

1. Support Vector Machine (SVM) and support vectors.
2. Linearity, Non-linearity, Dimensions and planes.
3. Kernel and kernel methods.

- A Suppor Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification and regression problems.
- Generally it is used for classification problem.
- SVM constructs a line or a hyperplane in a high or infinite dimensional space which is used for classification, regression, or outlier detection.
- SVM makes sure that the data is separated with the widest margin.

Fig. 4.9.1: Data Split

SVMs maximise the margin around the separating hyperplane.

Fig. 4.9.2 : Distance between support vectors and the line

Support vectors

Support vectors are the data points that lie farthest to the decision surface (or hyperplane). These data points are most difficult to classify.

They have a direct impact on the optimal location of the decision surface.

4.9.1 Linearity and Non-linearity

- On a plane, a linear function is a function where the graph is a straight line. The line can go in any direction, but it is always a straight line.
- But a non-linear function has a shape that is not a straight line.

Fig. 4.9.3

4.9.2 Dimensions

A dimension of something is a particular aspect of it.

Examples

Width, depth and height are dimensions.

A line on a plane is one dimension. Considering the edges of a square has two dimensions and a cube has three dimensions.

4.9.3 Planes and Hyperplane

- In one dimension, a hyperplane is called a point.
- In two dimensions, it is a line.
- In three dimensions, it is a plane and in more dimensions, we call it as a hyperplane.

4.9.4 Kernel

- A Kernel is a method of placing a two dimensional plane into a higher dimensional space, so that it is curves in the higher dimensional space.
- In simple terms, a kernel is a function from the low dimensional space into a higher dimensional space. Thus we come to know that **SVM** works better with **two dimensional space** which are **linearly separable.** But for a non-linear data, SVM finds it difficult to classify the data.
- The easy solution here is to use the **Kernel Trick.**

4.9.5 Kernel Trick

- A **Kernel trick** is a simple method where a **Non linear** data is projected onto a higher-dimensional space so as to make it easier to classify the data. Then it is linearly divided by a plane.
- This is mathematically achieved by Lagrangian formula using Lagrangian multipliers.

4.9.6 Applications of Kernel Trick

There are various applications of Kernel trick and few of them are :

(i) Geostatistics

- It is a branch of statistics concentrating on spatial or spatiotemporal datasets.
- It was originally created to predict the probability distributions of ore grading at mining operations. Now it is applied in diverse disciplines including petroleum geology, hydrogeology, hydrology, meteorology, oceanography, geochemistry, geometallurgy, geography, forestry, environmental control, landscape ecology, soil science, and agriculture (specifically in precision farming)

(ii) Inverse distance weighting

- Type of deterministic method for multivariate interpolation with a known scattered set of points.
- The values assigned to unknown points are calculated with a weighted average of the values existing at the known points.

(iii) 3D reconstruction

An interdisciplinary field that involves molecular biology, genetics, computer science, mathematics and statistics. Software tools and methods are develop to understand biological data better.

(iv) Chemoinformatics

Application of computational and informational techniques over the field of chemistry to solve a wide range of problems.

(v) Information extraction

It is referred as IE. It is a method of automated extraction or retrieval of structured information from an unstructured and semi-structured text documents, databases and websites.

(vi) Handwriting recognition

It is referred to as HWR. It is the ability of a computer system to receive and interpret the handwritten input comprehensibly from different sources such as letter paper documents, photographs, touch-screens and other devices.

4.9.7 The Kernel Trick (Mathematically)

Our basic idea of **Kernel Trick** is to find the plane which can separate, classify or split the data with maximum margin is also called **street width.**

The distance from the point (x, y) to a line $Ax + By + C = 0$ is $\frac{|Ax + By + C|}{\sqrt{A^2 + B^2}}$

In order to maximise the margin, The distance between H_0 and H_1 is then $\frac{|w \cdot x + b|}{\|W\|} = \frac{1}{\|W\|}$, so the total distance between H_1 and H_2 is $\frac{2}{\|W\|}$.

Kernel Trick Mathematically

- In order to maximise the margin, we need to minimise $\|W\|$. Thus we are trying to optimise the margin or street width by maximising the distance by maximising the distance between support vectors.
- An optimisation problem consists of either maximising or minimising a real valued function by choosing input values from within an allowed set and computing the value of the function.
- Also we achieve the same by keeping in mind that the support vectors should be away from the street and not on or in between the street. Hence we can say that this is a typical constrained optimisation problem or situation. This can be solved by Lagrangian formula or Lagrangian multipliers.

Fig. 4.9.4

4.9.8 Few Popular Kernels

The most tricky and important part of using **'Kernel Trick'** is to choose the right Kernel function because it is very challenging to visualise the data in n-dimensional space.

Few popular kernels are :

(i) **Fisher kernel :** It is a kernel function that analyses and measures the similarity of two objects. This is done on the basis of sets of measurements for each object and a statistical model.

(ii) **Graph kernel :** It is a kernel function that computer an inner product on graphs.

(iii) **Polymomial kernel :** It is a kernel commonly used with Suppor Vector Machines (SVMs). It is also used with other kernalised models that symbolises the similarity of vectors in a feature space over polynomials of the original variables, allowing learning of non-linear models.

(iv) **Radial Basis Function Kernel (RBF) :** It is a real-valued kernel function whose value depends only on the distance from the origin, or distance from some other point is called a centre.

Purpose of Kernel Trick

- Kernel trick offers a more efficient and less expensive way to transform data into higher dimensions.
- Application of the kernel trick is not limited to the SVM algorithm. Any computations involving the dot product (x, y) can utilise the kernel trick.
- Kernel trick transforms the problem from supervised to unsupervised learning.

- Kernel trick is used to reduce the computational complexity, because the model construction is performed on the kernel matrix instead of the initial data table.
- The most typical kernels include polynomial and radial basis functions.
- Kernel trick allows the inner product of mapping function instead of the data points. The trick is to identify the kernel functions which can be represented in place of the inner product of mapping functions.

4.10 SUPPORT VECTOR REGRESSION

- **Regression Analysis** is a statistical procedure that is used to identify the degree and type of a connection between one dependent variable and a set of other factors, known as independent variables.
- The goal of most **linear regression** models is to reduce the sum of squared errors.
- For example, Ordinary Least Squares (OLS) is a form of linear least squares approach used in statistics to estimate the unknown parameters in a linear regression model.
- **Lasso, Ridge and Elastic Net** are all OLS extensions with an extra penalty parameter that seeks to reduce the amount of features utilised in the final model, while minising complexity.
- The purpose is to lower the error of the test set, as with many models.
- The actual fact is : 'A support-vector regression fits' distinguishes it from other forms of regressions. The regression line or gradient in linear and polynomial regressions follows the path chosen by the user.
- A linear regression always follows the equation ($y = mx + C$), but a polynomial regression follows the equation ($y = mx^n + C$). But these cannot determine the optimum regression equation, which a support vector regression can do in short.
- It chooses the best fitting model equation mathematically and apply it to the model.

4.10.1 Support Vector Regression

- A component of support vector machines is support vector regression. Thus there is a notion known as support vector machine, which can be used to analyse both regression and classification data.
- When the support vector machine is used for classification, it is referred to as support vector classification, and when it is used for regression, it is referred to as support vector regression.
- The Support Vector Regression (SVR) uses the same ideas as the SVM for classification, with a few small differences. A margin of tolerance ($\in$ = epsilon) is supplied in the case of regression as an approximate estimate to the SVM. Apart from that, there is a more challenging reason : the algorithm is more complex and that it must be considered. But the basic idea remains the same : to reduce the error by customising the hyperplane to maximise the margin. The introduction of a insensitive zone around the function, known as the tube, allows SVM to be generalized to SVR.
- The optimisation problem is reformulated in this tube to discover the tube that best approximates the continuous valued function while balancing model complexity and prediction error. Thus, SVR is defined as an **optimisation** problem by first constructing a convex-insensitive loss function to be reduced and then determining the flattest tube that includes the majority of the training cases.
- As a result, the **loss function** and the geometrical parameters of the tube are combined to form a multiobjective function. Then, using suitable numerical optimisation methods, the convex optimisation, which has a unique solution, is solved. Support vectors, which are training samples that fall outside the tube's perimeter, are used to represent the hyperplane.
- In a supervised learning environment, the support vectors are the most influential cases, that determine the form of the tube, and the training and test data are assumed to be independent and identically distributed (iid), obtained from the same fixed but unknown **probability data distribution** function.

4.10.2 Important Terminologies in Support Vector Regression

Kernel, Hyper plane, boundary line, support vector are important terminologies in SVR.

1. Kernel

- The function for converting a lower-dimensional data set to a higher dimensional data set.
- A kernel aids in the search for a hyperplane in higher dimensional space while reducing the computing cost.
- When the size of the data grows larger, the computing cost usually rises. When we are unable to identify a separating hyperplane in a particular dimension and must shift to a higher dimension, this increase in dimension is necessary.

2. Hyper plane

This is the separating line between the data classes in SVM. Although, in SVR, we describe it as a line, that will assist us in predicting a continuous value or goal value.

3. Boundary line

- Other than hyper plane, there are two lines in SVM that produce a margin. The support vectors might be within or outside the boundary line. The two classes are separated by this line.
- The premise is same in SVR. A decision boundary line can be conceived of as a demarcation line (for simplicity), with positive examples on one side and negative examples on the other.
- The instances as this line can be characterized as either good or negative. The same SVM approach will be used in support vector regression as well.

4. Support vectors

- The data points closest to the border are listed here. The distance between the locations is little or negligible. Support vectors are locations that are outside the tube in SVR.
- The smaller the value of, the more points outside the tube there are, and hence more support vectors there are.

5. Conclusion

- SVR really proves to be better than deep learning methods in cases of limited datasets and also require much less time than its counter part.
- In comparison with other regression algorithms, SVR uses much less computation and has high accuracy and credibility.

4.10.3 Support Vector Regression

(1) What is support vector regression good for ?

SVR gives the flexibility to define how much error is acceptable in our model and will find an appropriate line (or hyperplane in higher dimensions) to fit the data.

(2) What is SVR algorithm in machine learning ?

Supervised machine learning models with associated learning algorithms that analyse data for classification and regression analysis are known as support vector regression.

SVR built is based on the concept of Support Vector Machine or SVM.

(3) When should one use SVR ?

In general, one can use SVR to solve the same problems one would use linear regression for. Unlike linear regression, though, SVR also allows one to model non-linear relationships between variables and provides the flexibility to adjust the models robustness by tuning hyper-parameters.

(4) Is SVR a parametric model ?

Neural networks and support vector regression (SVR) are two universal approximates. They are data-driven non-parametric models.

4.11 MULTICLASS CLASSIFICATION

Multiclass classification

- In machine learning, multiclass or multinomial classification is the problem of classifying instances into one of the three or more classes (classifying instances into one of two classes is called **binary classification**).
- Many classification algorithms, especially multinomial logistic regression, permit the use of more than two classes, some are by nature binary algorithms, and these can be turned into multinomial classifiers by a variety of strategies.

Remark

- Multiclass classification should not be confused with multi-label classification, where multiple labels are to be predicted for each instance.
- Our aim here is, how to evaluate multi-class performance, and how to build multiclass models out of binary models.
- The latter is necessary for some models, such as linear classifiers, that are primarily designed to separate two classes. Other models, like decision trees, handle any number of classes naturally.
- If we have k-classes, performance of a classifier can be assessed using a k-by-k contingency table.
- Consider the following three-class confusion matrix (plus marginal's)

	Predicted			
Actual	15	2	3	20
	7	15	8	30
	2	8	45	50
	24	20	56	100

The accuracy of this classier is $\frac{(15+15+45)}{100} = 0.75$ (comider maximum valve)

Now, we can calculate per-class precision and note : for first class this is $\frac{15}{24} = 0.63$, and $\frac{15}{20} = 0.75$

For the second class $\frac{15}{20} = 0.75$ and $\frac{15}{30} = 0.50$ and for the third class $\frac{45}{56} = 0.80$ and $\frac{45}{50} = 0.90$.

We can average these number to obtain single precision and take the number for the whol classifier.

4.11.1 Popular Algorithms for Multiclass Classification

Popular algorithms that can be used for multi-class classification include :

(i) K-nearest neighbours (ii) Decision trees

(iii) Naive bayes (iv) Random forest

(v) Gradient boosting

4.11.2 Method of Multi-class Classification Approach

(i) Load dataset from the source.

(ii) Load the dataset into 'training' and 'test' data.

(iii) Train decision tree, SVM, and KNN classifiers on the training data.

(iv) Use the above classifiers to predict labels for the test data.

(v) Measure accuracy and visualise classification.

4.11.3 The Multiclass Data

In machine learning, multiclass or multinomial classification is the problem of classifying instances into one of three or more classes (classifying instances into one of two classes is called binary classification)

4.11.4 Applying Decision Tree to the Multiclass Data

- To apply decision tree is to assign a unique integer to each of your classes.
- All examples of class one will be assigned the valve y = 1, all the examples of class 2 will be assigned to valve y = 2, etc.
- After this decision classification tree can be trained.
- The tree then splits the observations according to the answer. There can be many terminal nodes leading to different 'classes' or even many nodes leading to the same class.
- For each split node, a best at tribute is to be chosen and a best split point (or continuous attribute), then impurity of left and right child node is to be computed.
- The child impurity could be computed based on gini index.

4.11.5 KNN for Multi-class Classification

The main advantage of KNN over other algorithms is that KNN can be used for multiclass classification. Therefore if the data consists of more than two labels or to classify the data into more than two categories then KNN can be a suitable algorithm.

4.11.6 KNN and SVM Comparison

- SVM takes care of outliers better than KNN.
- If training data is much larger than number of features (m >> h), KNN is better than SVM.
- SVM outperforms KNN when there are large features and lesser training data.
- The KNN or K-nearest neighbours called is called "It is because it does no training at "lazy". It is because it does no training at all when you supply the training data.
- At training time, all it is doing is storing the complete data set but it does not do any calculations at this point.

4.11.7 KNN and Decision Tree

- KNN and decision tree both can be used for regression and classification problems.
- Decision tree supports automatic feature interaction, whereas KNN does not.
- Decision tree can be faster, but KNN tends to be slower with large dataset,
- Both are non-parametric methods.
- The comparative analysis results indicate that the KNN classifier outperforms the result of the decision-tree classifier in the breast-cancer classification.

4.11.8 KNN and Random Forest

- Decision tree completes instantly with 85% accuracy.
- Random forest with 94% accuracy with very less running time and KNN with 96% accuracy with considerable running time and occupying the resources all along.

4.12 CATEGORISATION OF MULTI-CLASS

The multi-class classification techniques can be categorised into :

(i) transformation to binary

(ii) extension from binary

(iii) hierarchical classification

4.12.1 Transformation to Binary

Reducing the problem of multiclass classification to multiple binary classification can be categorised into (i) one v/s rest and (ii) one v/s one.

4.12.2 One-vs-rest

One-vs-rest (OvR or one-vs-all, OVA, or one-against-all, OAA) strategy involves training a single classifier per class, with samples as that class as positive samples and all other samples as negatives.

The training algorithm from a binary classification learner L is as follows :

Inputs

(i) L, a learner (training algorithm for binary classifiers)

(ii) Samples X

(iii) Labels y where $y_i \in (1, \ldots, K)$ is the label for the sample X_i

Output

A list of classifiers f_k for $k \in \{1, \ldots K)$

Procedure

For each K in $\{1, \ldots, K\}$

(a) Construct a new label vector z where $z_i = y_i$ if $y_i = K$ and $z_i = 0$, otherwise

(b) Apply L to X, z to obtain f_k.

4.12.3 One-vs-One

- In the one-vs-one (OvO) reduction, one trains $\frac{K(K-1)}{2}$ binary classifiers for a K-way multiclass problem, each receives the samples of a pair of classes from the original training set, and must learn to distinguish these two classes.
- At prediction time, a voting scheme is applied : All $\frac{K(K-1)}{2}$ classifiers are applied to an unseen sample and the class that got the highest number of '+1' predictions gets predicted by the combined classifier.

4.13 EXTENSION FROM BINARY

Here, existing binary classifiers are extended to solve multi-class classification problems.

Several algorithms are developed based on **Neural Networks, Decision trees,** K-nearest neighbours, naive Bayes, support vector machines and extreme learning machines to address multi-class classification problems.

4.13.1 Neural Networks

- Multiclass perceptrons provide a natural extension to the multi-class problem.
- Instead of having one neuron in the output layer, with binary output, we can have N binary neurons leading to multi-class classification.

4.13.2 Extreme Learning Machines

- Extreme Learning Machines (ELM) is a special case of single hidden layer Feed-Forward Neural Networks (SLFNS), where the input weights of hidden node biases can be chosen at random.
- Many variants and developments are made to the ELM for multiclass classification.

4.13.3 Naive Bayes

Naive Bayes is a successful classifier based upon the principle of maximum a posterior (MAP). This approach is naturally extensible to the case of having more than two classes; and it perform well inspite of the underlying simplifying assumption of **conditional independence.**

4.14 HIERARCHICAL CLASSIFICATION

- **Hierarchical classification** tackles the multi-class classification problem by dividing the output space into a tree.
- Each parent node is divided into multiple child nodes and the process is continued until each child node represents only one class.
- In the field of machine learning, hierarchical classification is referred to as **instance space decomposition.** It splits a complete **multi-class** problem into a set of smaller classification problems.
- A hierarchy is an organisational structure in which terms are ranked according to levels of importance.
- In a computing content, there are various types of hierarchical systems.
- Most **file systems,** for example, are based on a hierarchical model in which files are placed somewhere in a **hierarchical tree model.**
- A **file** is placed in a directory (folder in windows) or subdirectory at the desired place in the tree structure.
- The computer **memory** hierarchy ranks components in terms of response times, with **processor registers** at the top of the pyramid structure and **tape backup** at the bottom.
- Hierarchical classification is also called as taxonomical classification where organisms are classified into successive levels according to the complexity. Each level is called as taxonomic level.
- In this classification, kingdom is highest ranked followed by division, class, order, family, genus, and species is the lowest category.

Chapter Ends...

□□□

MODULE 5

CHAPTER 5 Learning with Clustering

Syllabus

Introduction to clustering with overview of distance metrics and major clustering approaches.

Graph Based Clustering: Clustering with minimal spanning tree, Model based Clustering: Expectation Maximization Algorithm, Density Based Clustering: DBSCAN.

5.1 Introduction to Clustering 5-3
5.1.1 Building the Similarity Graph 5-3
5.1.2 Projecting the Data onto a Lower Dimensional Space 5-3
5.1.3 K-means and Spectral Clustering 5-4
5.2 Evaluation methods based on Ground Truth Homogeneity 5-4
5.2.1 Trivial Homogeneity 5-4
5.2.2 Remark 5-4
5.2.3 Remarks 5-5
5.3 Adjusted Rand Index 5-6
5.4 Adjusted Rand Index 5-7
5.4.1 The Contingency Table 5-7
5.4.2 Definition 5-7
5.5 Concepts of Weak and Eager Learner 5-7
5.5.1 Example of Lazy Learner 5-8
5.6 Unsupervised Learning : Hierarchical Clustering 5-8
5.6.1 Hierarchical Clustering 5-8
5.6.2 Examples on Hierarchical Clustering 5-9
UEx. 5.6.3 (MU - May 16. 10 Marks) 5-14
UEx. 5.6.4 (MU - May 17. 10 Marks) 5-15
5.7 Unsupervised Learning : Association Rules 5-16
5.7.1 Introduction to Association Rules 5-16
5.7.2 Apriori Algorithm 5-16
5.7.3 Performance Measures 5-16

5.7.4 Issues in Machine Learning 5-19
UQ. What are the issues in Machine learning ? (MU - May 15, 5 Marks) 5-19
5.7.5 How to choose the right algorithm ? 5-19
UQ. Explain the steps required for selecting the right machine learning algorithm. (MU - May 16, 8 Marks) 5-19
5.7.6 Steps in Developing a Machine Learning Application 5-20
UQ. Explain the steps of developing Machine Learning applications. (MU - May 19, 10 Marks) 5-20
5.7.7 Applications of Machine Learning 5-21
UQ. Write short note on : Machine learning applications. (MU - May 16, May 17, 10 Marks) 5-21
5.8 Graph Based Clustering 5-22
5.8.1 Graph Clustering Algorithm 5-22
5.8.2 Method of Graph-based Clustering 5-22
5.9 Clustering with Minimal Spanning Tree 5-25
5.9.1 The Cluster Center-Initialisation Method 5-25
5.9.2 Definition : Geodesic Distance 5-26
5.9.3 Inconsistent Edge 5-27
5.9.4 Internal Clustering Validation Index 5-27
5.10 Model Based Clustering 5-27
5.10.1 Use of EM 5-28
5.10.2 EM Algorithm 5-28
GQ. Write short note on : EM algorithm. **(5 Marks)** 5-28
5.10.3 Advantages of EM Algorithm 5-28
5.10.4 Applications of EM-Algorithm 5-28
5.11 Expectation Maximisation Algorithm 5-29
5.11.1 Latent Variable Model 5-29
5.11.2 Use of Latent Variable Model 5-29
5.11.3 Use-Case of EM Algorithm 5-29
5.11.4 Multivariate Gaussian Model 5-29
5.11.5 Gaussian Mixture Models 5-29
5.12 Density Based Clustering 5-30
5.12.1 Irregularities in the Data 5-30
5.12.2 The basic Principle of DBSCAN Clustering 5-31
5.12.3 DBSCAN and K-means 5-31
- **Chapter Ends** 5-31

5.1 INTRODUCTION TO CLUSTERING

Spectral Clustering

- Spectral clustering is an algorithm which performs better than many traditional clustering algorithms in many cases.
- It treats each **data point as a graph node** and thus transforms the clustering problem into a graph partitioning problem.
- Implementation of spectral clustering consists of three fundamental steps.

5.1.1 Building the Similarity Graph

- This step builds the similarity graph in the form of an adjacency matrix which is represented by A.
- The adjacency matrix A can be constructed in the following manner.

The adjacency matrix A can be constructed in the following manner :

(i) Epsilon-neighbourhood Graph

- First a parameter epsilon is fixed. Then every point is connected to all the points which lie in its epsilon-radius.
- It all the distances between any two points are similar in scale then the distance between two points are not stored (i.e. weights of the edges)
- It is because they do not provide any additional information. In this case, the graph drawn or built is an undirected and weighted graph.

(ii) K-Nearest Neighbours

- Again, we fix the parameter k. Then for two vertices u and v, an edge is directed form u to v only if v is among the k-nearest neighbours of u. We follow the following approaches :
  - (a) Direct an edge from u to v and from v to u if either v is among the k-nearest neighbours of u **OR** u is among the k-nearest neighbours of v.
  - (b) An edge is to be directed from u to v **and** from v to u if v is among the k-nearest neighbours of u **AND** u is among the k-nearest neighbours of u **AND** u is among the k-nearest neighbours of v.

(iii) Fully-connected Graph

- Here, each point is connected with an undirected edge-weighted by the distance between the two points to every other point.
- Here, we use Gaussian similarity metric is used, to calculate the distance.

5.1.2 Projecting the Data onto a Lower Dimensional Space

- There is a possibility that members of the same cluster may be far away in the given dimensional space.
- Thus the dimensional space is reduced so that those points are closer in the reduced dimensional space and can be clustered together by a traditional clustering algorithm.
- It is carried out by **Graph Laplacian Matrix.**
- To compute it, we define the degree of a node.
- The degree of ith node is given by,

$$d_i = \sum_{j=1}^{n} (i, j) \in E^{wij}$$

Note that w_{ij} is the edge between the nodes i and j as defined in the adjacency matrix.

The degree matrix is defined as :

$$D_{ij} = \begin{cases} d_i & , \ i = j \\ 0 & , \ i \neq j \end{cases}$$

Then the Graph Laplacian matrix is defined as

$$L = D - A$$

- This matrix is then normalized.
- To reduce the dimensions, the eigenvalues and the respective eigenvectors are calculated.
- If the number of clusters is k then the first eigenvalues and their eigenvectors are taken and stacked into a matrix such that the eigenvectors are the columns.

3. Clustering the Data

- The process involves clustering the reduced data by using K-means clustering technique.
- First, each node is assigned a row of the normalized of the Graph-Laplacian-Matrix.
- Then this data is clustered using any traditional technique.

Properties

1. **Assumption-Less :** This clustering technique does not assume the data to follow some property. This makes this technique to answer a more-generic class of clustering problems.
2. **Ease of implementation and speed :** This algorithm is easier to implement than other clustering algorithms and is also very fast as it mainly consists of mathematical computations.
3. **Not-scalable :** This is time-consuming for dense datasets, because it involves the building of matrices and computation of eigenvalues and eigenvectors.

Working of spectral clustering

- In spectral clustering, the data points are treated as nodes of a graph. Thus, clustering is treated as a graph partitioning problem.
- The nodes are then mapped to a low dimensional space that can be easily segregated to form clusters.

5.1.3 K-means and Spectral Clustering

1. **Spectral clustering :** Data points as nodes of a connected graph and clusters are found by partitioning this graph. It is based on its spectral decomposition into subgraphs.
2. **K-means clustering :** It divides the objects into K-clusters such that some metric relative to the centroids of the clusters is minimised.

Advantage of spectral clustering

One advantage is its ability to cluster "points" which are not necessarily vectors and to use for this a "similarity", which is less-restrictive than a distance.

5.2 EVALUATION METHODS BASED ON GROUND TRUTH HOMOGENEITY

- One of the disadvantages of any clustering technique is it that it is difficult to evaluate its performance.
- To tackle this problem, the metric of V-measure was developed.
- The calculation of the V-measure requires the calculation of two terms :

1. **Homogeneity :** A perfectly homogeneous clustering is one where each cluster has data-points belonging to the same class label.

   Homogeneity describes the closeness of the clustering algorithm to this perfection.
2. **Completeness :** A perfectly complete clustering is one where all data-points belonging to the same class are clustered into the same cluster.

   Completeness describes the closeness of the clustering algorithm to this perfection.

5.2.1 Trivial Homogeneity

- It is the case when the number of clusters is equal to the number of data points and each point is in exactly one cluster.
- It takes place when homogeneity is highest while completeness is minimum.

Fig. 5.2.1

Trivial completeness

- It is the case when all the data points are clustered into one cluster.
- It is the extreme case when homogeneity is minimum and completeness is maximum.

Fig. 5.2.2

- We assume that each data point in the above diagram is of the different class label for Trivial Homogeneity and Trivial Completeness.

5.2.2 Remark

- The term homogeneous is different from completeness.
- In homogeneity the basic concept is of the respective cluster which we check whether in each cluster, each data point is of the same class label.
- In completeness, the basic concept is of the respective class label which we check whether data points of each class label is in the same cluster.

Fig. 5.2.3

- In the above diagram, the clustering is perfectly homogeneous since in each cluster the data points are of the same class label but it is not complete because not all data points of the same class label belong to the same class label.

Fig. 5.2.4

- In the above diagram, the clustering is perfectly complete. It is because all data points of the same class label belong to the same cluster.
- But it is not homogeneous because the 1st cluster contains data points of many class labels.
- Let us assume that there are N data samples, C different class labels, K clusters and a_{ck} number of data-points belonging to the class C and cluster K.
- Then the homogeneity h is given by the following :

$$h = 1 - \frac{H(C,K)}{H(C)}$$

Where

$$H(C,K) = -\sum_{K=1}^{N} \sum_{C=1}^{C} \frac{a_{ck}}{N} \log \left[\frac{a_{ck}}{\sum_{C=1}^{C} a_{ck}} \right]$$

$$\text{and } H(C) = -\sum_{C=1}^{C} \left[\frac{\sum_{K=1}^{K} a_{ck}}{C} \right] \log \left[\frac{\sum_{K=1}^{K} a_{ck}}{C} \right]$$

The completeness C is given by the formula

$$C = 1 - \frac{H(K,C)}{H(K)}$$

$$\text{Where } H(K,C) = -\sum_{C=1}^{C} \sum_{K=1}^{K} \frac{a_{ck}}{N} \log \left[\frac{a_{ck}}{\sum_{K=1}^{K} a_{ck}} \right]$$

$$\text{and } H(K) = -\sum_{K=1}^{K} \left[\frac{\sum_{C=1}^{C} a_{ck}}{C} \right] \cdot \log \left[\frac{\sum_{C=1}^{C} a_{ck}}{C} \right]$$

- Thus the weighted V-measure V_P is given by the following :

$$V\beta = \frac{(1+\beta)\, hC}{\beta h + C}$$

- The factor β can be adjusted in such a way that either the homogeneity is favoured or completeness of the algorithm is favoured.
- The advantage of this evaluation metric is :
  (i) It is independent of the number of class labels.
  (ii) The number of clusters.
  (iii) The size of the data.
  (iv) The clustering algorithm and
  (v) Is a reliable metric.

5.2.3 Remarks

(i) Homogeneity in clustering

- A clustering result satisfies homogeneity if all of its clusters contain only data points which are members of a single class.
- This metric is independent of the absolute values of the labels : a permutation of the class or cluster label values won't change the score value.

Fig. 5.2.5

(ii) Homogeneity score

- The score is useful to check whether the clustering algorithm meets an important requirement, i.e. a cluster should contain only samples belonging to a single class.
- It is defined as :

$$h = 1 - \frac{H(Y_{true} / Y_{pred})}{H(Y_{true})}$$

- It is bounded between 0 and 1, with low values indicating a low homogeneity.

5.3 ADJUSTED RAND INDEX

- The **Rand index** or **Rand measure** is a measure of the similarity between two **data clusterings.**
- The Rand Index is used in statistics and in particular in **data clustering.**
- A form of the Rand index is adjusted for the chance grouping of elements. It is called as **Adjusted Rand Index.**
- From a mathematical standpoint, Rand index is related to the **accuracy.** But it is also applicable when class labels are not used.

Fig. 5.3.1

- Example clustering for a dataset with the K means (left) and Mean shift (right) algorithm.

Definition : Rand Index

Consider a set of n elements.

$S = \{a_1, a_2, ..., a_n\}$ and two partitions of S to compare

$X = \{X_1, X_2, ... X_r\}$ a partition of S into r subsets and

$Y = \{Y_1, Y_2, ... Y_S\}$ a partition of S into s subsets.

With this we define

(i) a, the number of pairs of elements in S that are in the **same** subset in X and in the **same** subset in Y.

(ii) b, the number of pairs of elements in S that are in **different** subsets in X and in **different** subsets in Y.

(iii) c, the number of pairs of elements in S that are in the **same** subset in X and in **different** subsets in Y.

(iv) d, the number of pairs of elements in S that are in **different** subsets in X and in the **same** subset in Y.

The Rand index R is

$$R = \frac{a+b}{a+b+c+d} = \frac{a+b}{n\,C_2}$$

Clearly, (a + b) can be considered as the number of agreements between X and Y and c + d as the number of disagreements between X and Y.

(i) Note that the denominator is the total number of pairs, hence the Rand index represents the **frequency of occurrence** of agreements over the total pairs.

(ii) Or, the probability that X and Y will agree on a randomly chosen pair.

Note that, $n\,C_2 = \frac{n(n-1)}{2}$

(iii) Also rand index can be seen as a measure of the percentage of correct decisions made by the algorithm. It can also be computed as,

$$RI = \frac{TP + TN}{TP + FP + FN + TN}$$

Where

TP is the number of **true positives,**

TN is the number of **true negatives,**

FP is the number of **false positives,**

And FN is the number of **false negatives.**

Properties

(i) The rand index has a value between 0 and 1 with 0 indicates that the two data clusterings do not agree on any pair of points and

(ii) 1 indicating that the data clustering are exactly the same.

We define a, b, c, d in mathematical terms as follows :

(i) $a = |S^*|$, where

$S^* = \{(o_i, o_j) \mid oi, oj, \in X^k, o_i, o_j \in Y\lambda j$

(ii) $b = |S^*|$ where

$S^* = \{(o_i, o_j) \mid o_i \in X_{k1}; o_j \in X_{k2};$

$o_i \in Y_{\lambda 1}, o_j \in Y_\lambda{}^2\}$

(iii) $C = |S^*|$ where

$S^* = \{(o_i, o_j) \mid o_i, o_j \in X_k, o_i \in Y_{\lambda 1}, o_j \in Y_{\lambda 2}\}$

(iv) $d = |S^*|$ where

$S^* = \{(o_i, o_j) \mid o_i \in X_{k1}, o_j \in X_{k2}, o_i, o_j \in Y_\lambda\}$

For some

$1 \le i, j \le n, i \ne j, 1 \le k, k_1, k_2 \le r$

$k_1 \ne k_2,$

$1 \le \lambda, \lambda_1, \lambda_2 \le S, \lambda_1 \ne \lambda_2.$

Remark

(i) a is the number of pairs correctly labelled as belonging to the same subset (true positive) and

(ii) b is the number of pairs correctly labelled as belonging to different subsets.

5.4 ADJUSTED RAND INDEX

- The adjusted Rand index is the corrected for chance version of the **Rand index.**
- This correction decides the base line by using the expected similarity of all pair-wise comparisons between clusterings specified by a random model.
- The Rand index assumes the value between 0 and + 1, the adjusted Rand index may take negative values if the index is less than the expected index.

5.4.1 The Contingency Table

- Consider a set of n elements and two clustering (grouping or partitions) of these elements, namely,

  $X = \{X_1, X_2, \ldots X_r\}$ and

  $Y = \{Y_1, Y_2, \ldots Y_s\}$
- The overlap between X and Y can be mentioned in a contingency table $\{n_{ij}\}$, where each entry n_{ij} denotes the number of objects in common between X_i and Y_j.

  $n_{ij} = |X_i \cap Y_j|$

  We prepare a table

X \ Y	Y_1	Y_2	....	Y_S	Sums
X_1	n_{11}	n_{12}	...	n_{1S}	a_1
X_2	n_{21}	n_{22}	...	n_{2S}	a_2
⋮	⋮				⋮
X_r	n_{r1}	n_{r2}	...	n_{rs}	a_r
Sums	b_1	b_2	...	b_s	

5.4.2 Definition

The original adjusted Rand Index is given by (ARI)

$$ARI = \frac{\sum_{i \cdot j}\binom{n_{ij}}{2} - \frac{\left[\sum_i \binom{a_i}{2} \cdot \sum_j \binom{b_j}{2}\right]}{\binom{n}{2}}}{\frac{1}{2}\frac{\left[\sum_i \binom{ai}{2} + \sum_j \binom{bj}{2}\right] - \left[\sum_i \binom{ai}{2} \sum_j \binom{bj}{2}\right]}{\binom{n}{2}}}$$

Where n_{ij}, a_i, b_j are values from the contingency table

Note : $\binom{a_i}{2} = a_i C_2 = \frac{a_i(a_i - 1)}{2}$ etc.

☞ **Remarks**

1. Good adjusted Rand score (ARI)

The adjusted Rand index (ARI) should be as follows :

(i) $ARI \geq 0.90$, excellent recovery

(ii) $0.80 \leq ARI < 0.90$; good recovery

(iii) $0.65 \leq ARI < 0.80$, moderate recovery

$ARI < 0.65$: Poor recovery

2. Negative ARI

- Negative ARI says that the agreement is less than what is expected from a random result.
- This implies that the results are 'orthogonal' or 'complimentary' to some extent.

5.5 CONCEPTS OF WEAK AND EAGER LEARNER

- The distinction between easy learners and lazy learners is based on when the algorithm chooses from the data.
- A lazy learner delays choosing from the data, until it is asked to make a prediction, while an eager learner takes away from the data during training and uses this choice to make predictions.
- It does not directly compare queries with instances in the dataset.

☞ **Lazy learner**

(i) It just stores dataset **without** learning from it.

(ii) Start classifying data when it receives test data.

(iii) It takes less time for learning and more time for classifying data.

☞ **Eager Learner**

(i) When it receives data.set, it starts classifying (learning).

(ii) It does not wait for test-data to learn.

(iii) It takes long time learning and less time classifying data.

☞ **Examples**

In supervised learning :

Lazy

K-Nearest Neighbour, Case-brand Reasoning

Eager

Decision Tree, Naive Bayes, Artificial Neural Networks.

Eager and lazy learning in machine learning

- Eager learning methods construct general, explicit description of the target function. It is based on provided training examples.
- Lazy learning methods simply store the data and generalize the data only when an explicit request is made.
- Eager learners construct a classification model based on the given training data, even before receiving data for classification. It considers a single hypothesis that covers the entire instance space.
- In machine learning, lazy learning is a learning method in which generalisation of the training data is delayed until a query is made to the system.

Naive Bayes is an eager learner

- Naive Bayes classification algorithm is one of eager learning algorithms that promotes the class conditional independence.
- And it predicts the class label in the fastest manner.

5.5.1 Example of Lazy Learner

- Lazy learning refers to any machine learning process in which majority of computation defers to consultation time.
- The typical examples are :
  (i) Instance based learning and
  (ii) Lazy Bayesian Rules.
- Lazy learning stands in contrast to eager learning in which the majority of computation occurs at training time.
- K-NN is a lazy learner. It does not learn a discriminative function from the training data. Instead it memorises the training dataset.
- There is no training time in K-NN. The prediction step in K-NN is expensive.
- When we supply training data to this algorithm, the algorithm does not train itself at all.

Instance-based learning

- Instance based methods are referred to as lazy learning methods because they delay processing until a new instance is classified.
- The nearest neighbours of an instance are defined in terms of Euclidean distance.

Lazy decision tree

Lazy decision tree constructs a customised decision tree for each test, which consists of only a single path from the root to a leaf node.

Lazy algorithm

A lazy learning algorithm is a learning algorithm that can be applied by a lazy learning system (to solve a lazy learning task).

Lazy learner context

It lazily postpones any work until receiving testing records and only performs the work, which is necessary to predict its Target values.

5.6 UNSUPERVISED LEARNING : HIERARCHICAL CLUSTERING

5.6.1 Hierarchical Clustering

Agglomerative Hierarchical Clustering

- In agglomerative clustering initially each data point is considered as a single cluster. In the next step, pairs of clusters are merged or agglomerated.
- This step is repeated until all clusters have been merged in to a single cluster. At the end a single cluster remains that contains all the data points.
- Hierarchical clustering algorithms works in top-down manner or bottom-up manner. Hierarchical clustering is known as Hierarchical agglomerative clustering.
- In agglomerative, clustering is represented as a dendogram as in Fig. 5.6.1 where each merge is represented by a horizontal line.

Fig. 5.6.1 : Dendogram

- A clustering of the data objects is obtained by cutting the dendogram at the desired level, then each connected forms a cluster.
- The basic steps of Agglomerative hierarchical clustering are as follows :
  1. Compute the proximity matrix (distance matrix)
  2. Assume each data point as a cluster.
  3. Repeat
  4. Merge the two nearest clusters.
  5. Update the proximity matrix
  6. Until only a single cluster remains
- In Agglomerative hierarchical clustering proximity matrix is symmetric i.e., the number on lower half will be same as the numbers on top half.
- Different approaches to defining the distance between clusters distinguish the different algorithm's i.e., Single linkage, Complete linkage and Average linkage clusters.
- In single linkage, the distance between two clusters is considered to be equal to shortest distance from any member of one cluster to any member of other cluster.

Fig. 5.6.2

D (r, s) = Min {d (i, j), object i → cluster r and object j → cluster s

- In complete linkage, the distance between two clusters is considered to be equal to greatest distance from any member of one cluster to any member of other cluster.

D (r, s) = Max {d (i, j), object i→ cluster r and object j → cluster s

- In average linkage, we consider the distance between any two clusters A and B is taken to be equal to average of all distances between pairs of object i in A and j in B.ie., mean distance between elements of each other.

D (r, s) = Mean {d (i, j), object i→ cluster r and object j → cluster s

5.6.2 Examples on Hierarchical Clustering

Ex. 5.6.1 : The table shows the six data points. Use all link methods to find clusters. Use Euclidian distance measure.

	X	y
D_1	0.4	0.53
D_2	0.22	0.38
D_3	0.35	0.32
D_4	0.26	0.19
D_5	0.08	0.41
D_6	0.45	0.30

☑ **Soln. :**

First we will solve using single linkage

The distance of data point

$D_1 = (0.4, 0.53)$ to $D_2 = (0.22, 0.38)$ is,

$$D = \sqrt{(0.4-0.22)^2 + (0.53-0.38)^2} = 0.24$$

The distance of data point

$D_1 = (0.4, 0.53)$ to $D_3 = (0.35, 0.32)$ is,

$$D = \sqrt{(0.4-0.35)^2 + (0.53-0.32)^2} = 0.22$$

The distance of data point

$D_1 = (0.4, 0.53)$ to $D_4 = (0.26, 0.19)$ is,

$$D = \sqrt{(0.4-0.26)^2 + (0.53-0.19)^2} = 0.37$$

The distance of data point

$D_1 = (0.4, 0.53)$ to $D_5 = (0.08, 0.41)$ is,

$$D = \sqrt{(0.4 - 0.08)^2 + (0.53 - 0.41)^2} = 0.34$$

The distance of data point

$D_1 = (0.4, 0.53)$ to $D_6 = (0.45, 0.30)$ is,

$$D = \sqrt{(0.4 - 0.45)^2 + (0.53 - 0.30)^2} = 0.23$$

Similarly we will calculate all distances.

☞ **Distance matrix**

D_1	0					
D_2	0.24	0				
D_3	0.22	0.15	0			
D_4	0.37	0.20	0.15	0		
D_5	0.34	0.14	0.28	0.29	0	
D_6	0.23	0.25	0.11	0.22	0.39	0
	D_1	D_2	D_3	D_4	D_5	D_6

0.11 is smallest. D_3 and D_6 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance $((D_3, D_6), D_1)$ = min (distance (D_3, D_1), distance (D_6, D_1)) = min (0.22, 0.23) = 0.22

Distance $((D_3, D_6), D_2)$ = min (distance (D_3, D_2), distance (D_6, D_2)) = min (0.15, 0.25) = 0.15

Distance $((D_3, D_6), D_4)$ = min (distance (D_3, D_4), distance (D_6, D_4)) = min (0.15, 0.22) = 0.15

Distance $((D_3, D_6), D_5)$ = min (distance (D_3, D_5), distance (D_6, D_5)) = min (0.28, 0.39) = 0.28

Similarly we will calculate all distances.

☞ **Distance matrix**

D_1	0				
D_2	0.24	0			
(D_3, D_6)	0.22	0.15	0		
D_4	0.37	0.20	0.15	0	
D_5	0.34	0.14	0.28	0.29	0
	D_1	D_2	(D_3, D_6)	D_4	D_5

0.14 is smallest. D_2 and D_5 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance $((D_3, D_6), (D_2, D_5))$ = min (distance (D_3, D_2), distance (D_6, D_2), distance (D_3, D_5), distance (D_6, D_6))

= min (0.15, 0.25, 0.28, 0.29) = 0.15

Similarly, we will calculate all distances.

☞ **Distance matrix**

D_1	0			
(D_2, D_5)	0.24	0		
(D_3, D_6)	0.22	0.15	0	
D_4	0.37	0.20	0.15	0
	D_1	(D_2, D_5)	(D_3, D_6)	D_4

0.15 is smallest. (D_2, D_5) and (D_3, D_6) as well as D_4 and (D_3, D_6) have smallest distance. We can pick either one.

☞ **Distance matrix**

D_1	0		
(D_2, D_5, D_3, D_6)	0.22	0	
D_4	0.37	0.15	0
	D_1	(D_2, D_5, D_3, D_6)	D_4

0.15 is smallest. (D_2, D_5, D_3, D_6) and D_4 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

☞ **Distance matrix**

D_1	0	
$(D_2, D_5, D_3, D_6, D_4)$	0.22	**0**
	D_1	$(D_2, D_5, D_3, D_6, D_4)$

Now a single cluster remains $(D_2, D_5, D_3, D_6, D_4, D_1)$

Next, we represent the final dendogram for single linkage as,

Now we will solve using complete linkage

☞ **Distance matrix**

D_1	0					
D_2	0.24	0				
D_3	0.22	0.15	0			
D_4	0.37	0.20	0.15	0		
D_5	0.34	0.14	0.28	0.29	0	
D_6	0.23	0.25	0.11	0.22	0.39	0
	D_1	D_2	D_3	D_4	D_5	D_6

0.11 is smallest. D_3 and D_6 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance $((D_3, D_6), D_1)$ = max (distance (D_3, D_1), distance (D_6, D_1)) = max (0.22, 0.23) = 0.23

Similarly, we will calculate all distances.

☞ **Distance matrix**

D_1	0				
D_2	0.24	0			
(D_3, D_6)	0.23	0.25	0		
D_4	0.37	0.20	0.22	0	
D_5	0.34	0.14	0.39	0.29	0
	D_1	D_2	(D_3, D_6)	D_4	D_5

0.14 is smallest. D_2 and D_5 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

☞ **Distance matrix**

D_1	0			
(D_2, D_5)	0.34	0		
(D_3, D_6)	0.23	0.39	0	
D_4	0.37	0.29	0.22	0
	D_1	(D_2, D_5)	(D_3, D_6)	D_4

0.22 is smallest. Here (D_3, D_6) and D_4 have smallest distance. So, we combine these two in one cluster and recalculate distance matrix.

☞ **Distance matrix**

D_1	0		
(D_2, D_5)	0.34	0	
(D_3, D_6, D_4)	0.37	0.39	0
	D_1	(D_3, D_6, D_4)	(D_3, D_6, D_4)

0.34 is smallest. (D_2, D_5) and D_1 have smallest distance so, we combine these two in one cluster and recalculate distance matrix.

☞ **Distance matrix**

(D_2, D_5, D_1)	0	0
(D_3, D_6, D_4)	0.39	0
	(D_2, D_5, D_1)	(D_3, D_6, D_4)

Now a single cluster remains $(D_2, D_5, D_1, D_3, D_6, D_4)$

Next, we represent the final dendogram for complete linkage as,

Now we will solve using average linkage

☞ **Distance matrix**

D_1	0					
D_2	0.24	0				
D_3	0.22	0.15	0			
D_4	0.37	0.20	0.15	0		
D_5	0.34	0.14	0.28	0.29	0	
D_6	0.23	0.25	0.11	0.22	0.39	0
	D_1	D_2	D_3	D_4	D_5	D_6

0.11 is smallest. D_3 and D_6 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance $((D_3, D_6), D_1)$ = 1/2 (distance (D_3, D_1) + distance (D_6, D_1)) = 1/2 (0.22 + 0.23) = 0.23

Similarly, we will calculate all distances.

☞ **Distance matrix**

D_1	0				
D_2	0.24	0			
(D_3, D_6)	0.23	0.2	0		
D_4	0.37	0.20	0.19	0	
D_5	0.34	0.14	0.34	0.29	0
	D_1	D_2	(D_3, D_6)	D_4	D_5

0.14 is smallest. D_2 and D_5 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

☞ **Distance matrix**

D_1	0			
(D_2, D_5)	0.29	0		
(D_3, D_6)	0.22	0.27	0	
D_4	0.37	0.22	0.15	0
	D_1	(D_2, D_5)	(D_3, D_6)	D_4

(D_3, D_6) and D_4 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

☞ **Distance matrix**

D_1	0		
(D_2, D_5)	0.24	0	
(D_3, D_6, D_4)	0.27	0.26	0
	D_1	(D_2, D_5)	(D_3, D_6, D_4)

0.24 is smallest. (D_2, D_5) and D_1 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

☞ **Distance matrix**

(D_2, D_5, D_1)	0	0
(D_3, D_6, D_4)	0.26	0
	(D_2, D_5, D_1)	(D_3, D_6, D_4)

Now a single cluster remains $(D_2, D_5, D_1, D_3, D_6, D_4)$

Next, we represent the final dendogram for average linkage as,

Ex. 5.6.2 : Apply single linkage, complete linkage and average linkage on the following distance matrix and draw dendogram.

☑ **Soln. :**

First we will solve using single linkage

☞ **Distance matrix**

P_1	0				
P_2	2	0			
P_3	6	3	0		
P_4	10	9	7	0	
P_5	9	8	5	4	0
	P_1	P_1	P_3	P_4	P_5

2 is smallest. P_1 and P_2 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance $((P_1, P_2), P_3)$ = min (distance (P, P_3), distance (P_2, P_3)) = min (6, 3) = 3

Similarly, we will calculate all distances.

Distance matrix

(P_1, P_2)	0			
P_3	3	0		
P_4	9	7	0	
P_5	8	5	4	0
	(P_1, P_2)	P_3	P_4	P_5

3 is smallest. (P_1, P_2) and P_3 have smallest distanc So, we combine this two in one cluster and recalcula distance matrix.

Distance $((P_1, P_2, P_3), P_4))$ = min (distance (P_1, P_4), distance (P_2, P_4), distance (P_3, P_4)) = min (9, 7) = 7

Similarly, we will calculate all distances.

Distance matrix

(P_1, P_2, P_3)	0		
P_4	7	0	
P_5	5	4	0
	(P_1, P_2, P_3)	P_4	P_5

4 is smallest. P_4 and P_5 have smallest distance.

Distance matrix

(P_1, P_2, P_3)	0	
(P_4, P_5)	5	0
	(P_1, P_2, P_3)	(P_4, P_5)

Now a single cluster remains $(P_1, P_2, P_3, P_4, P_5)$

Next, we represent the final dendogram for single linkage as,

Now we will solve using complete linkage

Distance matrix

P_1	0				
P_2	2	0			
P_3	6	3	0		
P_4	10	9	7	0	
P_5	9	8	5	4	0
	P_1	P_2	P_3	P_4	P_5

2 is smallest. P_1 and P_2 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance $((P_1, P_2), P_3)$ = max (distance (P_1, P_3), distance (P_2, P_3)) = max (6, 3) = 6

Similarly, we will calculate all distances.

Distance matrix

(P_1, P_2)	0			
P_3	6	0		
P_4	10	7	0	
P_5	9	5	4	0
	(P_1, P_2)	P_3	P_4	P_5

4 is smallest. P_4 and P_5 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance matrix

(P_1, P_2)	0		
P_3	6	0	
(P_4, P_5)	10	7	0
	(P_1, P_2)	P_3	(P_4, P_5)

6 is smallest. (P_1, P_2) and P_3 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance matrix

(P_1, P_2, P_3)	0	
(P_4, P_5)	10	0
	(P_1, P_2, P_3)	(P_4, P_5)

Now a single cluster remains $(P_1, P_2, P_3, P_4, P_5)$

Next, we represent the final dendogram for complete linkage as,

Now we will solve using average linkage

☞ **Distance matrix**

P_1	0				
P_2	2	0			
P_3	6	3	0		
P_4	10	9	7	0	
P_5	9	8	5	4	0
	P_1	P_2	P_3	P_4	P_5

2 is smallest. P_1 and P_2 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance $((P_1, P_2), P_3) = 1/2$ (distance (P_1, P_3), distance $(P_2, P_3)) = 1/2\ (6, 3) = 4.5$

Similarly, we will calculate all distances.

☞ **Distance matrix**

(P_1, P_2)	0			
P_3	4.5	0		
P_4	9.5	7	0	
P_5	8.5	5	4	0
	(P_1, P_2)	P_3	P_4	P_5

4 is smallest. P_4 and P_5 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

☞ **Distance matrix**

(P_1, P_2)	0		
P_3	4.5	0	
(P_4, P_5)	9	6	0
	(P_1, P_2)	P_3	(P_4, P_5)

4.5 is smallest. (P_1, P_2) and P_3 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

☞ **Distance matrix**

(P_1, P_2, P_3)	0	
(P_4, P_5)	8	0
	(P_1, P_2, P_3)	(P_4, P_5)

Now a single cluster remains $(P_1, P_2, P_3, P_4, P_5)$

Next, we represent the final dendogram for average linkage as,

UEx. 5.6.3 **MU - May 16, 10 Marks**

Apply Agglomerative clustering algorithm on given data and draw dendrogram. Show three clusters with its allocated points. Use single link method.

	A	B	C	D	E	F
A	0	$\sqrt{2}$	$\sqrt{10}$	$\sqrt{17}$	$\sqrt{5}$	$\sqrt{20}$
B	$\sqrt{2}$	0	$\sqrt{8}$	3	1	$\sqrt{18}$
C	$\sqrt{10}$	$\sqrt{8}$	0	$\sqrt{5}$	$\sqrt{5}$	2
D	$\sqrt{17}$	1	$\sqrt{5}$	0	2	3
E	$\sqrt{5}$	1	$\sqrt{5}$	2	0	$\sqrt{13}$
F	$\sqrt{20}$	$\sqrt{18}$	2	3	$\sqrt{13}$	0

☑ **Soln. :**

☞ **Distance matrix**

A	0					
B	1.414	0				
C	3.162	2.828	0			
D	4.123	1	2.236	0		
E	2.236	1	2.236	2	0	
F	4.472	4.242	2	3	3.6	0
	A	B	C	D	E	F

1 is smallest. B, D and B, E have smallest distance. We can select anyone. So, we combine B, D in one cluster and recalculate distance matrix using single linkage.

Distance matrix

A	0				
B,D	1.414	0			
C	3.162	2.236	0		
E	2.26	1	2.236	0	
F	4.472	3	2	3.6	0
	A	B,D	C	E	F

1 is smallest. B, D and E have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance matrix

A	0			
B,D,E	1.414	0		
C	3.162	1	0	
F	4.472	3	2	0
	A	B,D,E	C	F

1 is smallest. B, D, E and C are combined together.

Distance matrix

A	0		
B,D,E,C	1.414	0	
F	4.472	2	0
	A	B,D,E,C	F

In the questions three clusters are asked with their allocated points. Three clusters are A, (B, D, E, C) and F.

UEx. 5.6.4 MU - May 17, 10 Marks

For the given set of points identify clusters using complete link and average link using Agglomerative clustering.

	A	B
P_1	1	1
P_2	1.5	1.5
P_3	5	5
P_4	3	4
P_5	4	4
P_6	3	3.5

Soln. :

First we will solve using complete linkage

Distance matrix

p1	0					
P2	0.707	0				
P3	5.656	4.949	0			
P4	3.605	2.915	2.236	0		
P5	4.242	3.535	1.414	1	0	
P6	5.201	2.5	1.802	0.5	1.118	0
	P1	P2	P3	P4	P5	P6

0.5 is smallest. P4 and P6 have smallest distance. We can select anyone. So, we combine this in one cluster and recalculate distance matrix using complete linkage.

Distance matrix

P1	0				
P2	0.707	0			
P3	5.656	4.949	0		
P4,P6	5.201	2.5	1.802	0	
P5	4.242	3.535	1.414	1.118	0
	P1	P2	P3	P4,P6	P5

0.707 is smallest. P1 and P2 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance matrix

P1,P2	0			
P3	5.656	0		
P4,P6	5.201	2.236	0	
P5	4.242	1.414	1.118	0
	P1,P2	P3	P4,P6	P5

1.118 is smallest. P4, P6 and P5 are combined together.

Distance matrix

P1,P2	0		
P3	5.656	0	
P4,P5,P6	5.201	2.236	0
	P1,P2	P3	P4,P5,P6

2.236 is smallest. P4, P5, P6 and P3 are combined together.

P1,P2	0	
P3,P4,P5,P6	5.656	0
	P1,P2	P3,P4,P5,P6

Next we will combine all clusters in a single cluster.

Now we will solve using average linkage

Distance matrix

P1	0					
P2	0.707	0				
P3	5.656	4.949	0			
P4	3.605	2.915	2.236	0		
P5	4.242	3.535	1.414	1	0	
P6	5.201	2.5	1.802	0.5	1.118	0
	P1	P2	P3	P4	P5	P6

0.5 is smallest. P4 and P6 have smallest distance. We can select anyone .So, we combine this in one cluster and recalculate distance matrix using complete linkage.

Distance matrix

P1	0				
P2	0.707	0			
P3	5.656	4.949	0		
P4,P6	4.403	2.707	2.019	0	
P5	4.242	3.535	1.414	1.059	0
	P1	P2	P3	P4,P6	P5

0.707 is smallest. P1 and P2 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance matrix

P1,P2	0			
P3	5.302	0		
P4,P6	3.55	2.019	0	
P5	3.888	1.414	1.059	0
	P1, P2	P3	P4, P6	P5

1.059 is smallest. P4, P6 and P5 are combined together.

Distance matrix

P1,P2	0		
P3	5.302	0	
P4,P5,P6	3.66	1.817	0
	P1,P2	P3	P4,P5,P6

1.817 is smallest. P4,P5,P6 and P3 are combined together.

P1,P2	0	
P3,P4,P5,P6	4.07	0
	P1,P2	P3,P4,P5,P6

Next we will combine all clusters in a single cluster.

5.7 UNSUPERVISED LEARNING : ASSOCIATION RULES

5.7.1 Introduction to Association Rules

- Associations are any things that are generally occurred together. Let us understand this with the help of an example, as a shop owner we are interested in finding out the products that are generally sell together.
- We will simplify this further, if there is a customer who purchases milk then it is more likely that he will also purchase bread. From this we can say that the customers who buy milk tend to also buy bread.
- In a bakery shop most clients will buy cake. This means that there will be many frequent item sets involving cake, such as {candle, cake}.
- This might suggest the construction of an association rule **if candle then cake** – however, this is predictable given that {cake} is already a frequent item set (and clearly at least as frequent as {candle, cake}).
- Of more interest would be the converse rule **if cake then candle** which expresses that a considerable proportion of the people buying cake also buy a candle.

5.7.2 Apriori Algorithm

- Frequent item sets are used to create association rules in Apriori algorithm. It is created to work on the databases that contain transactions. Association rules are used to identify whether the two items are strongly connected to each other or not.

- Breadth first search and hash tree methods are used to find out the item sets in an optimum manner. Iterations are applied to find the frequent item sets from the huge amount of database.
- In market basket analysis mostly Apriori algorithm is used. The algorithm is used to find the products that can be bought together. Apriori algorithm can also be used in the medical discipline to find reactions of drugs on the patients.

What is Frequent Item set ?

- Item sets having support more than the set minimum support or threshold value are called as Frequent item sets. Let us see an example, if there are P and Q frequent item sets, then individually P and Q should also be the frequent item set.
- Let us assume there are the two transactions: P = {3,4,6,7,8}, and Q = {4,6,9}, in these two transactions, 4 and 6 are the frequent item sets.

Apriori Algorithm Working

Apriori algorithm works according to the following steps,

1. Find the support of item sets in the transactional database. Choose the minimum support and confidence.
2. From the transaction consider all supports with more support value than the minimum or chosen support value.
3. Identify all the rules of these subsets which are having more confidence value than the threshold or minimum confidence.
4. Sort the rules as the decreasing order of lift.

- **Example :** Let us assume that we are having following dataset that contains various transactions. We want to identify the frequent item sets and create the association rules with Apriori algorithm. In these example minimum support 2 and minimum confidence 50% is given.

Transaction ID	Item sets
T1	1,2
T2	2,4
T3	2,3
T4	1,2,4
T5	1,3
T6	2,3
T7	1,3
T8	1,2,3,5
T9	1,2,3

Soln. :

Step 1 : Calculating S1 and I1 :

- Initially we will generate a table that contains support count of each item set in the given dataset. This table is called the Candidate set or S1.

Itemset	Support_count
1	6
2	7
3	5
4	2
5	1

- Next, we will consider all the item sets that are having more support count than the Minimum Support i.e. 2. This will give us the table for the frequent item set I1.
- Here all the item sets are having more or same support count than the minimum support, except the E, so E item set will be removed.

Itemset	Support_count
1	6
2	7
3	5
4	2

Step 2 : Calculating S2 and I2

- Next, we will create S2 using I1. In S2, we will generate the pair of the item sets of I1 in the form of subsets.
- Once we create the subsets, we will again find the support count from the main transaction table. Here we will check how many times these pairs have occurred together in the given dataset. So, we will get the following table for S2 :

Itemset	Support_count
{1,2}	4
{1,3}	4
{1,4}	1
{2,3}	4
{2,4}	2
{3,5}	0

- Now we will compare S2 Support count with the minimum support count, and after doing this, the item set with less support count will be removed from the table S2. It will give us the following table for I2

Itemset	Support_count
{1,2}	4
{1,3}	4
{2,3}	4
{2,4}	2

Step 3 : Calculation of S3, and I3:

- For S3, we will follow the same two processes, but now we will form the S3 table with subsets of three item sets together, and will calculate the support count from the dataset. It will give the following table:

Itemset	Support_count
{1,2,3}	2
{2,3,4}	1
{1,3,4}	0
{1,2,4}	0

- Next we will generate the I3 table. From S3 table we can see that there is only one combination of itemset that has support count same as that of minimum support count. So, the I3 will have only one combination, i.e., {1,2,3}.

Step 4 : Identifying the association rules for the subsets:

- To create the association rules,, we will create a new table with the possible rules from the above combination {1,2,3}.
- We will compute the Confidence using formula sup (1 ^ 2)/ 1 for all the rules.
- After computing the confidence value for all rules, we will exclude the rules that have less confidence than the minimum threshold(50%).
- Consider the below table :

Rules	Support	Confidence
1 ^2 → 3	2	Sup{(1 ^2) ^3}/sup(1 ^2) = 2/4 = 0.5 = 50%
2^3 → 1	2	Sup{(2^3) ^1}/sup(2 ^3) = 2/4 = 0.5 = 50%
1^3 → 2	2	Sup{(1 ^3) ^2}/sup(1 ^3) = 2/4 = 0.5 = 50%
3→ 1 ^2	2	Sup{(3^(1 ^2)}/sup(3) = 2/5 = 0.4 = 40%
1→ 2^3	2	Sup{(1^(2 ^3)}/sup(1) = 2/6 = 0.33 = 33.33%
2→ 2^3	2	Sup{(2^(2 ^3)}/sup(2) = 2/7 = 0.28 = 28%

- As the given threshold or minimum confidence is 50%, so the first three rules 1^ 2 → 3, 2 ^ 3 → 1, and 1 ^ 3 → 2 can be considered as the strong association rules for the given problem.

5.7.3 Performance Measures

- Association rule mining is a kind of unsupervised learning which checks the dependency of one data item on other data item.
- Based on dependency items are mapped to each other so that it can be more profitable.
- This method identifies some interesting relations or associations that exist between the variables of dataset. For Association rule mining Apriori algorithm can be used.
- Association rule mining works on the concept of If and Else Statement, such as if X then Y. In this the If part is known as *antecedent*, and then part is known as *Consequent*.
- Such types of relationships in which we can find out some association or relation among two items is called as ***single cardinality***. In this rules are created. Here if

the number of items are more then cardinality also increases accordingly.

- Due to this to measure the associations between more numbers of data items, several metrics are used. These metrics are explained bellow,

1. **Support**

Support is defined as the number of occurrences of an item in the dataset. It is also defined as the fraction of the transaction T that contains the itemset I. If there are I datasets, then for transactions T, it can be written as:

Support (I) = Freq (I) / T

2. **Confidence**

Confidence represents how likely the rule has been found to be true. We can also say that how many times the items P and Q appears together in the dataset when the occurrence of P is already given. It is the ratio of the transaction that contains P and Q to the number of records that contain P.

Confidence = Freq (P, Q) / Freq (P)

3. **Lift**

Lift is defined as the strength of any rule.

Lift = Support (P, Q) / (Support (P) * Support (Q))

It is the ratio of the observed support measure and expected support if P and Q are independent of each other. It has three possible values:

- Lift= 1: It defines the probability of appearance of antecedent and consequent independent to each other.
- Lift>1: It defines the degree by which the two item sets are dependent on each other.
- Lift<1: It describes if there are any alternative for other items.

5.7.4 Issues in Machine Learning

UQ. What are the issues in Machine learning ? **(MU - May 15, 5 Marks)**

1. Which algorithm we have to select to learn general target functions from specific training dataset? What should be the settings for particular algorithms, so as to converge to the desired function, given sufficient training data? Which algorithms perform best for which type of problems and representations?
2. How much training data is sufficient? What should be the general amount of data that can be found to relate the confidence in learned hypotheses to the amount training experience and the character of the learner's hypothesis space?
3. Prior knowledge held by the learner is used at which time and manner to guide the process of generalizing from examples? If we have approximately correct knowledge, will it helpful even when it is only approximately correct?
4. What is the best strategy for choosing a useful next training experience, and how does the choice of this strategy after the complexity of the learning problem?
5. To reduce the task of learning to one or more function approximation problems, what will be the best approach? What specific functions should the system attempt to learn? Can this process itself be automated?
6. To improve the knowledge representation and to learn the target function, how can the learner automatically alter its representation?

5.7.5 How to choose the right algorithm ?

UQ. Explain the steps required for selecting the right machine learning algorithm. **(MU - May 16, 8 Marks)**

With all the different algorithms available in machine learning, how can you select which one to use ? First you need to focus on your goal.

What are you trying to get out of this? What data do you have or can you collect? Secondly you have to consider the data.

1. **Goal :** If you are trying to predict or forecast a target value, then you need to look into supervised learning. Otherwise, you have to use unsupervised learning.

   (a) If you have chosen **supervised** learning, then next you need to focus on what's your **target** value?

   If **target value** is **discrete** (e.g. Yes/ No, 1 /2/3, A/B/C), then use **Classification**.

   If **target value** is continuous i.e. Number of values (e.g. 0 – 100, – 99 to 99), then use **Regression.**

   (b) If you have chosen **unsupervised** learning, then next you need to focus on what is your **aim?**

   If you want to **fit your data** into some **discrete groups,** then use **Clustering**

If you want to **find numerical estimate** of how strong the fit into each group, then use **density estimation algorithm**

2. **Data** : Are the features continuous or nominal ? Are there missing values in features ? If yes, what is a reason for missing values? Are there outliers in the data? To narrow the algorithm selection process, all of these features of your data can help you.

Table 5.7.1 : Selection of Algorithm

	Supervised Learning	Unsupervised Learning
Discrete	Classification	Clustering
Continuous	Regression	Density Estimation

5.7.6 Steps in Developing a Machine Learning Application

UQ. Explain the steps of developing Machine Learning applications. **(MU - May 19, 10 Marks)**

1. Collection of Data

You could collect the samples from a website and extracting data.

- From RSS feed or an API
- From device to collect wind speed measurement
- Publicly available data.

2. Preparation of the input data

- Once you have the input data, you need to check whether it's in a useable format or not.
- Some algorithm can accept target variables and features as string; some need them to be integers. Some algorithm accepts features in a special format.

3. Analyse the input data

- Looking at the data you have passed in a text editor to check collection and preparation of input data steps are properly working and you don't have a bunch of empty values.
- You can also check at the data to find out if you can see any patterns or if there is anything obvious, such as a few data points greatly differ from remaining set of the data. Plotting data in 1, 2 or 3 dimensions can also help.
- Distil pmultiple dimensions down to 2/3 so that you can visualize the data.

4. The importance of this step is that it makes you understand that you don't have any garbage value coming in.

5. Train the algorithm

- Good clean data from the first two steps is given as input to the algorithm. The algorithm extracts information or knowledge. This knowledge is mostly stored in a format that is readily useable by machine for next 2 steps.
- In case of unsupervised learning, training step is not there because target value is not present. Complete data is used in the next step.

6. Test the algorithm

- In this step the information learned in the previous step is used. When you are checking an algorithm, you will test it to find out whether it works properly or not. In supervised case, you have some known values that can be used to evaluate the algorithm.
- In case of unsupervised, you may have to use some other matrices to evaluate the success. In either case, if you are not satisfied, you can again go back to step 4, change some things and test again.
- Mostly problem occurs in collection or preparation of data and you will have to go back to step 1.

7. Use it

In this step a real program is developed to do some task, and once again it is checked if all the previous steps worked as you expected. You might encounter some new data and have to revisit step 1-5.

Fig. 5.7.1 : Typical example of Machine Learning Application

5.7.7 Applications of Machine Learning

UQ. Write short note on : Machine learning applications. **(MU - May 16, May 17, 10 Marks)**

(1) Learning Associations

- A supermarket chain-one an example of retail application of machine learning is basket analysis, which is finding associations between products bought by customers :
- If people who buy P typically also buy Q and if there is a customer who buys Q and does not buy P, he or she is a potential P customer. Once we identify such customers, we can target them for cross-selling.
- In finding an association rule, we are interested in learning a conditional probability of the form P (Q/P) where Q is the product we would like to condition on P, which are the product / products which we know that customer has already purchased.

$$P(\text{Milk} / \text{Bread}) = 0.7$$

- It implies that 70% of customers who buy bread also buy milk

(2) Classification

- A credit is an amount of money loaned by a financial institution.
- It is important for the bank to be able to predict in advance the risk associated with a loan. Which is the probability that the customer will default and not pay the whole amount back?
- In credit scoring, the bank calculates the risk given the amount of credit and the information about the customer. (Income, savings, collaterals, profession, age, past financial history). The aim is to infer a general rule from this data, coding the association between a customer's attributes and his risk.
- Machine Learning system fits a model to the past data to be able to calculate the risk for a new application and then decides to accept or refuse it accordingly.

  If income > Q_1 and savings > Q_2

  Then low – risk ELES high – risk
- Other classification examples are Optical character recognition, face recognition, medical diagnosis, speech recognition and biometric.

Fig. 5.7.2 : Classification for credit scoring

(3) Regression

- Suppose we want to design a system that can predict the price of a flat.
- Let's take the inputs as the area of the flat, location and purchase year and other information that affects the rate of flat.
- The output is the price of the flat. The applications where output is numeric are regression problems.
- Let X represents flat features and Y is the price of flat. We can collect training data by surveying past purchased transactions and the Machine Learning algorithm fits a function to this data to learn Y as a function of X for the suitable values of W and W_0.

$$Y = w^*x + w_0$$

Fig. 5.7.3 : Regression for prediction of price of flat

(4) Unsupervised Learning

- One of the important unsupervised learning problem is clustering. In clustering dataset is partitioned in to meaningful sub classes known as clusters. For example, suppose you want to decorate your home using given items.

- Now you will classify them using unsupervised learning (no prior knowledge) and this classification can be on the basis of color of items, shape of items, material used for items, type of items or whatever way you would like.

(5) Reinforcement Learning

- There are some of the applications where output of system is a sequence of actions. In such applications the sequence of correct actions instead of single action is important in order to reach goal.
- An action is said to be good if it is part of good policy. Machine learning program generates a policy by learning previous good action sequences. Such methods are called reinforcement methods.
- A good example of reinforcement learning is chess playing. In artificial intelligence and machine learning, one of the most important research area is game playing.
- Games can be are easily described but at the same time, they are quite difficult to play well. Let's take a example of chess that has limited number of rules, but the game is very difficult because for each state there can be large number of possible moves.
- Another application of reinforcement learning is robot navigation. The robot can move in all possible directions at any point of time.
- The algorithm should reach goal state from an initial state by learning the correct sequence of actions after conducting number of trial runs.
- When the system has unreliable and partial sensory information, it makes reinforcement learning complex. Let's take an example of robot with incomplete camera information. Here robot does not know its exact location.

5.8 GRAPH BASED CLUSTERING

- Graph clustering is an important subject, and deals with clustering with graphs.
- The data of a clustering problem can be represented as a graph where each element to be clustered is represented as a node and the distance between two elements is modeled by a certain weight on the edge linking the nodes.
- Thus in graph clustering, elements within a cluster are connected to each other but have no connection to elements outside that cluster.

5.8.1 Graph Clustering Algorithm

- The HCS (Highly Connected Subgraphs) clustering algorithm (also known as the HCS algorithm, and other names such as highly connected clusters /components /Kernels) is an algorithm based on graph connectivity for cluster analysis.
- It works by representing the similarity data in a **similarity graph**, and then finding all the highly connected subgraphs.
- It does not make any prior assumptions on the number of clusters.
- The HCS algorithm gives a clustering solution, which is inherently meaningful in the application domain.

5.8.2 Method of Graph-based Clustering

(1) Transform the data into a graph representation :

(i) Vertices are the data points to be clustered

(ii) Edges are weighted and based on similarity between data points.

Fig. 5.8.1

(2) Clustering as graph partitioning

Two things are required :

(i) An objective function to determine what would be the best way to 'cut' the edges of a graph.

Fig. 5.8.2

(ii) An algorithm to find the optimal partition (optimal according to objective function).

(3) Objective function for partitioning

Suppose we want to partition the set of vertices V into two sets : V_1 and V_2. One possible objective function is to minimise graph cut :

$$\text{Cut } (V_1, V_2) \sum_{i \in V_1, j \in V_2} W_{ij} \text{ ; } W_{ij} \text{ is weight of the}$$

edge between nodes i and j

Cut = 0.2 Cut = 0.4

Fig. 5.8.3

(4) **Objective function for partitioning limitation of** minimising graph cut

Cut = 0.1

Fig. 5.8.4

The optimal solution might be to split up a single node from the rest of the graph!

Not a desirable solution,

(5) Objective function for partitioning

Our aim is not to only to have 'minimise the graph', but also look for "balanced" clusters.

$$\text{Ratio Cut } (V_1, V_2) = \frac{\text{Cut } (V_1, V_2)}{|V_1|} + \frac{\text{Cut } (V_1, V_2)}{|V_2|}$$

$$\text{Normalised Cut } (V_1, V_2) = \frac{\text{Cut } (V_1, V_2)}{\sum_{i \in V_1} d_i} + \frac{\text{Cut } (V_1, V_2)}{\sum_{j \in V_2} d_j}$$

$$\text{Where } d_i = \sum_j W_{ij}$$

V_1 and V_2 are the set of nodes in partitions 1 and 2;

$|V_1|$ is the number of nodes in partition V_1

Fig. 5.8.5

(a) Example

Cut = 0.1

Fig. 5.8.6

Cut = 0.2

Fig. 5.8.7

Ratio cut $= \frac{0.1}{1} + \frac{0.1}{5} = 0.12$

Normalised cut

$= \frac{0.1}{0.1} + \frac{0.1}{1.5} = 1.07$

Ratio cut $= \frac{0.2}{3} + \frac{0.2}{3} = 0.13$

Normalised cut

$\frac{0.2}{1} + \frac{0.2}{0.6} = 0.53$

(b) Example : If graph is unweighted (or has to same edge weight).

Fig. 5.8.8

Fig. 5.8.9

Cut = 1

Ratio cut $= \frac{1}{1} + \frac{1}{5} = 1.2$

Normalised cut

$= \frac{1}{1} + \frac{1}{9} = 1.11$

Cut = 2

Ratio cut $= \frac{1}{3} + \frac{1}{3} = 0.67$

Normalised cut

$\frac{1}{5} + \frac{1}{5} = 0.2$

(6) Algorithm for graph partitioning

Method of minimising the objective function :

(i) We can use a heuristic (greedy) approach to do this.

(ii) An elegant way to optimise the function is by using ideas from spectral graph theory. This leads to a class of algorithms known as **spectral clustering**.

(7) Spectral clustering

Spectral properties of a graph :

(i) We find eigenvalues and eigenvectors of the adjacency matrix. They can be used to represent a graph.

(ii) There exists a relationship between spectral properties of a graph and the graph partitioning problem.

Method

(i) Start with a similarity/adjacency matrix, W, of a graph :

(ii) Define a diagonal matrix D

$$D_{ij} = \begin{cases} \sum_{k=1}^{n} W_{ik} & , \text{ if } i = j \\ 0 & , \text{ otherwise} \end{cases}$$

If W is a binary 0 – 1 matrix, then D_{ii} represents the degree of node i.

Preliminaries

Two clusters

$$W = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Two block-diagonal matrices

Fig. 5.8.10

(8) Graph Laplacian matrix

$$W = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Two block-matrices

Laplacian, L = D – W

Laplacian also has a block structure

$$L = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & -1 & -1 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}$$

Fig. 5.8.11

(9) Properties of graph Laplacian

(i) $L = (D - W)$ is a symmetric matrix

(ii) L is a positive semi-definite matrix

It means all eigenvalues are non-negative i.e. ≥ 0.

(10) Spectral clustering

Consider a data-set with N data points :

(i) Construct an $N \times N$ similarity matrix W.

(ii) Compute the $N \times N$ Laplacian matrix, $L = D - W$

(iii) Compute the K "Smallest" eigenvectors of L :

(a) Each eigenvector V_1 is an $N \times 1$ Column vector.

(b) Create a matrix V containing eigen vectors V_1, V_2, ..., V_k as column (one may exclude the first eigenvector)

(iv) Cluster the rows of V using K-means or other clustering algorithms into K-clusters.

Example

Fig. 5.8.12

Remark

Spectral properties of a graph (i.e. eigen-values and eigenvectors) contain information about clustering structure.

5.9 CLUSTERING WITH MINIMAL SPANNING TREE

There are two Euclidean minimum spanning tree based clustering algorithms :

(1) a k-constrained (2) an unconstrained algorithm

(i) K-constrained clustering algorithm produces a K-partition of a set of points for any given k.

- The algorithm constructs a minimum spanning tree of a set of representative points and removes edges that satisfy a predefined criterion. The process is repeated until K clusters are produced.
- Our unconstrained clustering algorithm partitions a point set into a group of clusters by maximally reducing the overall standard deviation of the edges in the Euclidean minimum spanning tree. It is constructed from a given point set, without prescribing the number of clusters.
- The minimal spanning tree (MST-) based clustering method can identify clusters of arbitrary shape by removing inconsistent edges.
- The definition of the inconsistent edges is a major issue that has to be cleared in all MST – based clustering algorithms.
- Here we discuss MST-based clustering algorithm through the 'cluster centre initialisation algorithm', called as CCiMST.
- CCiMST is based on geodesic distance and dual densities of the points.
- We shall demonstrate that the inconsistent edge is located on the shortest path between the cluster centres, so that we can find the inconsistent edge with the length of the edges as well as densities of their endpoints on the shortest path.

5.9.1 The Cluster Center-Initialisation Method

- Among the cluster center initialisation methods, density peaks clustering (DPC) has been widely used.
- Cluster center-initialisation method is based on DPC; so we briefly describe DPC.

- In DPC, we assume that the cluster centers are characterised by a higher density than their neighbours and by a relatively large distance from points with higher densities.

DPC uses two quantities :

(i) the density ρ_i of point x_i and

(ii) its distance δ_i from points of higher densities.

The density ρ_i of point x_i is defined as :

$$\rho_i = \sum_j \exp\left[-\frac{D(x_i, x_j)^2}{2\sigma^2}\right],$$

Where, $D(x_i, x_j)$ is the Euclidean distance between points x_i and x_j and σ^2 is variance.

(1) Input : Data set

$X = \{x_i, x_i \in R^P, i = 1, 2, \ldots, n\}$, the total number of points n, a predefined parameter δ.

(2) Output : The value of σ.

(3) Begin

(4) Calculate and sort the pairwise distance between points in ascending order, that is, $\left\{d_1, d_2, \ldots, d_{\frac{n(n-1)}{2}}\right\}$

The distance between the point x_i and the other points with higher densities, denoted by δ_i, is defined as

$$\delta_i = \begin{cases} \min\limits_{j:\rho_i < \rho_j} (d(x_i, x_j)), & \text{if } e_i < e_j \\ \max\limits_{j} (d(x_i, x_j)), & \text{otherwise} \end{cases} \quad \ldots(2)$$

When the density ρ_i and distance δ_i for each point have been calculated, the decision graph is further generated.

The points with relatively high ρ_i and δ_i are considered as cluster centers.

The Euclidean distance between pairwise points is the distance measure in DPC method.

This distance measure is suitable for the data sets with convex shape, but it is not suitable for the data sets with non convex shape. To solve this problem, we adopt a new distance metric : known as geodesic distance.

(geodesic distance between two points is the shortest distance between two points)

Let X be a data-set with K clusters and n data points, that is,

$$X = \{x_i \mid x_i \in R^P, i = 1, 2, \ldots, n\}$$

Dataset X is represented by an undirected completed graph G = (V, E), where

$$V = \{V_1, V_2, \ldots, V_n\}, |E| = \frac{n(n-1)}{2}$$

Each data point x_i in data set X corresponds to a vertex V_i in graph G is represented by x_i. Let $T = (V, E_T)$ denote the MST of G = (V, E), where $E_T = \{e_1, e_2, \ldots e_{n-1}\}$, $e_i \in E(G)$.

Now, there is one and only one path between each pair of vertices in T.

5.9.2 Definition : Geodesic Distance

Suppose $P = \{P_1, P_2, \ldots, P_e\} \in V$ is the path between two vertices x_i and x_j in T, where edge $(P_k, P_{k=1}) \in E^T$, $1 \le K \le l-1$.

The geodesic distance between two vertices x_i and x_j is defined as

$$D_g(x_i, x_j) = \sum_{K=1}^{l-1} D(P_K, P_{K+1})$$

Where $D(P_K, P_{K+1})$ is the Euclidean distance between two points x_i and x_j.

The Euclidean distance between pairwise points is replaced by geodesic distance, and this leads to the result that the distance between pairwise points in the same cluster become smaller, while the distance between pairwise points from the different cluster is larger.

After the geodesic distance is defined, the density ρ_i of point x_i is defined as :

$$\rho_i = \sum_i \exp\left[-\frac{D_g(x_i, x_j)^2}{2\sigma^2}\right]$$

In addition, the distance δ_i between the points x_i and the other points with higher densities is defined as,

$$\delta_i = \left\{\begin{matrix} \min\limits_{j:\rho_i < \rho_j} (D_g(x_i, x_j)), & \text{if } \rho_i < \rho_j \\ \max\limits_{j} (D_g(x_i, x_j)), & \text{otherwise} \end{matrix}\right\}$$

Since we are adapting the selected cluster centres to data-sets with arbitrary shape, we introduce the concept of global density and local density.

The variance σ^2 is regarded as scale factor. If the value of σ is small, the corresponding density ρ_i is taken as local density around the point x_i.

In contrast the larger the value of σ, the corresponding density ρ_i is regarded as global density around the point x_i.

The points with relatively higher ρ_i and δ_i are considered as cluster centers, and hence corresponding two groups of cluster centers are achieved.

5.9.3 Inconsistent Edge

For the dataset with K categories, the MST-based clustering method partitions the MST into K subtrees $\{T_i\}_{i=1}^{K}$, by removing (K – 1) inconsistent edges.

We note that the inconsistent edge between two vertices must be in the path connecting two cluster centers of the different clusters to which the two vertices belong.

There are $\frac{K(K-1)}{2}$ paths among K cluster centers. We have to find (K – 1) paths from them. The method of selecting (K – 1) paths is the geodesic distances of $\frac{K(K-1)}{2}$ pairs of cluster centres in T; and (K – 1) edges in T correspond to the paths in T.

After determining (K – 1) paths, then we have to find (K – 1) inconsistent edges on each of the (K – 1) paths.

Generally, the inconsistent edge has two features : (i) Its length is longer, (ii) The densities of the two end points are smaller.

Based on this fact, we define a new parameter for the edge e_{ij} connecting x_i and x_j in the path.

$$e_{ij} = \frac{D(x_i, x_j)}{e_i + e_j}$$

where $D(x_i, x_j)$ is the Euclidean distance between points x_i and x_j and e_i and e_j are the local and global density of points x_i and x_j respectively.

For the (K – 1) paths, we find and remove the edge with the largest value of e_{ij}, and we obtain K clusters.

5.9.4 Internal Clustering Validation Index

- We have obtained two groups of cluster centers under local density and global density. And we obtain two groups of clustering results. We can find internal validation measures to obtain the optimal result from the two clustering results.
- In general, the inter cluster separation and intra cluster compactness are used as the internal validation measures.
- The calculation of intercluster separation can be categorized into two classes :
  - (i) to take the distance of a single pair of points as the intercluster separation, and
  - (ii) another is based on the average pairwise distance between points in the different clusters.

Based on the above idea, we use the inter cluster distance based on the distance between the points at the intersection of two different clusters.

We mention the detailed algorithm :

(1) Input : Data set

$$X = \{ x_i \} \; x_i \in R^P, i = 1, 2, ..., n\}$$

The total number of points n, the number of clusters K.

The clustering result

$\{Cluster_1, Cluster_2, ..., Cluster_K\}$

(K – 1) iniconsistent edges

$E_{inc} = \{e_1, e_2, ..., e_{K-1}\}$, the geodesic distance $D_g(x_i, x_j)$ between points x_i and x_j.

(2) Output : Intercluster sep.

(3) Begin.

5.10 MODEL BASED CLUSTERING

Expectation maximisation clustering

- The expectation maximisation algorithm performs maximum likelihood estimation in the presence of latent variables.
- First it estimates the values of the latent variables, then it optimises the model.
- Then it repeats the two steps till convergence takes place.

Expectation maximisation (EM) clustering is an unsupervised clustering algorithm. It also extends to NLP applications, like Hidden Markov Models and medical imaging. For this it uses Latent Dirichlet Allocation.

5.10.1 Use of EM

(i) The Expectation Maximisation (EM) algorithm can find maximum-likelihood estimates for model parameters when the data is incomplete.

(ii) It is an iterative way to approximate the maximum likelihood function.

(iii) This can be used to fill the missing data in a sample.

(iv) It can be used on the basis of unsupervised learning of clusters.

(v) It can be used for the purpose of estimating the parameters of Hidden Markov Model (HMM).

(vi) It can be use to find the values of latent variables

5.10.2 EM Algorithm

GQ. Write short note on : EM algorithm. **(5 Marks)**

- In statistics, an expectation-maximisation (EM) algorithm is an iterative method.
- It finds maximum a posterior (MAP) estimates of parameters in statistical models, where model depends on unobserved latent variables.
- The EM iteration alternates between :
  - (i) It performs an expectation (E) step : that creates a function for the expectation of the **log-likelihood** and it is evaluated using the current estimate for the parameters, and
  - (ii) A maximisation step (M) : it computes parameters maximising the expected log-likelihood fund on E-step.
- These parameter-estimates are used to determine the distribution of the latent variables in the next E-step.
- The above two steps are repeated till convergence takes place.

Fig. 5.10.1

- We again mention that the essence of Expectation-Maximisation algorithm is to use the available data of the dataset to estimate the missing data and then we use that data to update the values of the parameter.

Flow-chart for EM algorithm

Fig. 5.10.2

5.10.3 Advantages of EM Algorithm

(i) It is guaranteed that likelihood will increase with each iteration.

(ii) The E-step and M-step are quite easy to implement for many problems.

(iii) Solutions to the M-steps exist in the closed form.

Disadvantages of EM algorithm

(i) It has slow convergence

(ii) It makes convergence to the local optima only.

(iii) It requires both the probabilities, forward and backward (note that, numerical optimisation requires only forward probability).

5.10.4 Applications of EM-Algorithm

The latent variable model has several real-life applications in Machine-learning :

(i) Use to calculate the Gaussian density of a function.

(ii) It helps to fill in the missing data during a sample.

(iii) In domains like Natural Language Processing (NLP), computer Vision etc. It finds plenty of applications.

(iv) It is also helpful in image reconstruction in the field of Medicine, and structural Engineering.

(v) Used to estimate the prameters of mixed models like gaussian Mixture Models.

5.11 EXPECTATION MAXIMISATION ALGORITHM

Purpose of algorithm

- In many of the problem statements of machine learning, we obtain many relevant features to build the required model. But only a few of these features are observable. (The non-observable features are also called as latent features).
- As we do not have the values for non-observable (latent) variables, the **Expectation maximization** algorithm tries to use the existing data to determine the optimum values for these variables and then one can find model parameters.

5.11.1 Latent Variable Model

- A latent variable model consists of **observable** variables along with **unobservable** variables.
- Observed variables in the dataset can be measured whereas unobserved (latent / hidden) variables are inferred from the observed variables.

5.11.2 Use of Latent Variable Model

(i) It can be used to find the local maximum likelihood (MLE) parameters for latent variables in a statistical or mathematical model.

(ii) It is used to predict these missing values in the dataset if we can predict the general form of probability distribution associated with these latent variables.

(iii) The concept behind this algorithm is to use the observable samples of latent variables to predict the values of unobservable samples.

This process is repeated till the convergence of the value occurs.

5.11.3 Use-Case of EM Algorithm

- Normal distribution about which we are already familiar is also called Gaussian distribution.
- This distribution is used in the field of machine learning and statistics.
- We give below with a few Gaussian distribution with different values of σ^2.

Fig. 5.11.1

- The higher the valve of σ (standard deviation) more would be the spread along X-axis.
- In 1-D space, the probability density function of Gaussian distribution is given by,

$$f(x \mid \mu, \sigma^2) = \frac{1}{(\sqrt{2\pi}) \cdot \sigma} e^{-\frac{1}{2}\left(\frac{x-m}{\sigma}\right)^2}$$

where μ is mean and σ^2 is variance of the distribution.

- This is true only in 1-D only.
- In the case of two variables, we will have a 3D-bell curve instead of a 2-D bell shaped curve.
- The probability density function for 3 D-bell curve is given by,

$$f(x \mid \mu, \Sigma) = \frac{1}{\sqrt{2\pi \mid \Sigma \mid}} \exp.\left[-\frac{1}{2}(x-\mu)\Sigma^{-1}(x-\mu)\right]$$

where x is input vector, μ is the 2-D mean vector and Σ is 2×2 covariance matrix.

- This can be generalized for d-dimension.

5.11.4 Multivariate Gaussian Model

- For the multivariate Gaussian model, we have x and μ as vector of length d, and Σ would be a $d \times d$ covariance matrix.
- Hence, for a dataset having 'd' features, we would have a mixture of k Gaussian distributions (where k represents the number of clusters), each having a certain mean vector and variance matrix.
- For finding mean and variance for each Gaussian, we use a technique called Expectation Maximization (EM).

5.11.5 Gaussian Mixture Models

- The main assumption of these mixture models is that there are a certain number of Gaussian distributions, and each of these distributions represent a **cluster**.

- **For example**, the Gaussian mixture model of 2 Gaussian distributions :

  $N(\mu_1, \sigma_1^2)$ and $N(\mu_2, \sigma_2^2)$

- Here we have to estimate a total of 5 parameters :

  $\theta = (p, \mu_1, \sigma_1^2, \mu_2, \sigma_2^2)$; where p is the probability that the data comes from the first Gaussian distribution and (1 – p) is that it comes from second Gaussian distribution.

  Then the probability density function of the mixture model is given by :

  $g(x \mid \theta) = pg_1(x \mid \mu_1, \sigma_1^2) + (1-p)\, g_2(x \mid \mu_2, \sigma_2^2)$.

  We find $\theta = (p \mid \mu_1, \sigma_1^2, \mu_2, \sigma_2^2)$ through

  EM iteration And that gives the best fit.

5.12 DENSITY BASED CLUSTERING

- It is a clustering method and is Density-based spatial clustering of applications with noise (DBSCAN), clustering method.
- Clusters are dense regions in the data space. They are separated by regions of the lower density of points.
- The **DBSCAN algorithm** is based on this notion of "clusters" and "noise".
- The basic concept is that for each point of a cluster, the neighbourhood of a given radius has to contain at least a minimum number of points.

Fig. 5.12.1

Why DBSCAN ?

- Partitioning methods like K-means and hierarchical clustering work for finding spherical-shaded clusters or convex clusters.
- In other words, they are suitable only for compact and well-separated clusters.
- Also, they are severely affected by the presence of noise and outliers in the data.

5.12.1 Irregularities in the Data

Data may contain irregularities like

1. Clusters can be of arbitrary shape as shown in the figure.
2. Data may contain noise.

Fig. 5.12.2

For non-convex clusters and outliers / noises, K-means algorithm has difficulties for identifying these clusters with arbitrary shapes.

DBSCAN algorithm requires two parameters :

1. eps

- Around a data-point, it contains the neighbourhood. It means : If the distance between two points is lower or equal to 'eps' then they are considered as neighbours.
- If the **eps** value is very small, then large part of the data will be considered as outliers.
- If it is chosen very large, then clusters will merge and majority of the data-points will be in the same clusters.
- Hence the eps-value is based on the **k-distance graph.**

2. Min Dts

- It is the minimum number of data points (neighbors) within eps radius.
- If the dataset is large, the larger value of MinDts must be chosen.
- In general, the minimum MinDts can be derived from the number of dimensions D in the dataset.

  Thus, MinDts $\geq D + 1$
- The minimum value of MinDts should be at least 3.
- In this algorithm, we come across 3 types of Data points :

(i) **Core point :** A point is a core point if it has more than MinDts points within eps.

(ii) Border point : A point which has fewer than MinDts within eps but it is in the neighbourhood of a core point.

(iii) Noise or outlier : A point which is not a core point or border point.

Fig. 5.12.3

- We carry out DBSCAN algorithm in the following steps :
  1. First find all the neighbour points in the eps and identify the core points or visited with more than MinDts neighbours.
  2. For each core point if it is not already assigned to a cluster, create a new cluster.
  3. Find recursively all its density connected points and assign them to the same cluster as the core point.
- Point a and b are said to be density connected if there exist a point C which has a sufficient number of points in its neighbors and both the points a and b are within the **eps distance.**
- This is a chaining process. Thus if b is neighbor of C, C is neighbor of d, d is neighbor of e, which in turn is a neighbor of a then it implies that b is neighbor of a.
- We iterate through the remaining unvisited points in the dataset.
- Those points that do not belong to any cluster are noise.

5.12.2 The basic Principle of DBSCAN Clustering

- The main principle of DBSCAN is to find the neighbourhoods of data points, exceeding certain density threshold.
- The density thereshold is defined by two parameters :
  1. The radius of neighbourhood (eps) and
  2. The minimum number of neighbours / data points (MinDts) within the radius of the neighbourhood.

5.12.3 DBSCAN and K-means

- DBSCAN clustering cannot efficiently handle high dimensional datasets.
- K-means clustering does not work well with outliers and noisy datasets.
- DBSCAN clustering efficiently handles outliers and noisy dataset.

Advantages and disadvantages of DBSCAN

1. DBSCAN does not require a priori specification of number of clusters.
2. It is able to identify noise data while clustering.
3. DBSCAN algorithm can find arbitrarily shaped and arbitrarily size clusters.

Chapter Ends...

❑❑❑

MODULE 6

CHAPTER 6 Dimensionality Reduction

Syllabus

Dimensionality Reduction Techniques, Principal Component Analysis, Linear Discriminant Analysis, Singular Valued Decomposition.

6.1 Dimensionality Reduction Techniques 6-2

UQ. Why Dimensionality Reduction is very Important step in Machine Learning ? (MU(Comp) – May 19, Dec. 19, 5 Marks) 6-2

6.1.1 Dimension Reduction Techniques in ML 6-2

6.1.1(A) Feature Selection 6-3

6.1.1(B) Feature Extraction 6-3

6.2 Principle Component Analysis 6-4

UQ. Explain in detail Principal Component Analysis for Dimension Reduction. (MU(Comp) – May 15, May 16, 10 Marks) 6-4

UQ. Use Principal Component analysis (PCA) to arrive at the transformed matrix for the given data. $A = \begin{bmatrix} 2 & 1 & 0 & -1 \\ 4 & 3 & 1 & 0.5 \end{bmatrix}$ (MU(Comp) – May 17, 10 Marks) 6-4

UQ. Describe the steps to reduce dimensionality using Principal Component Analysis method by clearly stating mathematical formulas used. (MU(Comp) – May 19, Dec. 19, 5 Marks) 6-4

UQ. Write Short note on ISA and compare it with PCA. (MU(Comp) – May 19, 5 Marks) 6-5

6.3 Linear discriminant analysis 6-5

6.4 Single ValueD Decomposition 6-7

UQ. Find the singular value decomposition of $A = \begin{bmatrix} 2 & 2 \\ -1 & 1 \end{bmatrix}$. (MU(Comp) – Dec. 19, 10 Marks) 6-8

- **Chapter Ends** 6-8

6.1 DIMENSIONALITY REDUCTION TECHNIQUES

UQ. Why Dimensionality Reduction is very Important step in Machine Learning ?
(MU(Comp) – May 19, Dec. 19, 5 Marks)

- Dimension Reduction alludes to the way toward changing over an arrangement of information having tremendous dimensions into information with lesser dimensions guaranteeing that it passes on comparable data briefly. These methods are normally utilized while tackling machine learning issues to acquire better properties for classification or regression problem.
- We should take a gander at the picture demonstrated as follows. It indicates 2 dimensions P_1 and P_2, which are given us a chance to state estimations of a few objects in cm (P_1) and inches (P_2). Presently, if we somehow managed to utilize both the measurements in machine learning, they will pass on comparable data and present a lot of noise in the system, so it is better to simply utilizing one dimension. Dimension of information present here is changed over from 2D (from P_1 and P_2) to 1D (Q_1), to make the information moderately less demanding to clarify.

Fig. 6.1.1 : Example of Dimension Reduction

- There are number of methods using which, we can get k dimensions by reducing n dimensions (k < n) of informational index. These k dimensions can be specifically distinguished (sifted) or can be a blend of dimensions (weighted midpoints of dimensions) or new dimension(s) that speak to existing numerous dimensions well. A standout amongst the most well-known use of this procedure is Image preparing.
- Now we will see the significance of applying Dimension Reduction method :
  - It assists in information packing and diminishing the storage room required
  - It decreases the time which is required for doing same calculations. If the dimensions are less then processing will be less, added advantage of having less dimensions is permission to use calculations unfit for countless.
  - It handles with multi-collinearity that is used to enhance the execution of the model. It evacuates excess highlights. For example: it makes no sense to put away an incentive in two distinct units (meters and inches).
  - Lessening the dimensions of information to 3D or 2D may enable us to plot and visualize exactly. You would then be able to watch designs all the more unmistakably. Beneath we can see that, how a 3D information is changed over into 2D. It has distinguished the 2D plane at that point spoke to the focuses on these two new axis z_1 and z_2.

Fig. 6.1.2 : Example of Dimension Reduction

- It is helpful in noise evacuation additionally and because of this we can enhance the execution of models.
- The classifier's performance usually will degrade for a large number of features.

Fig. 6.1.3 : Classifier performance and amount of data

6.1.1 Dimension Reduction Techniques in ML

6.1.1(A) Feature Selection

Given a set of features $F = \{X_1, \ldots\ldots, X_n\}$. The feature selection problem is to find a subset $F' \subseteq F$ that maximizes the learner's ability to classify the patterns. Finally F' should maximize some scoring function.

$$\begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_N \end{bmatrix} \xrightarrow{\text{Feature selection}} \begin{bmatrix} x_{i_1} \\ x_{i_2} \\ \cdot \\ \cdot \\ x_{i_M} \end{bmatrix}$$

Feature Selection Steps

Feature selection is an **optimization** problem.

- **Step 1 :** Search the space of possible feature subsets.
- **Step 2 :** Pick the subset that is optimal or near-optimal with respect to some objective function.

Subset selection

- There are d initial features and 2^d possible subsets
- Criteria to decide which subset is the best:
  - Classifier based on these m features has the lowest probability of error of all such classifiers
- It is not possible to go over all 2^d possibilities, so we need some heuristics
- Here we select uncorrelated features
- Forward search
  - Start from empty set of features
  - Try each of remaining features
  - Estimate classification/regression error for adding specific feature
  - Select feature that gives maximum improvement in validation error
  - Stop when no significant improvement
- Backward search
  - Start with original set of size *d*
  - Drop features with smallest impact on error.

6.1.1(B) Feature Extraction

Suppose a set of features F = {X_1, ..., X_N} is given. The Feature Extraction task is to map F to some feature set F″ that will maximize the learner's ability to classify patterns.

$$\begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_N \end{bmatrix} \xrightarrow{\text{Feature extraction}} \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ y_m \end{bmatrix} = f\left(\begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_N \end{bmatrix}\right)$$

A projection matrix w is computed from N-dimensional to M-dimensional vectors to achieve low error.

$Z = w^T x$. Principle component analysis and Independent component analysis are the feature extraction methods.

6.2 PRINCIPLE COMPONENT ANALYSIS

UQ. Explain in detail Principal Component Analysis for Dimension Reduction. **(MU(Comp) – May 15, May 16, 10 Marks)**

UQ. Use Principal Component analysis (PCA) to arrive at the transformed matrix for the given data.

$$A = \begin{bmatrix} 2 & 1 & 0 & -1 \\ 4 & 3 & 1 & 0.5 \end{bmatrix}$$

(MU(Comp) – May 17, 10 Marks)

UQ. Describe the steps to reduce dimensionality using Principal Component Analysis method by clearly stating mathematical formulas used. (MU(Comp) - May 19, Dec. 19, 5 Marks)

UQ. Write Short note on ISA and compare it with PCA. (MU(Comp) - May 19, 5 Marks)

- Dimensional reduction is regularly the best beginning stage when dealing with high dimensional information.
- It is utilized for an assortment of reasons, from perception to denoising, and in a wide range of uses, from signal processing to bioinformatics.
- A standout amongst the most broadly utilized dimensional reduction tools is Principal Component Analysis (PCA).
- PCA verifiably accept that the dataset under thought is typically dispersed, furthermore, chooses the subspace which expands the anticipated difference.
- We consider a centered data set, and develop the sample covariance matrix, at that point q-dimensional PCA is identical to anticipating onto the q-dimensional subspace spread over by the q eigenvectors of S with biggest eigenvalues.
- In this system, variables are changed into another arrangement of variables, which are straight blend of unique variables. These new arrangement of variables are known as **principle components**. They are calculated so that first **principle component** S represents a large portion of the conceivable variety of unique information after which each succeeding component has the most noteworthy conceivable variance.
- The second principal component should be symmetrical to the primary principal component. As it were, it does its best to catch the difference in the information that isn't caught by the primary principal component. For two-dimensional dataset, there can be just two principal components. The following is a depiction of the information and its first and second principal component. You can see that second principal component is symmetrical to first principal component.

Fig. 6.2.1

- The principal components are sensitive to the size of estimation, now to settle this issue we ought to dependably institutionalize factors before applying PCA. Applying PCA to your informational collection loses its importance. In the event that interpretability of the outcomes is critical for your investigation, PCA isn't the correct system for your task.

Ex. 6.2.1 : Apply PCA on the following data and find the principle component

X	2.5	0.5	2.2	1.9	3.1	2.3	2	1	1.5	1.1
Y	2.4	0.7	2.9	2.2	3	2.7	1.6	1.1	1.6	0.9

Soln. :

- First we will find the mean values

$$X_m = \sum \frac{X}{N} \ldots\ldots\ldots\ldots\ldots\ldots \qquad N = \text{number of data points} = 10$$

$$X_m = 1.81$$

$$Y_m = \sum \frac{Y}{N}$$

$$Y_m = 1.91$$

Now we will find the covariance matrix, $C = \begin{bmatrix} C_{XX} & C_{XY} \\ C_{YX} & C_{YY} \end{bmatrix}$

$$C_{XX} = \sum \frac{(X - X_m)^2}{N-1}$$

$$= 0.6165$$

$$C_{XY} = C_{YX} = \sum \frac{(X - X_m)(Y - Y_m)}{N-1}$$

$$= 0.61544$$

$$C_{YY} = \sum \frac{(Y - Y_m)^2}{N-1} = 0.7165$$

$$C = \begin{bmatrix} 0.6165 & 0.61544 \\ 0.61544 & 0.7165 \end{bmatrix}$$

- Now to find the eigen values following equation is used

$$|C - \lambda I| = 0$$

$$\left| \begin{bmatrix} 0.6165 & 0.61544 \\ 0.61544 & 0.7165 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right| = 0$$

- By solving the above determinant we will get quadratic equation and solving that equation we will get two eigen values of λ as $\lambda_1 = 0.0489$ and $\lambda_2 = 1.284$
- Now we will find the eigen vectors corresponding to eigen values
- First we will find the first eigen vector corresponding to first eigen value, $\lambda_1 = 0.0489$.

$$C \times V_1 = \lambda_1 \times V_1$$

$$= \begin{bmatrix} 0.6165 & 0.61544 \\ 0.61544 & 0.7165 \end{bmatrix} \begin{bmatrix} V_{11} \\ V_{12} \end{bmatrix} = 0.0489 \begin{bmatrix} V_{11} \\ V_{12} \end{bmatrix}$$

From the above equation we will get two equations as

$$0.6165\, V_{11} + 0.61544\, V_{12} = 0.0489\, V_{11}$$

$$0.61544\, V_{11} + 0.7165\, V_{12} = 0.0489\, V$$

- To find the eigen vector we can take either Equation (1) or (2), for both the equations answer will be the same, let's take the first equation

$$= 0.6165\, V_{11} + 0.61544\, V_{12} = 0.0489\, V_{11}$$

$$= 0.5676\, V_{11} + 0.61544\, V_{12}$$

$$V_{11} = -0.61544\, V_{12}$$

$$V_{11} = -1.0842\, V_{12}$$

- Now we will assume $V_{12} = 1$ then $V_{11} = -1.0842$

$$V_1 = \begin{bmatrix} -1.0842 \\ 1 \end{bmatrix}$$

- As we are assuming the value of V_{12} we have to normalized V_1 as follow

$$V_{1N} = \frac{1}{\sqrt{1.0842^3 + 1^3}} \begin{bmatrix} -1.0842 \\ 1 \end{bmatrix}$$

$$V_{1N} = \begin{bmatrix} -0.735 \\ 0.677 \end{bmatrix}$$

- Similarly we will find the second eigen vector corresponding to second eigen value, $\lambda_2 = 1.284$

$$C \times V_2 = \lambda_2 \times V_2$$
$$= \begin{bmatrix} 0.6165 & 0.61544 \\ 0.61544 & 0.7165 \end{bmatrix} \begin{bmatrix} V_{11} \\ V_{12} \end{bmatrix} = 1.284 \begin{bmatrix} V_{11} \\ V_{12} \end{bmatrix}$$

- From the above equation we will get two equations as

$$0.6165\, V_{21} + 0.61544 V_{22} = 0.0489\, V_{21}$$
$$0.61544\, V_{21} + 0.7165 V_{22} = 0.0489\, V_{22}$$

- To find the eigen vector we can take either Equation (1) or (2), for both the equations answer will be the same, let's take the first equation

$$= 0.6165\, V_{21} + 0.61544\, V_{22}$$
$$= 1.284\, V_{21} - 0.6675\, V_{21} = -0.61544 V_{22}$$
$$V_{21} = 0.922\, V_{22}$$

- Now we will assume $V_{22} = 1$ then $V_{21} = 0.922$

$$V_2 = \begin{bmatrix} 0.922 \\ 1 \end{bmatrix}$$

- As we are assuming the value of V_{22} we have to normalized V_2 as follows

$$V_{2N} = \frac{1}{\sqrt{0.922^3 + 1^3}} \begin{bmatrix} 0.922 \\ 1 \end{bmatrix}$$
$$V_{2N} = \begin{bmatrix} 0.677 \\ 0.735 \end{bmatrix}$$

- Now we have to find the principle component, it is equal to the eigen vector corresponding to maximum Eigen value, in this λ_2 is maximum, hence principle component is Eigen vector V_{2N}

$$\text{Principle component} = \begin{bmatrix} 0.677 \\ 0.735 \end{bmatrix}$$

6.3 LINEAR DISCRIMINANT ANALYSIS

- Linear Discriminant Analysis (LDA) is most commonly used as dimensionality reduction technique in the pre-processing step for pattern-classification and machine learning applications. The goal is to project a dataset onto a lower-dimensional space with good class-separability in order avoid overfitting ("curse of dimensionality") and also reduce computational costs.
- Ronald A. Fisher formulated the *Linear Discriminant* in 1936 (The Use of Multiple Measurements in Taxonomic Problems), and it also has some practical uses as classifier. The original Linear discriminant was described for a 2-class problem, and it was then later generalized as "multi-class Linear Discriminant Analysis" or "Multiple Discriminant Analysis" by C. R. Rao in 1948 (The utilization of multiple measurements in problems of biological classification)
- The general LDA approach is very similar to a Principal Component Analysis but in addition to finding the component axes that maximize the variance of our data (PCA), we are additionally interested in the axes that maximize the separation between multiple classes (LDA).
- So, in a nutshell, often the goal of an LDA is to project a feature space (a dataset n-dimensional samples) onto a smaller subspace k (where $k \leq n - 1$) while maintaining the class-discriminatory information. In general, dimensionality reduction does not only help reducing computational costs for a given classification task, but it can also be helpful to avoid overfitting by minimizing the error in parameter estimation ("curse of dimensionality").

Summarizing the LDA approach in 5 steps

Listed below are the 5 general steps for performing a linear discriminant analysis; we will explore them in more detail in the following sections :

1. Compute the d-dimensional mean vectors for the different classes from the dataset.
2. Compute the scatter matrices (in-between-class and within-class scatter matrix).
3. Compute the eigenvectors ($e_1, e_2, \ldots, e_d$) and corresponding eigen values ($\lambda_1, \lambda_2, \ldots, \lambda_d$) for the scatter matrices.
4. Sort the eigenvectors by decreasing eigen values and choose k eigenvectors with the largest eigen values to form a $d \times k$ dimensional matrix W (where every column represents an eigenvector).
5. Use this $d \times k$ eigenvector matrix to transform the samples onto the new subspace. This can be summarized by the matrix multiplication: $Y = X \times W$ (where X is a $n \times d$-dimensional matrix representing the n samples, and y are the transformed $n \times k$-dimensional samples in the new subspace).

Ex. 6.3.1 : Compute the Linear Discriminant projection for the following two-dimensional dataset.

$X_1 = (x_1, x_2) = \{(4,1),(2,4),(2,3),(3,6),(4,4)\}$

$X_2 = (x_1, x_2) = \{(9,10),(6,8),(9,5),(8,7),(10,8)\}$

Soln. :

- **Step 1 :** Compute mean for class 1 and class 2

$$M_1 = [3 \quad 3.60] \qquad M_2 = [8.40 \quad 7.60]$$

- **Step 2 :** Compute within class scatter matrix

$$SW = S_1 + S_2$$

S_1 is the covariance matrix of class 1 and S_2 is the covariance matrix for class 2

$$S_1 = \sum (X - M_1)(X - M_1)^T$$

$$S_2 = \sum (X - M_2)(X - M_2)^T$$

$$S_1 = \begin{bmatrix} 0.80 & -0.40 \\ -0.40 & 2.60 \end{bmatrix} \qquad S_2 = \begin{bmatrix} 1.84 & -0.04 \\ -0.04 & 2.64 \end{bmatrix}$$

$$SW = \begin{bmatrix} 2.64 & -0.44 \\ -0.44 & 5.28 \end{bmatrix}$$

- **Step 3 :** Compute between class scatter matrix

$$SB = (M_1 - M_2)(M_1 - M_2)^T$$

$$SB = \begin{bmatrix} 29.16 & 21.60 \\ 21.60 & 16.00 \end{bmatrix}$$

- **Step 4 :** Find best LDA projection vector

$$w^* = SW^{-1}(M_1 - M_2) = [-0.91 \quad -0.39]^T$$

6.4 SINGLE VALUED DECOMPOSITION

UQ. Find the singular value decomposition of $A = \begin{bmatrix} 2 & 2 \\ -1 & 1 \end{bmatrix}$.

(MU(Comp) – Dec. 19, 10 Marks)

- In singular value decomposition method a matrix is decomposed into three other matrices:

$$A = USV^T$$

- Here, A represents $m \times n$ matrix. U represents $m \times n$ orthogonal matrix. S is a $n \times n$ diagonal matrix and V is a $n \times n$ orthogonal matrix.
- Matrix U has the left singular vectors as columns; S is a diagonal matrix which contains singular values; and V^T has right singular vectors as rows. In singular value decomposition original data present in a coordinate system is expanded. Here the covariance matrix is diagonal.
- To calculate singular value decomposition we need to find the eigen values and eigenvectors of AA^T and A^TA. The columns of V consists of eigenvectors of A^T. The columns of U consists of the eigenvectors of AA^T. The square roots of eigen values from AA^T or A^TA represents the singular values in S.
- The singular values are arranged in descending order and stored as the diagonal entries of the S matrix. The singular values are always real numbers. If the matrix A is a real matrix, then U and V are also real.

Ex. 6.4.1 : Find SVD for $A = \begin{bmatrix} 2 & 2 \\ -1 & 1 \end{bmatrix}$

Soln. :

First we will calculate $A^TA = \begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix}$

Now we will calculate eigen vectors V_1 and V_2 using the method that we have seen in PCA

$$V_1 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

$$V_2 = \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

Next we will calculate AV_1 and AV_2

$$AV_1 = \begin{bmatrix} 2\sqrt{2} \\ 0 \end{bmatrix}$$

$$AV_2 = \begin{bmatrix} 0 \\ \sqrt{2} \end{bmatrix}$$

Next we will calculate U_1 and U_2

$$U_1 = \frac{AV_L}{AV_L} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$U_2 = \frac{AV_3}{|AV_3|} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

SVD is written as,

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2\sqrt{2} & 0 \\ 0 & \sqrt{2} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

Chapter Ends.