Chapter 1

Introduction to Machine Learning

"Computers are able to see, hear and learn. Welcome to the future."
— Dave Waters

Machine Learning (ML) is a promising and flourishing field. It can enable top management of an organization to extract the knowledge from the data stored in various archives of the business organizations to facilitate decision making. Such decisions can be useful for organizations to design new products, improve business processes, and to develop decision support systems.

Learning Objectives

- Explore the basics of machine learning
- Introduce types of machine learning
- Provide an overview of machine learning tasks
- State the components of the machine learning algorithm
- Explore the machine learning process
- Survey some machine learning applications

1.1 NEED FOR MACHINE LEARNING

Business organizations use huge amount of data for their daily activities. Earlier, the full potential of this data was not utilized due to two reasons. One reason was data being scattered across different archive systems and organizations not being able to integrate these sources fully. Secondly, the lack of awareness about software tools that could help to unearth the useful information from data. Not anymore! Business organizations have now started to use the latest technology, machine learning, for this purpose.

Machine learning has become so popular because of three reasons:

1. High volume of available data to manage: Big companies such as Facebook, Twitter, and YouTube generate huge amount of data that grows at a phenomenal rate. It is estimated that the data approximately gets doubled every year.

2. Second reason is that the cost of storage has reduced. The hardware cost has also dropped. Therefore, it is easier now to capture, process, store, distribute, and transmit the digital information.
3. Third reason for popularity of machine learning is the availability of complex algorithms now. Especially with the advent of deep learning, many algorithms are available for machine learning.

With the popularity and ready adaption of machine learning by business organizations, it has become a dominant technology trend now. Before starting the machine learning journey, let us establish these terms - data, information, knowledge, intelligence, and wisdom. A knowledge pyramid is shown in Figure 1.1.

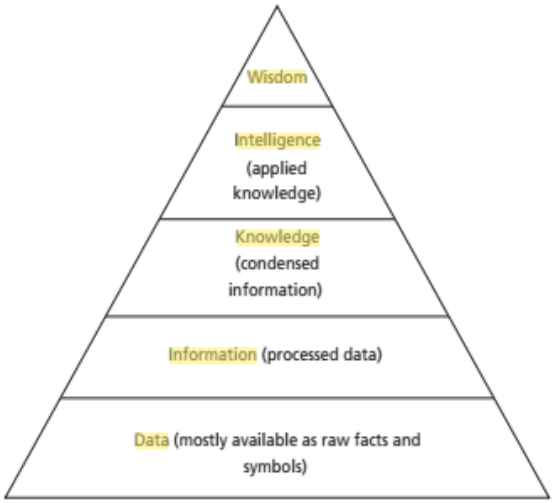


Figure 1.1: The Knowledge Pyramid

What is data? All facts are data. Data can be numbers or text that can be processed by a computer. Today, organizations are accumulating vast and growing amounts of data with data sources such as flat files, databases, or data warehouses in different storage formats.

Processed data is called information. This includes patterns, associations, or relationships among data. For example, sales data can be analyzed to extract information like which is the fast selling product. Condensed information is called knowledge. For example, the historical patterns and future trends obtained in the above sales data can be called knowledge. Unless knowledge is extracted, data is of no use. Similarly, knowledge is not useful unless it is put into action. Intelligence is the applied knowledge for actions. An actionable form of knowledge is called intelligence. Computer systems have been successful till this stage. The ultimate objective of knowledge pyramid is wisdom that represents the maturity of mind that is, so far, exhibited only by humans.

Here comes the need for machine learning. The objective of machine learning is to process these archival data for organizations to take better decisions to design new products, improve the business processes, and to develop effective decision support systems.

1.2 MACHINE LEARNING EXPLAINED

Machine learning is an important sub-branch of Artificial Intelligence (AI). A frequently quoted definition of machine learning was by Arthur Samuel, one of the pioneers of Artificial Intelligence. He stated that *"Machine learning is the field of study that gives the computers ability to learn without being explicitly programmed."*

The key to this definition is that the systems should learn by itself without explicit programming. How is it possible? It is widely known that to perform a computation, one needs to write programs that teach the computers how to do that computation.

In conventional programming, after understanding the problem, a detailed design of the program such as a flowchart or an algorithm needs to be created and converted into programs using a suitable programming language. This approach could be difficult for many real-world problems such as puzzles, games, and complex image recognition applications. Initially, artificial intelligence aims to understand these problems and develop general purpose rules manually. Then, these rules are formulated into logic and implemented in a program to create intelligent systems. This idea of developing intelligent systems by using logic and reasoning by converting an expert's knowledge into a set of rules and programs is called an expert system. An expert system like MYCIN was designed for medical diagnosis after converting the expert knowledge of many doctors into a system. However, this approach did not progress much as programs lacked real intelligence. The word MYCIN is derived from the fact that most of the antibiotics' names end with 'mycin'.

The above approach was impractical in many domains as programs still depended on human expertise and hence did not truly exhibit intelligence. Then, the momentum shifted to machine learning in the form of data driven systems. The focus of AI is to develop intelligent systems by using data-driven approach, where data is used as an input to develop intelligent models. The models can then be used to predict new inputs. Thus, the aim of machine learning is to learn a model or set of rules from the given dataset automatically so that it can predict the unknown data correctly.

As humans take decisions based on an experience, computers make models based on extracted patterns in the input data and then use these data-filled models for prediction and to take decisions. For computers, the learnt model is equivalent to human experience. This is shown in Figure 1.2.

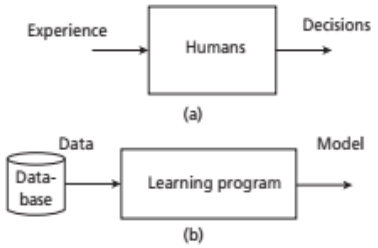


Figure 1.2: (a) A Learning System for Humans (b) A Learning System for Machine Learning

Often, the quality of data determines the quality of experience and, therefore, the quality of the learning system. In statistical learning, the relationship between the input x and output y is

modeled as a function in the form $y = f(x)$. Here, f is the learning function that maps the input x to output y. Learning of function f is the crucial aspect of forming a model in statistical learning. In machine learning, this is simply called mapping of input to output.

The learning program summarizes the raw data in a model. Formally stated, a model is an explicit description of patterns within the data in the form of:

1. Mathematical equation
2. Relational diagrams like trees/graphs
3. Logical if/else rules, or
4. Groupings called clusters

In summary, a model can be a formula, procedure or representation that can generate data decisions. The difference between pattern and model is that the former is local and applicable only to certain attributes but the latter is global and fits the entire dataset. For example, a model can be helpful to examine whether a given email is spam or not. The point is that the model is generated automatically from the given data.

Another pioneer of AI, Tom Mitchell's definition of machine learning states that, *"A computer program is said to learn from experience E, with respect to task T and some performance measure P, if its performance on T measured by P improves with experience E."* The important components of this definition are experience E, task T, and performance measure P.

For example, the task T could be detecting an object in an image. The machine can gain the knowledge of object using training dataset of thousands of images. This is called experience E. So, the focus is to use this experience E for this task of object detection T. The ability of the system to detect the object is measured by performance measures like precision and recall. Based on the performance measures, course correction can be done to improve the performance of the system.

Models of computer systems are equivalent to human experience. Experience is based on data. Humans gain experience by various means. They gain knowledge by rote learning. They observe others and imitate it. Humans gain a lot of knowledge from teachers and books. We learn many things by trial and error. Once the knowledge is gained, when a new problem is encountered, humans search for similar past situations and then formulate the heuristics and use that for prediction. But, in systems, experience is gathered by these steps:

1. Collection of data
2. Once data is gathered, abstract concepts are formed out of that data. Abstraction is used to generate concepts. This is equivalent to humans' idea of objects, for example, we have some idea about how an elephant looks like.
3. Generalization converts the abstraction into an actionable form of intelligence. It can be viewed as ordering of all possible concepts. So, generalization involves ranking of concepts, inferencing from them and formation of heuristics, an actionable aspect of intelligence. Heuristics are educated guesses for all tasks. For example, if one runs or encounters a danger, it is the resultant of human experience or his heuristics formation. In machines, it happens the same way.
4. Heuristics normally works! But, occasionally, it may fail too. It is not the fault of heuristics as it is just a 'rule of thumb'. The course correction is done by taking evaluation measures. Evaluation checks the thoroughness of the models and to-do course correction, if necessary, to generate better formulations.

1.3 MACHINE LEARNING IN RELATION TO OTHER FIELDS

Machine learning uses the concepts of Artificial Intelligence, Data Science, and Statistics primarily. It is the resultant of combined ideas of diverse fields.

1.3.1 Machine Learning and Artificial Intelligence

Machine learning is an important branch of AI, which is a much broader subject. The aim of AI is to develop intelligent agents. An agent can be a robot, humans, or any autonomous systems. Initially, the idea of AI was ambitious, that is, to develop intelligent systems like human beings. The focus was on logic and logical inferences. It had seen many ups and downs. These down periods were called AI winters.

The resurgence in AI happened due to development of data driven systems. The aim is to find relations and regularities present in the data. Machine learning is the subbranch of AI, whose aim is to extract the patterns for prediction. It is a broad field that includes learning from examples and other areas like reinforcement learning. The relationship of AI and machine learning is shown in Figure 1.3. The model can take an unknown instance and generate results.

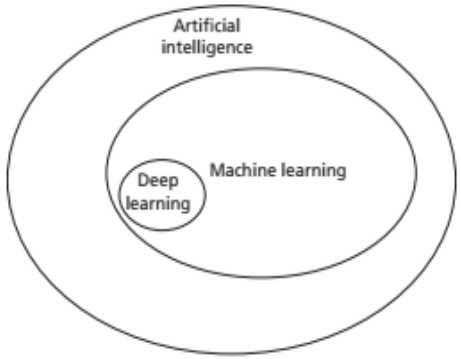


Figure 1.3: Relationship of AI with Machine Learning

Deep learning is a subbranch of machine learning. In deep learning, the models are constructed using neural network technology. Neural networks are based on the human neuron models. Many neurons form a network connected with the activation functions that trigger further neurons to perform tasks.

1.3.2 Machine Learning, Data Science, Data Mining, and Data Analytics

Data science is an 'Umbrella' term that encompasses many fields. Machine learning starts with data. Therefore, data science and machine learning are interlinked. Machine learning is a branch of data science. Data science deals with gathering of data for analysis. It is a broad field that includes:

Big Data Data science concerns about collection of data. Big data is a field of data science that deals with data's following characteristics:

1. Volume: Huge amount of data is generated by big companies like Facebook, Twitter, YouTube.
2. Variety: Data is available in variety of forms like images, videos, and in different formats.
3. Velocity: It refers to the speed at which the data is generated and processed.

Big data is used by many machine learning algorithms for applications such as language translation and image recognition. Big data influences the growth of subjects like Deep learning. Deep learning is a branch of machine learning that deals with constructing models using neural networks.

Data Mining Data mining's original genesis is in the business. Like while mining the earth one gets into precious resources, it is often believed that unearthing of the data produces hidden information that otherwise would have eluded the attention of the management. Nowadays, many consider that data mining and machine learning are same. There is no difference between these fields except that data mining aims to extract the hidden patterns that are present in the data, whereas, machine learning aims to use it for prediction.

Data Analytics Another branch of data science is data analytics. It aims to extract useful knowledge from crude data. There are different types of analytics. Predictive data analytics is used for making predictions. Machine learning is closely related to this branch of analytics and shares almost all algorithms.

Pattern Recognition It is an engineering field. It uses machine learning algorithms to extract the features for pattern analysis and pattern classification. One can view pattern recognition as a specific application of machine learning.

These relations are summarized in Figure 1.4.

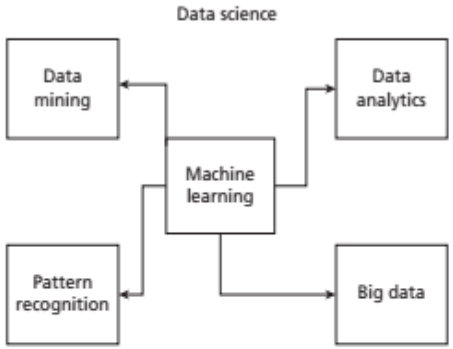


Figure 1.4: Relationship of Machine Learning with Other Major Fields

1.3.3 Machine Learning and Statistics

Statistics is a branch of mathematics that has a solid theoretical foundation regarding statistical learning. Like machine learning (ML), it can learn from data. But the difference between statistics and ML is that statistical methods look for regularity in data called patterns. Initially, statistics sets a hypothesis and performs experiments to verify and validate the hypothesis in order to find relationships among data.

Statistics requires knowledge of the statistical procedures and the guidance of a good statistician. It is mathematics intensive and models are often complicated equations and involve many assumptions. Statistical methods are developed in relation to the data being analysed. In addition, statistical methods are coherent and rigorous. It has strong theoretical foundations and interpretations that require a strong statistical knowledge.

Machine learning, comparatively, has less assumptions and requires less statistical knowledge. But, it often requires interaction with various tools to automate the process of learning.

Nevertheless, there is a school of thought that machine learning is just the latest version of 'old Statistics' and hence this relationship should be recognized.

1.4 TYPES OF MACHINE LEARNING

What does the word 'learn' mean? Learning, like adaptation, occurs as the result of interaction of the program with its environment. It can be compared with the interaction between a teacher and a student. There are four types of machine learning as shown in Figure 1.5.

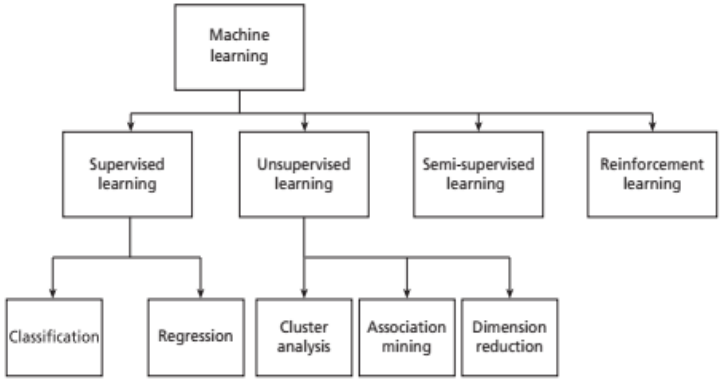


Figure 1.5: Types of Machine Learning

Before discussing the types of learning, it is necessary to discuss about data.

Labelled and Unlabelled Data Data is a raw fact. Normally, data is represented in the form of a table. Data also can be referred to as a data point, sample, or an example. Each row of the table represents a data point. Features are attributes or characteristics of an object. Normally, the columns of the table are attributes. Out of all attributes, one attribute is important and is called a label. Label is the feature that we aim to predict. Thus, there are two types of data – labelled and unlabelled.

Labelled Data To illustrate labelled data, let us take one example dataset called Iris flower dataset or Fisher's Iris dataset. The dataset has 50 samples of Iris – with four attributes, length and width of sepals and petals. The target variable is called class. There are three classes – Iris setosa, Iris virginica, and Iris versicolor.

The partial data of Iris dataset is shown in Table 1.1.

Table 1.1: Iris Flower Dataset

S.No.	Length of Petal	Width of Petal	Length of Sepal	Width of Sepal	Class
1.	5.5	4.2	1.4	0.2	Setosa
2.	7	3.2	4.7	1.4	Versicolor
3.	7.3	2.9	6.3	1.8	Virginica

A dataset need not be always numbers. It can be images or video frames. Deep neural networks can handle images with labels. In the following Figure 1.6, the deep neural network takes images of dogs and cats with labels for classification.

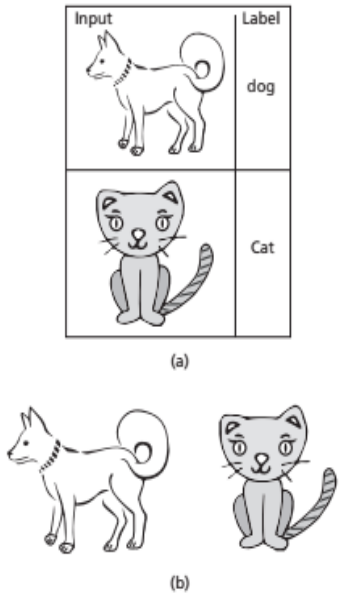


Figure 1.6: (a) Labelled Dataset (b) Unlabelled Dataset

In unlabelled data, there are no labels in the dataset.

1.4.1 Supervised Learning

Supervised algorithms use labelled dataset. As the name suggests, there is a supervisor or teacher component in supervised learning. A supervisor provides labelled data so that the model is constructed and generates test data.

In supervised learning algorithms, learning takes place in two stages. In layman terms, during the first stage, the teacher communicates the information to the student that the student is supposed to master. The student receives the information and understands it. During this stage, the teacher has no knowledge of whether the information is grasped by the student.

This leads to the second stage of learning. The teacher then asks the student a set of questions to find out how much information has been grasped by the student. Based on these questions,

the student is tested, and the teacher informs the student about his assessment. This kind of learning is typically called supervised learning.

Supervised learning has two methods:

1. Classification
2. Regression

Classification

Classification is a supervised learning method. The input attributes of the classification algorithms are called independent variables. The target attribute is called label or dependent variable. The relationship between the input and target variable is represented in the form of a structure which is called a classification model. So, the focus of classification is to predict the 'label' that is in a discrete form (a value from the set of finite values). An example is shown in Figure 1.7 where a classification algorithm takes a set of labelled data images such as dogs and cats to construct a model that can later be used to classify an unknown test image data.

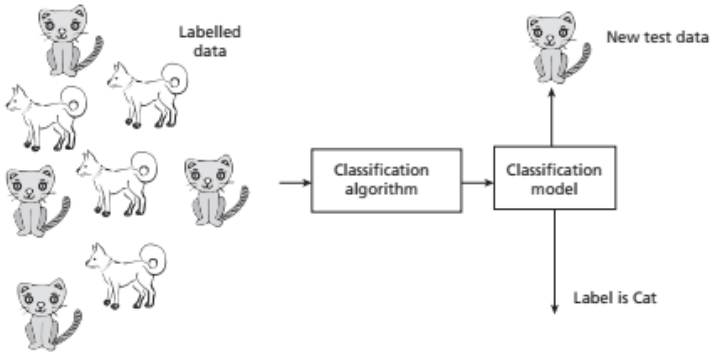


Figure 1.7: An Example Classification System

In classification, learning takes place in two stages. During the first stage, called training stage, the learning algorithm takes a labelled dataset and starts learning. After the training set, samples are processed and the model is generated. In the second stage, the constructed model is tested with test or unknown sample and assigned a label. This is the classification process.

This is illustrated in the above Figure 1.7. Initially, the classification learning algorithm learns with the collection of labelled data and constructs the model. Then, a test case is selected, and the model assigns a label.

Similarly, in the case of Iris dataset, if the test is given as (6.3, 2.9, 5.6, 1.8, ?), the classification will generate the label for this. This is called classification. One of the examples of classification is – Image recognition, which includes classification of diseases like cancer, classification of plants, etc.

The classification models can be categorized based on the implementation technology like decision trees, probabilistic methods, distance measures, and soft computing methods. Classification models can also be classified as generative models and discriminative models. Generative models deal with the process of data generation and its distribution. Probabilistic models are examples of

generative models. Discriminative models do not care about the generation of data. Instead, they simply concentrate on classifying the given data.

Some of the key algorithms of classification are:

- Decision Tree
- Random Forest
- Support Vector Machines
- Naïve Bayes
- Artificial Neural Network and Deep Learning networks like CNN

Regression Models

Regression models, unlike classification algorithms, predict continuous variables like price. In other words, it is a number. A fitted regression model is shown in Figure 1.8 for a dataset that represent weeks input x and product sales y.

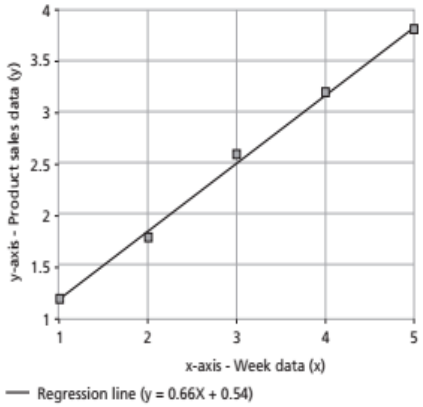


Figure 1.8: A Regression Model of the Form $y = ax + b$

The regression model takes input x and generates a model in the form of a fitted line of the form $y = f(x)$. Here, x is the independent variable that may be one or more attributes and y is the dependent variable. In Figure 1.8, linear regression takes the training set and tries to fit it with a line – product sales = 0.66 × Week + 0.54. Here, 0.66 and 0.54 are all regression coefficients that are learnt from data. The advantage of this model is that prediction for product sales (y) can be made for unknown week data (x). For example, the prediction for unknown eighth week can be made by substituting x as 8 in that regression formula to get y.

One of the most important regression algorithms is linear regression that is explained in the next section.

Both regression and classification models are supervised algorithms. Both have a supervisor and the concepts of training and testing are applicable to both. What is the difference between classification and regression models? The main difference is that regression models predict continuous variables such as product price, while classification concentrates on assigning labels such as class.

1.4.2 Unsupervised Learning

The second kind of learning is by self-instruction. As the name suggests, there are no supervisor or teacher components. In the absence of a supervisor or teacher, self-instruction is the most common kind of learning process. This process of self-instruction is based on the concept of trial and error.

Here, the program is supplied with objects, but no labels are defined. The algorithm itself observes the examples and recognizes patterns based on the principles of grouping. Grouping is done in ways that similar objects form the same group.

Cluster analysis and Dimensional reduction algorithms are examples of unsupervised algorithms.

Cluster Analysis

Cluster analysis is an example of unsupervised learning. It aims to group objects into disjoint clusters or groups. Cluster analysis clusters objects based on its attributes. All the data objects of the partitions are similar in some aspect and vary from the data objects in the other partitions significantly.

Some of the examples of clustering processes are — segmentation of a region of interest in an image, detection of abnormal growth in a medical image, and determining clusters of signatures in a gene database.

An example of clustering scheme is shown in Figure 1.9 where the clustering algorithm takes a set of dogs and cats images and groups it as two clusters-dogs and cats. It can be observed that the samples belonging to a cluster are similar and samples are different radically across clusters.

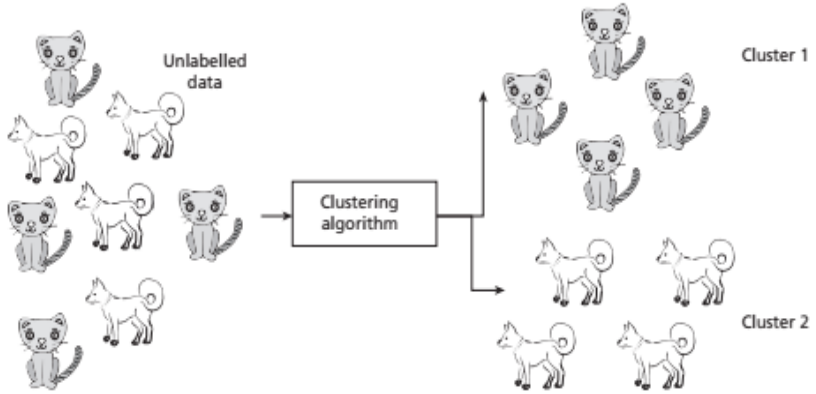


Figure 1.9: An Example Clustering Scheme

Some of the key clustering algorithms are:

- k-means algorithm
- Hierarchical algorithms

Dimensionality Reduction

Dimensionality reduction algorithms are examples of unsupervised algorithms. It takes a higher dimension data as input and outputs the data in lower dimension by taking advantage of the variance of the data. It is a task of reducing the dataset with few features without losing the generality.

The differences between supervised and unsupervised learning are listed in the following Table 1.2.

Table 1.2: Differences between Supervised and Unsupervised Learning

S.No.	Supervised Learning	Unsupervised Learning
1.	There is a supervisor component	No supervisor component
2.	Uses Labelled data	Uses Unlabelled data
3.	Assigns categories or labels	Performs grouping process such that similar objects will be in one cluster

1.4.3 Semi-supervised Learning

There are circumstances where the dataset has a huge collection of unlabelled data and some labelled data. Labelling is a costly process and difficult to perform by the humans. Semi-supervised algorithms use unlabelled data by assigning a pseudo-label. Then, the labelled and pseudo-labelled dataset can be combined.

1.4.4 Reinforcement Learning

Reinforcement learning mimics human beings. Like human beings use ears and eyes to perceive the world and take actions, reinforcement learning allows the agent to interact with the environment to get rewards. The agent can be human, animal, robot, or any independent program. The rewards enable the agent to gain experience. The agent aims to maximize the reward.

The reward can be positive or negative (Punishment). When the rewards are more, the behavior gets reinforced and learning becomes possible.

Consider the following example of a Grid game as shown in Figure 1.10.

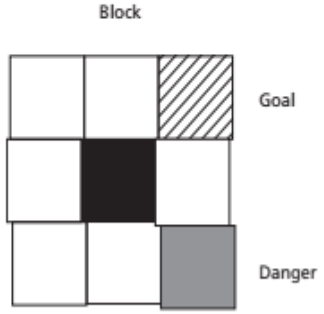


Figure 1.10: A Grid game

In this grid game, the gray tile indicates the danger, black is a block, and the tile with diagonal lines is the goal. The aim is to start, say from bottom-left grid, using the actions left, right, top and bottom to reach the goal state.

To solve this sort of problem, there is no data. The agent interacts with the environment to get experience. In the above case, the agent tries to create a model by simulating many paths and finding rewarding paths. This experience helps in constructing a model.

It can be said in summary, compared to supervised learning, there is no supervisor or labelled dataset. Many sequential decisions need to be taken to reach the final decision. Therefore, reinforcement algorithms are reward-based, goal-oriented algorithms.

Scan for information on *'Important Machine Learning Algorithms'*

1.5 CHALLENGES OF MACHINE LEARNING

What are the challenges of machine learning? Let us discuss about them now.

Problems that can be Dealt with Machine Learning

Computers are better than humans in performing tasks like computation. For example, while calculating the square root of large numbers, an average human may blink but computers can display the result in seconds. Computers can play games like chess, GO, and even beat professional players of that game.

However, humans are better than computers in many aspects like recognition. But, deep learning systems challenge human beings in this aspect as well. Machines can recognize human faces in a second. Still, there are tasks where humans are better as machine learning systems still require quality data for model construction. The quality of a learning system depends on the quality of data. This is a challenge. Some of the challenges are listed below:

1. Problems – Machine learning can deal with the 'well-posed' problems where specifications are complete and available. Computers cannot solve 'ill-posed' problems.

   Consider one simple example (shown in Table 1.3):

   Table 1.3: An Example

Input (x_1, x_2)	Output (y)
1, 1	1
2, 1	2
3, 1	3
4, 1	4
5, 1	5

   Can a model for this test data be multiplication? That is, $y = x_1 \times x_2$. Well! It is true! But, this is equally true that y may be $y = x_1 \div x_2$, or $y = x_1^{x_2}$. So, there are three functions that fit the data. This means that the problem is ill-posed. To solve this problem, one needs more example to check the model. Puzzles and games that do not have sufficient specification may become an ill-posed problem and scientific computation has many ill-posed problems.

2. Huge data – This is a primary requirement of machine learning. Availability of a quality data is a challenge. A quality data means it should be large and should not have data problems such as missing data or incorrect data.
3. High computation power – With the availability of Big Data, the computational resource requirement has also increased. Systems with *Graphics Processing Unit* (GPU) or even *Tensor Processing Unit* (TPU) are required to execute machine learning algorithms. Also, machine learning tasks have become complex and hence time complexity has increased, and that can be solved only with high computing power.
4. Complexity of the algorithms – The selection of algorithms, describing the algorithms, application of algorithms to solve machine learning task, and comparison of algorithms have become necessary for machine learning or data scientists now. Algorithms have become a big topic of discussion and it is a challenge for machine learning professionals to design, select, and evaluate optimal algorithms.
5. Bias/Variance – Variance is the error of the model. This leads to a problem called bias/variance tradeoff. A model that fits the training data correctly but fails for test data, in general lacks generalization, is called overfitting. The reverse problem is called underfitting where the model fails for training data but has good generalization. Overfitting and underfitting are great challenges for machine learning algorithms.

1.6 MACHINE LEARNING PROCESS

The emerging process model for the data mining solutions for business organizations is CRISP-DM. Since machine learning is like data mining, except for the aim, this process can be used for machine learning. CRISP-DM stands for Cross Industry Standard Process – Data Mining. This process involves six steps. The steps are listed below in Figure 1.11.

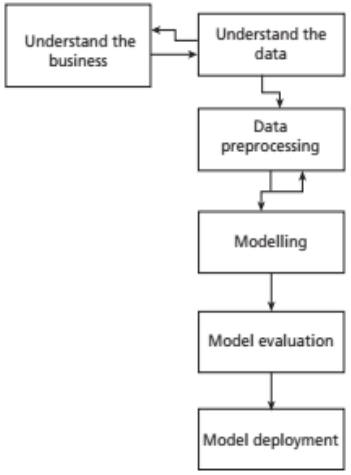


Figure 1.11: A Machine Learning/Data Mining Process

1. Understanding the business – This step involves understanding the objectives and requirements of the business organization. Generally, a single data mining algorithm is enough for giving the solution. This step also involves the formulation of the problem statement for the data mining process.
2. Understanding the data – It involves the steps like data collection, study of the characteristics of the data, formulation of hypothesis, and matching of patterns to the selected hypothesis.
3. Preparation of data – This step involves producing the final dataset by cleaning the raw data and preparation of data for the data mining process. The missing values may cause problems during both training and testing phases. Missing data forces classifiers to produce inaccurate results. This is a perennial problem for the classification models. Hence, suitable strategies should be adopted to handle the missing data.
4. Modelling – This step plays a role in the application of data mining algorithm for the data to obtain a model or pattern.
5. Evaluate – This step involves the evaluation of the data mining results using statistical analysis and visualization methods. The performance of the classifier is determined by evaluating the accuracy of the classifier. The process of classification is a fuzzy issue. For example, classification of emails requires extensive domain knowledge and requires domain experts. Hence, performance of the classifier is very crucial.
6. Deployment – This step involves the deployment of results of the data mining algorithm to improve the existing process or for a new situation.

1.7 MACHINE LEARNING APPLICATIONS

Machine Learning technologies are used widely now in different domains. Machine learning applications are everywhere! One encounters many machine learning applications in the day-to-day life. Some applications are listed below:

1. Sentiment analysis – This is an application of natural language processing (NLP) where the words of documents are converted to sentiments like happy, sad, and angry which are captured by emoticons effectively. For movie reviews or product reviews, five stars or one star are automatically attached using sentiment analysis programs.
2. Recommendation systems – These are systems that make personalized purchases possible. For example, Amazon recommends users to find related books or books bought by people who have the same taste like you, and Netflix suggests shows or related movies of your taste. The recommendation systems are based on machine learning.
3. Voice assistants – Products like Amazon Alexa, Microsoft Cortana, Apple Siri, and Google Assistant are all examples of voice assistants. They take speech commands and perform tasks. These chatbots are the result of machine learning technologies.
4. Technologies like Google Maps and those used by Uber are all examples of machine learning which offer to locate and navigate shortest paths to reduce time.

The machine learning applications are enormous. The following Table 1.4 summarizes some of the machine learning applications.

Chapter 2

Understanding Data

"Torture the data, and it will confess to anything."

— **Ronald Coase**

Machine learning algorithms involve large datasets. Hence, it is necessary to understand the data and datasets before applying machine learning algorithms. This chapter aims to introduce the concepts necessary to understand data better.

Learning Objectives

- Understand data and types of data
- Know about data management
- Learn the basics of dataset
- Identify the data types
- Introduce basics of descriptive analytics
- Explore data visualization techniques
- Introduce univariate data analysis
- Understand basics of bivariate and multivariate data analysis

2.1 WHAT IS DATA?

Scan for information on 'Machine Learning and Importance of Linear Algebra', 'Matrices and Tensors', 'Sampling Techniques', 'Information Theory', 'Evaluation of Classifier Model' and 'Additional Examples'

All facts are data. In computer systems, bits encode facts present in numbers, text, images, audio, and video. Data can be directly human interpretable (such as numbers or texts) or diffused data such as images or video that can be interpreted only by a computer. Today, business organizations are accumulating vast and growing amounts of data of the order of gigabytes, tera bytes, exabytes. A byte is 8 bits. A bit is either 0 or 1. A kilo byte (KB) is 1024 bytes, one mega byte (MB) is approximately 1000 KB, one giga byte is approximately 1,000,000 KB, 1000 giga bytes is one tera byte and 1000000 tera bytes is one Exa byte.

Data is available in different data sources like flat files, databases, or data warehouses. It can either be an operational data or a non-operational data. Operational data is the one that is encountered in normal business procedures and processes. For example, daily sales data is operational data, on the other hand, non-operational data is the kind of data that is used for decision making.

Data by itself is meaningless. It has to be processed to generate any information. A string of bytes is meaningless. Only when a label is attached like height of students of a class, the data becomes meaningful. Processed data is called information that includes patterns, associations, or relationships among data. For example, sales data can be analyzed to extract information like which product was sold larger in the last quarter of the year.

Elements of Big Data

Data whose volume is less and can be stored and processed by a small-scale computer is called 'small data'. These data are collected from several sources, and integrated and processed by a small-scale computer. Big data, on the other hand, is a larger data whose volume is much larger than 'small data' and is characterized as follows:

1. Volume – Since there is a reduction in the cost of storing devices, there has been a tremendous growth of data. Small traditional data is measured in terms of gigabytes (GB) and terabytes (TB), but Big Data is measured in terms of petabytes (PB) and exabytes (EB). One exabyte is 1 million terabytes.
2. Velocity – The fast arrival speed of data and its increase in data volume is noted as velocity. The availability of IoT devices and Internet power ensures that the data is arriving at a faster rate. Velocity helps to understand the relative growth of big data and its accessibility by users, systems and applications.
3. Variety – The variety of Big Data includes:
   - Form – There are many forms of data. Data types range from text, graph, audio, video, to maps. There can be composite data too, where one media can have many other sources of data, for example, a video can have an audio song.
   - Function – These are data from various sources like human conversations, transaction records, and old archive data.
   - Source of data – This is the third aspect of variety. There are many sources of data. Broadly, the data source can be classified as open/public data, social media data and multimodal data. These are discussed in Section 2.3.1 of this chapter.

   Some of the other forms of Vs that are often quoted in the literature as characteristics of Big data are:
4. Veracity of data – Veracity of data deals with aspects like conformity to the facts, truthfulness, believability, and confidence in data. There may be many sources of error such as technical errors, typographical errors, and human errors. So, veracity is one of the most important aspects of data.
5. Validity – Validity is the accuracy of the data for taking decisions or for any other goals that are needed by the given problem.
6. Value – Value is the characteristic of big data that indicates the value of the information that is extracted from the data and its influence on the decisions that are taken based on it.

Thus, these 6 Vs are helpful to characterize the big data. The data quality of the numeric attributes is determined by factors like precision, bias, and accuracy. Precision is defined as the closeness of repeated measurements. Often, standard deviation is used to measure the precision. Bias is a systematic result due to erroneous assumptions of the algorithms or procedures. Accuracy is the degree of measurement of errors that refers to the closeness of measurements to the true value of the quantity. Normally, the significant digits used to store and manipulate indicate the accuracy of the measurement.

2.1.1 Types of Data

In Big Data, there are three kinds of data. They are structured data, unstructured data, and semi-structured data.

Structured Data

In structured data, data is stored in an organized manner such as a database where it is available in the form of a table. The data can also be retrieved in an organized manner using tools like SQL.

The structured data frequently encountered in machine learning are listed below:

Record Data A dataset is a collection of measurements taken from a process. We have a collection of objects in a dataset and each object has a set of measurements. The measurements can be arranged in the form of a matrix. Rows in the matrix represent an object and can be called as entities, cases, or records. The columns of the dataset are called attributes, features, or fields. The table is filled with observed data. Also, it is better to note the general jargons that are associated with the dataset. Label is the term that is used to describe the individual observations.

Data Matrix It is a variation of the record type because it consists of numeric attributes. The standard matrix operations can be applied on these data. The data is thought of as points or vectors in the multidimensional space where every attribute is a dimension describing the object.

Graph Data It involves the relationships among objects. For example, a web page can refer to another web page. This can be modeled as a graph. The modes are web pages and the hyperlink is an edge that connects the nodes.

Ordered Data Ordered data objects involve attributes that have an implicit order among them.

The examples of ordered data are:

1. Temporal data – It is the data whose attributes are associated with time. For example, the customer purchasing patterns during festival time is sequential data. Time series data is a special type of sequence data where the data is a series of measurements over time.
2. Sequence data – It is like sequential data but does not have time stamps. This data involves the sequence of words or letters. For example, DNA data is a sequence of four characters – A T G C.
3. Spatial data – It has attributes such as positions or areas. For example, maps are spatial data where the points are related by location.

Unstructured Data

Unstructured data includes video, image, and audio. It also includes textual documents, programs, and blog data. It is estimated that 80% of the data are unstructured data.

Semi-Structured Data

Semi-structured data are partially structured and partially unstructured. These include data like XML/JSON data, RSS feeds, and hierarchical data.

2.1.2 Data Storage and Representation

Once the dataset is assembled, it must be stored in a structure that is suitable for data analysis. The goal of data storage management is to make data available for analysis. There are different approaches to organize and manage data in storage files and systems from flat file to data warehouses. Some of them are listed below:

Flat Files These are the simplest and most commonly available data source. It is also the cheapest way of organizing the data. These flat files are the files where data is stored in plain ASCII or EBCDIC format. Minor changes of data in flat files affect the results of the data mining algorithms. Hence, flat file is suitable only for storing small dataset and not desirable if the dataset becomes larger.

Some of the popular spreadsheet formats are listed below:

- CSV files – CSV stands for comma–separated value files where the values are separated by commas. These are used by spreadsheet and database applications. The first row may have attributes and the rest of the rows represent the data.
- TSV files – TSV stands for Tab separated values files where values are separated by Tab.

Both CSV and TSV files are generic in nature and can be shared. There are many tools like Google Sheets and Microsoft Excel to process these files.

Database System It normally consists of database files and a database management system (DBMS). Database files contain original data and metadata. DBMS aims to manage data and improve operator performance by including various tools like database administrator, query processing, and transaction manager. A relational database consists of sets of tables. The tables have rows and columns. The columns represent the attributes and rows represent tuples. A tuple corresponds to either an object or a relationship between objects. A user can access and manipulate the data in the database using SQL.

Different types of databases are listed below:

1. A transactional database is a collection of transactional records. Each record is a transaction. A transaction may have a time stamp, identifier and a set of items, which may have links to other tables. Normally, transactional databases are created for performing associational analysis that indicates the correlation among the items.
2. Time-series database stores time related information like log files where data is associated with a time stamp. This data represents the sequences of data, which represent values or events obtained over a period (for example, hourly, weekly or yearly) or repeated time span. Observing sales of product continuously may yield a time-series data.

3. Spatial databases contain spatial information in a raster or vector format. Raster formats are either bitmaps or pixel maps. For example, images can be stored as a raster data. On the other hand, the vector format can be used to store maps as maps use basic geometric primitives like points, lines, polygons and so forth.

World Wide Web (WWW) It provides a diverse, worldwide online information source. The objective of data mining algorithms is to mine interesting patterns of information present in WWW.

XML (eXtensible Markup Language) It is both human and machine interpretable data format that can be used to represent data that needs to be shared across the platforms.

Data Stream It is dynamic data, which flows in and out of the observing environment. Typical characteristics of data stream are huge volume of data, dynamic, fixed order movement, and real-time constraints.

RSS (Really Simple Syndication) It is a format for sharing instant feeds across services.

JSON (JavaScript Object Notation) It is another useful data interchange format that is often used for many machine learning algorithms.

2.2 BIG DATA ANALYTICS AND TYPES OF ANALYTICS

The primary aim of data analysis is to assist business organizations to take decisions. For example, a business organization may want to know which is the fastest selling product, in order for them to market activities. Data analysis is an activity that takes the data and generates useful information and insights for assisting the organizations.

Data analysis and data analytics are terms that are used interchangeably to refer to the same concept. However, there is a subtle difference. Data analytics is a general term and data analysis is a part of it. Data analytics refers to the process of data collection, preprocessing and analysis. It deals with the complete cycle of data management. Data analysis is just analysis and is a part of data analytics. It takes historical data and does the analysis. Data analytics, instead, concentrates more on future and helps in prediction.

There are four types of data analytics:

1. Descriptive analytics
2. Diagnostic analytics
3. Predictive analytics
4. Prescriptive analytics

Descriptive Analytics It is about describing the main features of the data. After data collection is done, descriptive analytics deals with the collected data and quantifies it. It is often stated that analytics is essentially statistics. There are two aspects of statistics – Descriptive and Inference. Descriptive analytics only focuses on the description part of the data and not the inference part.

Diagnostic Analytics It deals with the question – 'Why?'. This is also known as causal analysis, as it aims to find out the cause and effect of the events. For example, if a product is not selling, diagnostic analytics aims to find out the reason. There may be multiple reasons and associated effects are analyzed as part of it.

Predictive Analytics It deals with the future. It deals with the question – 'What will happen in future given this data?'. This involves the application of algorithms to identify the patterns to predict the future. The entire course of machine learning is mostly about predictive analytics and forms the core of this book.

Prescriptive Analytics It is about the finding the best course of action for the business organizations. Prescriptive analytics goes beyond prediction and helps in decision making by giving a set of actions. It helps the organizations to plan better for the future and to mitigate the risks that are involved.

2.3 BIG DATA ANALYSIS FRAMEWORK

For performing data analytics, many frameworks are proposed. All proposed analytics frameworks have some common factors. Big data framework is a layered architecture. Such an architecture has many advantages such as genericness. A 4-layer architecture has the following layers:

1. Date connection layer
2. Data management layer
3. Data analytics later
4. Presentation layer

Data Connection Layer It has data ingestion mechanisms and data connectors. Data ingestion means taking raw data and importing it into appropriate data structures. It performs the tasks of ETL process. By ETL, it means extract, transform and load operations.

Data Management Layer It performs preprocessing of data. The purpose of this layer is to allow parallel execution of queries, and read, write and data management tasks. There may be many schemes that can be implemented by this layer such as data-in-place, where the data is not moved at all, or constructing data repositories such as data warehouses and pull data on-demand mechanisms.

Data Analytic Layer It has many functionalities such as statistical tests, machine learning algorithms to understand, and construction of machine learning models. This layer implements many model validation mechanisms too. The processing is done as shown in Box 2.1.

Box 2.1: Types of Processing

Cloud Computing

Cloud computing is an emerging technology which is basically a business service model or simply called as pay-per-usage model. The term 'Cloud' refers to the Internet that provides sharing of processing power, applications, storage and services. It offers different kinds of services such as **Iaas**, **Paas**, and **SaaS**.

SaaS (Software as a Service) enables users to access software applications from the cloud. **Paas** (Platform as a Service) provides users the platform to develop and run their applications. **Iaas** (Infrastructure as a Service) enables users to access the infrastructure required to run their applications, storage, operating systems, etc.

(*Continued*)

The cloud services can be deployed in four most commonly used deployment models such as Public Cloud, Private Cloud, Community Cloud, and Hybrid Cloud based on the service model, organization, geographic location, etc. The **Public Cloud** is accessible to the public and is owned by a vendor, who offers the services of the cloud to the users. **Private Cloud** is a privately-owned cloud where the user or an organization owns the cloud and only the user or employees of that organization have access to the cloud, thereby making data and transactions secure. In **Community Cloud,** the infrastructure is owned jointly by different organizations. The **Hybrid Cloud** is the combination of two or more cloud types. The characteristics of cloud computing are:

1. Shared Infrastructure – Sharing of physical services, storage, and networking capabilities
2. Dynamic Provisioning – Resources assigned dynamically, based on demands
3. Dynamic Scaling – Expansion and contraction of service capability
4. Network Access – Needs to be accessed across the internet
5. Utility-based Metering – Uses metering to provide reporting and billing information
6. Multitenancy – Serves multiple customers
7. Reliability – Customer reliable service

Grid Computing

Grid Computing is a parallel and distributed computing framework consisting of a network of computers offering a super computing service as a single virtual supercomputer. This high-performance computing is required to perform specialized tasks that require a high computing power and a single computer cannot provide enough computing resources. The grid computing model forms a grid by connecting tens of thousands of nodes as a cluster that runs on an operating system. In this model, the resources are pooled together and the load is shared across multiple nodes to accomplish a task more quickly. This grid is constructed by middleware software that evenly distributes the task to several nodes connected in the grid. The individual nodes perform the task independently and in parallel which are then integrated to complete the large-scale task. This model of computing is best suited for applications that are complex and can be computed in parallel.

H-Computing (High Performance Computing or HPC)

It enables to perform complex tasks at high speed. It aggregates computing power in such a way that provides much higher performance to solve complex problems in science, engineering, research or business. It leverages parallel processing techniques for solving complex computational problems. HPC system achieves this sustained performance through concurrent use of computing resources. An HPC system combines the computing power of thousands of compute nodes that work in parallel to complete tasks faster. The system comprises three key components called compute, network and storage. The architecture of HPC consists of compute servers that are networked together to form a cluster. Software programs are run in parallel on the servers in the cluster and are networked to the data storage to capture the output. These components work together to complete a task.

Presentation Layer It has mechanisms such as dashboards, and applications that display the results of analytical engines and machine learning algorithms.

Thus, the Big Data processing cycle involves data management that consists of the following steps.

1. Data collection
2. Data preprocessing
3. Applications of machine learning algorithm
4. Interpretation of results and visualization of machine learning algorithm

This is an iterative process and is carried out on a permanent basis to ensure that data is suitable for data mining.

Application and interpretation of machine learning algorithms constitute the basis for the rest of the book. So, primarily, data collection and data preprocessing are covered as part of this chapter.

The following section covers data collection in detail.

2.3.1 Data Collection

The first task of gathering datasets are the collection of data. It is often estimated that most of the time is spent for collection of good quality data. A good quality data yields a better result. It is often difficult to characterize a 'Good data'. 'Good data' is one that has the following properties:

1. Timeliness – The data should be relevant and not stale or obsolete data.
2. Relevancy – The data should be relevant and ready for the machine learning or data mining algorithms. All the necessary information should be available and there should be no bias in the data.
3. Knowledge about the data – The data should be understandable and interpretable, and should be self-sufficient for the required application as desired by the domain knowledge engineer.

Broadly, the data source can be classified as open/public data, social media data and multimodal data.

1. Open or public data source – It is a data source that does not have any stringent copyright rules or restrictions. Its data can be primarily used for many purposes. Government census data are good examples of open data:
   - Digital libraries that have huge amount of text data as well as document images
   - Scientific domains with a huge collection of experimental data like genomic data and biological data
   - Healthcare systems that use extensive databases like patient databases, health insurance data, doctors' information, and bioinformatics information
2. Social media – It is the data that is generated by various social media platforms like Twitter, Facebook, YouTube, and Instagram. An enormous amount of data is generated by these platforms.
3. Multimodal data – It includes data that involves many modes such as text, video, audio and mixed types. Some of them are listed below:

   - Image archives contain larger image databases along with numeric and text data
   - The World Wide Web (WWW) has huge amount of data that is distributed on the Internet.

   These data are heterogeneous in nature.

2.3.2 Data Preprocessing

In real world, the available data is 'dirty'. By this word 'dirty', it means:

- Incomplete data
- Inaccurate data
- Outlier data
- Data with missing values
- Data with inconsistent values
- Duplicate data

Data preprocessing improves the quality of the data mining techniques. The raw data must be preprocessed to give accurate results. The process of detection and removal of errors in data is called data cleaning. Data wrangling means making the data processable for machine learning algorithms. Some of the data errors include human errors such as typographical errors or incorrect measurement and structural errors like improper data formats. Data errors can also arise from omission and duplication of attributes. Noise is a random component and involves distortion of a value or introduction of spurious objects. Often, the noise is used if the data is a spatial or temporal component. Certain deterministic distortions in the form of a streak are known as artifacts.

Consider, for example, the following patient Table 2.1. The 'bad' or 'dirty' data can be observed in this table.

Table 2.1: Illustration of 'Bad' Data

Patient ID	Name	Age	Date of Birth (DoB)	Fever	Salary
1.	John	21		Low	–1500
2.	Andre	36		High	Yes
3.	David	5	10/10/1980	Low	" "
4.	Raju	136		High	Yes

It can be observed that data like Salary = ' ' is incomplete data. The DoB of patients, John, Andre, and Raju, is the missing data. The age of David is recorded as '5' but his DoB indicates it is 10/10/1980. This is called inconsistent data.

Inconsistent data occurs due to problems in conversions, inconsistent formats, and difference in units. Salary for John is –1500. It cannot be less than '0'. It is an instance of noisy data. Outliers are data that exhibit the characteristics that are different from other data and have very unusual values. The age of Raju cannot be 136. It might be a typographical error. It is often required to distinguish between noise and outlier data.

Outliers may be legitimate data and sometimes are of interest to the data mining algorithms. These errors often come during data collection stage. These must be removed so that machine learning algorithms yield better results as the quality of results is determined by the quality of input data. This removal process is called data cleaning.

Missing Data Analysis

The primary data cleaning process is missing data analysis. Data cleaning routines attempt to fill up the missing values, smoothen the noise while identifying the outliers and correct the inconsistencies of the data. This enables data mining to avoid overfitting of the models.

The procedures that are given below can solve the problem of missing data:

1. Ignore the tuple – A tuple with missing data, especially the class label, is ignored. This method is not effective when the percentage of the missing values increases.
2. Fill in the values manually – Here, the domain expert can analyse the data tables and carry out the analysis and fill in the values manually. But, this is time consuming and may not be feasible for larger sets.
3. A global constant can be used to fill in the missing attributes. The missing values may be 'Unknown' or be 'Infinity'. But, some data mining results may give spurious results by analysing these labels.
4. The attribute value may be filled by the attribute value. Say, the average income can replace a missing value.
5. Use the attribute mean for all samples belonging to the same class. Here, the average value replaces the missing values of all tuples that fall in this group.
6. Use the most possible value to fill in the missing value. The most probable value can be obtained from other methods like classification and decision tree prediction.

Some of these methods introduce bias in the data. The filled value may not be correct and could be just an estimated value. Hence, the difference between the estimated and the original value is called an error or bias.

Removal of Noisy or Outlier Data

Noise is a random error or variance in a measured value. It can be removed by using binning, which is a method where the given data values are sorted and distributed into equal frequency bins. The bins are also called as buckets. The binning method then uses the neighbor values to smooth the noisy data.

Some of the techniques commonly used are 'smoothing by means' where the mean of the bin removes the values of the bins, 'smoothing by bin medians' where the bin median replaces the bin values, and 'smoothing by bin boundaries' where the bin value is replaced by the closest bin boundary. The maximum and minimum values are called bin boundaries. Binning methods may be used as a discretization technique. Example 2.1 illustrates this principle.

Example 2.1: Consider the following set: S = {12, 14, 19, 22, 24, 26, 28, 31, 34}. Apply various binning techniques and show the result.

Solution: By equal-frequency bin method, the data should be distributed across bins. Let us assume the bins of size 3, then the above data is distributed across the bins as shown below:

Bin 1 : 12 , 14, 19

Bin 2 : 22, 24, 26

Bin 3 : 28, 31, 32

By smoothing bins method, the bins are replaced by the bin means. This method results in:

Bin 1 : 15, 15, 15

Bin 2 : 24, 24, 24

Bin 3 : 30.3, 30.3, 30.3

Using smoothing by bin boundaries method, the bins' values would be like:

Bin 1 : 12, 12, 19

Bin 2 : 22, 22, 26

Bin 3 : 28, 32, 32

As per the method, the minimum and maximum values of the bin are determined, and it serves as bin boundary and does not change. Rest of the values are transformed to the nearest value. It can be observed in Bin 1, the middle value 14 is compared with the boundary values 12 and 19 and changed to the closest value, that is 12. This process is repeated for all bins.

Data Integration and Data Transformations

Data integration involves routines that merge data from multiple sources into a single data source. So, this may lead to redundant data. The main goal of data integration is to detect and remove redundancies that arise from integration. Data transformation routines perform operations like normalization to improve the performance of the data mining algorithms. It is necessary to transform data so that it can be processed. This can be considered as a preliminary stage of data conditioning. Normalization is one such technique. In normalization, the attribute values are scaled to fit in a range (say 0–1) to improve the performance of the data mining algorithm. Often, in neural networks, these techniques are used. Some of the normalization procedures used are:

1. *Min-Max*
2. *z*-Score

***Min-Max* Procedure** It is a normalization technique where each variable V is normalized by its difference with the minimum value divided by the range to a new range, say 0–1. Often, neural networks require this kind of normalization. The formula to implement this normalization is given as:

$$min\text{-}max = \frac{V - min}{max - min} \times (new\ max - new\ min) + new\ min \tag{2.1}$$

Here *max-min* is the range. *Min* and *max* are the minimum and maximum of the given data, *new max* and *new min* are the minimum and maximum of the target range, say 0 and 1.

Example 2.2: Consider the set: V = {88, 90, 92, 94}. Apply *Min-Max* procedure and map the marks to a new range 0–1.

Solution: The minimum of the list V is 88 and maximum is 94. The *new min* and *new max* are 0 and 1, respectively. The mapping can be done using Eq. (2.1) as:

For marks 88,

$$min\text{-}max = \frac{88 - 88}{94 - 88} \times (1 - 0) + 0 = 0$$

Similarly, other marks can be computed as follows:

For marks 90,

$$min\text{-}max = \frac{90 - 88}{94 - 88} \times (1 - 0) + 0 = 0.33$$

For marks 92,

$$min\text{-}max = \frac{92 - 88}{94 - 88} \times (1 - 0) + 0 = \frac{4}{6} = 0.66$$

For marks 94,

$$min\text{-}max = \frac{94 - 88}{94 - 88} \times (1 - 0) + 0 = \frac{6}{6} = 1$$

So, it can be observed that the marks {88, 90, 92, 94} are mapped to the new range {0, 0.33, 0.66, 1}.

Thus, the *Min-Max* normalization range is between 0 and 1.

z-Score Normalization This procedure works by taking the difference between the field value and mean value, and by scaling this difference by standard deviation of the attribute.

$$V* = V - \mu/\sigma \tag{2.2}$$

Here, σ is the standard deviation of the list V and μ is the mean of the list V.

Example 2.3: Consider the mark list $V = \{10, 20, 30\}$, convert the marks to z-score.

Solution: The mean and Sample Standard deviation (σ) values of the list V are 20 and 10, respectively. So the z-scores of these marks are calculated using Eq. (2.2) as:

$$z\text{-score of } 10 = \frac{10 - 20}{10} = -\frac{10}{10} = -1$$

$$z\text{-score of } 20 = \frac{20 - 20}{10} = \frac{0}{10} = 0$$

$$z\text{-score of } 30 = \frac{30 - 20}{10} = \frac{10}{10} = 1$$

Hence, the z-score of the marks 10, 20, 30 are –1, 0 and 1, respectively.

What is the use of z-scores? z-scores are used to detect outlier detection. If the data value z-score function is either less than –3 or greater than +3, then it is possibly an outlier. The major disadvantage of z-score function is that it is extremely sensitive to outliers as it is dependent on mean.

Data Reduction

Data reduction reduces data size but produces the same results. There are different ways in which data reduction can be carried out such as data aggregation, feature selection, and dimensionality reduction.

2.4 DESCRIPTIVE STATISTICS

Descriptive statistics is a branch of statistics that does dataset summarization. It is used to summarize and describe data. Descriptive statistics are just descriptive and do not go beyond that. In other words, descriptive statistics do not bother too much about machine learning algorithms and its functioning.

Data visualization is a branch of study that is useful for investigating the given data. Mainly, the plots are useful to explain and present data to customers.

Descriptive analytics and data visualization techniques help to understand the nature of the data, which further helps to determine the kinds of machine learning or data mining tasks that can be applied to the data. This step is often known as Exploratory Data Analysis (EDA). The focus of EDA is to understand the given data and to prepare it for machine learning algorithms. EDA includes descriptive statistics and data visualization.

Let us discuss descriptive statistics with the fundamental concepts of datatypes.

Dataset and Data Types

A dataset can be assumed to be a collection of data objects. The data objects may be records, points, vectors, patterns, events, cases, samples or observations. These records contain many attributes. An attribute can be defined as the property or characteristics of an object.

For example, consider the following database shown in sample Table 2.2.

Table 2.2: Sample Patient Table

Patient ID	Name	Age	Blood Test	Fever	Disease
1.	John	21	Negative	Low	No
2.	Andre	36	Positive	High	Yes

Every attribute should be associated with a value. This process is called measurement. The type of attribute determines the data types, often referred to as measurement scale types. The data types are shown in Figure 2.1.

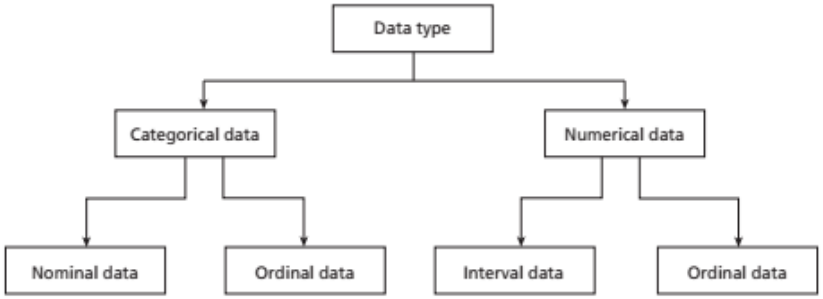


Figure 2.1: Types of Data

Broadly, data can be classified into two types:

1. Categorical or qualitative data
2. Numerical or quantitative data

Categorical or Qualitative Data The categorical data can be divided into two types. They are nominal type and ordinal type.

- Nominal Data – In Table 2.2, patient ID is nominal data. Nominal data are symbols and cannot be processed like a number. For example, the average of a patient ID does not make any statistical sense. Nominal data type provides only information but has no ordering among data. Only operations like (=, ≠) are meaningful for these data. For example, the patient ID can be checked for equality and nothing else.
- Ordinal Data – It provides enough information and has natural order. For example, Fever = {Low, Medium, High} is an ordinal data. Certainly, low is less than medium and medium is less than high, irrespective of the value. Any transformation can be applied to these data to get a new value.

Numeric or Qualitative Data It can be divided into two categories. They are interval type and ratio type.

- Interval Data – Interval data is a numeric data for which the differences between values are meaningful. For example, there is a difference between 30 degree and 40 degree. Only the permissible operations are + and –.
- Ratio Data – For ratio data, both differences and ratio are meaningful. The difference between the ratio and interval data is the position of zero in the scale. For example, take the Centigrade-Fahrenheit conversion. The zeroes of both scales do not match. Hence, these are interval data.

Another way of classifying the data is to classify it as:

1. Discrete value data
2. Continuous data

Discrete Data This kind of data is recorded as integers. For example, the responses of the survey can be discrete data. Employee identification number such as 10001 is discrete data.

Continuous Data It can be fitted into a range and includes decimal point. For example, age is a continuous data. Though age appears to be discrete data, one may be 12.5 years old and it makes sense. Patient height and weight are all continuous data.

Third way of classifying the data is based on the number of variables used in the dataset. Based on that, the data can be classified as univariate data, bivariate data, and multivariate data. This is shown in Figure 2.2.

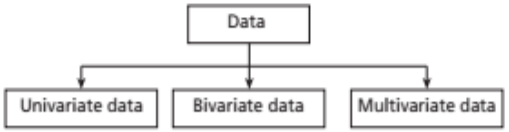


Figure 2.2: Types of Data Based on Variables

In case of univariate data, the dataset has only one variable. A variable is also called as category. Bivariate data indicates that the number of variables used are two and multivariate data uses three or more variables.

This chapter primarily deals with univariate data in detail with just an overview of bivariate and multivariate data.

2.5 UNIVARIATE DATA ANALYSIS AND VISUALIZATION

Scan for information on *'Measures of Frequency'* and also for *'Additional Examples'*

Univariate analysis is the simplest form of statistical analysis. As the name indicates, the dataset has only one variable. A variable can be called as a category. Univariate does not deal with cause or relationships. The aim of univariate analysis is to describe data and find patterns.

Univariate data description involves finding the frequency distributions, central tendency measures, dispersion or variation, and shape of the data.

2.5.1 Data Visualization

To understand data, graph visualization is must. Data visualization helps to understand data. It helps to present information and data to customers. Some of the graphs that are used in univariate data analysis are bar charts, histograms, frequency polygons and pie charts.

The advantages of the graphs are presentation of data, summarization of data, description of data, exploration of data, and to make comparisons of data. Let us consider some forms of graphs now:

Bar Chart A Bar chart (or Bar graph) is used to display the frequency distribution for variables. Bar charts are used to illustrate discrete data. The charts can also help to explain the counts of nominal data. It also helps in comparing the frequency of different groups.

The bar chart for students' marks {45, 60, 60, 80, 85} with Student ID = {1, 2, 3, 4, 5} is shown below in Figure 2.3.

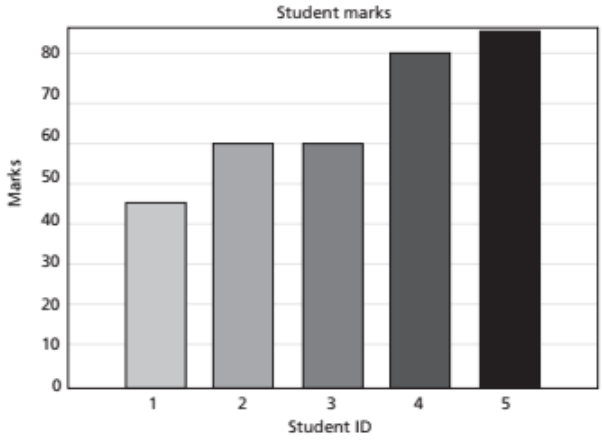


Figure 2.3: Bar Chart

Pie Chart These are equally helpful in illustrating the univariate data. The percentage frequency distribution of students' marks {22, 22, 40, 40, 70, 70, 70, 85, 90, 90} is below in Figure 2.4.

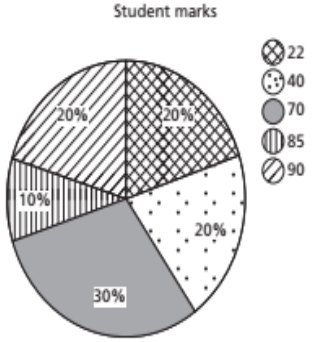


Figure 2.4: Pie Chart

It can be observed that the number of students with 22 marks are 2. The total number of students are 10. So, 2/10 × 100 = 20% space in a pie of 100% is allotted for marks 22 in Figure 2.4.

Histogram It plays an important role in data mining for showing frequency distributions. The histogram for students' marks {45, 60, 60, 80, 85} in the group range of 0–25, 26–50, 51–75, 76–100 is given below in Figure 2.5. One can visually inspect from Figure 2.5 that the number of students in the range 76–100 is 2.

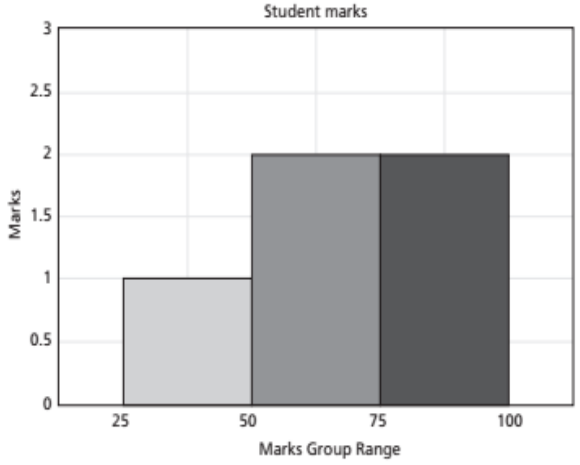


Figure 2.5: Sample Histogram of English Marks

Histogram conveys useful information like nature of data and its mode. Mode indicates the peak of dataset. In other words, histograms can be used as charts to show frequency, skewness present in the data, and shape.

Dot Plots These are similar to bar charts. They are less clustered as compared to bar charts, as they illustrate the bars only with single points. The dot plot of English marks for five students with ID as {1, 2, 3, 4, 5} and marks {45, 60, 60, 80, 85} is given in Figure 2.6. The advantage is that by visual inspection one can find out who got more marks.

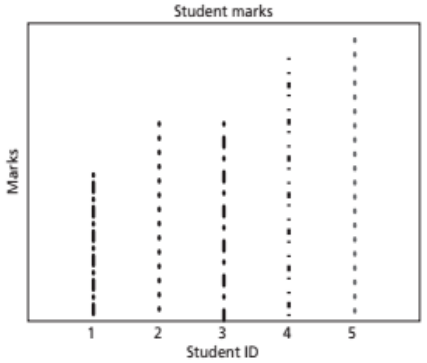


Figure 2.6: Dot Plots

2.5.2 Central Tendency

One cannot remember all the data. Therefore, a condensation or summary of the data is necessary. This makes the data analysis easy and simple. One such summary is called central tendency. Thus, central tendency can explain the characteristics of data and that further helps in comparison. Mass data have tendency to concentrate at certain values, normally in the central location. It is called measure of central tendency (or averages). This represents the first order of measures. Popular measures are mean, median and mode.

1. Mean – Arithmetic average (or mean) is a measure of central tendency that represents the 'center' of the dataset. This is the commonest measure used in our daily conversation such as average income or average traffic. It can be found by adding all the data and dividing the sum by the number of observations. Mathematically, the *average* of all the values in the sample (population) is denoted as $\bar{x}$. Let $x_1, x_2, \cdots, x_N$ be a set of 'N' values or observations, then the arithmetic mean is given as:

$$\bar{x} = \frac{x_1 + x_2 + \cdots + x_N}{N} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2.3}$$

For example, the mean of the three numbers 10, 20, and 30 is $\frac{10 + 20 + 30}{3} = \frac{60}{3} = 20$

- Weighted mean – Unlike arithmetic mean that gives the weightage of all items equally, weighted mean gives different importance to all items as the item importance varies. Hence, different weightage can be given to items.

In case of frequency distribution, mid values of the range are taken for computation. This is illustrated in the following computation.

In weighted mean, the mean is computed by adding the product of proportion and group mean. It is mostly used when the sample sizes are unequal.

- Geometric mean – Let $x_1, x_2, \ldots, x_N$ be a set of 'N' values or observations. Geometric mean is the N^{th} root of the product of N items. The formula for computing geometric mean is given as follows:

$$\text{Geometric mean} = \left(\prod_{i=1}^{n} x_i\right)^{\frac{1}{N}} = \sqrt[N]{x_1 \times x_2 \times \cdots \times x_N} \tag{2.4}$$

Here, n is the number of items and x_i are values. For example, if the values are 6 and 8, the geometric mean is given as $\sqrt[2]{6 \times 8} = \sqrt{48}$. In larger cases, computing geometric mean is difficult. Hence, it is usually calculated as:

$$\text{Anti-log of } \frac{\log(x_1) + \log(x_2) + \cdots + \log(x_N)}{N} \tag{2.5}$$

$$= \text{anti-log } \frac{\sum_{i=1}^{n} \log(x_i)}{N} \tag{2.6}$$

The problem of mean is its extreme sensitiveness to noise. Even small changes in the input affect the mean drastically. Hence, often the top 2% is chopped off and then the mean is calculated for a larger dataset.

2. Median – The middle value in the distribution is called median. If the total number of items in the distribution is odd, then the middle value is called median. If the numbers are even, then the average value of two items in the centre is the median. It can be observed that the median is the value where x_i is divided into two equal halves, with half of the values being lower than the median and half higher than the median. A median class is that class where $(N/2)^{th}$ item is present.

In the continuous case, the median is given by the formula:

$$\text{Median} = L_1 + \frac{\frac{N}{2} - cf}{f} \times i \tag{2.7}$$

Median class is that class where $N/2^{th}$ item is present. Here, i is the class interval of the median class and L_1 is the lower limit of median class, f is the frequency of the median class, and cf is the cumulative frequency of all classes preceding median.

3. Mode – Mode is the value that occurs more frequently in the dataset. In other words, the value that has the highest frequency is called mode. Mode is only for discrete data and is not applicable for continuous data as there are no repeated values in continuous data.

The procedure for finding the mode is to calculate the frequencies for all the values in the data, and mode is the value (or values) with the highest frequency. Normally, the dataset is classified as unimodal, bimodal and trimodal with modes 1, 2 and 3, respectively.

2.5.3 Dispersion

The spreadout of a set of data around the central tendency (mean, median or mode) is called dispersion. Dispersion is represented by various ways such as range, variance, standard deviation, and standard error. These are second order measures. The most common measures of the dispersion data are listed below:

Range Range is the difference between the maximum and minimum of values of the given list of data.

Standard Deviation The mean does not convey much more than a middle point. For example, the following datasets {10, 20, 30} and {10, 50, 0} both have a mean of 20. The difference between these two sets is the spread of data.

Standard deviation is the average distance from the mean of the dataset to each point. The formula for sample standard deviation is given by:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N-1}} \tag{2.8}$$

Here, N is the size of the population, x_i is observation or value from the population and μ is the population mean. Often, $N-1$ is used instead of N in the denominator of Eq. (2.8). The reason is that for larger real-world, the division by $N-1$ gives an answer closer to the actual value.

Quartiles and Inter Quartile Range It is sometimes convenient to subdivide the dataset using coordinates. Percentiles are about data that are less than the coordinates by some percentage of the total value. k^{th} percentile is the property that the k% of the data lies at or below X_i. For example, median is 50^{th} percentile and can be denoted as $Q_{0.50}$. The 25^{th} percentile is called first quartile (Q_1) and the 75^{th} percentile is called third quartile (Q_3).

Another measure that is useful to measure dispersion is Inter Quartile Range (IQR). The IQR is the difference between Q_3 and Q_1.

$$\text{Interquartile percentile} = Q_3 - Q_1 \tag{2.9}$$

Outliers are normally the values falling apart at least by the amount 1.5 × IQR above the third quartile or below the first quartile.

Interquartile is defined by $Q_{0.75} - Q_{0.25}$. (2.10)

Example 2.4: For patients' age list {12, 14, 19, 22, 24, 26, 28, 31, 34}, find the IQR.

Solution: The median is in the fifth position. In this case, 24 is the median. The first quartile is median of the scores below the mean i.e., {12, 14, 19, 22}. Hence, it's the median of the list below 24. In this case, the median is the average of the second and third values, that is, $Q_{0.25} = 16.5$. Similarly, the third quartile is the median of the values above the median, that is {26, 28, 31, 34}. So, $Q_{0.75}$ is the average of the seventh and eighth score. In this case, it is 28 + 31/2 = 59/2 = 29.5.

Hence, the IQR using Eq. (2.10) is:

$$= Q_{0.75} - Q_{0.25}$$
$$= 29.5 - 16.5 = 13$$

The half of IQR is called semi-quartile range. The Semi Inter Quartile Range (SIQR) is given as:

$$\text{SIQR} = \frac{1}{2} \times \text{IQR}$$
$$= \frac{1}{2} \times 13 = 6.5 \quad (2.11)$$

Five-point Summary and Box Plots The median, quartiles Q_1 and Q_3, and minimum and maximum written in the order < Minimum, Q_1, Median, Q_3, Maximum > is known as five-point summary.

Box plots are suitable for continuous variables and a nominal variable. Box plots can be used to illustrate data distributions and summary of data. It is the popular way for plotting five number summaries. A Box plot is also known as a Box and whisker plot.

The box contains bulk of the data. These data are between first and third quartiles. The line inside the box indicates location – mostly median of the data. If the median is not equidistant, then the data is skewed. The whiskers that project from the ends of the box indicate the spread of the tails and the maximum and minimum of the data value.

Example 2.5: Find the 5-point summary of the list {13, 11, 2, 3, 4, 8, 9}.

Solution: The minimum is 2 and the maximum is 13. The Q_1, Q_2 and Q_3 are 3, 8 and 11, respectively. Hence, 5-point summary is {2, 3, 8, 11, 13}, that is, {minimum, Q_1, median, Q_3, maximum}.

Box plots are useful for describing 5-point summary. The Box plot for the set is given in Figure 2.7.

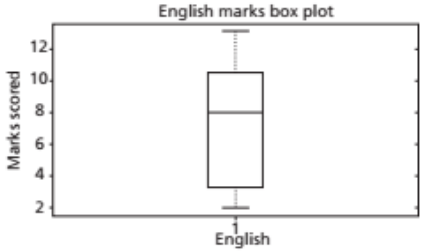


Figure 2.7: Box Plot for English Marks

2.5.4 Shape

Skewness and Kurtosis (called moments) indicate the symmetry/asymmetry and peak location of the dataset.

Skewness

The measures of direction and degree of symmetry are called measures of third order. Ideally, skewness should be zero as in ideal normal distribution. More often, the given dataset may not have perfect symmetry (consider the following Figure 2.8).

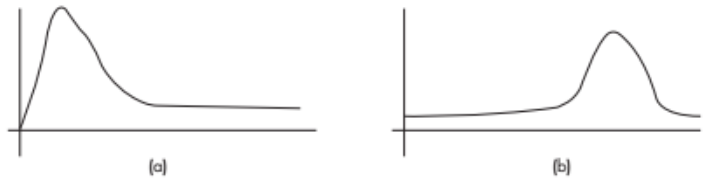


Figure 2.8: (a) Positive Skewed and (b) Negative Skewed Data

The dataset may also either have very high values or extremely low values. If the dataset has far higher values, then it is said to be skewed to the right. On the other hand, if the dataset has far more low values then it is said to be skewed towards left. If the tail is longer on the left-hand side and hump on the right-hand side, it is called positive skew. Otherwise, it is called negative skew.

The given dataset may have an equal distribution of data. The implication of this is that if the data is skewed, then there is a greater chance of outliers in the dataset. This affects the mean and median. Hence, this may affect the performance of the data mining algorithm. A perfect symmetry means the skewness is zero. In the case of skew, the median is greater than the mean. In positive skew, the mean is greater than the median.

Generally, for negatively skewed distribution, the median is more than the mean. The relationship between skew and the relative size of the mean and median can be summarized by a convenient numerical skew index known as Pearson 2 skewness coefficient.

$$\frac{3 \times (\mu - median)}{\sigma} \quad (2.12)$$

Also, the following measure is more commonly used to measure skewness. Let $X_1, X_2, \cdots, X_N$ be a set of 'N' values or observations then the skewness can be given as:

$$\frac{1}{N} \times \sum_{i=1}^{N} \frac{(x_i - \mu)^3}{\sigma^3} \quad (2.13)$$

Here, μ is the population mean and σ is the population standard deviation of the univariate data. Sometimes, for bias correction instead of N, $N - 1$ is used.

Kurtosis

Kurtosis also indicates the peaks of data. If the data is high peak, then it indicates higher kurtosis and vice versa.

Kurtosis is the measure of whether the data is heavy tailed or light tailed relative to normal distribution. It can be observed that normal distribution has bell-shaped curve with no long tails. Low kurtosis tends to have light tails. The implication is that there is no outlier data. Let $x_1, x_2, \cdots, x_N$ be a set of 'N' values or observations. Then, kurtosis is measured using the formula given below:

$$\frac{\sum_{i=1}^{N}(x_i - \bar{x})^4 / N}{\sigma^4} \quad (2.14)$$

It can be observed that $N - 1$ is used instead of N in the numerator of Eq. (2.14) for bias correction. Here, $\bar{x}$ and σ are the mean and standard deviation of the univariate data, respectively.

Some of the other useful measures for finding the shape of the univariate dataset are mean absolute deviation (MAD) and coefficient of variation (CV).

Mean Absolute Deviation (MAD)

MAD is another dispersion measure and is robust to outliers. Normally, the outlier point is detected by computing the deviation from median and by dividing it by MAD. Here, the absolute deviation between the data and mean is taken. Thus, the absolute deviation is given as:

$$|x - \mu| \tag{2.15}$$

The sum of the absolute deviations is given as $\Sigma|x - \mu|$

Therefore, the mean absolute deviation is given as: $\frac{\Sigma|x - \mu|}{N}$ (2.16)

Coefficient of Variation (CV)

Coefficient of variation is used to compare datasets with different units. CV is the ratio of standard deviation and mean, and %CV is the percentage of coefficient of variations.

2.5.5 Special Univariate Plots

The ideal way to check the shape of the dataset is a stem and leaf plot. A stem and leaf plot are a display that help us to know the shape and distribution of the data. In this method, each value is split into a 'stem' and a 'leaf'. The last digit is usually the leaf and digits to the left of the leaf mostly form the stem. For example, marks 45 are divided into stem 4 and leaf 5 in Figure 2.9.

The stem and leaf plot for the English subject marks, say, {45, 60, 60, 80, 85} is given in Figure 2.9.

Stem	Leaf
4	5
5	
6	0 0
7	
8	0 5

Figure 2.9: Stem and Leaf Plot for English Marks

It can be seen from Figure 2.9 that the first column is stem and the second column is leaf. For the given English marks, two students with 60 marks are shown in stem and leaf plot as stem-6 with 2 leaves with 0.

As discussed earlier, the ideal shape of the dataset is a bell-shaped curve. This corresponds to normality. Most of the statistical tests are designed only for normal distribution of data. A *Q-Q* plot can be used to assess the shape of the dataset. The *Q-Q* plot is a 2D scatter plot of an univariate data against theoretical normal distribution data or of two datasets – the quartiles of the first and second datasets. The normal *Q-Q* plot for marks x = [13 11 2 3 4 8 9] is given below in Figure 2.10.

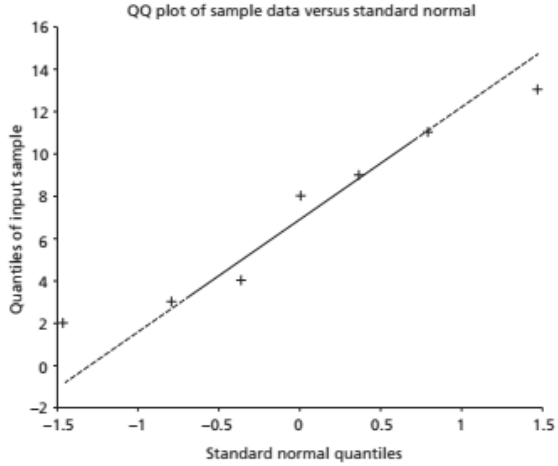


Figure 2.10: Normal *Q-Q* Plot

Ideally, the points fall along the reference line (45 Degree) if the data follows normal distribution. If the deviation is more, then there is greater evidence that the datasets follow some different distribution, that is, other than the normal distribution shape. In such a case, careful analysis of the statistical investigations should be carried out before interpretation.

This skewness, kurtosis, mean absolute deviation and coefficient of variation help in assessing the univariate data.

2.6 BIVARIATE DATA AND MULTIVARIATE DATA

Bivariate Data involves two variables. Bivariate data deals with causes of relationships. The aim is to find relationships among data. Consider the following Table 2.3, with data of the temperature in a shop and sales of sweaters.

Table 2.3: Temperature in a Shop and Sales Data

Temperature (in centigrade)	Sales of Sweaters (in thousands)
5	200
10	150
15	140
20	75
22	60
23	55
25	20

Here, the aim of bivariate analysis is to find relationships among variables. The relationships can then be used in comparisons, finding causes, and in further explorations. To do that, graphical display of the data is necessary. One such graph method is called scatter plot.

Scatter plot is used to visualize bivariate data. It is useful to plot two variables with or without nominal variables, to illustrate the trends, and also to show differences. It is a plot between explanatory and response variables. It is a 2D graph showing the relationship between two variables.

The scatter plot (Refer Figure 2.11) indicates strength, shape, direction and the presence of Outliers. It is useful in exploratory data before calculating a correlation coefficient or fitting regression curve.

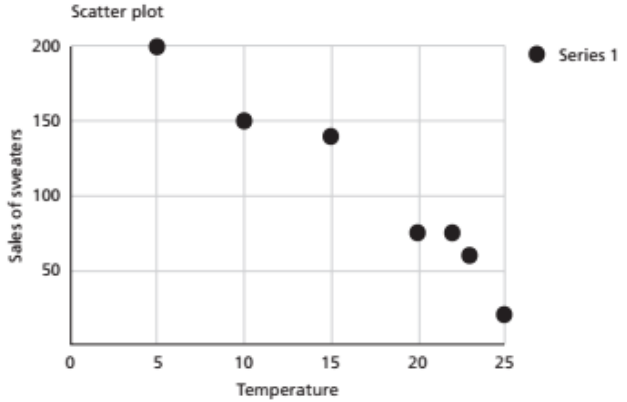


Figure 2.11: Scatter Plot

Line graphs are similar to scatter plots. The Line Chart for sales data is shown in Figure 2.12.

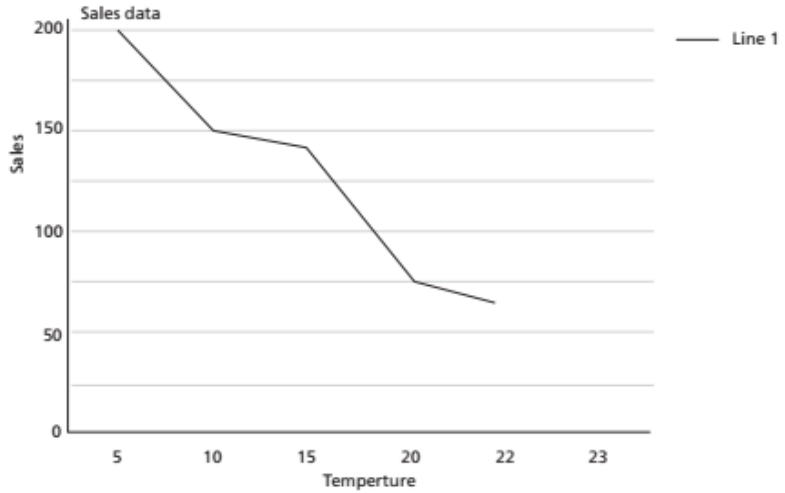


Figure 2.12: Line Chart

2.6.1 Bivariate Statistics

Covariance and Correlation are examples of bivariate statistics. Covariance is a measure of joint probability of random variables, say X and Y. Generally, random variables are represented in capital letters. It is defined as covariance(X, Y) or COV(X, Y) and is used to measure the variance between two dimensions. The formula for finding co-variance for specific x, and y are:

$$\text{cov}(X,Y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(X))(y_i - E(Y)) \tag{2.17}$$

Here, x_i and y_i are data values from X and Y. $E(X)$ and $E(Y)$ are the mean values of x_i and y_i. N is the number of given data. Also, the COV(X, Y) is same as COV(Y, X).

Example 2.6: Find the covariance of data $X = \{1, 2, 3, 4, 5\}$ and $Y = \{1, 4, 9, 16, 25\}$.

Solution: Mean(X) = $E(X) = \frac{15}{5} = 3$, Mean(Y) = $E(Y) = \frac{55}{5} = 11$. The covariance is computed using Eq. (2.17) as:

$$\frac{(1-3)(1-11) + (2-3)(4-11) + (3-30)(9-11) + (4-3)(16-11) + (5-3)(25-11)}{5} = 12$$

The covariance between X and Y is 12. It can be normalized to a value between −1 and +1. This is done by dividing it by the correlation of variables. This is called Pearson correlation coefficient. Sometimes, $N - 1$ is also can be used instead of N. In that case, the covariance is 60/4 = 15.

Correlation

The Pearson correlation coefficient is the most common test for determining any association between two phenomena. It measures the strength and direction of a *linear* relationship between the x and y variables.

The correlation indicates the relationship between dimensions using its sign. The sign is more important than the actual value.

1. If the value is positive, it indicates that the dimensions increase together.
2. If the value is negative, it indicates that while one-dimension increases, the other dimension decreases.
3. If the value is zero, then it indicates that both the dimensions are independent of each other.

If the dimensions are correlated, then it is better to remove one dimension as it is a redundant dimension.

If the given attributes are $X = (x_1, x_2, \cdots, x_N)$ and $Y = (y_1, y_2, \cdots, y_N)$, then the Pearson correlation coefficient, that is denoted as r, is given as:

$$r = \frac{COV(X,Y)}{\sigma_X \sigma_Y} \tag{2.18}$$

where, σ_X, σ_Y are the standard deviations of X and Y.

Example 2.7: Find the correlation coefficient of data $X = \{1, 2, 3, 4, 5\}$ and $Y = \{1, 4, 9, 16, 25\}$.

Solution: The mean values of X and Y are $\frac{15}{5} = 3$ and $\frac{55}{5} = 11$. The standard deviations of X and Y are 1.41 and 8.6486, respectively. Therefore, the correlation coefficient is given as ratio of covariance (12 from the previous problem 2.5) and standard deviation of x and y as per Eq. (2.18) as:

$$r = \frac{12}{1.41 \times 8.6486} \approx 0.984$$

2.7 MULTIVARIATE STATISTICS

In machine learning, almost all datasets are multivariable. Multivariate data is the analysis of more than two observable variables, and often, thousands of multiple measurements need to be conducted for one or more subjects.

The multivariate data is like bivariate data but may have more than two dependant variables. Some of the multivariate analysis are regression analysis, principal component analysis, and path analysis.

Id	*Attribute* 1	*Attribute* 2	*Attribute* 3
1	1	4	1
2	2	5	2
3	3	6	1

The mean of multivariate data is a mean vector and the mean of the above three attributes is given as (2, 7.5, 1.33). The variance of multivariate data becomes the covariance matrix. The mean vector is called centroid and variance is called dispersion matrix. This is discussed in the next section.

Multivariate data has three or more variables. The aim of the multivariate analysis is much more. They are regression analysis, factor analysis and multivariate analysis of variance that are explained in the subsequent chapters of this book.

Heatmap

Heatmap is a graphical representation of 2D matrix. It takes a matrix as input and colours it. The darker colours indicate very large values and lighter colours indicate smaller values. The advantage of this method is that humans perceive colours well. So, by colour shaping, larger values can be perceived well. For example, in vehicle traffic data, heavy traffic regions can be differentiated from low traffic regions through heatmap.

In Figure 2.13, patient data highlighting weight and health status is plotted. Here, X-axis is weights and Y-axis is patient counts. The dark colour regions highlight patients' weights vs patient counts in health status.

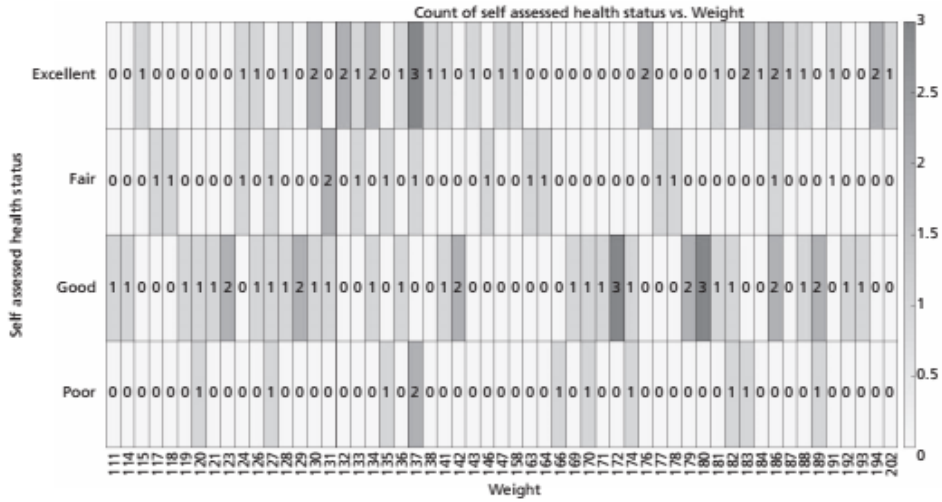


Figure 2.13: Heatmap for Patient Data

Pairplot

Pairplot or scatter matrix is a data visual technique for multivariate data. A scatter matrix consists of several pair-wise scatter plots of variables of the multivariate data. All the results are presented in a matrix format. By visual examination of the chart, one can easily find relationships among the variables such as correlation between the variables.

A random matrix of three columns is chosen and the relationships of the columns is plotted as a pairplot (or scattermatrix) as shown below in Figure 2.14.

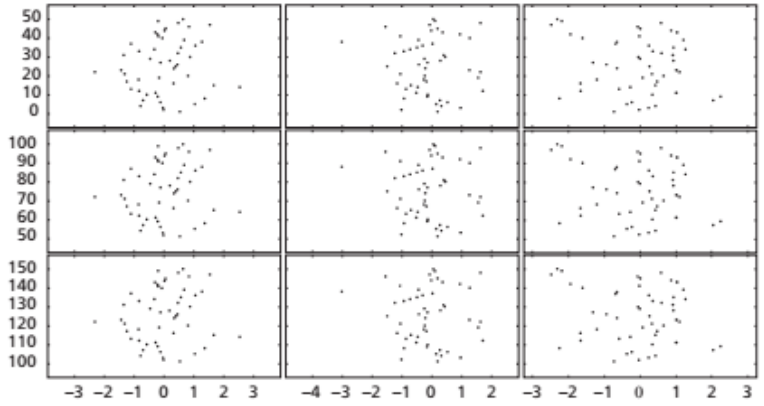


Figure 2.14: Pairplot for Random Data

2.8 ESSENTIAL MATHEMATICS FOR MULTIVARIATE DATA

Machine learning involves many mathematical concepts from the domain of Linear algebra, Statistics, Probability and Information theory. The subsequent sections discuss important aspects of linear algebra and probability.

'Linear Algebra' is a branch of mathematics that is central for many scientific applications and other mathematical subjects. While all branches of mathematics are crucial for machine learning, linear algebra plays a major large role as it is the mathematics of data. Linear algebra deals with linear equations, vectors, matrices, vector spaces and transformations. These are the driving forces of machine learning and machine learning cannot exist without these data types.

Let us discuss some of the important concepts of linear algebra now.

2.8.1 Linear Systems and Gaussian Elimination for Multivariate Data

A linear system of equations is a group of equations with unknown variables.

Let $Ax = y$, then the solution x is given as:

$$x = y/A = A^{-1} y \tag{2.19}$$

This is true if y is not zero and A is not zero. The logic can be extended for N-set of equations with 'n' unknown variables.

It means if $A = \begin{pmatrix} a_{11} & a_{12} \cdots a_{1n} \\ \vdots & \vdots \quad \vdots \quad \vdots \\ a_{1n} & a_{2n} \cdots a_{nn} \end{pmatrix}$ and $y = (y_1 \;\; y_2 \ldots y_n)$, then the unknown variable x can be computed as:

$$x = y/A = A^{-1} y \tag{2.20}$$

If there is a unique solution, then the system is called consistent independent. If there are various solutions, then the system is called consistent dependant. If there are no solutions and if the equations are contradictory, then the system is called inconsistent.

For solving large number of system of equations, Gaussian elimination can be used. The procedure for applying Gaussian elimination is given as follows:

1. Write the given matrix.
2. Append vector y to the matrix A. This matrix is called augmentation matrix.
3. Keep the element a_{11} as pivot and eliminate all a_{11} in second row using the matrix operation, $R_2 - \left(\frac{a_{21}}{a_{11}}\right)$, here R_2 is the second row and $\left(\frac{a_{21}}{a_{11}}\right)$ is called the multiplier. The same logic can be used to remove a_{11} in all other equations.
4. Repeat the same logic and reduce it to reduced echelon form. Then, the unknown variable as:

$$x_n = \frac{y_{nn}}{a_{nn}} \tag{2.21}$$

5. Then, the remaining unknown variables can be found by back-substitution as:

$$x_{n-1} = \frac{y_{n-1} - a_{n-1} \times x_n}{a_{(n-1)(n-1)}} \tag{2.22}$$

This part is called backward substitution.

To facilitate the application of Gaussian elimination method, the following row operations are applied:

1. Swapping the rows
2. Multiplying or dividing a row by a constant
3. Replacing a row by adding or subtracting a multiple of another row to it

These concepts are illustrated in Example 2.8.

Example 2.8: Solve the following set of equations using Gaussian Elimination method.

$$2x_1 + 4x_2 = 6$$
$$4x_1 + 3x_2 = 7$$

Solution: Rewrite this in matrix form as follows:

$$\begin{pmatrix} 2 & 4 & | & 6 \\ 4 & 3 & | & 7 \end{pmatrix}$$
$$\sim \begin{pmatrix} 2 & 4 & | & 6 \\ 4 & 3 & | & 7 \end{pmatrix} R_1 = \frac{R_1}{2}$$

Apply the transformation by dividing the row 1 by 2. There are no general guidelines of row operations other than reducing the given matrix to row echelon form. The operator ~ means reducing to. The above matrix can further be reduced as follows:

$$\sim \begin{pmatrix} 1 & 2 & | & 3 \\ 4 & 3 & | & 7 \end{pmatrix} R_2 = R_2 - 4R_1$$
$$\sim \begin{pmatrix} 1 & 2 & | & 3 \\ 0 & -5 & | & -5 \end{pmatrix} R_2 = R_2/-5$$
$$\sim \begin{pmatrix} 1 & 2 & | & 3 \\ 0 & 1 & | & 1 \end{pmatrix} R_1 = R_1 - 2R_2$$
$$\sim \begin{pmatrix} 1 & 0 & | & 1 \\ 0 & 1 & | & 1 \end{pmatrix}$$

Therefore, in the reduced echelon form, it can be observed that:

$$x_2 = 1$$
$$x_1 = 1$$

2.8.2 Matrix Decompositions

It is often necessary to reduce a matrix to its constituent parts so that complex matrix operations can be performed. These methods are also known as matrix factorization methods.

The most popular matrix decomposition is called eigen decomposition. It is a way of reducing the matrix into eigen values and eigen vectors.

Then, the matrix A can be decomposed as:

$$A = Q \Lambda Q^T \tag{2.23}$$

where, Q is the matrix of eigen vectors, Λ is the diagonal matrix and Q^T is the transpose of matrix Q.

LU Decomposition

One of the simplest matrix decompositions is *LU* decomposition where the matrix *A* can be decomposed matrices:

$$A = LU$$

Here, *L* is the lower triangular matrix and *U* is the upper triangular matrix. The decomposition can be done using Gaussian elimination method as discussed in the previous section. First, an identity matrix is augmented to the given matrix. Then, row operations and Gaussian elimination is applied to reduce the given matrix to get matrices *L* and *U*.

Example 2.9 illustrates the application of Gaussian elimination to get *LU*.

Example 2.9: Find *LU* decomposition of the given matrix:

$$A = \begin{pmatrix} 1 & 2 & 4 \\ 3 & 3 & 2 \\ 3 & 4 & 2 \end{pmatrix}$$

Solution: First, augment an identity matrix and apply Gaussian elimination. The steps are as shown in:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 4 \\ 3 & 3 & 2 \\ 3 & 4 & 2 \end{bmatrix} \quad \boxed{\text{Initial Matrix}}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 4 \\ 0 & -3 & -10 \\ 3 & 4 & 2 \end{bmatrix} \quad \boxed{R_2 = R_2 - 3R_1}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 4 \\ 0 & -3 & -10 \\ 0 & -2 & -10 \end{bmatrix} \quad \boxed{R_3 = R_3 - 3R_1}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 3 & \frac{2}{3} & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 4 \\ 0 & -3 & -10 \\ 0 & 0 & \frac{-10}{3} \end{bmatrix} \quad \boxed{R_3 = R_3 - \frac{2}{3}R_2}$$

Now, it can be observed that the first matrix is *L* as it is the lower triangular matrix whose values are the determiners used in the reduction of equations above such as 3, 3 and 2/3. The second matrix is *U*, the upper triangular matrix whose values are the values of the reduced matrix because of Gaussian elimination.

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 3 & \frac{2}{3} & 1 \end{pmatrix} \text{ and } U = \begin{pmatrix} 1 & 2 & 4 \\ 0 & -3 & -10 \\ 0 & 0 & -\frac{10}{3} \end{pmatrix}.$$

It can be cross verified that the multiplication of *LU* yields the original matrix *A*. What are the applications of *LU* decomposition? Some of the applications are finding matrix inverses and determinant. If the order of the matrix is large, then this method can be used.

2.8.3 Machine Learning and Importance of Probability and Statistics

Machine learning is linked with statistics and probability. Like linear algebra, statistics is the heart of machine learning. The importance of statistics needs to be stressed as without statistics; analysis of data is difficult. Probability is especially important for machine learning. Any data can be assumed to be generated by a probability distribution. Machine learning datasets have multiple data that are generated by multiple distributions. So, a knowledge of probability distribution and random variables are must for better understanding of the machine learning concepts.

What is the objective for an experiment? This is about hypothesis or constructing a model. Machine learning has many models. So, the theory of hypothesis, construction of models, evaluating the models using hypothesis testing, significance of the model and evaluation of the model are all linking factors of machine learning and probability theory. Similarly, the construction of datasets using sampling theory is also required.

The subsequent section will deal with the important concepts of statistics and probability.

Probability Distributions

A probability distribution of a variable, say *X*, summarizes the probability associated with *X*'s events. Distribution is a parameterized mathematical function. In other words, distribution is a function that describes the relationship between the observations in a sample space.

Consider a set of data. The data is said to follow a distribution if it obeys a mathematical function that characterizes that distribution. The function can be used to calculate the probability of individual observations.

Probability distributions are of two types:

1. Discrete probability distribution
2. Continuous probability distribution

The relationships between the events for a continuous random variable and their probabilities is called a continuous probability distribution. It is summarized as Probability Density Function (PDF). PDF calculates the probability of observing an instance. The plot of PDF shows the shape of the distribution. Cumulative Distributive Function (CDF) computes the probability of an observation ≤ value.

Both PDF and CDF are continuous values. The discrete equivalent of PDF in discrete distribution is called Probability Mass Function (PMF).

The probability of an event cannot be detected directly. It should be computed as the area under the curve for a small interval around the specific outcome. This is defined as CDF.

Let us discuss some of the distributions that are encountered in machine learning.

Continuous Probability Distributions Normal, Rectangular, and Exponential distributions fall under this category.

1. Normal Distribution – Normal distribution is a continuous probability distribution. This is also known as gaussian distribution or bell-shaped curve distribution. It is the most common distribution function. The shape of this distribution is a typical bell-shaped curve. In normal distribution, data tends to be around a central value with no bias on left or right. The heights of the students, blood pressure of a population, and marks scored in a class can be approximated using normal distribution.

   PDF of the normal distribution is given as:

$$f(x, \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.24}$$

   Here, μ is mean and σ is the standard deviation. Normal distribution is characterized by two parameters – mean and variance.

   Mostly, one uses the normal distribution curve of mean 0 and a *SD* of 1. In normal distribution, mean, median and mode are same. The distribution extends from $-\infty$ to $+\infty$. Standard deviation is how the data is spread out.

   One important concept associated with normal distribution is *z*-score. It can be computed as:

   $z = \frac{x - \mu}{\sigma}$. When μ is zero and σ is 1, *z*-score is same as x. This is useful to normalize the data.

   Most of the statistical tests expect data to follow normal distribution. To check it, normality tests are used. Normality test of the data can be done by Q-Q plot where CDF of one random variable follows CDF of normal distribution. Then, quantity of one distribution is plotted against other distributions. If they are same, then the plot closely follows the straight line from bottom-left to top-right.

2. Rectangular Distribution – This is also known as uniform distribution. It has equal probabilities for all values in the range a, b. The uniform distribution is given as follows:

$$P(X = x) = \begin{cases} \frac{1}{b-a} & \text{for } a \le x \le b \\ 0 & \text{Otherwise} \end{cases} \tag{2.25}$$

3. Exponential Distribution – This is a continuous uniform distribution. This probability distribution is used to describe the time between events in a Poisson process. Exponential distribution is another special case of Gamma distribution with a fixed parameter of 1. This distribution is helpful in modelling of time until an event occurs.

   The PDF is given as follows:

$$f(x, \lambda) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x \ge 0 \quad (\lambda > 0) \\ 0 & \text{if } x < 0 \end{cases} \tag{2.26}$$

   Here, x is a random variable and λ is called rate parameter. The mean and standard deviation of exponential distribution is given as β, where, $\beta = \frac{1}{\lambda}$.

Discrete Distribution Binomial, Poisson, and Bernoulli distributions fall under this category.

1. Binomial Distribution – Binomial distribution is another distribution that is often encountered in machine learning. It has only two outcomes: success or failure. This is also called Bernoulli trial.

   The objective of this distribution is to find probability of getting success k out of n trials. The way to get success out of k out of n number of trials is given as:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{2.27}$$

   The binomial distribution function is given as follows, where p is the probability of success and probability of failure is $(1 - p)$. The probability of success in a certain number of trials is given as:

$$p^k(1-p)^{n-k} \text{ or } p^k q^{n-k} \tag{2.28}$$

   Combining both, one gets PDF of binomial distribution as:

$$\binom{n}{k} p^k (1-p)^{n-k} \tag{2.29}$$

   Here, p is the probability of each choice, k is the number of choices, and n is the total number of choices. The mean of binomial distribution is given below:

$$\mu = n \times p \tag{2.30}$$

   And the variance is given as:

$$\sigma^2 = np(1-p) \tag{2.31}$$

   Hence, the standard deviation is given as:

$$\sigma = \sqrt{np(1-p)} \tag{2.32}$$

2. Poisson Distribution – It is another important distribution that is quite useful. Given an interval of time, this distribution is used to model the probability of a given number of events k. The mean rule λ is inclusive of previous events. Some of the examples of Poisson distribution are number of emails received, number of customers visiting a shop and the number of phone calls received by the office.

   The PDF of Poisson distribution is given as follows:

$$f(X = x; \lambda) = Pr[X = x] = \frac{e^{-\lambda}\lambda^x}{x!} \tag{2.33}$$

   Here, x is the number of times the event occurs and λ is the mean number of times an event occurs.

   The mean is the population mean at number of emails received and the standard deviation is $\sqrt{\lambda}$.

3. Bernoulli Distribution – This distribution models an experiment whose outcome is binary. The outcome is positive with p and negative with $1 - p$. The PMF of this distribution is given as:

$$f(k; p) = \begin{cases} q = 1 - p & \text{if } k = 0 \\ p & \text{if } k = 1. \end{cases} \tag{2.34}$$

   The mean is p and variance is $p(1 - p) = q$

Density Estimation

Let there be a set of observed values $x_1, x_2, \cdots, x_n$ from a larger set of data whose distribution is not known. Density estimation is the problem of estimating the density function from an observed data. The estimated density function, denoted as, $p(x)$ can be used to value directly for any unknown data, say x_t as $p(x_t)$. If its value is less than ε, then x_t is not an outlier or anomaly data. Else, it is categorized as an anomaly data.

There are two types of density estimation methods, namely parametric density estimation and non-parametric density estimation.

Parametric Density Estimation It assumes that the data is from a known probabilistic distribution and can be estimated as $p(x \mid \Theta)$, where, Θ is the parameter. Maximum likelihood function is a parametric estimation method.

Maximum Likelihood Estimation For a sample of observations, one can estimate the probability distribution. This is called density estimation. Maximum Likelihood Estimation (MLE) is a probabilistic framework that can be used for density estimation. This involves formulating a function called likelihood function which is the conditional probability of observing the observed samples and distribution function with its parameters. For example, if the observations are $X = \{x_1, x_2, \cdots, x_n\}$, then density estimation is the problem of choosing a PDF with suitable parameters to describe the data. MLE treats this problem as a search or optimization problem where the probability should be maximized for the joint probabilities of X and its parameter, θ.

For example, this is expressed as $p(X; \theta)$, where, $X = \{x_1, x_2, \cdots, x_n\}$

The likelihood of observing the data is given as a function $L(X; \theta)$. The objective of MLE is to maximize this function as $max\ L(X; \theta)$.

The joint probability of this problem can be restated as $\prod_{i=1}^{n} p(x_i; \theta)$.

The computation of the above formula is unstable and the hence the problem is restated as maximum of log conditional probability given θ. This is given as:

$$\sum_{i=1}^{n} \log p(x_i; \theta) \tag{2.35}$$

Instead of maximizing, one can minimize this function as:

$min = -\sum_{i=1}^{n} \log p(x_i; \theta)$ as, often, minimization is preferred over maximization.

This is called negative log-likelihood function.

The relevance of this theory of MLE for machine learning is that MLE can solve the problem of predictive modelling in machine learning. Regression problem is treated in Chapter 5 using the principle of least-square approach. The same problem can be viewed from MLE perspective too. If one assumes that the regression problem can be framed as predicting output y given input x, then for $p(y/x)$, the MLE framework can be applied as:

$$max \sum \log(y \mid x_i, h) \tag{2.36}$$

Here, h is the linear regression model. If Gaussian distribution is assumed as it is an obvious fact that most of the data follow Gaussian distribution, then MLE can be stated as:

$$\max \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{y_i - h(x_i;\beta)}{2\sigma^2}} \tag{2.37}$$

Here, β is the regression coefficient and x_i is the given sample. One can maximize this function or minimize the negative log likelihood function to provide a solution for linear regression problem. The Eq. (2.37) yields the same answer of the least-square approach.

Stochastic gradient descent (SGD) is an algorithm that is normally used to maximize or minimize any function. This algorithm can be used to minimize the above function to get the results effectively. Hence, many statistical packages use this approach for solving linear regression problem. Theoretically, any model can be used instead of linear regression in the above equation. Logistic regression is another model that can be used in MLE framework. Logistic regression is discussed in Chapter 5 of this book.

Gaussian Mixture Model and Expectation-Maximization (EM) Algorithm In machine learning, clustering is one of the important tasks. It is discussed in Chapter 13. MLE framework is quite useful for designing model-based methods for clustering data. A model is a statistical method and data is assumed to be generated by a distribution model with its parameter, θ. There may be many distributions involved and that is why it is called as mixture model. Since, Gaussian are normally assumed for data, this mixture model is categorized as Gaussian Mixture Model (GMM).

The EM algorithm is one algorithm that is commonly used for estimating the MLE in the presence of latent or missing variables. What is a latent variable? Let us assume that the dataset includes weights of boys and girls. Considering the fact that the boys' weights would be slightly more than the weights of the girls, one can assume that the larger weights are generated by one gaussian distribution with one set of parameters while girls' weights are generated with another set of parameters. There is an influence of gender in the data, but it is not directly present or observable. These are called latent variables. The EM algorithm is effective for estimating the PDF in the presence of latent variables.

Generally, there can be many unspecified distributions with different set of parameters. The EM algorithm has two stages:

1. Expectation (E) Stage – In this stage, the expected PDF and its parameters are estimated for each latent variable.
2. Maximization (M) stage – In this, the parameters are optimized using the MLE function.

This process is iterative, and the iteration is continued till all the latent variables are fitted by probability distributions effectively along with the parameters.

Non-parametric Density Estimation A non-parametric estimation can be generative or discriminative. Parzen window is a generative estimation method that finds $p(x \mid \Theta)$ as conditional density. Discriminative methods directly compute $p(\Theta \mid x)$ as posteriori probability. Parzen window and k-Nearest Neighbour (KNN) rule are examples of non-parametric density estimation. Let us discuss about them now.

Parzen Window Let there be 'n' samples, $X = \{x_1, x_2, \cdots, x_n\}$

The samples are drawn independently, called as identically independent distribution. Let R be the region that covers 'k' samples of total 'n' samples. Then, the probability density function is given as:

$$p = k/n \tag{2.38}$$

The estimate is given as:

$$p(x) = \frac{k/n}{V} \tag{2.39}$$

where, V is the volume of the region R. If R is the hypercube centered at x and h is the length of the hypercube, the volume V is h^2 for 2D square cube and h^3 for 3D cube.

The Parzen window is given as follows:

$$\varphi\left(\frac{x_i - x}{h}\right) = \begin{cases} 1 \text{ if } \dfrac{|x_{ik} - x_k|}{h} < \dfrac{1}{2} \\ 0 \text{ otherwise} \end{cases} \tag{2.40}$$

The window indicates if the sample is inside the region or not. The Parzen probability density function estimate using Eq. (2.40) is given as:

$$p(x) = \frac{k/n}{V} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{V_n}\varphi\left(\frac{x_i - x}{h}\right) \tag{2.41}$$

This window can be replaced by any other function too. If Gaussian function is used, then it is called Gaussian density function.

KNN Estimation The KNN estimation is another non-parametric density estimation method. Here, the initial parameter k is determined and based on that k-neighbours are determined. The probability density function estimate is the average of the values that are returned by the neighbours.

2.9 OVERVIEW OF HYPOTHESIS

Data collection alone is not enough. Data must be interpreted to give a conclusion. The conclusion should be a structured outcome. This assumption of the outcome is called a hypothesis. Statistical methods are used to confirm or reject the hypothesis. The assumption of the statistical test is called null hypothesis. It is also called as hypothesis zero ($H0$). In other words, hypothesis is the existing belief. The violation of this hypothesis is called first hypothesis ($H1$) or hypothesis one. This is the hypothesis the researcher is trying to establish.

There are two types of hypothesis tests, parametric and non-parametric. Parametric tests are based on parameters such as mean and standard deviation. Non-parametric tests are dependent on characteristics such as independence of events or data following certain distribution.

Statistical tests help to:

1. Define null and alternate hypothesis
2. Describe the hypothesis using parameters
3. Identify the statistical test and statistics

4. Decide the criteria called significance value α
5. Compute p-value (probability value)
6. Take the final decision of accepting or rejecting the hypothesis based on the parameters

Hypothesis testing is particularly important as it is an integral part of the learning algorithms. Generally, the data size is small. So, one may have to know whether the hypothesis will work for additional samples and how accurate it is. No matter how effective the statistical tests are, two kinds of errors are involved, that are Type I and Type II.

Type I error is the incorrect rejection of a true null hypothesis and is called false positive. Type II error is the incomplete failure of rejecting a false hypothesisand is called false negative.

During these calculations, one must include the size of the data sample. Degree of freedom indicates the number of independent pieces of information used for the test. It is indicated as v. The mean or variance can be used to indicate the degree of freedom.

Hypothesis Testing

Let us define two important errors called sample error and true (or actual error). Let us assume that D is the unknown distribution, Target function is $f(x)$: $x \geq \{0, 1\}$, x is the instance, $h(x)$ is the hypothesis, and sample set is S that derives the samples on instances drawn from X. Then, the actual error is denoted as:

$$error_D(h) = pr_{x \in D}\{f(x) \neq h(x)\} \tag{2.42}$$

In other words, true error is the probability that the hypothesis will misclassify an instance that is drawn at random. The point is that population is very large and hence it is not possible to determine true error and can only be estimated. So, another error is called sample error or estimator.

Sample error is with respect to sample S. It is the probability for instances drawn from X, that is, the fractions of S that are misclassified. The sample error is given as follows:

$$error_S(h) = \frac{1}{n}\sum_{x \in S}\delta(f(x), h(x)) \tag{2.43}$$

Here, $\delta(f(x), h(x))$ is 1 if $f(x) \neq h(x)$, otherwise is zero.

The data associated with the sample has two problems. One is bias and another is variance.

Since the dataset is small in size, the data may be overfitting. To generalize the concept of the population with smaller dataset is difficult. If S is the sample set, then the data is biased and can be estimated as $E[error_S(h)] - error_D(h)$. Another data error is variance, that is, deviation of $error_S(h)$ from $error_D(h)$. Here, errors are sample error and true or actual error.

p-value

Statistical tests can be performed to either accept or reject the null hypothesis. This is done by the value called p-value or probability value. It indicates the probability of hypothesis being true. The p-value is used to interpret or quantify the test. For example, a statistical test result may give a value of 0.03. Then, one can compare it with the level 0.05. As $0.03 < 0.05$, the result is assumed to be significant. This means that the variables tested are not independent. Here, 0.05 is called significant level. In general, significant level is called α and p-value is compared with α. If p-value $\leq \alpha$, then the hypothesis $H1$ is rejected and if p-value $> \alpha$, then the hypothesis $H0$ is rejected.

Confidence Intervals

The acceptance or rejection of the hypothesis can also be done using confidence interval. The confidence interval is computed as:

Confidence interval = 1 – significant level (2.44)

Confidence level is the range of values that indicates the location of true mean. Confidence intervals indicate the confidence of the result. If the confidence level is 90%, then it infers that there is 90% of chance that the true mean lies in this range and remaining 10% indicates that true mean in not present. For finding this, one requires mean and standard deviation. Then, x can be given as $mean(x) \pm z \times \frac{s}{\sqrt{N}}$. Here, s is the standard deviation, N is the number of samples, and z is the value associated with 90% and is called % of confidence. This is also called as margin of error.

Sample error is the unbiased estimate of true error. If no information is provided, then both errors are the same. It is, however, often safe to suggest a margin of confidence associated with the hypothesis. The hypothesis with 95% confidence about the sample error can be given as follows:

$$error_S(h) \pm 1.96\sqrt{\frac{error_S(h)(1-error_S(h))}{n}} \tag{2.45}$$

This 1.96 indicates the 95% confidence of the error. The number 1.96 can be replaced by any number that is associated with different levels of confidence. The procedure to estimate the difference between two hypothesis, say h_1 and h_2, is as follows:

1. A parameter d can be chosen to estimate the error of two hypothesis:
2. $d = error_D(h_1) - error_D(h_2)$ (2.46)

   Here, there are two hypothesis h_1 and h_2 tested on two sample sets s_1 and s_2. Similarly, n_1 and n_2 are randomly drawn number of samples.
3. The estimator $\hat{d}$ can be estimated as the difference as:

   $\hat{d} = error_D(h_1) - error_D(h_2)$
4. The confidence intervals can be used for the estimator also as follows:

$$\hat{d} \pm z_n\sqrt{\frac{error_{s_1}(h_1)-(1-error_{s_1}(h_1))}{n_1}+\frac{error_{s_2}(h_2)-(1-error_{s_2}(h_2))}{n_2}} \tag{2.47}$$

Sometimes, it is desirable to find interval L and U such that N% of the probability falls in this interval.

Let us discuss about comparing two algorithms based on the hypothesis. This is done based on statistical tests. Some of the important statistical tests are discussed in the next section.

2.9.1 Comparing Learning Methods

Some of the methods for comparing the learning programs are given below:

Z-test

Z-test assumes normal distribution of data whose population variation is known. The sample size is assumed to be large. The focus is to test the population mean. The z-statistic is given as:

$$Z = \frac{X-\mu}{\sqrt{\frac{\sigma^2}{n}}} \tag{2.48}$$

Here, X is the input data, and n is number of data elements. μ and σ are mean and standard deviation of X, respectively.

Example 2.10: Let 12 be the population mean (μ) with the population variance (σ^2) of 2. Consider the sample $X = \{1, 2, 3, 4, 5\}$. Apply z-test and show whether the result is significant.

Solution: The sample mean of $X = \frac{15}{5} = 3$ and the number of samples is $n = 5$. Substituting in Eq. (2.48) gives:

$$Z = \frac{X-\mu}{\frac{\sigma}{\sqrt{n}}} = -10.06$$

By checking the critical value at significance 0.05, one can find that the null hypothesis H0 is rejected.

t-test and Paired t-test

t-test is a hypothesis test and checks if the difference between two samples' mean is real or by chance. Here, data is continuous and randomly selected. There will only be small number of samples and variance between groups is real. The t-test statistics follows t-distribution under null hypothesis and is used when the number of samples <30. The distribution follows t-distribution rather than Gaussian distribution. It indicates whether two groups are different or not.

One Sample Test In this test, the mean of one group is checked against the set average that can be either theoretical value or population mean. So, the procedure is:

- Select a group
- Compute average
- Compare it with theoretical value and compute t-statistic:

$$t = \frac{m-\mu}{\frac{s}{\sqrt{n}}} \tag{2.49}$$

Here, t is t-statistic, m is the mean of the group, μ is the theoretical value or population mean, s is the standard deviation, and n is the group size or sample size.

Independent Two Sample *t*-test t-statistic for two groups A and B is computed as follows:

$$t = \frac{mean(A)-mean(B)}{\sqrt{\frac{(N_1-1)s_1^2+(N_2-1)s_2^2}{N_1+N_2-2}\left(\frac{1}{N_1}+\frac{1}{N_2}\right)}} \tag{2.50}$$

Here, mean(A) and mean(B) are for two different samples. N_1 and N_2 are sample sizes of two groups A and B. s^2 is the variance of the two samples and the degree of freedom is given as $N_1 + N_2 - 2$. Then, t-statistic is compared with the t-critical value.

Let there be one group {8, 6, 7, 6, 3} and another group {7, 8, 8, 8, 2}. The number of elements in each group (n) are 5. For the first group, the mean and variance are 6 and 3.5, respectively. For the second group, the mean and variance are 6.6 and 6.8, respectively. The degree of freedom is number of samples $N - 1 = 4$. The t-statistic formula can be applied for the above data and its value is –0.42. The critical value is 0.343. Since the value is less that the critical value, one can say the test is insignificant.

Paired *t*-test It is used to evaluate the hypothesis before and after intervention. The fact is that these samples are not independent. For example, consider the case of an effect of medication for a diabetic patient. The sequence is that first the sugar is tested, then the medication is done, and again the sugar test is conducted to study the effect of medication. In short, in paired t-test, the data is taken from the same subject twice. In an unpaired t-test, the samples are taken independently. In this only one group is involved. The t-statistic is computed as:

$$t = \frac{m - \mu}{\frac{s}{\sqrt{n}}}$$

Here, t is t-statistic, m is the mean of the group, μ is the theoretical value or population mean, s is the standard deviation, and n is the group size or sample size.

Chi-Square Test

Chi-Square test is a non-parametric test. The goodness-of-fit test statistics follows a Chi-Square distribution under null hypothesis and measures the statistical significance between observed frequency and expressed frequency, and each observation is independent of each other and follows normal distribution. This is used to compare the observed frequency of some observation (such as, frequency of buying different brands of fans) with the expected frequency of the observation (of buying of each fan by customers) and decide whether any statistical difference exists. This comparison is used to calculate the value of the Chi-Square statistic as:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{2.51}$$

Here, E is the expected frequency, O is the observed frequency and the degree of freedom is $C - 1$, where, C is number of categories. The Chi-Square test allows us to detect the duplication of data and helps to remove the redundancy of values.

Example 2.11: Consider the following Table 2.4, where the machine learning course registration is done by both boys and girls. There are 50 boys and 50 girls in the class and the registration of the course is given in the table. Apply Chi-Square test and find out whether any differences exist between boys and girls for course registration.

Table 2.4: Observed Data

Gender	Registered	Not Registered	Total
Boys	35	15	50
Girls	25	25	50
Total	60	40	100

Solution: Let the null hypothesis be H_0 when there is no difference between boys and girls and H_1 be the alternate hypothesis when there is a significant difference between boys and girls. For applying the Chi-Square test based on the observations, the expectation should be obtained by multiplying the total boys X registered/Total and Total girls X not registered/Total as shown in Table 2.5.

Table 2.5: Expected Data

Gender	Registered	Not Registered	Total
Boys	$\frac{50 \times 60}{100} = 30$	$\frac{50 \times 40}{100} = 20$	50
Girls	$\frac{50 \times 60}{100} = 30$	$\frac{50 \times 40}{100} = 20$	50
Total	60	40	100

The Chi-Statistic is obtained using Eq. (2.51) as follows:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} = \frac{(35-30)^2}{30} + \frac{(15-20)^2}{20} + \frac{(25-30)^2}{30} + \frac{(25-20)^2}{20} = 4.166$$ for degree of freedom = number of categories –1 = 2 – 1 = 1. The p value for this statistic is 0.0412. This is less than 0.05. Therefore, the result is significant.

2.10 FEATURE ENGINEERING AND DIMENSIONALITY REDUCTION TECHNIQUES

Features are attributes. Feature engineering is about determining the subset of features that form an important part of the input that improves the performance of the model, be it classification or any other model in machine learning.

Feature engineering deals with two problems – Feature Transformation and Feature Selection. Feature transformation is extraction of features and creating new features that may be helpful in increasing performance. For example, the height and weight may give a new attribute called Body Mass Index (BMI).

Feature subset selection is another important aspect of feature engineering that focuses on selection of features to reduce the time but not at the cost of reliability.

The subset selection reduces the dataset size by removing irrelevant features and constructs a minimum set of attributes for machine learning. If the dataset has n attributes, then time complexity is extremely high as n dimensions need to be processed for the given dataset. For n attributes, there are 2^n possible subsets. If the value of n is high, the problem becomes intractable. This is called 'curse of dimensionality'. Since, as the number of dimensions increases, the time complexity increases. The remedy is that some of the components that do not contribute much can be deleted. This results in the reduction of dimensionality. Choosing optimal attributes becomes a graph search problem. Typically, the feature subset selection problem uses greedy approach by looking for the best choice at the time using locally optimal choice while hoping that it would lead to global optimal solutions.

The features can be removed based on two aspects:

1. Feature relevancy – Some features contribute more for classification than other features. For example, a mole on the face can help in face detection than common features like nose. In simple words, the features should be relevant. The relevancy of the features can be determined based on information measures such as mutual information, correlation-based features like correlation coefficient and distance measures. Distance measures are discussed in Chapter 13 of this book.
2. Feature redundancy – Some features are redundant. For example, when a database table has a field called Date of birth, then age field is not relevant as age can be computed easily from date of birth. This helps in removing the column age that leads to reduction of dimension one.

So, the procedure is:

1. Generate all possible subsets
2. Evaluate the subsets and model performance
3. Evaluate the results for optimal feature selection

Filter-based selection uses statistical measures for assessing features. In this approach, no learning algorithm is used. Correlation and information gain measures like mutual information and entropy are all examples of this approach.

Wrapper-based methods use classifiers to identify the best features. These are selected and evaluated by the learning algorithms. This procedure is computationally intensive but has superior performance.

Let us discuss some of the important algorithms that fall under this category.

2.10.1 Stepwise Forward Selection

This procedure starts with an empty set of attributes. Every time, an attribute is tested for statistical significance for best quality and is added to the reduced set. This process is continued till a good reduced set of attributes is obtained.

2.10.2 Stepwise Backward Elimination

This procedure starts with a complete set of attributes. At every stage, the procedure removes the worst attribute from the set, leading to the reduced set.

Combined Approach Both forward and reverse methods can be combined so that the procedure can add the best attribute and remove the worst attribute.

2.10.3 Principal Component Analysis

The idea of the principal component analysis (PCA) or KL transform is to transform a given set of measurements to a new set of features so that the features exhibit high information packing properties. This leads to a reduced and compact set of features. Basically, this elimination is made possible because of the information redundancies. This compact representation is of a reduced dimension.

Consider a group of random vectors of the form:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

The *mean vector* of the set of random vectors is defined as:

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

The operator E refers to the expected value of the population. This is calculated theoretically using the probability density functions (PDF) of the elements x_i and the joint probability density functions between the elements x_i and x_j. From this, the covariance matrix can be calculated as:

$$C = E\{(x - m_x)(x - m_x)^T\} \tag{2.52}$$

For M random vectors, when M is large enough, the mean vector and covariance matrix can be approximately calculated as:

$$m_x = \frac{1}{M}\sum_{k=1}^{M} x_k \tag{2.53}$$

$$A = \frac{1}{M}\sum_{k=1}^{M} x_k x_k^T - m_x m_x^T \tag{2.54}$$

This covariance matrix is real and symmetric. If e_i and λ_i (where, $i = 1, 2, \ldots, n$) be the set of eigen vectors and corresponding eigen values of the covariance matrix, the eigen values can be arranged in a descending order so that $\lambda_i \geq \lambda_{i+1}$ for $i = 1, 2, \ldots, n - 1$. The corresponding eigen vectors are calculated. Based on this, the transform kernel is constructed. Let the transform kernel be A. Then, the matrix rows are formed from the eigen vectors of the covariance matrix.

The mapping of the vectors x to y using the transformation can now be described as:

$$y = A(x - m_x) \tag{2.55}$$

This transform is also called as Karhunen-Loeve or Hoteling transform. The original vector x can now be reconstructed as follows:

$$x = A^T y + m_x \tag{2.56}$$

The goal of PCA is to reduce the set of attributes to a newer, smaller set that captures the variance of the data. The variance is captured by fewer components, which would give the same result as the original, with all the attributes. Here, instead of using all the eigen vectors of the covariance matrix, only a small set that exhibits the variance can be used. By using a small set, the KL transform can achieve maximum compression of the available data.

If K largest eigen values are used, the recovered information would be:

$$x = A_K^T y + m_x \tag{2.57}$$

The advantages of PCA are immense. It reduces the attribute list by eliminating all irrelevant attributes. The PCA algorithm is as follows:

1. The target dataset x is obtained
2. The mean is subtracted from the dataset. Let the mean be m. Thus, the adjusted dataset is $X - m$. The objective of this process is to transform the dataset with zero mean.
3. The covariance of dataset x is obtained. Let it be C.

4. Eigen values and eigen vectors of the covariance matrix are calculated.
5. The eigen vector of the highest eigen value is the principal component of the dataset. The eigen values are arranged in a descending order. The feature vector is formed with these eigen vectors in its columns.

   Feature vector = {eigen vector$_1$, eigen vector$_2$, … , eigen vector$_n$}
6. Obtain the transpose of feature vector. Let it be A.
7. PCA transform is $y = A \times (x - m)$, where x is the input dataset, m is the mean, and A is the transpose of the feature vector.

The original data can be retrieved using the formula given below:

$$\text{Original data } (f) = \{(A)^{-1} \times y\} + m \tag{2.58}$$

$$= \{(A)^{T} \times y\} + m \tag{2.59}$$

The new data is a dimensionally reduced matrix that represents the original data. Therefore, PCA is effective in removing the attributes that do not contribute. If the original data is required, it can be obtained with no loss of information. Scree plot is a visualization technique to visualize the principal components or variables that play a more important role as compared to other attributes. Scree plot is a visualization technique to visualize the principal components visually. For a randomly selected dataset of 246 attributes, PCA is applied and its scree plot is shown in Figure 2.15. The scree plot indicates that only 6 out of 246 attributes are important.

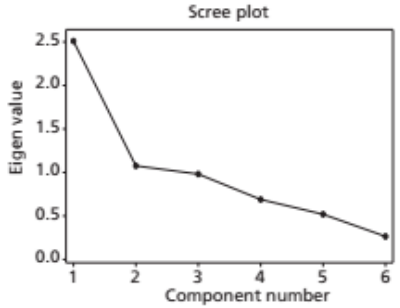


Figure 2.15: Scree Plot

From Figure 2.15, one can infer the relevance of the attributes. The scree plot indicates that the first attribute is more important than all other attributes.

Example 2.12: Let the data points be $\begin{pmatrix}2\\6\end{pmatrix}$ and $\begin{pmatrix}1\\7\end{pmatrix}$. Apply PCA and find the transformed data. Again, apply the inverse and prove that PCA works.

Solution: One can combine two vectors into a matrix as follows:

The mean vector can be computed as Eq. (2.53) as follows:

$$\mu = \begin{pmatrix}\frac{2+1}{2}\\ \frac{6+7}{2}\end{pmatrix} = \begin{pmatrix}1.5\\6.5\end{pmatrix}$$

As part of PCA, the mean must be subtracted from the data to get the adjusted data:

$$x_1 = \begin{pmatrix}2-1.5\\6-6.5\end{pmatrix} = \begin{pmatrix}0.5\\-0.5\end{pmatrix}$$

$$x_2 = \begin{pmatrix}1-1.5\\7-6.5\end{pmatrix} = \begin{pmatrix}-0.5\\0.5\end{pmatrix}$$

One can find the covariance for these data vectors. The covariance can be obtained using Eq. (2.54):

$$m_1 = \begin{pmatrix}0.5\\-0.5\end{pmatrix}\begin{pmatrix}0.5 & -0.5\end{pmatrix} = \begin{pmatrix}0.25 & -0.25\\-0.25 & 0.25\end{pmatrix}$$

$$m_2 = \begin{pmatrix}-0.5\\0.5\end{pmatrix}\begin{pmatrix}-0.5 & 0.5\end{pmatrix} = \begin{pmatrix}0.25 & -0.25\\-0.25 & 0.25\end{pmatrix}$$

The final covariance matrix is obtained by adding these two matrices as:

$$C = \begin{pmatrix}0.5 & -0.5\\-0.5 & 0.5\end{pmatrix}$$

The eigen values and eigen vectors of matrix C can be obtained (left as an exercise) as $\lambda_1 = 1$, $\lambda_1 = 0$. The eigen vectors are $\begin{pmatrix}-1\\1\end{pmatrix}$ and $\begin{pmatrix}1\\1\end{pmatrix}$. The matrix A can be obtained by packing the eigen vector of these eigen values (after sorting it) of matrix C. For this problem, $A = \begin{pmatrix}-1 & 1\\1 & 1\end{pmatrix}$. The transpose of A, $A^T = \begin{pmatrix}-1 & 1\\1 & 1\end{pmatrix}$ is also the same matrix as it is an orthogonal matrix. The matrix can be normalized by diving each elements of the vector, by the norm of the vector to get:

$$A = \begin{pmatrix}-\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{pmatrix}$$

One can check that the PCA matrix A is orthogonal. A matrix is orthogonal is $A^{-1} = A$ and $AA^{-1} = I$.

$$AA^T = \begin{pmatrix}-\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{pmatrix}\begin{pmatrix}-\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{pmatrix}$$

$$= \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}$$

The transformed matrix y using Eq. (2.55) is given as:

$$y = A \times (x - m)$$

Recollect that $(x-m)$ is the adjusted matrix.

$$y = A(x-m) = \begin{pmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 0.5 & -0.5 \\ -0.5 & 0.5 \end{pmatrix}$$

$$= \begin{pmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} \left(\textit{for convenience } 0.5 = \frac{1}{2} \right)$$

$$= \begin{pmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 \end{pmatrix}$$

One can check the original matrix can be retrieved from this matrix as:

$$\{(A)^T \times y\} + m$$

$$x = A^T y + m = \begin{pmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 1.5 \\ 6.5 \end{pmatrix}$$

$$= \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} + \begin{pmatrix} 1.5 \\ 6.5 \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 6 & 7 \end{pmatrix}$$

Therefore, one can infer the original is obtained without any loss of information.

2.10.4 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is also a feature reduction technique like PCA. The focus of LDA is to project higher dimension data to a line (lower dimension data). LDA is also used to classify the data. Let there be two classes, c_1 and c_2. Let μ_1 and μ_2 be the mean of the patterns of two classes. The mean of the class c_1 and c_2 can be computed as:

$$\mu_1 = \frac{1}{N_1} \sum_{x_i \in c_1}^{n} x_i \text{ and } \mu_2 = \frac{1}{N_2} \sum_{x_i \in c_2}^{n} x_i$$

The aim of LDA is to optimize the function:

$$J(V) = \frac{V^T \sigma_B V}{V^T \sigma_W V} \tag{2.60}$$

where, V is the linear projection and σ_B and σ_W are class scatter matrix and within scatter matrix, respectively. For the two-class problem, these matrices are given as:

$$\sigma_B = N_1(\mu_1 - \mu)(\mu_1 - \mu)^T + N_2(\mu_2 - \mu)(\mu_2 - \mu)^T \tag{2.61}$$

$$\sigma_W = \sum_{x_i \in c_1} (x_i - \mu_1)(x_i - \mu_1)^T + \sum_{x_i \in c_2} (x_i - \mu_2)(x_i - \mu_2)^T \tag{2.62}$$

The maximization of $J(V)$ should satisfy the equation:

$$\sigma_B V = \lambda \sigma_W V \text{ or } \sigma_W^{-1} \sigma_B V = \lambda V \tag{2.63}$$

As $\sigma_B V$ is always in the direction of $(\mu_1 - \mu_2)$, V can be given as:

$$V = \sigma_W^{-1}(\mu_1 - \mu_2) \tag{2.64}$$

Let $V = \{v_1, v_2, \cdots, v_d\}$ be the generalized eigen vectors of σ_B and σ_W, where, d is the largest eigen values as in PCA. The transformation of x is then given as:

$$y = V_d^T x \tag{2.65}$$

Like in PCA, the largest eigen values can be retained to have projections.

2.10.5 Singular Value Decomposition

Singular Value Decomposition (SVD) is another useful decomposition technique. Let A be the matrix, then the matrix A can be decomposed as:

$$A = USV^T \tag{2.66}$$

Here, A is the given matrix of dimension $m \times n$, U is the orthogonal matrix whose dimension is $m \times n$, S is the diagonal matrix of dimension $n \times n$, and V is the orthogonal matrix. The procedure for finding decomposition matrix is given as follows:

1. For a given matrix, find AA^T
2. Find eigen values of AA^T
3. Sort the eigen values in a descending order. Pack the eigen vectors as a matrix U.
4. Arrange the square root of the eigen values in diagonal. This matrix is diagonal matrix, S.
5. Find eigen values and eigen vectors for A^TA. Find the eigen value and pack the eigen vector as a matrix called V.

Thus, $A = USV^T$. Here, U and V are orthogonal matrices. The columns of U and V are left and right singular values, respectively. SVD is useful in compression, as one can decide to retain only a certain component instead of the original matrix A as:

$$a_{ij} = \sum_{k=1}^{n} u_{ik} s_k v_{jk} \tag{2.67}$$

Based on the choice of retention, the compression can be controlled.

Example 2.13: Find SVD of the matrix:

$$A = \begin{pmatrix} 1 & 2 \\ 4 & 9 \end{pmatrix}$$

Solution: The first step is to compute:

$$AA^T = \begin{pmatrix} 1 & 2 \\ 4 & 9 \end{pmatrix} \begin{pmatrix} 1 & 4 \\ 2 & 9 \end{pmatrix} = \begin{pmatrix} 5 & 22 \\ 22 & 97 \end{pmatrix}$$

The eigen value and eigen vector of this matrix can be calculated to get U. The eigen values of this matrix are 0.0098 and 101.9902.

The eigen vectors of this matrix are:

$$u_1 = \begin{pmatrix} 0.2268 \\ 1 \end{pmatrix}$$
$$u_2 = \begin{pmatrix} -4.4086 \\ 1 \end{pmatrix}$$

These vectors are normalized to get the vectors respectively as:

$$u_1 = \begin{pmatrix} 0.2212 \\ 0.9752 \end{pmatrix}$$
$$u_2 = \begin{pmatrix} -0.9752 \\ 0.2212 \end{pmatrix}$$

The matrix U can be obtained by concatenating the above vector as:

$$U = [u_1, u_2] = \begin{pmatrix} 0.2212 & -0.9752 \\ 0.9752 & 0.2212 \end{pmatrix}$$

The matrix V can be obtained by finding A^TA. It is $\begin{pmatrix} 17 & 38 \\ 38 & 85 \end{pmatrix}$. The eigen values are 0.0098 and 101.9902. The eigen vectors can be found as follows:

$$v_1 = \begin{pmatrix} 0.447 \\ 1 \end{pmatrix} \text{ when } \lambda = 101.99$$

$$v_2 = \begin{pmatrix} -2.236 \\ 1 \end{pmatrix} \text{ when } \lambda = 0.0098$$

The above can be normalized as follows:

$$v_1 = \begin{pmatrix} 0.4082 \\ 0.9129 \end{pmatrix}$$
$$v_2 = \begin{pmatrix} -0.9129 \\ 0.4082 \end{pmatrix}$$

The matrix V can be obtained by concatenating the above vector as:

$$V = [v_1 \ \ v_2] = \begin{pmatrix} 0.4081 & -0.9129 \\ 0.9129 & 0.4082 \end{pmatrix}$$

The matrix S can be found as the diagonal matrix as:

$$S = \begin{pmatrix} \sqrt{101.9902} & 0 \\ 0 & \sqrt{0.0098} \end{pmatrix} = \begin{pmatrix} 10.099 & 0 \\ 0 & 0.099 \end{pmatrix}$$

Therefore, the matrix decomposition $A = U\,SV^T$ is complete.

The main advantage of SVD is compression. A matrix, say an image, can be decomposed and selectively only certain components can be retained by making all other elements zero. This reduces the contents of image while retaining the quality of the image. SVD is useful in data reduction too.

Chapter 3

Basics of Learning Theory

"Predicting the future isn't magic, it's artificial intelligence."
— Dave Waters

Learning is a process by which one can acquire knowledge and construct new ideas or concepts based on the experiences. Machine learning is an intelligent way of learning general concept from training examples without writing a program. There are many machine learning algorithms through which computers can intelligently learn from past data or experiences, identify patterns, and make predictions when new data is fed. This chapter deals with an introduction of concept learning of hypotheses space and modelling of machine learning.

Learning Objectives

- Understand the basics of learning theory and the key elements of machine learning
- Introduce Concept learning to obtain an abstraction and generalization from the data
- Learn about hypothesis representation language and to explore searching the hypothesis space using Find-S algorithm and its limitations
- Study about version spaces, List-Then-Eliminate algorithm and Candidate Elimination algorithm
- Introduce Inductive bias, a set of prior assumptions considered by a learning algorithm beyond the training data in order to perform induction
- Discuss the tradeoff between the two factors called bias and variance which exist when modelling a machine learning algorithm
- Understand the different challenges in model selection and model evaluation
- Study popular re-sampling model selection method called Cross-Validation (*K*-fold, LOOCV, etc.) to tune machine learning model
- Introduce the various learning frameworks and learn to evaluate hypothesis

3.1 INTRODUCTION TO LEARNING AND ITS TYPES

The process of acquiring knowledge and expertise through study, experience, or being taught is called as learning. Generally, humans learn in different ways. To make machines learn, we need to simulate the strategies of human learning in machines. But, will the computers learn? This question has been raised over many centuries by philosophers, mathematicians and logicians.

First let us address the question - What sort of tasks can the computers learn? This depends on the nature of problems that the computers can solve. There are two kinds of problems – well-posed and ill-posed. Computers can solve only well-posed problems, as these have well-defined specifications and have the following components inherent to it.

1. Class of learning tasks (T)
2. A measure of performance (P)
3. A source of experience (E)

The standard definition of learning proposed by Tom Mitchell is that a program can learn from E for the task T, and P improves with experience E. Let us formalize the concept of learning as follows:

Let x be the input and $\mathcal{X}$ be the input space, which is the set of all inputs, and Y is the output space, which is the set of all possible outputs, that is, yes/no.

Let D be the input dataset with examples, $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$ for n inputs.

Let the unknown target function be $f: \mathcal{X} \rightarrow Y$, that maps the input space to output space. The objective of the learning program is to pick a function, $g: \chi \rightarrow Y$ to approximate hypothesis f. All the possible formulae form a hypothesis space. In short, let H be the set of all formulae from which the learning algorithm chooses. The choice is good when the hypothesis g replicates f for all samples. This is shown in Figure 3.1.

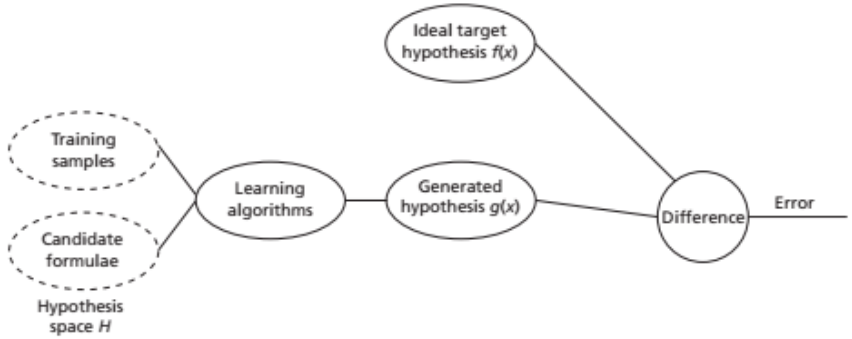


Figure 3.1: Learning Environment

It can be observed that training samples and target function are dependent on the given problem. The learning algorithm and hypothesis set are independent of the given problem. Thus, learning model is informally the hypothesis set and learning algorithm. Thus, learning model can be stated as follows:

Learning Model = Hypothesis Set + Learning Algorithm

Let us assume a problem of predicting a label for a given input data. Let D be the input dataset with both positive and negative examples. Let y be the output with class 0 or 1. The simple learning model can be given as:

$$\sum_{i=1}^{D} x_i w_i > Threshold, \text{ belongs to class 1 and}$$

$$\sum_{i=1}^{D} x_i w_i < Threshold, \text{ belongs to another class}$$

This can be put into a single equation as follows:

$$h(x) = sign\left(\left(\sum_{i=1}^{D} x_i w_i\right) + b\right)$$

where, $x_1, x_2, \cdots, x_d$ are the components of the input vector, $w_1, w_2, \ldots, w_d$ are the weights and +1 and –1 represent the class. This simple model is called perception model. One can simplify this by making $w_0 = b$ and fixing it as 1, then the model can further be simplified as:

$$h(x) = sign(w^T x).$$

This is called perception learning algorithm. The formal learning models are later discussed in Section 3.7 of this chapter.

Classical and Adaptive Machine Learning Systems

A classical machine learning system has components such as Input, Process and Output. The input values are taken from the environment directly. These values are processed and a hypothesis is generated as output model. This model is then used for making predictions. The predicted values are consumed by the environment.

In contrast to the classical systems, adaptive systems interact with the input for getting labelled data as direct inputs are not available. This process is called reinforcement learning. In reinforcement learning, a learning agent interacts with the environment and in return gets feedback. Based on the feedback, the learning agent generates input samples for learning, which are used for generating the learning model. Such learning agents are not static and change their behaviour according to the external signal received from the environment. The feedback is known as reward and learning here is the ability of the learning agent adapting to the environment based on the reward. These are the characteristics of an adaptive system.

Learning Types

There are different types of learning. Some of the different learning methods are as follows:

1. **Learn by memorization** or learn by repetition also called as ***rote learning*** is done by memorizing without understanding the logic or concept. Although rote learning is basically learning by repetition, in machine learning perspective, the learning occurs by simply comparing with the existing knowledge for the same input data and producing the output if present.
2. **Learn by examples** also called as **learn by experience** or previous knowledge acquired at some time, is like finding an ***analogy***, which means performing ***inductive learning*** from observations that formulate a general concept. Here, the learner learns by inferring a general rule from the set of observations or examples. Therefore, inductive learning is also called as ***discovery learning***.

3. **Learn by being taught** by an expert or a teacher, generally called as ***passive learning***. However, there is a special kind of learning called ***active learning*** where the learner can interactively query a teacher/expert to label unlabelled data instances with the desired outputs.
4. **Learning by critical thinking,** also called as ***deductive learning***, deduces new facts or conclusion from related known facts and information.
5. **Self learning,** also called as ***reinforcement learning***, is a ***self-directed learning*** that normally learns from mistakes punishments and rewards.
6. **Learning to solve problems** is a type of ***cognitive learning*** where learning happens in the mind and is possible by devising a methodology to achieve a goal. Here, the learner initially is not aware of the solution or the way to achieve the goal but only knows the goal. The learning happens either directly from the initial state by following the steps to achieve the goal or indirectly by inferring the behaviour.
7. **Learning by generalizing explanations**, also called as *explanation-based learning (EBL)*, is another learning method that exploits domain knowledge from experts to improve the accuracy of learned concepts by supervised learning.

Acquiring general concept from specific instances of the training dataset is the main challenge of machine learning.

Scan for the figure showing *'Types of Learning'*

3.2 INTRODUCTION TO COMPUTATION LEARNING THEORY

There are many questions that have been raised by mathematicians and logicians over the time taken by computers to learn. Some of the questions are as follows:

1. How can a learning system predict an unseen instance?
2. How do the hypothesis *h* is close to *f*, when hypothesis *f* itself is unknown?
3. How many samples are required?
4. Can we measure the performance of a learning system?
5. Is the solution obtained local or global?

These questions are the basis of a field called 'Computational Learning Theory' or in short (COLT). It is a specialized field of study of machine learning. COLT deals with formal methods used for learning systems. It deals with frameworks for quantifying learning tasks and learning algorithms. It provides a fundamental basis for study of machine learning. It deals with Probably Approximate Learning (PAC) and Vapnik-Chervonenkis (VC) dimensions. The focus of PAC is the quantification of the computational difficulty of learning tasks and algorithms and the computation capacity quantification is the focus of VC dimension.

Computational Learning Theory uses many concepts from diverse areas such as Theoretical Computer Science, Artificial Intelligence and Statistics. The core concept of COLT is the concept of

learning framework. One such important framework is PAC. The learning framework is discussed in a detailed manner in Section 3.7. COLT focuses on supervised learning tasks. Since the complexity of analyzing is difficult, normally, binary classification tasks are considered for analysis.

3.3 DESIGN OF A LEARNING SYSTEM

A system that is built around a learning algorithm is called a learning system. The design of systems focuses on these steps:

1. Choosing a training experience
2. Choosing a target function
3. Representation of a target function
4. Function approximation

Training Experience

Let us consider designing of a chess game. In direct experience, individual board states and correct moves of the chess game are given directly. In indirect system, the move sequences and results are only given. The training experience also depends on the presence of a supervisor who can label all valid moves for a board state. In the absence of a supervisor, the game agent plays against itself and learns the good moves, if the training samples cover all scenarios, or in other words, distributed enough for performance computation. If the training samples and testing samples have the same distribution, the results would be good.

Determine the Target Function

The next step is the determination of a target function. In this step, the type of knowledge that needs to be learnt is determined. In direct experience, a board move is selected and is determined whether it is a good move or not against all other moves. If it is the best move, then it is chosen as: $B \rightarrow M$, where, B and M are legal moves. In indirect experience, all legal moves are accepted and a score is generated for each. The move with largest score is then chosen and executed.

Determine the Target Function Representation

The representation of knowledge may be a table, collection of rules or a neural network. The linear combination of these factors can be coined as:

$$V = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3$$

where, x_1, x_2 and x_3 represent different board features and w_0, w_1, w_2 and w_3 represent weights.

Choosing an Approximation Algorithm for the Target Function

The focus is to choose weights and fit the given training samples effectively. The aim is to reduce the error given as:

$$E \equiv \sum_{\substack{\text{Training} \\ \text{Samples}}} \left[V_{train}(b) - \hat{V}(b) \right]^2$$

Here, b is the sample and $\hat{V}(b)$ is the predicted hypothesis. The approximation is carried out as:

- Computing the error as the difference between trained and expected hypothesis. Let error be $error(b)$.
- Then, for every board feature x_i, the weights are updated as:

$$w_i = w_i + \mu \times error(b) \times x_i$$

Here, μ is the constant that moderates the size of the weight update.

Thus, the learning system has the following components:

- A Performance system to allow the game to play against itself.
- A Critic system to generate the samples.
- A Generalizer system to generate a hypothesis based on samples.
- An Experimenter system to generate a new system based on the currently learnt function. This is sent as input to the performance system.

3.4 INTRODUCTION TO CONCEPT LEARNING

Concept learning is a learning strategy of acquiring abstract knowledge or inferring a general concept or deriving a category from the given training samples. It is a process of abstraction and generalization from the data.

Concept learning helps to classify an object that has a set of common, relevant features. Thus, it helps a learner compare and contrast categories based on the similarity and association of positive and negative instances in the training data to classify an object. The learner tries to simplify by observing the common features from the training samples and then apply this simplified model to the future samples. This task is also known as learning from experience.

Each concept or category obtained by learning is a Boolean valued function which takes a true or false value. For example, humans can identify different kinds of animals based on common relevant features and categorize all animals based on specific sets of features. The special features that distinguish one animal from another can be called as a concept. This way of learning categories for object and to recognize new instances of those categories is called as concept learning. It is formally defined as inferring a Boolean valued function by processing training instances.

Concept learning requires three things:

1. Input – Training dataset which is a set of training instances, each labeled with the name of a concept or category to which it belongs. Use this past experience to train and build the model.
2. Output – Target concept or Target function f. It is a mapping function $f(x)$ from input x to output y. It is to determine the specific features or common features to identify an object. In other words, it is to find the hypothesis to determine the target concept. For e.g., the specific set of features to identify an elephant from all animals.
3. Test – New instances to test the learned model.

Formally, Concept learning is defined as–"Given a set of hypotheses, the learner searches through the hypothesis space to identify the best hypothesis that matches the target concept".

Consider the following set of training instances shown in Table 3.1.

Table 3.1: Sample Training Instances

S.No.	Horns	Tail	Tusks	Paws	Fur	Color	Hooves	Size	Elephant
1.	No	Short	Yes	No	No	Black	No	Big	Yes
2.	Yes	Short	No	No	No	Brown	Yes	Medium	No
3.	No	Short	Yes	No	No	Black	No	Medium	Yes
4.	No	Long	No	Yes	Yes	White	No	Medium	No
5.	No	Short	Yes	Yes	Yes	Black	No	Big	Yes

Here, in this set of training instances, the independent attributes considered are 'Horns', 'Tail', 'Tusks', 'Paws', 'Fur', 'Color', 'Hooves' and 'Size'. The dependent attribute is 'Elephant'. The target concept is to identify the animal to be an Elephant.

Let us now take this example and understand further the concept of hypothesis.

Target Concept: Predict the type of animal - For example –'Elephant'.

3.4.1 Representation of a Hypothesis

A *hypothesis* '*h*' approximates a target function '*f*' to represent the relationship between the independent attributes and the dependent attribute of the training instances. The hypothesis is the predicted approximate model that best maps the inputs to outputs. Each hypothesis is represented as a conjunction of attribute conditions in the antecedent part.

For example, (Tail = Short) ∧ (Color = Black)….

The set of hypothesis in the search space is called as hypotheses. Hypotheses are the plural form of hypothesis. Generally '*H*' is used to represent the hypotheses and '*h*' is used to represent a candidate hypothesis.

Each attribute condition is the constraint on the attribute which is represented as attribute-value pair. In the antecedent of an attribute condition of a hypothesis, each attribute can take value as either '?' or 'φ' or can hold a single value.

- "?" denotes that the attribute can take any value [e.g., Color = ?]
- "φ" denotes that the attribute cannot take any value, i.e., it represents a null value [e.g., Horns = φ]
- Single value denotes a specific single value from acceptable values of the attribute, i.e., the attribute 'Tail' can take a value as 'short' [e.g., Tail = Short]

For example, a hypothesis '*h*' will look like,

	Horns	Tail	Tusks	Paws	Fur	Color	Hooves	Size
h =	<No	?	Yes	?	?	Black	No	Medium>

Given a test instance x, we say $h(x) = 1$, if the test instance x satisfies this hypothesis h.

The training dataset given above has 5 training instances with 8 independent attributes and one dependent attribute. Here, the different hypotheses that can be predicted for the target concept are,

	Horns	Tail	Tusks	Paws	Fur	Color	Hooves	Size
h =	<No	?	Yes	?	?	Black	No	Medium>

(or)

	Horns	Tail	Tusks	Paws	Fur	Color	Hooves	Size
h =	<No	?	Yes	?	?	Black	No	Big>

The task is to predict the best hypothesis for the target concept (an elephant). The most general hypothesis can allow any value for each of the attribute.

It is represented as:

<?, ?, ?, ?, ?, ?, ?, ?>. This hypothesis indicates that any animal can be an elephant.

The most specific hypothesis will not allow any value for each of the attribute <φ, φ, φ, φ, φ, φ, φ, φ>. This hypothesis indicates that no animal can be an elephant.

The target concept mentioned in this example is to identify the conjunction of specific features from the training instances to correctly identify an elephant.

Example 3.1: Explain Concept Learning Task of an Elephant from the dataset given in Table 3.1.

Given,

Input: 5 instances each with 8 attributes

Target concept/function 'c': Elephant → {Yes, No}

Hypotheses H: Set of hypothesis each with conjunctions of literals as propositions [i.e., each literal is represented as an attribute-value pair]

Solution: The hypothesis 'h' for the concept learning task of an Elephant is given as:

h =	<No	Short	Yes	?	?	Black	No	?>

This hypothesis h is expressed in propositional logic form as below:

(Horns = No) ∧ (Tail = Short) ∧ (Tusks = Yes) ∧ (Paws = ?) ∧ (Fur = ?) ∧ (Color = Black) ∧ (Hooves = No) ∧ (Size = ?)

Output: Learn the hypothesis 'h' to predict an 'Elephant' such that for a given test instance x,

$$h(x) = c(x)$$

This hypothesis produced is also called as concept description which is a model that can be used to classify subsequent instances.

Thus, concept learning can also be called as *Inductive Learning* that tries to induce a general function from specific training instances. This way of learning a hypothesis that can produce an approximate target function with a sufficiently large set of training instances can also approximately classify other unobserved instances and is called as inductive learning hypothesis. We can only determine an approximate target function because it is very difficult to find an exact target function with the observed training instances. That is why a hypothesis is an approximate target function that best maps the inputs to outputs.

3.4.2 Hypothesis Space

Hypothesis space is the set of all possible hypotheses that approximates the target function f. In other words, the set of all possible approximations of the target function can be defined as hypothesis space. From this set of hypotheses in the hypothesis space, a machine learning algorithm would determine the best possible hypothesis that would best describe the target function or best fit the outputs. Generally, a hypothesis representation language represents a larger hypothesis space. Every machine learning algorithm would represent the hypothesis space in a different manner about the function that maps the input variables to output variables. For example, a regression algorithm represents the hypothesis space as a linear function whereas a decision tree algorithm represents the hypothesis space as a tree.

The set of hypotheses that can be generated by a learning algorithm can be further reduced by specifying a language bias.

The subset of hypothesis space that is consistent with all-observed training instances is called as **Version Space.** Version space represents the only hypotheses that are used for the classification.

For example, each of the attribute given in the Table 3.1 has the following possible set of values.

Horns - Yes, No

Tail - Long, Short

Tusks - Yes, No

Paws - Yes, No

Fur - Yes, No

Color - Brown, Black, White

Hooves - Yes, No

Size - Medium, Big

Considering these values for each of the attribute, there are $(2 \times 2 \times 2 \times 2 \times 2 \times 3 \times 2 \times 2) = 384$ distinct instances covering all the 5 instances in the training dataset.

So, we can generate $(4 \times 4 \times 4 \times 4 \times 4 \times 5 \times 4 \times 4) = 81{,}920$ distinct hypotheses when including two more values [?, φ] for each of the attribute. However, any hypothesis containing one or more φ symbols represents the empty set of instances; that is, it classifies every instance as negative instance. Therefore, there will be $(3 \times 3 \times 3 \times 3 \times 3 \times 4 \times 3 \times 3 + 1) = 8{,}749$ distinct hypotheses by including only '?' for each of the attribute and one hypothesis representing the empty set of instances. Thus, the hypothesis space is much larger and hence we need efficient learning algorithms to search for the best hypothesis from the set of hypotheses.

Hypothesis ordering is also important wherein the hypotheses are ordered from the most specific one to the most general one in order to restrict searching the hypothesis space exhaustively.

3.4.3 Heuristic Space Search

Heuristic search is a search strategy that finds an optimized hypothesis/solution to a problem by iteratively improving the hypothesis/solution based on a given heuristic function or a cost measure. Heuristic search methods will generate a possible hypothesis that can be a solution in

the hypothesis space or a path from the initial state. This hypothesis will be tested with the target function or the goal state to see if it is a real solution. If the tested hypothesis is a real solution, then it will be selected. This method generally increases the efficiency because it is guaranteed to find a better hypothesis but may not be the best hypothesis. It is useful for solving tough problems which could not solved by any other method. The typical example problem solved by heuristic search is the travelling salesman problem.

Several commonly used heuristic search methods are hill climbing methods, constraint satisfaction problems, best-first search, simulated-annealing, A* algorithm, and genetic algorithms.

3.4.4 Generalization and Specialization

In order to understand about how we construct this concept hierarchy, let us apply this general principle of generalization/specialization relation. By generalization of the most specific hypothesis and by specialization of the most general hypothesis, the hypothesis space can be searched for an approximate hypothesis that matches all positive instances but does not match any negative instance.

Searching the Hypothesis Space

There are two ways of learning the hypothesis, consistent with all training instances from the large hypothesis space.

1. Specialization – General to Specific learning
2. Generalization – Specific to General learning

Scan for information on *'Additional Examples on Generalization and Specialization'*

Generalization – Specific to General Learning This learning methodology will search through the hypothesis space for an approximate hypothesis by generalizing the most specific hypothesis.

Example 3.2: Consider the training instances shown in Table 3.1 and illustrate Specific to General Learning.

Solution: We will start from all false or the most specific hypothesis to determine the most restrictive specialization. Consider only the positive instances and generalize the most specific hypothesis. Ignore the negative instances.

This learning is illustrated as follows:

The most specific hypothesis is taken now, which will not classify any instance to true.

h = <φ φ φ φ φ φ φ φ>

Read the first instance $I1$, to generalize the hypothesis h so that this positive instance can be classified by the hypothesis $h1$.

$I1$:	No	Short	Yes	No	No	Black	No	Big	**Yes (Positive instance)**
$h1$ =	<No	Short	Yes	No	No	Black	No	Big>	

When reading the second instance $I2$, it is a negative instance, so ignore it.

$I2$:	Yes	Short	No	No	No	Brown	Yes	Medium	**No (Negative instance)**
$h2$ =	<No	Short	Yes	No	No	Black	No	Big>	

Similarly, when reading the third instance $I3$, it is a positive instance so generalize $h2$ to $h3$ to accommodate it. The resulting $h3$ is generalized.

$I3$:	No	Short	Yes	No	No	Black	No	Medium	**Yes (Positive instance)**
$h3$ =	<No	Short	Yes	No	No	Black	No	?>	

Ignore $I4$ since it is a negative instance.

$I4$:	No	Long	No	Yes	Yes	White	No	Medium	**No (Negative instance)**
$h4$ =	<No	Short	Yes	No	No	Black	No	?>	

When reading the fifth instance $I5$, $h4$ is further generalized to $h5$.

$I5$:	No	Short	Yes	Yes	Yes	Black	No	Big	**Yes (Positive instance)**
$h5$ =	<No	Short	Yes	?	?	Black	No	?>	

Now, after observing all the positive instances, an approximate hypothesis $h5$ is generated which can now classify any subsequent positive instance to true.

Specialization – General to Specific Learning This learning methodology will search through the hypothesis space for an approximate hypothesis by specializing the most general hypothesis.

Example 3.3: Illustrate learning by Specialization – General to Specific Learning for the data instances shown in Table 3.1.

Solution: Start from the most general hypothesis which will make true all positive and negative instances.

Initially,

h = <?	?	?	?	?	?	?	?	?>

h is more general to classify all instances to true.

$I1$:	No	Short	Yes	No	No	Black	No	Big	**Yes (Positive instance)**
$h1$ =	<?	?	?	?	?	?	?	?>	

$I2$:	Yes	Short	No	No	No	Brown	Yes	Medium	**No (Negative instance)**
$h2$ =	<No	?	?	?	?	?	?	?>	
	<?	?	Yes	?	?	?	?	?>	
	<?	?	?	?	?	Black	?	?>	
	<?	?	?	?	?	?	No	?>	
	<?	?	?	?	?	?	?	Big>	

$h2$ imposes constraints so that it will not classify a negative instance to true.

$I3$:	No	Short	Yes	No	No	Black	No	Medium	**Yes (Positive instance)**
$h3$ =	<No	?	?	?	?	?	?	?>	
	<?	?	Yes	?	?	?	?	?>	

<? ? ? ? ? Black ? ?>
<? ? ? ? ? ? No ?>
<? ? ? ? ? ? ? Big>

*I*4: No Long No Yes Yes White No Medium **No (Negative instance)**

*h*4 = <? ? Yes ? ? ? ? ?>
<? ? ? ? ? Black ? ?>
<? ? ? ? ? ? ? Big>

Remove any hypothesis inconsistent with this negative instance.

*I*5: No Short Yes Yes Yes Black No Big **Yes (Positive instance)**

*h*5 = <? ? Yes ? ? ? ? ?>
<? ? ? ? ? Black ? ?>
<? ? ? ? ? ? ? Big>

Thus, *h*5 is the hypothesis space generated which will classify the positive instances to true and negative instances to false.

3.4.5 Hypothesis Space Search by Find-S Algorithm

Find-S algorithm is guaranteed to converge to the most specific hypothesis in ***H*** that is consistent with the positive instances in the training dataset. Obviously, it will also be consistent with the negative instances. Thus, this algorithm considers only the positive instances and eliminates negative instances while generating the hypothesis. It initially starts with the most specific hypothesis.

Algorithm 3.1: Find-S

Input: Positive instances in the Training dataset

Output: Hypothesis *'h'*

1. Initialize '*h*' to the most specific hypothesis.
   h = <φ φ φ φ φ ….>
2. Generalize the initial hypothesis for the first positive instance [Since '*h*' is more specific].
3. For each subsequent instances:
   If it is a positive instance,
      Check for each attribute value in the instance with the hypothesis '*h*'.
         If the attribute value is the same as the hypothesis value, then do nothing,
         Else if the attribute value is different than the hypothesis value, change it to '?' in '*h*'.
   Else if it is a negative instance,
      Ignore it.

Example 3.4: Consider the training dataset of 4 instances shown in Table 3.2. It contains the details of the performance of students and their likelihood of getting a job offer or not in their final semester. Apply the Find-S algorithm.

Table 3.2: Training Dataset

CGPA	Interactiveness	Practical Knowledge	Communication Skills	Logical Thinking	Interest	Job Offer
≥9	Yes	Excellent	Good	Fast	Yes	Yes
≥9	Yes	Good	Good	Fast	Yes	Yes
≥8	No	Good	Good	Fast	No	No
≥9	Yes	Good	Good	Slow	No	Yes

Solution:

Step 1: Initialize '*h*' to the most specific hypothesis. There are 6 attributes, so for each attribute, we initially fill '*φ*' in the initial hypothesis '*h*'.

h = <φ φ φ φ φ φ>

Step 2: Generalize the initial hypothesis for the first positive instance. *I*1 is a positive instance, so generalize the most specific hypothesis '*h*' to include this positive instance. Hence,

*I*1: ≥9 Yes Excellent Good Fast Yes **Positive instance**

h = <≥9 Yes Excellent Good Fast Yes>

Step 3: Scan the next instance *I*2, since *I*2 is a positive instance. Generalize '*h*' to include positive instance *I*2. For each of the non-matching attribute value in '*h*' put a '?' to include this positive instance. The third attribute value is mismatching in '*h*' with *I*2, so put a '?'.

*I*2: ≥9 Yes Good Good Fast Yes **Positive instance**

h = <≥9 Yes ? Good Fast Yes>

Now,scan *I*3. Since it is a negative instance, ignore it. Hence, the hypothesis remains the same without any change after scanning *I*3.

I3: ≥8 No Good Good Fast No **Negative instance**

h = <≥9 Yes ? Good Fast Yes>

Now scan *I*4. Since it is a positive instance, check for mismatch in the hypothesis '*h*' with *I*4. The 5th and 6th attribute value are mismatching, so add '?' to those attributes in '*h*'.

I4: ≥9 Yes Good Good Slow No **Positive instance**

h = <≥9 Yes ? Good ? ?>

Now, the final hypothesis generated with Find-S algorithm is:

h = <≥9 Yes ? Good ? ?>

It includes all positive instances and obviously ignores any negative instance.

Limitations of Find-S Algorithm

1. Find-S algorithm tries to find a hypothesis that is consistent with positive instances, ignoring all negative instances. As long as the training dataset is consistent, the hypothesis found by this algorithm may be consistent.
2. The algorithm finds only one unique hypothesis, wherein there may be many other hypotheses that are consistent with the training dataset.

3. Many times, the training dataset may contain some errors; hence such inconsistent data instances can mislead this algorithm in determining the consistent hypothesis since it ignores negative instances.

Hence, it is necessary to find the set of hypotheses that are consistent with the training data including the negative examples. To overcome the limitations of Find-S algorithm, Candidate Elimination algorithm was proposed to output the set of all hypotheses consistent with the training dataset.

3.4.6 Version Spaces

The version space contains the subset of hypotheses from the hypothesis space that is consistent with all training instances in the training dataset.

Scan for information on *'Additional Examples on Version Spaces'*

List-Then-Eliminate Algorithm

The principle idea of this learning algorithm is to initialize the version space to contain all hypotheses and then eliminate any hypothesis that is found inconsistent with any training instances. Initially, the algorithm starts with a version space to contain all hypotheses scanning each training instance. The hypotheses that are inconsistent with the training instance are eliminated. Finally, the algorithm outputs the list of remaining hypotheses that are all consistent.

Algorithm 3.2: List-Then-Eliminate

Input: Version Space – a list of all hypotheses

Output: Set of consistent hypotheses

1. Initialize the version space with a list of hypotheses.
2. For each training instance,
   - remove from version space any hypothesis that is inconsistent.

This algorithm works fine if the hypothesis space is finite but practically it is difficult to deploy this algorithm. Hence, a variation of this idea is introduced in the Candidate Elimination algorithm.

Version Spaces and the Candidate Elimination Algorithm

Version space learning is to generate all consistent hypotheses around. This algorithm computes the version space by the combination of the two cases namely,

- Specific to General learning – Generalize *S* to include the positive example
- General to Specific learning – Specialize *G* to exclude the negative example

Using the Candidate Elimination algorithm, we can compute the version space containing all (and only those) hypotheses from *H* that are consistent with the given observed sequence of training instances. The algorithm defines two boundaries called '*general boundary*' which is a set of all hypotheses that are the most general and '*specific boundary*' which is a set of all hypotheses that are the most specific. Thus, the algorithm limits the version space to contain only those hypotheses that are most general and most specific. Thus, it provides a compact representation of List-then algorithm.

Algorithm 3.3: Candidate Elimination

Input: Set of instances in the Training dataset

Output: Hypothesis *G* and *S*

1. Initialize *G*, to the maximally general hypotheses.
2. Initialize *S*, to the maximally specific hypotheses.
   - Generalize the initial hypothesis for the first positive instance.
3. For each subsequent new training instance,
   - If the instance is **positive**,
     - Generalize *S* to include the positive instance,
       - Check the attribute value of the positive instance and *S*,
         - If the attribute value of positive instance and *S* are different, fill that field value with '?'.
         - If the attribute value of positive instance and *S* are same, then do no change.
     - Prune *G* to exclude all inconsistent hypotheses in *G* with the positive instance.
   - If the instance is **negative,**
     - Specialize *G* to exclude the negative instance,
       - Add to *G* all minimal specializations to exclude the negative example and be consistent with *S*.
         - If the attribute value of *S* and the negative instance are different, then fill that attribute value with *S* value.
         - If the attribute value of *S* and negative instance are same, no need to update '*G*' and fill that attribute value with '?'.
     - Remove from *S* all inconsistent hypotheses with the negative instance.

Generating Positive Hypothesis 'S' If it is a positive example, refine *S* to include the positive instance. We need to generalize *S* to include the positive instance. The hypothesis is the conjunction of '*S*' and positive instance. When generalizing, for the first positive instance, add to *S* all minimal generalizations such that *S* is filled with attribute values of the positive instance. For the subsequent positive instances scanned, check the attribute value of the positive instance and *S* obtained in the

previous iteration. If the attribute values of positive instance and *S* are different, fill that field value with a '?'. If the attribute values of positive instance and *S* are same, no change is required.

If it is a negative instance, it skips.

Generating Negative Hypothesis 'G' If it is a negative instance, refine *G* to exclude the negative instance. Then, prune *G* to exclude all inconsistent hypotheses in *G* with the positive instance. The idea is to add to *G* all minimal specializations to exclude the negative instance and be consistent with the positive instance. Negative hypothesis indicates general hypothesis.

If the attribute values of positive and negative instances are different, then fill that field with positive instance value so that the hypothesis does not classify that negative instance as true. If the attribute values of positive and negative instances are same, then no need to update '*G*' and fill that attribute value with a '?'.

Generating Version Space – [Consistent Hypothesis] We need to take the combination of sets in '*G*' and check that with '*S*'. When the combined set fields are matched with fields in '*S*', then only that is included in the version space as consistent hypothesis.

Example 3.4: Consider the same set of instances from the training dataset shown in Table 3.3 and generate version space as consistent hypothesis.

Solution:

Step 1: Initialize '*G*' boundary to the maximally general hypotheses,

G = <? ? ? ? ? ?>

Step 2: Initialize '*S*' boundary to the maximally specific hypothesis. There are 6 attributes, so for each attribute, we initially fill 'φ' in the hypothesis '*S*'.

S = <φ φ φ φ φ φ>

Generalize the initial hypothesis for the first positive instance. *I*1 is a positive instance; so generalize the most specific hypothesis '*S*' to include this positive instance. Hence,

*I*1:	≥9	Yes	Excellent	Good	Fast	Yes	**Positive instance**
*S*1 = <	≥9	Yes	Excellent	Good	Fast	Yes>	
*G*1 = <?	?	?	?	?	?>		

Step 3:

Iteration 1

Scan the next instance *I*2. Since *I*2 is a positive instance, generalize '*S*1' to include positive instance *I*2. For each of the non-matching attribute value in '*S*1', put a '?' to include this positive instance. The third attribute value is mismatching in '*S*1' with *I*2, so put a '?'.

*I*2:	≥9	Yes	Good	Good	Fast	Yes	**Positive instance**
*S*2 = <	≥9	Yes	?	Good	Fast	Yes>	

Prune *G*1 to exclude all inconsistent hypotheses with the positive instance. Since *G*1 is consistent with this positive instance, there is no change. The resulting *G*2 is,

*G*2 = <? ? ? ? ? ?>

Iteration 2

Now Scan *I*3,

*I*3:	≥8	No	Good	Good	Fast	No	**Negative instance**

Since it is a negative instance, specialize *G*2 to exclude the negative example but stay consistent with *S*2. Generate hypothesis for each of the non-matching attribute value in *S*2 and fill with the attribute value of *S*2. In those generated hypotheses, for all matching attribute values, put a '?'. The first, second and 6th attribute values do not match, hence '3' hypotheses are generated in *G*3.

There is no inconsistent hypothesis in *S*2 with the negative instance, hence *S*3 remains the same.

*G*3 = <	≥9	?	?	?	?	?>
	<?	Yes	?	?	?	?>
	<?	?	?	?	?	Yes>
*S*3 = <	≥9	Yes	?	Good	Fast	Yes>

Iteration 3

Now Scan *I*4. Since it is a positive instance, check for mismatch in the hypothesis '*S*3' with *I*4. The 5th and 6th attribute value are mismatching, so add '?' to those attributes in '*S*4'.

*I*4:	≥9	Yes	Good	Good	Slow	No	**Positive instance**
*S*4 = <	≥9	Yes	?	Good	?	?>	

Prune *G*3 to exclude all inconsistent hypotheses with the positive instance *I*4.

*G*3 = <	≥9	?	?	?	?	?>	
	<?	Yes	?	?	?	?>	
	<?	?	?	?	?	Yes>	**Inconsistent**

Since the third hypothesis in *G*3 is inconsistent with this positive instance, remove the third one. The resulting *G*4 is,

*G*4 = <	≥9	?	?	?	?	?>
	<?	Yes	?	?	?	?>

Using the two boundary sets, *S*4 and *G*4, the version space is converged to contain the set of consistent hypotheses.

The final version space is,

<	≥9	Yes	?	?	?	?>
<	≥9	?	?	Good	?	?>
<?	Yes	?	Good	?	?>	

Thus, the algorithm finds the version space to contain only those hypotheses that are most general and most specific.

The diagrammatic representation of deriving the version space is shown in Figure 3.2.

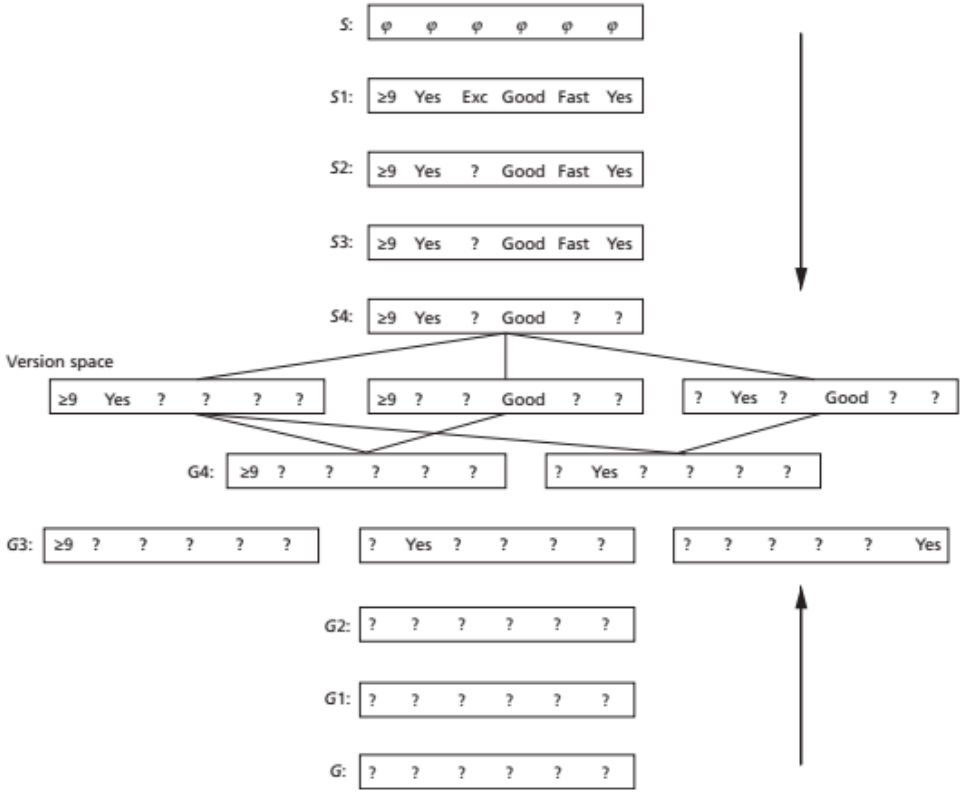


Figure 3.2: Deriving the Version Space

Chapter 4

Similarity-based Learning

"Anyone who stops learning is old, whether at twenty or eighty."
— Henry Ford

Similarity-based Learning is a supervised learning technique that predicts the class label of a test instance by gauging the similarity of this test instance with training instances. Similarity-based learning refers to a family of instance-based learning which is used to solve both classification and regression problems. Instance-based learning makes prediction by computing distances or similarities between test instance and specific set of training instances local to the test instance in an incremental process. In contrast to other learning mechanisms, it considers only the nearest instance or instances to predict the class of unseen instances. This learning methodology improves the performance of classification since it uses only a specific set of instances as incremental learning task. Similarity-based classification is useful in various fields such as image processing, text classification, pattern recognition, bio informatics, data mining, information retrieval, natural language processing, etc. A practical application of this learning is predicting daily stock index price changes. This chapter provides an insight of how different similarity-based models predict the class of a new instance.

Learning Objectives

- Understand the fundamentals of Instance based learning
- Know about the concepts of Nearest-Neighbor Learning using the algorithm called *k*-Nearest-Neighbors (*k*-NN)
- Learn about Weighted *k*-Nearest-Neighbor classifier that chooses the neighbors by using the weighted distance
- Gain knowledge about Nearest Centroid classifier, a simple alternative to *k*-NN classifiers
- Understand Locally Weighted Regression (LWR) that approximates the linear functions of all *k* neighbors to minimize the error while prediction

4.1 INTRODUCTION TO SIMILARITY OR INSTANCE-BASED LEARNING

Similarity-based classifiers use similarity measures to locate the nearest neighbors and classify a test instance which works in contrast with other learning mechanisms such as decision trees or neural networks. Similarity-based learning is also called as Instance-based learning/Just-in time learning since it does not build an abstract model of the training instances and performs lazy learning when classifying a new instance. This learning mechanism simply stores all data and uses it only when it needs to classify an unseen instance. The advantage of using this learning is that processing occurs only when a request to classify a new instance is given. This methodology is particularly useful when the whole dataset is not available in the beginning but collected in an incremental manner.

The drawback of this learning is that it requires a large memory to store the data since a global abstract model is not constructed initially with the training data. Classification of instances is done based on the measure of similarity in the form of distance functions over data instances. Several distance metrics are used to estimate the similarity or dissimilarity between instances required for clustering, nearest neighbor classification, anomaly detection, and so on. Popular distance metrics used are Hamming distance, Euclidean distance, Manhattan distance, Minkowski distance, Cosine similarity, Mahalanobis distance, Pearson's correlation or correlation similarity, Mean squared difference, Jaccard coefficient, Tanimoto coefficient, etc.

Generally, Similarity-based classification problems formulate the features of test instance and training instances in Euclidean space to learn the similarity or dissimilarity between instances.

4.1.1 Differences Between Instance- and Model-based Learning

An *instance* is an entity or an example in the training dataset. It is described by a set of features or attributes. One attribute describes the class label or category of an instance. *Instance-based methods* learn or predict the class label of a test instance only when a new instance is given for classification and until then it delays the processing of the training dataset.

It is also referred to as *lazy learning* methods since it does not generalize any model from the training dataset but just keeps the training dataset as a knowledge base until a new instance is given. In contrast, *model-based learning*, generally referred to as *eager learning*, tries to generalize the training data to a model before receiving test instances. Model-based machine learning describes all assumptions about the problem domain in the form of a model. These algorithms basically learn in two phases, called training phase and testing phase. In training phase, a model is built from the training dataset and is used to classify a test instance during the testing phase. Some examples of models constructed are decision trees, neural networks and Support Vector Machines (SVM), etc.

The differences between Instance-based Learning and Model-based Learning are listed in Table 4.1.

Table 4.1: Differences between Instance-based Learning and Model-based Learning

Instance-based Learning	Model-based Learning
Lazy Learners	Eager Learners
Processing of training instances is done only during testing phase	Processing of training instances is done during training phase

(*Continued*)

Instance-based Learning	Model-based Learning
No model is built with the training instances before it receives a test instance	Generalizes a model with the training instances before it receives a test instance
Predicts the class of the test instance directly from the training data	Predicts the class of the test instance from the model built
Slow in testing phase	Fast in testing phase
Learns by making many local approximations	Learns by creating global approximation

Instance-based learning also comes under the category of memory-based models which normally compare the given test instance with the trained instances that are stored in memory. Memory-based models classify a test instance by checking the similarity with the training instances.

Some examples of Instance-based learning algorithms are:

1. *k*-Nearest Neighbor (*k*-NN)
2. Variants of Nearest Neighbor learning
3. Locally Weighted Regression
4. Learning Vector Quantization (LVQ)
5. Self-Organizing Map (SOM)
6. Radial Basis Function (RBF) networks

In this chapter, we will discuss about certain instance-based learning algorithms such as *k*-Nearest Neighbor (*k*-NN), Variants of Nearest Neighbor learning, and Locally Weighted Regression learning.

Self-Organizing Map (SOM) and Radial Basis Function (RBF) networks are discussed along with the concepts of artificial neural networks discussed in Chapter 10 since they could be referred only after the understanding of neural networks.

These instance-based methods have serious limitations about the range of feature values taken. Moreover, they are sensitive to irrelevant and correlated features leading to misclassification of instances.

4.2 NEAREST-NEIGHBOR LEARNING

A natural approach to similarity-based classification is *k*-Nearest-Neighbors (*k*-NN), which is a non-parametric method used for both classification and regression problems. It is a simple and powerful non-parametric algorithm that predicts the category of the test instance according to the '*k*' training samples which are closer to the test instance and classifies it to that category which has the largest probability. A visual representation of this learning is shown in Figure 4.1. There are two classes of objects called C_1 and C_2 in the given figure. When given a test instance T, the category of this test instance is determined by looking at the class of $k = 3$ nearest neighbors. Thus, the class of this test instance T is predicted as C_2.

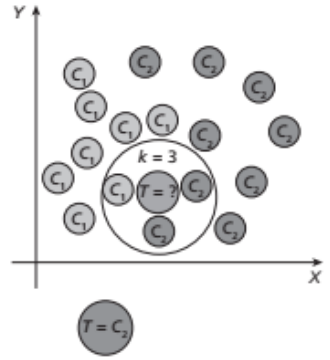


Figure 4.1: Visual Representation of *k*-Nearest Neighbor Learning

The algorithm relies on the assumption that similar objects are close to each other in the feature space. *k*-NN performs instance-based learning which just stores the training data instances and learning instances case by case. The model is also 'memory-based' as it uses training data at time when predictions need to be made. It is a lazy learning algorithm since no prediction model is built earlier with training instances and classification happens only after getting the test instance.

The algorithm classifies a new instance by determining the '*k*' most similar instances (i.e., *k* nearest neighbors) and summarizing the output of those '*k*' instances. If the target variable is discrete then it is a classification problem, so it selects the most common class value among the '*k*' instances by a majority vote. However, if the target variable is continuous then it is a regression problem, and hence the mean output variable of the '*k*' instances is the output of the test instance.

The most popular distance measure such as Euclidean distance is used in *k*-NN to determine the '*k*' instances which are similar to the test instance. The value of '*k*' is best determined by tuning with different '*k*' values and choosing the '*k*' which classifies the test instance more accurately.

Algorithm 4.1: *k*-NN

Inputs: Training dataset *T*, distance metric *d*, Test instance *t*, the number of nearest neighbors *k*

Output: Predicted class or category

Prediction: For test instance *t*,

1. For each instance *i* in *T*, compute the distance between the test instance *t* and every other instance *i* in the training dataset using a distance metric (Euclidean distance).
   [Continuous attributes - Euclidean distance between two points in the plane with coordinates (x_1, y_1) and (x_2, y_2) is given as dist $((x_1, y_1), (x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$]
   [Categorical attributes (Binary) - Hamming Distance: If the value of the two instances is same, the distance *d* will be equal to 0 otherwise *d* = 1.]
2. Sort the distances in an ascending order and select the first *k* nearest training data instances to the test instance.
3. Predict the class of the test instance by majority voting (if target attribute is discrete valued) or mean (if target attribute is continuous valued) of the *k* selected nearest instances.

Example 4.1: Consider the student performance training dataset of 8 data instances shown in Table 4.2 which describes the performance of individual students in a course and their CGPA obtained in the previous semesters. The independent attributes are CGPA, Assessment and Project. The target variable is 'Result' which is a discrete valued variable that takes two values 'Pass' or 'Fail'. Based on the performance of a student, classify whether a student will pass or fail in that course.

Table 4.2: Training Dataset *T*

S.No.	CGPA	Assessment	Project Submitted	Result
1.	9.2	85	8	Pass
2.	8	80	7	Pass
3.	8.5	81	8	Pass

(*Continued*)

S.No.	CGPA	Assessment	Project Submitted	Result
4.	6	45	5	Fail
5.	6.5	50	4	Fail
6.	8.2	72	7	Pass
7.	5.8	38	5	Fail
8.	8.9	91	9	Pass

Solution: Given a test instance (6.1, 40, 5) and a set of categories {Pass, Fail} also called as classes, we need to use the training set to classify the test instance using Euclidean distance.

The task of classification is to assign a category or class to an arbitrary instance.

Assign $k = 3$.

Step 1: Calculate the Euclidean distance between the test instance (6.1, 40, and 5) and each of the training instances as shown in Table 4.3.

Table 4.3: Euclidean Distance

S.No.	CGPA	Assessment	Project Submitted	Result	Euclidean Distance
1.	9.2	85	8	Pass	$\sqrt{(9.2-6.1)^2+(85-40)^2+(8-5)^2}$ = 45.2063
2.	8	80	7	Pass	$\sqrt{(8-6.1)^2+(80-40)^2+(7-5)^2}$ = 40.09501
3.	8.5	81	8	Pass	$\sqrt{(8.5-6.1)^2+(81-40)^2+(8-5)^2}$ = 41.17961
4.	6	45	5	Fail	$\sqrt{(6-6.1)^2+(45-40)^2+(5-5)^2}$ = 5.001
5.	6.5	50	4	Fail	$\sqrt{(6.5-6.1)^2+(50-40)^2+(4-5)^2}$ = 10.05783
6.	8.2	72	7	Pass	$\sqrt{(8.2-6.1)^2+(72-40)^2+(7-5)^2}$ = 32.13114
7.	5.8	38	5	Fail	$\sqrt{(5.8-6.1)^2+(38-40)^2+(5-5)^2}$ = 2.022375
8.	8.9	91	9	Pass	$\sqrt{(8.9-6.1)^2+(91-40)^2+(9-5)^2}$ = 51.23319

Step 2: Sort the distances in the ascending order and select the first 3 nearest training data instances to the test instance. The selected nearest neighbors are shown in Table 4.4.

Table 4.4: Nearest Neighbors

Instance	Euclidean Distance	Class
4	5.001	Fail
5	10.05783	Fail
7	2.022375	Fail

Here, we take the 3 nearest neighbors as instances 4, 5 and 7 with smallest distances.

Step 3: Predict the class of the test instance by majority voting.

The class for the test instance is predicted as 'Fail'.

Data normalization/standardization is required when data (features) have different ranges or a wider range of possible values when computing distances and to transform all features to a specific range. This is probably done to eliminate the influence of one feature over another (i.e., to give all features equal chances). For example if one feature has values in the range of [0–1] and another feature has values in the range of [0–100], then the second feature will influence more even if there is a small variation than the first feature.

k-NN classifier performance is strictly affected by three factors such as the number of nearest neighbors (i.e., selection of *k*), distance metric and decision rule.

If the *k* value selected is small then it may result in overfitting or less stable and if it is big then it may include many irrelevant points from other classes. The choice of the distance metric selected also plays a major role and it depends on the type of the independent attributes in the training dataset.

The *k*-NN classification algorithm best suits lower dimensional data asin a high-dimensional space the nearest neighbors may not be very close at all.

4.3 WEIGHTED *K*-NEAREST-NEIGHBOR ALGORITHM

The Weighted *k*-NN is an extension of *k*-NN. It chooses the neighbors by using the weighted distance. The *k*-Nearest Neighbor (*k*-NN) algorithm has some serious limitations as its performance is solely dependent on choosing the *k* nearest neighbors, the distance metric used and the decision rule. However, the principle idea of Weighted *k*-NN is that *k* closest neighbors to the test instance are assigned a higher weight in the decision as compared to neighbors that are farther away from the test instance. The idea is that weights are inversely proportional to distances.

The selected *k* nearest neighbors can be assigned uniform weights, which means all the instances in each neighborhood are weighted equally or weights can be assigned by the inverse of their distance. In the second case, closer neighbors of a query point will have a greater influence than neighbors which are further away.

Algorithm 4.2: Weighted *k*-NN

Inputs: Training dataset '*T*', Distance metric '*d*', Weighting function *w*(*i*), Test instance '*t*', the number of nearest neighbors '*k*'

(*Continued*)

Output: Predicted class or category

Prediction: For test instance *t*,

1. For each instance '*i*' in Training dataset *T*, compute the distance between the test instance *t* and every other instance '*i*' using a distance metric (Euclidean distance).
   [Continuous attributes - Euclidean distance between two points in the plane with coordinates (x_1, y_1) and (x_2, y_2) is given as dist $((x_1, y_1), (x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$]
   [Categorical attributes (Binary) - Hamming Distance: If the values of two instances are the same, the distance *d* will be equal to 0. Otherwise *d* = 1.]
2. Sort the distances in the ascending order and select the first '*k*' nearest training data instances to the test instance.
3. Predict the class of the test instance by weighted voting technique (Weighting function *w*(*i*)) for the *k* selected nearest instances:
   - Compute the inverse of each distance of the '*k*' selected nearest instances.
   - Find the sum of the inverses.
   - Compute the weight by dividing each inverse distance by the sum. (Each weight is a vote for its associated class).
   - Add the weights of the same class.
   - Predict the class by choosing the class with the maximum vote.

Example 4.2: Consider the same training dataset given in Table 4.1. Use Weighted *k*-NN and determine the class.

Solution:

Step 1: Given a test instance (7.6, 60, 8) and a set of classes {Pass, Fail}, use the training dataset to classify the test instance using Euclidean distance and weighting function.

Assign *k* = 3. The distance calculation is shown in Table 4.5.

Table 4.5: Euclidean Distance

S.No.	CGPA	Assessment	Project Submitted	Result	Euclidean Distance
1.	9.2	85	8	Pass	$\sqrt{(9.2-7.6)^2+(85-60)^2+(8-8)^2}$ = 25.05115
2.	8	80	7	Pass	$\sqrt{(8-7.6)^2+(80-60)^2+(7-8)^2}$ = 20.02898
3.	8.5	81	8	Pass	$\sqrt{(8.5-7.6)^2+(81-60)^2+(8-8)^2}$ = 21.01928

(*Continued*)

S.No.	CGPA	Assessment	Project Submitted	Result	Euclidean Distance
4.	6	45	5	Fail	$\sqrt{(6-7.6)^2+(45-60)^2+(5-8)^2}$ = 15.38051
5.	6.5	50	4	Fail	$\sqrt{(6.5-7.6)^2+(50-60)^2+(4-8)^2}$ = 10.82636
6.	8.2	72	7	Pass	$\sqrt{(8.2-7.6)^2+(72-60)^2+(7-8)^2}$ = 12.05653
7.	5.8	38	5	Fail	$\sqrt{(5.8-7.6)^2+(38-60)^2+(5-8)^2}$ = 22.27644
8.	8.9	91	9	Pass	$\sqrt{(8.9-7.6)^2+(91-60)^2+(9-8)^2}$ = 31.04336

Step 2: Sort the distances in the ascending order and select the first 3 nearest training data instances to the test instance. The selected nearest neighbors are shown in Table 4.6.

Table 4.6: Nearest Neighbors

Instance	Euclidean Distance	Class
4	15.38051	Fail
5	10.82636	Fail
6	12.05653	Pass

Step 3: Predict the class of the test instance by weighted voting technique from the 3 selected nearest instances.

- Compute the inverse of each distance of the 3 selected nearest instances as shown in Table 4.7.

Table 4.7: Inverse Distance

Instance	Euclidean Distance	Inverse Distance	Class
4	15.38051	0.06502	Fail
5	10.82636	0.092370	Fail
6	12.05653	0.08294	Pass

- Find the sum of the inverses.

  Sum = 0.06502 + 0.092370 + 0.08294 = 0.24033

- Compute the weight by dividing each inverse distance by the sum as shown in Table 4.8.

Table 4.8: Weight Calculation

Instance	Euclidean Distance	Inverse Distance	Weight = Inverse distance/Sum	Class
4	15.38051	0.06502	0.270545	Fail
5	10.82636	0.092370	0.384347	Fail
6	12.05653	0.08294	0.345109	Pass

- Add the weights of the same class.

  Fail = 0.270545 + 0.384347 = 0.654892

  Pass = 0.345109

- Predict the class by choosing the class with the maximum vote.

The class is predicted as '**Fail**'.

4.4 NEAREST CENTROID CLASSIFIER

A simple alternative to *k*-NN classifiers for similarity-based classification is the Nearest Centroid Classifier. It is a simple classifier and also called as Mean Difference classifier. The idea of this classifier is to classify a test instance to the class whose centroid/mean is closest to that instance.

Algorithm 4.3: Nearest Centroid Classifier

Inputs: Training dataset *T*, Distance metric *d*, Test instance *t*

Output: Predicted class or category

1. Compute the mean/centroid of each class.
2. Compute the distance between the test instance and mean/centroid of each class (Euclidean Distance).
3. Predict the class by choosing the class with the smaller distance.

Example 4.3: Consider the sample data shown in Table 4.9 with two features *x* and *y*. The target classes are 'A' or 'B'. Predict the class using Nearest Centroid Classifier.

Table 4.9: Sample Data

X	*Y*	Class
3	1	**A**
5	2	**A**
4	3	**A**
7	6	**B**
6	7	**B**
8	5	**B**

Solution:

Step 1: Compute the mean/centroid of each class. In this example there are two classes called 'A' and 'B'.

Centroid of class 'A' = (3 + 5 + 4, 1 + 2 + 3)/3 = (12, 6)/3 = (4, 2)

Centroid of class 'B' = (7 + 6 + 8, 6 + 7 + 5)/3 = (21, 18)/3 = (7, 6)

Now given a test instance (6, 5), we can predict the class.

Step 2: Calculate the Euclidean distance between test instance (6, 5) and each of the centroid.

$$\text{Euc\_Dist}[(6, 5) ; (4, 2)] = \sqrt{(6-4)^2 + (5-2)^2} = \sqrt{13} = 3.6$$

$$\text{Euc\_Dist}[(6, 5) ; (7, 6)] = \sqrt{(6-7)^2 + (5-6)^2} = \sqrt{2} = 1.414$$

The test instance has smaller distance to class B. Hence, the class of this test instance is predicted as 'B'.

4.5 LOCALLY WEIGHTED REGRESSION (LWR)

Locally Weighted Regression (LWR) is a non-parametric supervised learning algorithm that performs local regression by combining regression model with nearest neighbor's model. LWR is also referred to as a memory-based method as it requires training data while prediction but uses only the training data instances locally around the point of interest. Using nearest neighbors algorithm, we find the instances that are closest to a test instance and fit linear function to each of those 'k' nearest instances in the local regression model. The key idea is that we need to approximate the linear functions of all 'k' neighbors that minimize the error such that the prediction line is no more linear but rather it is a curve.

Ordinary linear regression finds out a linear relationship between the input x and the output y.

Given training dataset T,

Hypothesis function $h_\beta(x)$, the predicted target output is a linear function where β_0 is the intercept and β_1 is the coefficient of x.

It is given in Eq. (4.1) as,

$$h_\beta(x) = \beta_0 + \beta_1 x \tag{4.1}$$

The cost function is such that it minimizes the error difference between the predicted value $h_\beta(x)$ and true value 'y' and it is given as in Eq. (4.2).

$$J(\beta) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\beta(x_i) - y_i\right)^2 \tag{4.2}$$

where 'm' is the number of instances in the training dataset.

Now the cost function is modified for locally weighted linear regression including the weights only for the nearest neighbor points. Hence, the cost function is given as in Eq. (4.3).

$$J(\beta) = \frac{1}{2}\sum_{i=1}^{m} w_i\left(h_\beta(x_i) - y_i\right)^2 \tag{4.3}$$

where w_i is the weight associated with each x_i.

The weight function used is a Gaussian kernel that gives a higher value for instances that are close to the test instance, and for instances far away, it tends to zero but never equals to zero.

w_i is computed in Eq. (4.4) as,

$$w_i = e^{\frac{-(x_i - x)^2}{2\tau^2}} \tag{4.4}$$

where, τ is called the bandwidth parameter and controls the rate at which w_i reduces to zero with distance from x_i.

Example 4.4: Consider a simple example with four instances shown in Table 4.10 and apply locally weighted regression.

Table 4.10: Sample Table

S.No.	Salary (in lakhs)	Expenditure (in thousands)
1.	5	25
2.	1	5
3.	2	7
4.	1	8

Solution: Using linear regression model assuming we have computed the parameters:

$\beta_0 = 4.72$, $\beta_1 = 0.62$

Given a test instance with $x = 2$, the predicted y' is:

$y' = \beta_0 + \beta_1 x = 4.72 + 0.62 \times 2 = 5.96$

Applying the nearest neighbor model, we choose $k = 3$ closest instances.

Table 4.11 shows the Euclidean distance calculation for the training instances.

Table 4.11: Euclidean Distance Calculation

S.No.	x = Salary (in lakhs)	y = Expenditure (in thousands)	Euclidean Distance
1.	5	25	$\sqrt{(5-2)^2} = 3$
2.	1	5	$\sqrt{(1-2)^2} = 1$
3.	2	7	$\sqrt{(2-2)^2} = 0$
4.	1	8	$\sqrt{(1-2)^2} = 1$

Instances 2, 3 and 4 are closer with smaller distances.

The mean value = (5 + 7 + 8)/3 = 20/3 = 6.67.

Using Eq. (4.4) compute the weights for the closest instances, using the Gaussian kernel,

$$w_i = e^{\frac{-(x_i - x)^2}{2\tau^2}}$$

Hence the weights of the closest instances is computed as follows,

Weight of Instance 2 is:

$$w_2 = e^{\frac{-(x_2 - x)^2}{2\tau^2}} = e^{\frac{-(1-2)^2}{2\times 0.4^2}} = e^{-3.125} = 0.043$$

Weight of Instance 3 is:

$$w_3 = e^{\frac{-(x_3 - x)^2}{2\tau^2}} = e^{\frac{-(2-2)^2}{2\times 0.4^2}} = e^{0} = 1$$ [w_3 is closer hence gets a higher weight value]

Weight of Instance 4 is:

$$w_4 = e^{\frac{-(x_4 - x)^2}{2\tau^2}} = e^{\frac{-(1-2)^2}{2\times 0.4^2}} = e^{-3.125} = 0.043$$

The predicted output for the three closer instances is given as follows:

The predicted output of Instance 2 is:

$y_2' = h_\beta(x_2) = \beta_0 + \beta_1 x_2 = 4.72 + 0.62 \times 1 = 5.34$

The predicted output of Instance 3 is:

$y_3' = h_\beta(x_3) = \beta_0 + \beta_1 x_2 = 4.72 + 0.62 \times 2 = 5.96$

The predicted output of Instance 4 is:

$y_4' = h_\beta(x_4) = \beta_0 + \beta_1 x_2 = 4.72 + 0.62 \times 1 = 5.34$

The error value is calculated as:

$$J(\beta) = \frac{1}{2}\sum_{i=1}^{m} w_i (h_\beta(x_i) - y_i)^2 = \frac{1}{2}(0.043(5.34 - 5)^2 + 1(5.96 - 7)^2 + 0.043(5.34 - 8)^2) = 0.6953$$

Now, we need to adjust this cost function to minimize the error difference and get optimal β parameters.

Chapter 5

Regression Analysis

"Regression analysis is the hydrogen bomb of the statistics arsenal."
— Charles Wheelan, Naked Statistics: Stripping the Dread from the Data

Regression analysis is a supervised learning method for predicting continuous variables. The difference between classification and regression analysis is that regression methods are used to predict qualitative variables or continuous numbers unlike categorical variables or labels. It is used to predict linear or non-linear relationships among variables of the given dataset. This chapter deals with an introduction of regression and its various types.

Learning Objectives

- Understand the basics of regression analysis
- Introduce concepts of correlation and causation
- Learn about linear regression and its validation techniques
- Discuss about multiple linear regression
- Introduce logistic regression
- Study about the concept of regularization
- Study popular regression methods like Ridge, Lasso, and Elastic Net

5.1 INTRODUCTION TO REGRESSION

Regression analysis is the premier method of supervised learning. This is one of the most popular and oldest supervised learning technique. Given a training dataset D containing N training points (x_i, y_i), where $i = 1...N$, regression analysis is used to model the relationship between one or more independent variables x_i and a dependent variable y_i. The relationship between the dependent and independent variables can be represented as a function as follows:

$$y = f(x) \tag{5.1}$$

Here, the feature variable x is also known as an explanatory variable, exploratory variable, a predictor variable, an independent variable, a covariate, or a domain point. y is a dependent variable. Dependent variables are also called as labels, target variables, or response variables.

Regression analysis determines the change in response variables when one exploration variable is varied while keeping all other parameters constant. This is used to determine the relationship each of the exploratory variables exhibits. Thus, regression analysis is used for prediction and forecasting.

Regression is used to predict continuous variables or quantitative variables such as price and revenue. Thus, the primary concern of regression analysis is to find answer to questions such as:

1. What is the relationship between the variables?
2. What is the strength of the relationships?
3. What is the nature of the relationship such as linear or non-linear?
4. What is the relevance of the attributes?
5. What is the contribution of each attribute?

There are many applications of regression analysis. Some of the applications of regressions include predicting:

1. Sales of a goods or services
2. Value of bonds in portfolio management
3. Premium on insurance companies
4. Yield of crops in agriculture
5. Prices of real estate

5.2 INTRODUCTION TO LINEARITY, CORRELATION, AND CAUSATION

The quality of the regression analysis is determined by the factors such as correlation and causation.

Regression and Correlation

Correlation among two variables can be done effectively using a Scatter plot, which is a plot between explanatory variables and response variables. It is a 2D graph showing the relationship between two variables. The x-axis of the scatter plot is independent, or input or predictor variables and y-axis of the scatter plot is output or dependent or predicted variables. The scatter plot is useful in exploring data. Some of the scatter plots are shown in Figure 5.1. The Pearson correlation coefficient is the most common test for determining correlation if there is an association between two variables. The correlation coefficient is denoted by r. Correlation is discussed in Chapter 2 of this book. The positive, negative, and random correlations are given in Figure 5.1. In positive correlation, one variable change is associated with the change in another variable. In negative correlation, the relationship between the variables is reciprocal while in random correlation, no relationship exists between variables.

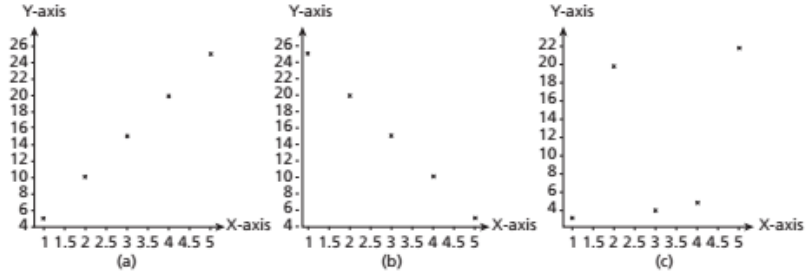


Figure 5.1: Examples of (a) Positive Correlation (b) Negative Correlation (c) Random Points with No Correlation

While correlation is about relationships among variables, say x and y, regression is about predicting one variable given another variable.

Regression and Causation

Causation is about causal relationship among variables, say x and y. Causation means knowing whether x causes y to happen or vice versa. x causes y is often denoted as x implies y. Correlation and Regression relationships are not same as causation relationship. For example, the correlation between economical background and marks scored does not imply that economic background causes high marks. Similarly, the relationship between higher sales of cool drinks due to a rise in temperature is not a causal relation. Even though high temperature is the cause of cool drinks sales, it depends on other factors too.

Linearity and Non-linearity Relationships

The linearity relationship between the variables means the relationship between the dependent and independent variables can be visualized as a straight line. The line of the form, $y = ax + b$ can be fitted to the data points that indicate the relationship between x and y. By linearity, it is meant that as one variable increases, the corresponding variable also increases in a linear manner. A linear relationship is shown in Figure 5.2 (a). A non-linear relationship exists in functions such as exponential function and power function and it is shown in Figures 5.2 (b) and 5.2 (c). Here, x-axis is given by x data and y-axis is given by y data.

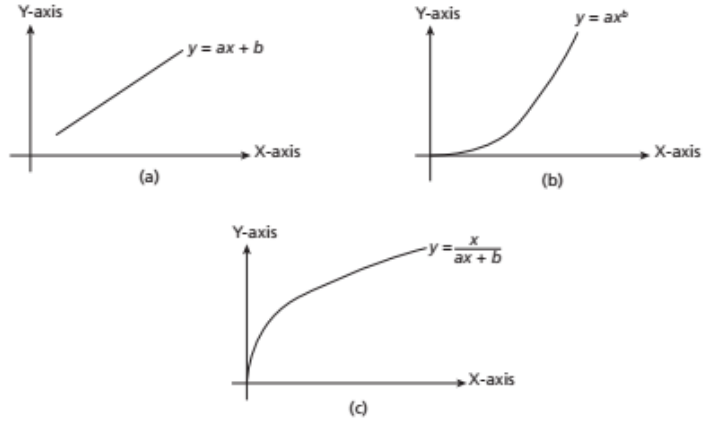


Figure 5.2: (a) Example of Linear Relationship of the Form $y = ax + b$ (b) Example of a Non-linear Relationship of the Form $y = ax^b$ (c) Examples of a Non-linear Relationship $y = \frac{x}{ax + b}$

The functions like exponential function ($y = ax^b$) and power function $\left(y = \frac{x}{ax + b}\right)$ are non-linear relationships between the dependent and independent variables that cannot be fitted in a line. This is shown in Figures 5.2 (b) and (c).

Types of Regression Methods

The classification of regression methods is shown in Figure 5.3.

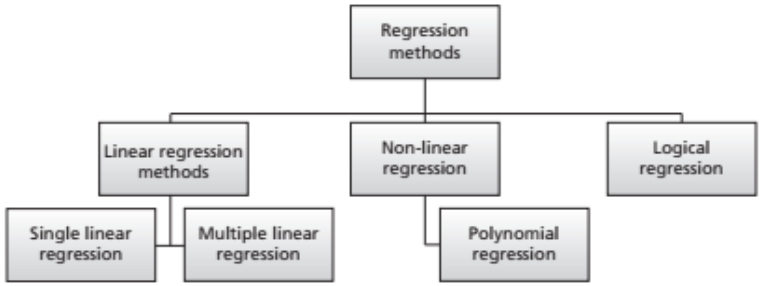


Figure 5.3: Types of Regression Methods

Linear Regression It is a type of regression where a line is fitted upon given data for finding the linear relationship between one independent variable and one dependent variable to describe relationships.

Multiple Regression It is a type of regression where a line is fitted for finding the linear relationship between two or more independent variables and one dependent variable to describe relationships among variables.

Polynomial Regression It is a type of non-linear regression method of describing relationships among variables where N^{th} degree polynomial is used to model the relationship between one independent variable and one dependent variable. Polynomial multiple regression is used to model two or more independent variables and one dependant variable.

Logistic Regression It is used for predicting categorical variables that involve one or more independent variables and one dependent variable. This is also known as a binary classifier.

Lasso and Ridge Regression Methods These are special variants of regression method where regularization methods are used to limit the number and size of coefficients of the independent variables.

Limitations of Regression Method

1. Outliers – Outliers are abnormal data. It can bias the outcome of the regression model, as outliers push the regression line towards it.
2. Number of cases – The ratio of independent and dependent variables should be at least 20 : 1. For every explanatory variable, there should be at least 20 samples. Atleast five samples are required in extreme cases.
3. Missing data – Missing data in training data can make the model unfit for the sampled data.
4. Multicollinearity – If exploratory variables are highly correlated (0.9 and above), the regression is vulnerable to bias. Singularity leads to perfect correlation of 1. The remedy is to remove exploratory variables that exhibit correlation more than 1. If there is a tie, then the tolerance (1 – R squared) is used to eliminate variables that have the greatest value.

5.3 INTRODUCTION TO LINEAR REGRESSION

In the simplest form, the linear regression model can be created by fitting a line among the scattered data points. The line is of the form given in Eq. (5.2).

$$y = a_0 + a_1 \times x + e \tag{5.2}$$

Here, a_0 is the intercept which represents the bias and a_1 represents the slope of the line. These are called regression coefficients. e is the error in prediction.

The assumptions of linear regression are listed as follows:

1. The observations (y) are random and are mutually independent.
2. The difference between the predicted and true values is called an error. The error is also mutually independent with the same distributions such as normal distribution with zero mean and constant variables.
3. The distribution of the error term is independent of the joint distribution of explanatory variables.
4. The unknown parameters of the regression models are constants.

The idea of linear regression is based on Ordinary Least Square (OLS) approach. This method is also known as ordinary least squares method. In this method, the data points are modelled using a straight line. Any arbitrarily drawn line is not an optimal line. In Figure 5.4, three data points and their errors (e_1, e_2, e_3) are shown. The vertical distance between each point and the line (predicted by the approximate line equation $y = a_0 + a_1x$) is called an error. These individual errors are added to compute the total error of the predicted line. This is called sum of residuals. The squares of the individual errors can also be computed and added to give a sum of squared error. The line with the lowest sum of squared error is called line of best fit.

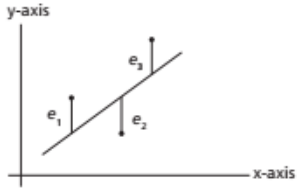


Figure 5.4: Data Points and their Errors

In another words, OLS is an optimization technique where the difference between the data points and the line is optimized.

Mathematically, based on Eq. (5.2), the line equations for points (x_1, x_2, …, x_n) are:

$$y_1 = (a_0 + a_1x_1) + e_1$$
$$y_2 = (a_0 + a_1x_2) + e_2$$
$$\vdots$$
$$y_n = (a_0 + a_1x_n) + e_n \tag{5.3}$$

In general, the error is given as: $e_i = y_i - (a_0 + a_1x_i)$ (5.4)

This can be extended into the set of equations as shown in Eq. (5.3).

Here, the terms $(e_1, e_2, \ldots, e_n)$ are error associated with the data points and denote the difference between the true value of the observation and the point on the line. This is also called as residuals. The residuals can be positive, negative or zero.

A regression line is the line of best fit for which the sum of the squares of residuals is minimum. The minimization can be done as minimization of individual errors by finding the parameters a_0 and a_1 such that:

$$E = \sum_{i=1}^{n} e_i = \sum_{i=1}^{n} (y_i - (a_0 + a_1 x_i)) \tag{5.5}$$

Or as the minimization of sum of absolute values of the individual errors:

$$E = \sum_{i=1}^{n} |e_i| = \sum_{i=1}^{n} |(y_i - (a_0 + a_1 x_i))| \tag{5.6}$$

Or as the minimization of the sum of the squares of the individual errors:

$$E = \sum_{i=1}^{n} (e_i)^2 = \sum_{i=1}^{n} (y_i - (a_0 + a_1 x_i))^2 \tag{5.7}$$

Sum of the squares of the individual errors, often preferred as individual errors (positive and negative errors), do not get cancelled out and are always positive, and sum of squares results in a large increase even for a small change in the error. Therefore, this is preferred for linear regression.

Therefore, linear regression is modelled as a minimization function as follows:

$$J(a_1, a_0) = \sum_{i=1}^{n} [y_i - f(x_i)]^2$$
$$= \sum_{i=1}^{n} [y_i - (a_0 + a_1 x_i)]^2 \tag{5.8}$$

Here, $J(a_0, a_1)$ is the criterion function of parameters a_0 and a_1. This needs to be minimized. This is done by differentiating and substituting to zero. This yields the coefficient values of a_0 and a_1. The values of estimates of a_0 and a_1 are given as follows:

$$a_1 = \frac{(\overline{xy}) - (\overline{x})(\overline{y})}{(\overline{x_i^2}) - (\overline{x})^2} \tag{5.9}$$

And the value of a_0 is given as follows:

$$a_0 = (\overline{y}) - a_1 \times \overline{x} \tag{5.10}$$

Let us consider a simple problem to illustrate the usage of the above concept.

Example 5.1: Let us consider an example where the five weeks' sales data (in Thousands) is given as shown below in Table 5.1. Apply linear regression technique to predict the 7th and 9th month sales.

Table 5.1: Sample Data

x_i (Week)	y_i (Sales in Thousands)
1	1.2
2	1.8
3	2.6
4	3.2
5	3.8

Solution: Here, there are 5 items, i.e., i = 1, 2, 3, 4, 5. The computation table is shown below (Table 5.2). Here, there are five samples, so i ranges from 1 to 5.

Table 5.2: Computation Table

x_i	y_i	$(x_i)^2$	$x_i \times y_i$
1	1.2	1	1.2
2	1.8	4	3.6
3	2.6	9	7.8
4	3.2	16	12.8
5	3.8	25	19
Sum = 15 Average of (x_i) $= \overline{x} = \frac{15}{5}$ = 3	Sum = 12.6 Average of (y_i) $= \overline{y} = \frac{12.6}{5}$ = 2.52	Sum = 55 Average of (x_i^2) $= \overline{x_i^2} = \frac{55}{5}$ = 11	Sum = 44.4 Average of $(x_i \times y_i)$ $= \overline{xy} = \frac{44.4}{5}$ = 8.88

Let us compute the slope and intercept now using Eq. (5.9) as:

$$a_1 = \frac{8.88 - 3(2.52)}{11 - 3^2} = 0.66$$
$$a_0 = 2.52 - 0.66 \times 3 = 0.54$$

The fitted line is shown in Figure 5.5.

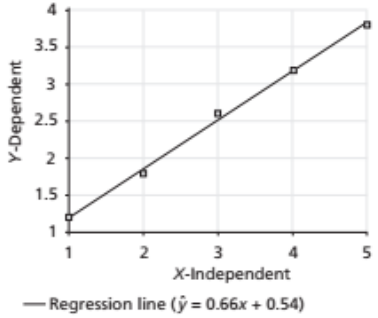


Figure 5.5: Linear Regression Model Constructed

Let us model the relationship as $y = a_0 + a_1 \times x$. Therefore, the fitted line for the above data is: $y = 0.54 + 0.66 \times x$.

The predicted 7th week sale would be (when $x = 7$), $y = 0.54 + 0.66 \times 7 = 5.16$ and the 12th month, $y = 0.54 + 0.66 \times 12 = 8.46$. All sales are in thousands.

Scan for *'Additional Examples'*

Linear Regression in Matrix Form

Matrix notations can be used for representing the values of independent and dependent variables. This is illustrated through Example 5.2.

The Eq. (5.3) can be written in the form of matrix as follows:

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & \vdots \\ 1 & x_n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix} \tag{5.11}$$

This can be written as:

$Y = Xa + e$, where X is an $n \times 2$ matrix, Y is an $n \times 1$ vector, a is a 2×1 column vector and e is an $n \times 1$ column vector.

Example 5.2: Find linear regression of the data of week and product sales (in Thousands) given in Table 5.3. Use linear regression in matrix form.

Table 5.3: Sample Data for Regression

x_i (Week)	y_i (Product Sales in Thousands)
1	1
2	3
3	4
4	8

Solution: Here, the dependent variable X is be given as:

$$x^T = [1\ 2\ 3\ 4]$$

And the independent variable is given as follows:

$$y^T = [1\ 3\ 4\ 8]$$

The data can be given in matrix form as follows:

$X = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{pmatrix}$. The first column can be used for setting bias.

and $Y = \begin{pmatrix} 1 \\ 3 \\ 4 \\ 8 \end{pmatrix}$

The regression is given as:

$$a = ((X^TX)^{-1}\ X^T)Y$$

The computation order of this equation is shown step by step as:

1. Computation of $(X^TX) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{pmatrix} \times \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{pmatrix} = \begin{pmatrix} 4 & 10 \\ 10 & 30 \end{pmatrix}$
2. Computation of matrix inverse of $(X^TX)^{-1} = \begin{pmatrix} 4 & 10 \\ 10 & 30 \end{pmatrix}^{-1} = \begin{pmatrix} 1.5 & -0.5 \\ -0.5 & 0.2 \end{pmatrix}$
3. Computation of $((X^TX)^{-1}X^T) = \begin{pmatrix} 1.5 & -0.5 \\ -0.5 & 0.2 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0.5 & 0 & -0.5 \\ -0.3 & -0.1 & 0.1 & 0.3 \end{pmatrix}$
4. Finally, $((X^TX)^{-1}X^T)Y = \begin{pmatrix} 1 & 0.5 & 0 & -0.5 \\ -0.3 & -0.1 & 0.1 & 0.3 \end{pmatrix} \times \begin{pmatrix} 1 \\ 3 \\ 4 \\ 8 \end{pmatrix} = \begin{pmatrix} -1.5 \\ 2.2 \end{pmatrix} \begin{pmatrix} Intercept \\ slope \end{pmatrix}$

Thus, the substitution of values in Eq. (5.11) using the previous steps yields the fitted line as $2.2\,x - 1.5$.

5.4 VALIDATION OF REGRESSION METHODS

The regression model should be evaluated using some metrics for checking the correctness. The following metrics are used to validate the results of regression.

Standard Error

Residuals or error is the difference between the actual (y) and predicted value ($\hat{y}$).

If the residuals have normal distribution, then the mean is zero and hence it is desirable. This is a measure of variability in finding the coefficients. It is preferable that the error be less than the coefficient estimate. The standard deviation of residuals is called residual standard error. If it is zero, then it means that the model fits the data correctly.

Mean Absolute Error (MAE)

MAE is the mean of residuals. It is the difference between estimated or predicted target value and actual target incomes. It can be mathematically defined as follows:

$$\text{MAE} = \frac{1}{n}\sum_{i=0}^{n-1}\left|y_i - \hat{y}_i\right| \tag{5.12}$$

Here, $\hat{y}$ is the estimated or predicted target output and y is the actual target output, and n is the number of samples used for regression analysis.

Mean Squared Error (MSE)

It is the sum of square of residuals. This value is always positive and closer to 0. This is given mathematically as:

$$\frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2 \tag{5.13}$$

Root Mean Square Error (RMSE)

The square root of the MSE is called RMSE. This is given as:

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2} \tag{5.14}$$

Relative MSE

Relative MSE is the ratio of the prediction ability of the $\hat{y}$ to the average of the trivial population. The value of zero indicates that the model is perfect and its value ranges between 0 and 1. If the value is more than 1, then the created model is not a good one. This is given as follows:

$$\text{RelMSE} = \frac{\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \overline{y}_i)^2} \tag{5.15}$$

Coefficient of Variation

Coefficient of variation is unit less and is given as:

$$\text{CV} = \frac{\text{RMSE}}{\overline{y}} \tag{5.16}$$

Example 5.3: Consider the following training set Table 5.4 for predicting the sales of the items.

Table 5.4: Training Item Table

Items x_i	Actual Sales (In Thousands) y_i
I_1	80
I_2	90
I_3	100
I_4	110
I_5	120

Consider two fresh items I_6 and I_7, whose actual values are 80 and 75, respectively. A regression model predicts the values of the items I_6 and I_7 as 75 and 85, respectively. Find MAE, MSE, RMSE, RelMSE and CV.

Solution: The test items' actual and prediction is given in Table 5.5 as:

Table 5.5: Test Item Table

Test Items	Actual Value y_i	Predicted Value $\overline{y}_i$
I_6	80	75
I_7	75	85

Mean Absolute Error (MAE) using Eq. (5.12) is given as:

$$\text{MAE} = \frac{1}{2} \times |80 - 75| + |75 - 85| = \frac{15}{2} = 7.5$$

Mean Squared Error (MSE) using Eq. (5.13) is given as:

$$\text{MSE} = \frac{1}{2} \times |80 - 75|^2 + |75 - 85|^2 = \frac{125}{2} = 62.5$$

Root Mean Square error using Eq. (5.14) is given as:

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{62.5} = 7.91$$

For finding RelMSE and CV, the training table should be used to find the average of y. The average of y is $\frac{80 + 90 + 100 + 110 + 120}{5} = \frac{500}{5} = 100$.

RelMSE using Eq. (5.15) can be computed as:

$$\text{RelMSE} = \frac{(80 - 75)^2 + (75 - 85)^2}{(80 - 100)^2 + (75 - 100)^2} = \frac{125}{1025} = 0.1219$$

CV can be computed using Eq. (5.16) as $\frac{\sqrt{62.5}}{100} = 0.08$.

Coefficient of Determination

To understand the coefficient of determination, one needs to understand the total variation of coefficients in regression analysis. The sum of the squares of the differences between the y-value of the data pair and the average of y is called total variation. Thus, the following variations can be defined.

The explained variation is given as:

$$= \Sigma(\hat{y}_i - \overline{y}_i)^2 \tag{5.17}$$

The unexplained variation is given as:

$$= \Sigma(y_i - \hat{y}_i)^2 \tag{5.18}$$

Thus, the total variation is equal to the explained variation and the unexplained variation.

The coefficient of determination r^2 is the ratio of the explained and total variations.

$$r^2 = \frac{\text{Explained variation}}{\text{Total variation}} \tag{5.19}$$

It is a measure of how many future samples are likely to be predicted by the regression model. Its value ranges from 1 to $-\infty$, where 1 is the most optimum value. It also signifies the proportion of variance. Here, r is the correlation coefficient. If $r = 0.95$, then r^2 is given as $0.95 \times 0.95 = 0.9025$. This means that 90% of the model can be explained by the relationship between x and y. The rest 10% is unexplained and that may be due to various reasons such as noise, chance, or error.

Standard Error Estimate

Standard error estimate is another useful measure of regression. It is the standard deviation of the observed values to the predicted values. This is given as:

$$s_e = \sqrt{\frac{\Sigma(y_i - \hat{y}_i)}{n - 2}} \tag{5.20}$$

Here, as usual, y_i is the observed value and $\hat{y}_i$ is the predicted value. Here, n is the number of samples.

Example 5.4: Let us consider the data given in the Table 5.3 with actual and predicted values. Find standard error estimate.

Solution: The observed value or the predicted value is given below in Table 5.6.

Table 5.6: Sample Data

x_i	y_i	Predicted Value	$(y - \hat{y})^2$
1	1.5	1.46	$(1.5 - 1.46)^2 = 0.0016$
2	2.9	2.02	$(2.9 - 2.02)^2 = 0.7744$
3	2.7	2.58	$(2.7 - 2.58)^2 = 0.0144$
4	3.1	3.14	$(3.1 - 3.14)^2 = 0.0016$

The sum of $(y - \hat{y})^2$ for all $i = 1, 2, 3$ and 4 (i.e., number of samples $n = 4$) is 0.792. The standard deviation error estimate as given in Eq. (5.20) is:

$$\sqrt{\frac{0.792}{4-2}} = \sqrt{0.396} = 0.629$$

Chapter 6

Decision Tree Learning

"Prediction is very difficult, especially if it's about the future."
— Niels Bohr

Decision Tree Learning is a widely used predictive model for supervised learning that spans over a number of practical applications in various areas. It is used for both classification and regression tasks. The decision tree model basically represents logical rules that predict the value of a target variable by inferring from data features. This chapter provides a keen insight into how to construct a decision tree and infer knowledge from the tree.

Learning Objectives

- Understand the structure of the decision tree
- Know about the fundamentals of Entropy
- Learn and understand popular univariate Decision Tree Induction algorithms such as ID3, C4.5, and multivariate decision tree algorithm such as CART
- Deal with continuous attributes using improved C4.5 algorithm
- Construct Classification and Regression Tree (CART) for classifying both categorical and continuous-valued target variables
- Construct regression trees where the target feature is a continuous-valued variable
- Understand the basics of validating and pruning of decision trees

6.1 INTRODUCTION TO DECISION TREE LEARNING MODEL

Decision tree learning model, one of the most popular supervised predictive learning models, classifies data instances with high accuracy and consistency. The model performs an *inductive inference* that reaches a general conclusion from observed examples. This model is variably used for solving complex classification applications.

Decision tree is a concept tree which summarizes the information contained in the training dataset in the form of a tree structure. Once the concept model is built, test data can be easily classified.

This model can be used to classify both categorical target variables and continuous-valued target variables. Given a training dataset X, this model computes a hypothesis function $f(X)$ as decision tree.

Inputs to the model are data instances or objects with a set of features or attributes which can be discrete or continuous and the output of the model is a decision tree which predicts or classifies the target class for the test data object.

In statistical terms, attributes or features are called as independent variables. The target feature or target class is called as response variable which indicates the category we need to predict on a test object.

The decision tree learning model generates a complete hypothesis space in the form of a tree structure with the given training dataset and allows us to search through the possible set of hypotheses which in fact would be a smaller decision tree as we walk through the tree. This kind of search bias is called as preference bias.

6.1.1 Structure of a Decision Tree

A decision tree has a structure that consists of a root node, internal nodes/decision nodes, branches, and terminal nodes/leaf nodes. The topmost node in the tree is the root node. Internal nodes are the test nodes and are also called as decision nodes. These nodes represent a choice or test of an input attribute and the outcome or outputs of the test condition are the branches emanating from this decision node. The branches are labelled as per the outcomes or output values of the test condition. Each branch represents a sub-tree or subsection of the entire tree. Every decision node is part of a path to a leaf node. The leaf nodes represent the labels or the outcome of a decision path. The labels of the leaf nodes are the different target classes a data instance can belong to.

Every path from root to a leaf node represents a logical rule that corresponds to a conjunction of test attributes and the whole tree represents a disjunction of these conjunctions. The decision tree model, in general, represents a collection of logical rules of classification in the form of a tree structure.

Decision networks, otherwise called as influence diagrams, have a directed graph structure with nodes and links. It is an extension of Bayesian belief networks that represents information about each node's current state, its possible actions, the possible outcome of those actions, and their utility. The concept of Bayesian Belief Network (BBN) is discussed in Chapter 9.

Figure 6.1 shows symbols that are used in this book to represent different nodes in the construction of a decision tree. A circle is used to represent a root node, a diamond symbol is used to represent a decision node or the internal nodes, and all leaf nodes are represented with a rectangle.

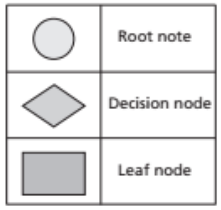


Figure 6.1: Nodes in a Decision Tree

A decision tree consists of two major procedures discussed below.

Building the Tree

Goal Construct a decision tree with the given training dataset. The tree is constructed in a top-down fashion. It starts from the root node. At every level of tree construction, we need to find the best split attribute or best decision node among all attributes. This process is recursive and continued until we end up in the last level of the tree or finding a leaf node which cannot be split further. The tree construction is complete when all the test conditions lead to a leaf node. The leaf node contains the target class or output of classification.

Output Decision tree representing the complete hypothesis space.

Knowledge Inference or Classification

Goal Given a test instance, infer to the target class it belongs to.

Classification Inferring the target class for the test instance or object is based on inductive inference on the constructed decision tree. In order to classify an object, we need to start traversing the tree from the root. We traverse as we evaluate the test condition on every decision node with the test object attribute value and walk to the branch corresponding to the test's outcome. This process is repeated until we end up in a leaf node which contains the target class of the test object.

Output Target label of the test instance.

Advantages of Decision Trees

1. Easy to model and interpret
2. Simple to understand
3. The input and output attributes can be discrete or continuous predictor variables.
4. Can model a high degree of nonlinearity in the relationship between the target variables and the predictor variables
5. Quick to train

Disadvantages of Decision Trees

Some of the issues that generally arise with a decision tree learning are that:

1. It is difficult to determine how deeply a decision tree can be grown or when to stop growing it.
2. If training data has errors or missing attribute values, then the decision tree constructed may become unstable or biased.
3. If the training data has continuous valued attributes, handling it is computationally complex and has to be discretized.
4. A complex decision tree may also be over-fitting with the training data.
5. Decision tree learning is not well suited for classifying multiple output classes.
6. Learning an optimal decision tree is also known to be NP-complete.

Example 6.1: How to draw a decision tree to predict a student's academic performance based on the given information such as class attendance, class assignments, home-work assignments, tests, participation in competitions or other events, group activities such as projects and presentations, etc.

Solution: The target feature is the student performance in the final examination whether he will pass or fail in the examination. The decision nodes are test nodes which check for conditions like 'What's the student's class attendance?', 'How did he perform in his class assignments?', 'Did he do his home assignments properly?' 'What about his assessment results?', 'Did he participate in competitions or other events?', 'What is the performance rating in group activities such as projects and presentations?'. Table 6.1 shows the attributes and set of values for each attribute.

Table 6.1: Attributes and Associated Values

Attributes	Values
Class attendance	Good, Average, Poor
Class assignments	Good, Moderate, Poor
Home-work assignments	Yes, No
Assessment	Good, Moderate, Poor
Participation in competitions or other events	Yes, No
Group activities such as projects and presentations	Yes, No
Exam Result	Pass, Fail

The leaf nodes represent the outcomes, that is, either 'pass', or 'fail'.

A decision tree would be constructed by following a set of if-else conditions which may or may not include all the attributes, and decision nodes outcomes are two or more than two. Hence, the tree is not a binary tree.

Note: *A decision tree is not always a binary tree. It is a tree which can have more than two branches.*

Example 6.2: Predict a student's academic performance of whether he will pass or fail based on the given information such as 'Assessment' and 'Assignment'. The following Table 6.2 shows the independent variables, Assessment and Assignment, and the target variable Exam Result with their values. Draw a binary decision tree.

Table 6.2: Attributes and Associated Values

Attributes	Values
Assessment	≥50, <50
Assignment	Yes, No
Exam Result	Pass, Fail

Solution: Consider the root node is 'Assessment'. If a student's marks are ≥50, the root node is branched to leaf node 'Pass' and if the assessment marks are <50, it is branched to another decision node. If the decision node in next level of the tree is 'Assignment' and if a student has submitted his assignment, the node branches to 'Pass' and if not submitted, the node branches to 'Fail'. Figure 6.2 depicts this rule.

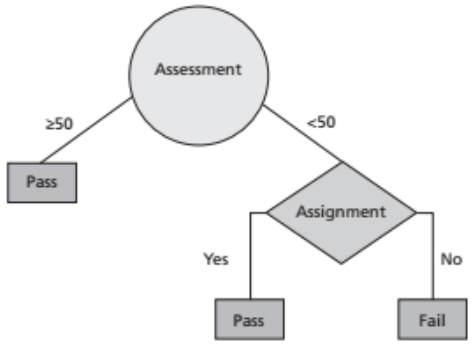


Figure 6.2: Illustration of a Decision Tree

This tree can be interpreted as a sequence of logical rules as follows:

if (Assessment ≥ 50) then 'Pass'
else if (Assessment < 50) then
    if (Assignment == Yes) then 'Pass'
    else if (Assignment == No) then 'Fail'

Now, if a test instance is given, such as a student has scored 42 marks in his assessment and has not submitted his assignment, then it is predicted with the decision tree that his exam result is 'Fail'.

Many algorithms exist which will be studied for constructing decision trees in the sections below.

6.1.2 Fundamentals of Entropy

Given the training dataset with a set of attributes or features, the decision tree is constructed by finding the attribute or feature that best describes the target class for the given test instances. The best split feature is the one which contains more information about how to split the dataset among all features so that the target class is accurately identified for the test instances. In other words, the best split attribute is more informative to split the dataset into sub datasets and this process is continued until the stopping criterion is reached. This splitting should be pure at every stage of selecting the best feature.

The best feature is selected based on the amount of information among the features which are basically calculated on probabilities. Quantifying information is closely related to information theory. In the field of information theory, the features are quantified by a measure called Shannon Entropy which is calculated based on the probability distribution of the events.

Entropy is the amount of uncertainty or randomness in the outcome of a random variable or an event. Moreover, entropy describes about the homogeneity of the data instances. The best feature is selected based on the entropy value. For example, when a coin is flipped, head or tail are the two outcomes, hence its entropy is lower when compared to rolling a dice which has got six outcomes. Hence, the interpretation is,

Higher the entropy → Higher the uncertainty

Lower the entropy → Lower the uncertainty

Similarly, if all instances are homogenous, say (1, 0), which means all instances belong to the same class (here it is positive) or (0, 1) where all instances are negative, then the entropy is 0. On the other hand, if the instances are equally distributed, say (0.5, 0.5), which means 50% positive and 50% negative, then the entropy is 1. If there are 10 data instances, out of which 6 belong to positive class and 4 belong to negative class, then the entropy is calculated as shown in Eq. (6.1),

$$\text{Entropy} = -\left[\frac{6}{10}\log_2\frac{6}{10} + \frac{4}{10}\log_2\frac{4}{10}\right] \tag{6.1}$$

It is concluded that if the dataset has instances that are completely homogeneous, then the entropy is 0 and if the dataset has samples that are equally divided (i.e., 50% – 50%), it has an entropy of 1. Thus, the entropy value ranges between 0 and 1 based on the randomness of the samples in the dataset. If the entropy is 0, then the split is pure which means that all samples in the set will partition into one class or category. But if the entropy is 1, the split is impure and the distribution of the samples is more random. The stopping criterion is based on the entropy value.

Let P be the probability distribution of data instances from 1 to n as shown in Eq. (6.2).

$$\text{So, } P = P_1 \ldots\ldots P_n \tag{6.2}$$

Entropy of P is the information measure of this probability distribution given in Eq. (6.3),

$$\begin{aligned}\text{Entropy\_Info}(P) &= \text{Entropy\_Info}(P_1 \ldots P_n)\\ &= -(P_1\log_2(P_1) + P_2\log_2(P_2) + \ldots\ldots + P_n\log_2(P_n))\end{aligned} \tag{6.3}$$

where, P_1 is the probability of data instances classified as class 1 and P_2 is the probability of data instances classified as class 2 and so on.

P_1 = |No of data instances belonging to class 1| / |Total no of data instances in the training dataset|

Entropy_Info(P) can be computed as shown in Eq. (6.4).

Thus,

$$\text{Entropy\_Info}(6, 4) \text{ is calculated as } -\left[\frac{6}{10}\log_2\frac{6}{10} + \frac{4}{10}\log_2\frac{4}{10}\right] \tag{6.4}$$

Mathematically, entropy is defined in Eq. (6.5) as:

$$\text{Entropy\_Info}(X) = \sum_{\text{xevalues}(X)} Pr[X = x] \cdot \log_2\frac{1}{Pr[X = x]} \tag{6.5}$$

$Pr[X = x]$ is the probability of a random variable X with a possible outcome x.

Note: $\log_2\frac{1}{Pr[X = x]} = -\log_2(Pr[X = x])$

Algorithm 6.1: General Algorithm for Decision Trees

1. Find the best attribute from the training dataset using an attribute selection measure and place it at the root of the tree.

(*Continued*)

2. Split the training dataset into subsets based on the outcomes of the test attribute and each subset in a branch contains the data instances or tuples with the same value for the selected test attribute.
3. Repeat step 1 and step 2 on each subset until we end up in leaf nodes in all the branches of the tree.
4. This splitting process is recursive until the stopping criterion is reached.

Stopping Criteria

The following are some of the common stopping conditions:

1. The data instances are homogenous which means all belong to the same class C_i and hence its entropy is 0.
2. A node with some defined minimum number of data instances becomes a leaf (Number of data instances in a node is between 0.25 and 1.00% of the full training dataset).
3. The maximum tree depth is reached, so further splitting is not done and the node becomes a leaf node.

6.2 DECISION TREE INDUCTION ALGORITHMS

There are many decision tree algorithms, such as ID3, C4.5, CART, CHAID, QUEST, GUIDE, CRUISE, and CTREE, that are used for classification in real-time environment. The most commonly used decision tree algorithms are ID3 (Iterative Dichotomizer 3), developed by J.R Quinlan in 1986, and C4.5 is an advancement of ID3 presented by the same author in 1993. CART, that stands for Classification and Regression Trees, is another algorithm which was developed by Breiman et al. in 1984.

The accuracy of the tree constructed depends upon the selection of the best split attribute. Different algorithms are used for building decision trees which use different measures to decide on the splitting criterion. Algorithms such as ID3, C4.5 and CART are popular algorithms used in the construction of decision trees. The algorithm ID3 uses 'Information Gain' as the splitting criterion whereas the algorithm C4.5 uses 'Gain Ratio' as the splitting criterion. The CART algorithm is popularly used for classifying both categorical and continuous-valued target variables. CART uses GINI Index to construct a decision tree.

Decision trees constructed using ID3 and C4.5 are also called as *univariate decision trees* which consider only one feature/attribute to split at each decision node whereas decision trees constructed using CART algorithm are *multivariate decision trees* which consider a conjunction of univariate splits. The details about univariate and multivariate data has been discussed in Chapter 2.

6.2.1 ID3 Tree Construction

ID3 is a supervised learning algorithm which uses a training dataset with labels and constructs a decision tree. ID3 is an example of univariate decision trees as it considers only one feature at each decision node. This leads to axis-aligned splits. The tree is then used to classify the future test instances. It constructs the tree using a greedy approach in a top-down fashion by identifying the best attribute at each level of the tree.

ID3 works well if the attributes or features are considered as discrete/categorical values. If some attributes are continuous, then we have to partition attributes or features to be decretized or nominal attributes or features.

The algorithm builds the tree using a purity measure called 'Information Gain' with the given training data instances and then uses the constructed tree to classify the test data. It is applied for training set with only nominal attributes or categorical attributes and with no missing values for classification. ID3 works well for a large dataset. If the dataset is small, overfitting may occur. Moreover, it is not accurate if the dataset has missing attribute values.

No pruning is done during or after construction of the tree and it is prone to outliers. C4.5 and CART can handle both categorical attributes and continuous attributes. Both C4.5 and CART can also handle missing values, but C4.5 is prone to outliers whereas CART can handle outliers as well.

Algorithm 6.2: Procedure to Construct a Decision Tree using ID3

1. Compute Entropy_Info Eq. (6.8) for the whole training dataset based on the target attribute.
2. Compute Entropy_Info Eq. (6.9) and Information_Gain Eq. (6.10) for each of the attribute in the training dataset.
3. Choose the attribute for which entropy is minimum and therefore the gain is maximum as the best split attribute.
4. The best split attribute is placed as the root node.
5. The root node is branched into subtrees with each subtree as an outcome of the test condition of the root node attribute. Accordingly, the training dataset is also split into subsets.
6. Recursively apply the same operation for the subset of the training set with the remaining attributes until a leaf node is derived or no more training instances are available in the subset.

Note: *We stop branching a node if entropy is 0.*

The best split attribute at every iteration is the attribute with the highest information gain.

Definitions

Let T be the training dataset.

Let A be the set of attributes $A = \{A_1, A_2, A_3, \ldots\ldots A_n\}$.

Let m be the number of classes in the training dataset.

Let P_i be the probability that a data instance or a tuple 'd' belongs to class C_i.

It is calculated as,

P_i = Total no of data instances that belongs to class C_i in T/Total no of tuples in the training set T (6.6)

Mathematically, it is represented as shown in Eq. (6.7).

$$P_i = \frac{\left|d_{C_i}\right|}{\left|T\right|} \tag{6.7}$$

Expected information or Entropy needed to classify a data instance d' in T is denoted as Entropy_Info(T) given in Eq. (6.8).

$$\text{Entropy\_Info}(T) = -\sum_{i=1}^{m} P_i \log_2 P_i \tag{6.8}$$

Entropy of every attribute denoted as Entropy_Info(T, A) is shown in Eq. (6.9) as:

$$\text{Entropy\_Info}(T, A) = \sum_{i=1}^{v} \frac{\left|A_i\right|}{\left|T\right|} \times \text{Entropy\_Info}\ (A_i) \tag{6.9}$$

where, the attribute A has got 'v' distinct values $\{a_1, a_2, \ldots. a_v\}$, $\left|A_i\right|$ is the number of instances for distinct value 'i' in attribute A, and Entropy_Info (A_i) is the entropy for that set of instances.

Information_Gain is a metric that measures how much information is gained by branching on an attribute A. In other words, it measures the reduction in impurity in an arbitrary subset of data.

It is calculated as given in Eq. (6.10):

$$\text{Information\_Gain}(A) = \text{Entropy\_Info}(T) - \text{Entropy\_Info}(T, A) \tag{6.10}$$

It can be noted that as entropy increases, information gain decreases. They are inversely proportional to each other.

Scan for *'Additional Examples'*

Example 6.3: Assess a student's performance during his course of study and predict whether a student will get a job offer or not in his final year of the course. The training dataset T consists of 10 data instances with attributes such as 'CGPA', 'Interactiveness', 'Practical Knowledge' and 'Communication Skills' as shown in Table 6.3. The target class attribute is the 'Job Offer'.

Table 6.3: Training Dataset T

S.No.	CGPA	Interactiveness	Practical Knowledge	Communication Skills	Job Offer
1.	≥9	Yes	Very good	Good	Yes
2.	≥8	No	Good	Moderate	Yes
3.	≥9	No	Average	Poor	No
4.	<8	No	Average	Good	No
5.	≥8	Yes	Good	Moderate	Yes
6.	≥9	Yes	Good	Moderate	Yes
7.	<8	Yes	Good	Poor	No
8.	≥9	No	Very good	Good	Yes
9.	≥8	Yes	Good	Good	Yes
10.	≥8	Yes	Average	Good	Yes

Solution:

Step 1:

Calculate the Entropy for the target class 'Job Offer'.

Entropy_Info(Target Attribute = Job Offer) = Entropy_Info(7, 3) =

$$= -\left[\frac{7}{10}\log_2\frac{7}{10} + \frac{3}{10}\log_2\frac{3}{10}\right] = -(-0.3599 + -0.5208) = 0.8807$$

Iteration 1:

Step 2:

Calculate the Entropy_Info and Gain(Information_Gain) for each of the attribute in the training dataset.

Table 6.4 shows the number of data instances classified with Job Offer as Yes or No for the attribute CGPA.

Table 6.4: Entropy Information for CGPA

CGPA	Job Offer = Yes	Job Offer = No	Total	Entropy
≥9	3	1	4	
≥8	4	0	4	**0**
<8	0	2	2	**0**

Entropy_Info(T, CGPA)

$$= \frac{4}{10}\left[-\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4}\right] + \frac{4}{10}\left[-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}\right] + \frac{2}{10}\left[-\frac{0}{2}\log_2\frac{0}{2} - \frac{2}{2}\log_2\frac{2}{2}\right]$$

$$= \frac{4}{10}(0.3111 + 0.4997) + 0 + 0$$

$$= 0.3243$$

Gain (CGPA) = 0.8807 − 0.3243

= 0.5564

Table 6.5 shows the number of data instances classified with Job Offer as Yes or No for the attribute Interactiveness.

Table 6.5: Entropy Information for Interactiveness

Interactiveness	Job Offer = Yes	Job Offer = No	Total	Entropy
YES	5	1	6	
NO	2	2	4	

$$\text{Entropy\_Info}(T, \text{Interactiveness}) = \frac{6}{10}\left[-\frac{5}{6}\log_2\frac{5}{6} - \frac{1}{6}\log_2\frac{1}{6}\right] + \frac{4}{10}\left[-\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4}\right]$$

$$= \frac{6}{10}(0.2191 + 0.4306) + \frac{4}{10}(0.4997 + 0.4997)$$

$$= 0.3898 + 0.3998 = 0.7896$$

Gain(Interactiveness) = 0.8807 − 0.7896

= 0.0911

Table 6.6 shows the number of data instances classified with Job Offer as Yes or No for the attribute Practical Knowledge.

Table 6.6: Entropy Information for Practical Knowledge

Practical Knowledge	Job Offer = Yes	Job Offer = No	Total	Entropy
Very Good	2	0	2	**0**
Average	1	2	3	
Good	4	1	5	

Entropy_Info(T, Practical Knowledge)

$$= \frac{2}{10}\left[-\frac{2}{2}\log_2\frac{2}{2} - \frac{0}{2}\log_2\frac{0}{2}\right] + \frac{3}{10}\left[-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right] + \frac{5}{10}\left[-\frac{4}{5}\log_2\frac{4}{5} - \frac{1}{5}\log_2\frac{1}{5}\right]$$

$$= \frac{2}{10}(0) + \frac{3}{10}(0.5280 + 0.3897) + \frac{5}{10}(0.2574 + 0.4641)$$

$$= 0 + 0.2753 + 0.3608$$

$$= 0.6361$$

Gain(Practical Knowledge) = 0.8807 − 0.6361

= 0.2446

Table 6.7 shows the number of data instances classified with Job Offer as Yes or No for the attribute Communication Skills.

Table 6.7: Entropy Information for Communication Skills

Communication Skills	Job Offer = Yes	Job Offer = No	Total
Good	4	1	5
Moderate	3	0	3
Poor	0	2	2

Entropy_Info(T, Communication Skills)

$$= \frac{5}{10}\left[-\frac{4}{5}\log_2\frac{4}{5} - \frac{1}{5}\log_2\frac{1}{5}\right] + \frac{3}{10}\left[-\frac{3}{3}\log_2\frac{3}{3} - \frac{0}{3}\log_2\frac{0}{3}\right] + \frac{2}{10}\left[-\frac{0}{2}\log_2\frac{0}{2} - \frac{2}{2}\log_2\frac{2}{2}\right]$$

$$= \frac{5}{10}(0.5280 + 0.3897) + \frac{3}{10}(0) + \frac{2}{10}(0)$$

$$= 0.3609$$

Gain(Communication Skills) = 0.8813 − 0.36096

= 0.5203

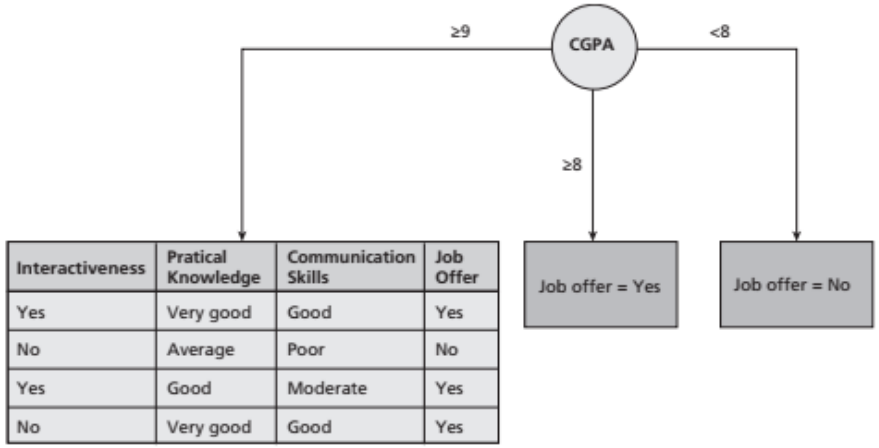
The Gain calculated for all the attributes is shown in Table 6.8:

Table 6.8: Gain

Attributes	Gain
CGPA	**0.5564**
Interactiveness	0.0911
Practical Knowledge	0.2246
Communication Skills	0.5203

Step 3: From Table 6.8, choose the attribute for which entropy is minimum and therefore the gain is maximum as the best split attribute.

The best split attribute is CGPA since it has the maximum gain. So, we choose CGPA as the root node. There are three distinct values for CGPA with outcomes ≥9, ≥8 and <8. The entropy value is 0 for ≥8 and <8 with all instances classified as Job Offer = Yes for ≥8 and Job Offer = No for <8. Hence, both ≥8 and <8 end up in a leaf node. The tree grows with the subset of instances with CGPA ≥9 as shown in Figure 6.3.



Interactiveness	Pratical Knowledge	Communication Skills	Job Offer
Yes	Very good	Good	Yes
No	Average	Poor	No
Yes	Good	Moderate	Yes
No	Very good	Good	Yes

Figure 6.3: Decision Tree After Iteration 1

Now, continue the same process for the subset of data instances branched with CGPA ≥ 9.

Iteration 2:

In this iteration, the same process of computing the Entropy_Info and Gain are repeated with the subset of training set. The subset consists of 4 data instances as shown in the above Figure 6.3.

Entropy_Info(T) = Entropy_Info(3, 1) =

$$= -\left[\frac{3}{4}\log_2\frac{3}{4} + \frac{1}{4}\log_2\frac{1}{4}\right]$$

$$= -(-0.3111 + -0.4997)$$

$$= 0.8108$$

$$\text{Entropy\_Info}(T, \text{Interactiveness}) = \frac{2}{4}\left[-\frac{2}{2}\log_2\frac{2}{2} - \frac{0}{2}\log_2\frac{0}{2}\right] + \frac{2}{4}\left[-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right]$$

$$= 0 + 0.4997$$

Gain(Interactiveness) = 0.8108 − 0.4997

= 0.3111

Entropy_Info(T, Practical Knowledge)

$$= \frac{2}{4}\left[-\frac{2}{2}\log_2\frac{2}{2} - \frac{0}{2}\log_2\frac{0}{2}\right] + \frac{1}{4}\left[-\frac{0}{1}\log_2\frac{0}{1} - \frac{1}{1}\log_2\frac{1}{1}\right] + \frac{1}{4}\left[-\frac{1}{1}\log_2\frac{1}{1} - \frac{0}{1}\log_2\frac{0}{1}\right]$$

= 0

Gain(Practical Knowledge) = 0.8108

Entropy_Info(T, Communication Skills)

$$= \frac{2}{4}\left[-\frac{2}{2}\log_2\frac{2}{2} - \frac{0}{2}\log_2\frac{0}{2}\right] + \frac{1}{4}\left[-\frac{0}{1}\log_2\frac{0}{1} - \frac{1}{1}\log_2\frac{1}{1}\right] + \frac{1}{4}\left[-\frac{1}{1}\log_2\frac{1}{1} - \frac{0}{1}\log_2\frac{0}{1}\right]$$

= 0

Gain(Communication Skills) = 0.8108

The gain calculated for all the attributes is shown in Table 6.9.

Table 6.9: Total Gain

Attributes	Gain
Interactiveness	0.3111
Practical Knowledge	**0.8108**
Communication Skills	**0.8108**

Here, both the attributes 'Practical Knowledge' and 'Communication Skills' have the same Gain. So, we can either construct the decision tree using 'Practical Knowledge' or 'Communication Skills'. The final decision tree is shown in Figure 6.4.

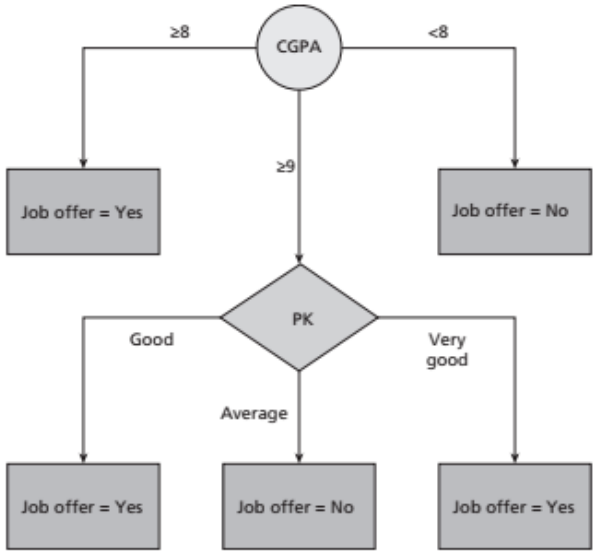


Figure 6.4: Final Decision Tree

6.2.2 C4.5 Construction

C4.5 is an improvement over ID3. C4.5 works with continuous and discrete attributes and missing values, and it also supports post-pruning. C5.0 is the successor of C4.5 and is more efficient and used for building smaller decision trees. C4.5 works with missing values by marking as '?', but these missing attribute values are not considered in the calculations.

The algorithm C4.5 is based on Occam's Razor which says that given two correct solutions, the simpler solution has to be chosen. Moreover, the algorithm requires a larger training set for better accuracy. It uses Gain Ratio as a measure during the construction of decision trees. ID3 is more biased towards attributes with larger values. For example, if there is an attribute called 'Register No' for students it would be unique for every student and will have distinct value for every data instance resulting in more values for the attribute. Hence, every instance belongs to a category and would have higher Information Gain than other attributes. To overcome this bias issue, C4.5 uses a purity measure Gain ratio to identify the best split attribute. In C4.5 algorithm, the Information Gain measure used in ID3 algorithm is normalized by computing another factor called Split_Info. This normalized information gain of an attribute called as Gain_Ratio is computed by the ratio of the calculated Split_Info and Information Gain of each attribute. Then, the attribute with the highest normalized information gain, that is, highest gain ratio is used as the splitting criteria.

As an example, we will choose the same training dataset shown in Table 6.3 to construct a decision tree using the C4.5 algorithm.

Given a Training dataset T,

The Split_Info of an attribute A is computed as given in Eq. (6.11):

$$\text{Split\_Info}(T, A) = -\sum_{i=1}^{v} \frac{|A_i|}{|T|} \times \log_2 \frac{|A_i|}{|T|} \tag{6.11}$$

where, the attribute A has got 'v' distinct values $\{a_1, a_2, \ldots a_v\}$, and $|A_i|$ is the number of instances for distinct value 'i' in attribute A.

The Gain_Ratio of an attribute A is computed as given in Eq. (6.12):

$$\text{Gain\_Ratio}(A) = \frac{\text{Info\_Gain}(A)}{\text{Split\_Info}(T, A)} \tag{6.12}$$

Algorithm 6.3: Procedure to Construct a Decision Tree using C4.5

1. Compute Entropy_Info Eq. (6.8) for the whole training dataset based on the target attribute.
2. Compute Entropy_Info Eq. (6.9), Info_Gain Eq. (6.10), Split_Info Eq. (6.11) and Gain_Ratio Eq. (6.12) for each of the attribute in the training dataset.
3. Choose the attribute for which Gain_Ratio is maximum as the best split attribute.
4. The best split attribute is placed as the root node.
5. The root node is branched into subtrees with each subtree as an outcome of the test condition of the root node attribute. Accordingly, the training dataset is also split into subsets.
6. Recursively apply the same operation for the subset of the training set with the remaining attributes until a leaf node is derived or no more training instances are available in the subset.

Example 6.4: Make use of Information Gain of the attributes which are calculated in ID3 algorithm in Example 6.3 to construct a decision tree using C4.5.

Solution:

Iteration 1:

Step 1: Calculate the Class_Entropy for the target class 'Job Offer'.

$$\text{Entropy\_Info(Target Attribute = Job Offer)} = \text{Entropy\_Info}(7, 3) = -\left[\frac{7}{10}\log_2\frac{7}{10} + \frac{3}{10}\log_2\frac{3}{10}\right]$$
$$= (-0.3599 + -0.5208)$$
$$= 0.8807$$

Step 2: Calculate the Entropy_Info, Gain(Info_Gain), Split_Info, Gain_Ratio for each of the attribute in the training dataset.

CGPA:

$$\text{Entropy Info}(T, \text{CGPA}) = \frac{4}{10}\left[-\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4}\right] + \frac{4}{10}\left[-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}\right] + \frac{2}{10}\left[-\frac{0}{2}\log_2\frac{0}{2} - \frac{2}{2}\log_2\frac{2}{2}\right]$$
$$= \frac{4}{10}(0.3111 + 0.4997) + 0 + 0$$
$$= 0.3243$$
$$\text{Gain(CGPA)} = 0.8807 - 0.3243$$
$$= 0.5564$$
$$\text{Split\_Info}(T, \text{CGPA}) = -\frac{4}{10}\log_2\frac{4}{10} - \frac{4}{10}\log_2\frac{4}{10} - \frac{2}{10}\log_2\frac{2}{10}$$
$$= 0.5285 + 0.5285 + 0.4641$$
$$= 1.5211$$
$$\text{Gain Ratio(CGPA)} = (\text{Gain(CGPA)})/(\text{Split\_Info}(T, \text{CGPA}))$$
$$= \frac{0.5564}{1.5211} = 0.3658$$

Interactiveness:

$$\text{Entropy Info}(T, \text{Interactiveness}) = \frac{6}{10}\left[-\frac{5}{6}\log_2\frac{5}{6} - \frac{1}{6}\log_2\frac{1}{6}\right] + \frac{4}{10}\left[-\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4}\right]$$
$$= \frac{6}{10}(0.2191 + 0.4306) + \frac{4}{10}(0.4997 + 0.4997)$$
$$= 0.3898 + 0.3998 = 0.7896$$
$$\text{Gain(Interactiveness)} = 0.8807 - 0.7896 = 0.0911$$
$$\text{Gain(Interactiveness)} = -\frac{6}{10}\log_2\frac{6}{10} - \frac{4}{10}\log_2\frac{4}{10} = 0.9704$$

$$\text{Gain\_Ratio(Interactiveness)} = \frac{\text{Gain(Interactiveness)}}{\text{Split\_Info}(T, \text{Interactiveness})} = \frac{0.0911}{0.9704} = 0.0939$$

Practical Knowledge:

$$\text{Entropy\_Info}(T, \text{Practical Knowledge}) = \frac{2}{10}\left[-\frac{2}{2}\log_2\frac{2}{2} - \frac{0}{2}\log_2\frac{0}{2}\right] + \frac{3}{10}\left[-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right] + \frac{5}{10}\left[-\frac{4}{5}\log_2\frac{4}{5} - \frac{1}{5}\log_2\frac{1}{5}\right]$$

$$= \frac{2}{10}(0) + \frac{3}{10}(0.5280 + 0.3897) + \frac{5}{10}(0.2574 + 0.4641)$$

$$= 0 + 0.2753 + 0.3608 = 0.6361$$

$$\text{Gain(Practical Knowledge)} = 0.8807 - 0.6361 = 0.2448$$

$$\text{Split\_Info}(T, \text{Practical Knowledge}) = -\frac{2}{10}\log_2\frac{2}{10} - \frac{5}{10}\log_2\frac{5}{10} - \frac{3}{10}\log_2\frac{3}{10} = 1.4853$$

$$\text{Gain\_Ratio(Practical Knowledge)} = \frac{\text{Gain(Practical Knowledge)}}{\text{Split\_Info}(T, \text{Practical Knowledge})} = \frac{0.2448}{1.4853} = 0.1648$$

Communication Skills:

$$\text{Entropy\_Info}(T, \text{Communication Skills}) = \frac{5}{10}\left[-\frac{4}{5}\log_2\frac{4}{5} - \frac{1}{5}\log_2\frac{1}{5}\right] + \frac{3}{10}\left[-\frac{3}{3}\log_2\frac{3}{3} - \frac{0}{3}\log_2\frac{0}{3}\right] + \frac{2}{10}\left[-\frac{0}{2}\log_2\frac{0}{2} - \frac{2}{2}\log_2\frac{2}{2}\right]$$

$$= \frac{5}{10}(0.5280 + 0.3897) + \frac{3}{10}(0) + \frac{2}{10}(0)$$

$$= 0.3609$$

$$\text{Gain(Communication Skills)} = 0.8813 - 0.36096 = 0.5202$$

$$\text{Split\_Info}(T, \text{Communication Skills}) = -\frac{5}{10}\log_2\frac{5}{10} - \frac{3}{10}\log_2\frac{3}{10} - \frac{2}{10}\log_2\frac{2}{10} = 1.4853$$

$$\text{Gain\_Ratio(Communication Skills)} = \frac{\text{Gain(Communication Skills)}}{\text{Split\_Info}(T, \text{Communication Skills})} = \frac{0.5202}{1.4853} = 0.3502$$

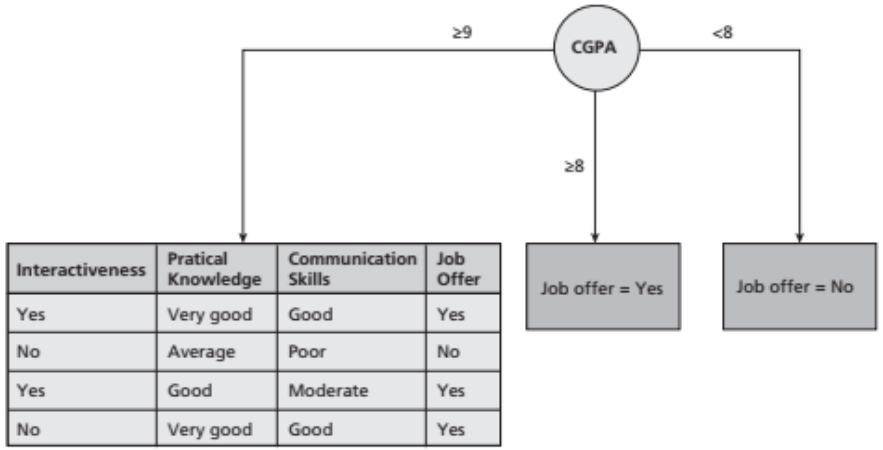
Table 6.10 shows the Gain_Ratio computed for all the attributes.

Table 6.10: Gain_Ratio

Attribute	Gain_Ratio
CGPA	**0.3658**
INTERACTIVENESS	0.0939
PRACTICAL KNOWLEDGE	0.1648
COMMUNICATION SKILLS	0.3502

Step 3: Choose the attribute for which Gain_Ratio is maximum as the best split attribute.

From Table 6.10, we can see that CGPA has highest gain ratio and it is selected as the best split attribute. We can construct the decision tree placing CGPA as the root node shown in Figure 6.5. The training dataset is split into subsets with 4 data instances.



Interactiveness	Pratical Knowledge	Communication Skills	Job Offer
Yes	Very good	Good	Yes
No	Average	Poor	No
Yes	Good	Moderate	Yes
No	Very good	Good	Yes

Figure 6.5: Decision Tree after Iteration 1

Iteration 2:

Total Samples: 4

Repeat the same process for this resultant dataset with 4 data instances.

Job Offer has 3 instances as Yes and 1 instance as No.

$$\text{Entropy\_Info(Target Class = Job Offer)} = -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = 0.3112 + 0.5 = 0.8112$$

Interactiveness:

$$\text{Entropy\_Info}(T, \text{Interactiveness}) = \frac{2}{4}\left[-\frac{2}{2}\log_2\frac{2}{2} - \frac{0}{2}\log_2\frac{0}{2}\right] + \frac{2}{4}\left[-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right] = 0 + 0.4997$$

$$\text{Gain(Interactiveness)} = 0.8108 - 0.4997 = 0.3111$$

$$\text{Split\_Info}(T, \text{Interactiveness}) = -\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4} = 0.5 + 0.5 = 1$$

$$\text{Gain\_Ratio(Interactiveness)} = \frac{\text{Gain(Interactiveness)}}{\text{Split\_Info}(T, \text{Interactiveness})} = \frac{0.3112}{1} = 0.3112$$

Practical Knowledge:

$$\text{Entropy\_Info}(T, \text{Practical Knowledge}) = \frac{2}{4}\left[-\frac{2}{2}\log_2\frac{2}{2} - \frac{0}{2}\log_2\frac{0}{2}\right] + \frac{1}{4}\left[-\frac{0}{1}\log_2\frac{0}{1} - \frac{1}{1}\log_2\frac{1}{1}\right] + \frac{1}{4}\left[-\frac{1}{1}\log_2\frac{1}{1} - \frac{0}{1}\log_2\frac{0}{1}\right] = 0$$

$$\text{Gain(Practical Knowledge)} = 0.8108$$

$$\text{Split\_Info}(T, \text{Practical Knowledge}) = -\frac{2}{4}\log_2\frac{2}{4} - \frac{1}{4}\log_2\frac{1}{4} - \frac{1}{4}\log_2\frac{1}{4} = 1.5$$

$$\text{Gain\_Ratio(Practical Knowledge)} = \frac{\text{Gain(Practical Knowledge)}}{\text{Split\_Info}(T, \text{Practical Knowledge})} = \frac{0.8108}{1.5} = 0.5408$$

Communication Skills:

$$\text{Entropy\_Info}(T, \text{Communication Skills}) = \frac{2}{4}\left[-\frac{2}{2}\log_2\frac{2}{2} - \frac{0}{2}\log_2\frac{0}{2}\right] + \frac{1}{4}\left[-\frac{0}{1}\log_2\frac{0}{1} - \frac{1}{1}\log_2\frac{1}{1}\right] + \frac{1}{4}\left[-\frac{1}{1}\log_2\frac{1}{1} - \frac{0}{1}\log_2\frac{0}{1}\right] = 0$$

$$\text{Gain(Communication Skills)} = 0.8108$$

$$\text{Split\_Info}(T, \text{Communication Skills}) = -\frac{2}{4}\log_2\frac{2}{4} - \frac{1}{4}\log_2\frac{1}{4} - \frac{1}{4}\log_2\frac{1}{4} = 1.5$$

$$\text{Gain\_Ratio(Communication Skills)} = \frac{\text{Gain(Practical Knowledge)}}{\text{Split\_Info}(T, \text{Practical Knowledge})} = \frac{0.8108}{1.5} = 0.5408$$

Table 6.11 shows the Gain_Ratio computed for all the attributes.

Table 6.11: Gain-Ratio

Attributes	Gain_Ratio
Interactiveness	0.3112
Practical Knowledge	0.5408
Communication Skills	0.5408

Both 'Practical Knowledge' and 'Communication Skills' have the highest gain ratio. So, the best splitting attribute can either be 'Practical Knowledge' or 'Communication Skills', and therefore, the split can be based on any one of these.

Here, we split based on 'Practical Knowledge'. The final decision tree is shown in Figure 6.6.

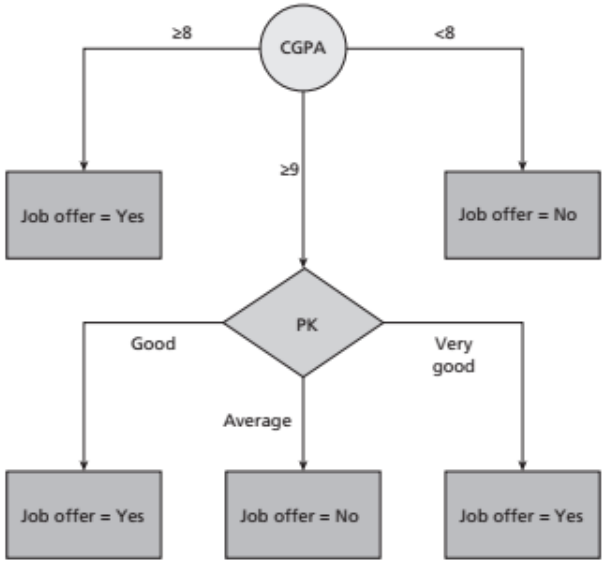


Figure 6.6: Final Decision Tree

Dealing with Continuous Attributes in C4.5

The C4.5 algorithm is further improved by considering attributes which are continuous, and a continuous attribute is discretized by finding a split point or threshold. When an attribute '*A*' has numerical values which are continuous, a threshold or best split point '*s*' is found such that the set of values is categorized into two sets such as $A < s$ and $A \geq s$. The best split point is the attribute value which has maximum information gain for that attribute.

Now, let us consider the set of continuous values for the attribute CGPA in the sample dataset as shown in Table 6.12.

Table 6.12: Sample Dataset

S.No.	CGPA	Job Offer
1.	9.5	Yes
2.	8.2	Yes
3.	9.1	No
4.	6.8	No
5.	8.5	Yes
6.	9.5	Yes
7.	7.9	No
8.	9.1	Yes
9.	8.8	Yes
10.	8.8	Yes

First, sort the values in an ascending order.

6.8	7.9	8.2	8.5	8.8	8.8	9.1	9.1	9.5	9.5

Remove the duplicates and consider only the unique values of the attribute.

6.8	7.9	8.2	8.5	8.8	8.8	9.1	9.5

Now, compute the Gain for the distinct values of this continuous attribute. Table 6.13 shows the computed values.

Table 6.13: Gain Values for CGPA

	6.8		7.9		8.2		8.5		8.8		9.1		9.5	
Range	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>
Yes	0	7	0	7	1	6	2	5	4	3	5	2	7	0
No	1	2	2	1	2	1	2	1	2	1	3	0	3	0
Entropy	0	0.7637	0	0.5433	0.9177	0.5913	1	0.6497	0.9177	0.8108	0.9538	0	0.8808	0
Entropy_Info (*S*, *T*)	0.6873		0.4346		0.6892		0.7898		0.8749		0.7630		0.8808	
Gain	0.1935		**0.4462**		0.1916		0.091		0.0059		0.1178		0	

For a sample, the calculations are shown below for a single distinct value say, CGPA ∈ 6.8.

$$\text{Entropy\_Info}(T, \text{Job\_Offer}) = -\left[\frac{7}{10}\log_2\frac{7}{10} + \frac{3}{10}\log_2\frac{3}{10}\right]$$
$$= -(-0.3599 + -0.5209)$$
$$= 0.8808$$
$$\text{Entropy}(7, 2) = -\left[\frac{7}{9}\log_2\frac{7}{9} + \frac{2}{9}\log_2\frac{2}{9}\right]$$
$$= -(-0.2818 + -0.4819)$$
$$= 0.7637$$
$$\text{Entropy\_Info}(T, \text{CGPA} \in 6.8) = \frac{1}{10} \times \text{Entropy}(0, 1) + \frac{9}{10}\text{Entropy}(7, 2)$$
$$= \frac{1}{10}\left[-\frac{0}{1}\log_2\frac{0}{1} - \frac{1}{1}\log_2\frac{1}{1}\right] + \frac{9}{10}\left[-\frac{7}{9}\log_2\frac{7}{9} - \frac{2}{9}\log_2\frac{2}{9}\right]$$
$$= 0 + \frac{9}{10}(0.7637)$$
$$= 0.6873$$
$$\text{Gain}(\text{CGPA} \in 6.8) = 0.8808 - 0.6873$$
$$= 0.1935$$

Similarly, the calculations are done for each of the distinct value for the attribute CGPA and a table is created. Now, the value of CGPA with maximum gain is chosen as the threshold value or the best split point. From Table 6.13, we can observe that CGPA with 7.9 has the maximum gain as 0.4462. Hence, CGPA ∈ 7.9 is chosen as the split point. Now, we can dicretize the continuous values of CGPA as two categories with CGPA ≤ 7.9 and CGPA > 7.9. The resulting discretized instances are shown in Table 6.14.

Table 6.14: Discretized Instances

S.No.	CGPA Continuous	CGPA Discretized	Job Offer
1.	9.5	>7.9	Yes
2.	8.2	>7.9	Yes
3.	9.1	>7.9	No
4.	6.8	≤7.9	No
5.	8.5	>7.9	Yes
6.	9.5	>7.9	Yes
7.	7.9	≤7.9	No
8.	9.1	>7.9	Yes
9.	8.8	>7.9	Yes
10.	8.8	>7.9	Yes

6.2.3 Classification and Regression Trees Construction

The Classification and Regression Trees (CART) algorithm is a multivariate decision tree learning used for classifying both categorical and continuous-valued target variables. CART algorithm is an example of multivariate decision trees that gives oblique splits. It solves both classification and regression problems. If the target feature is categorical, it constructs a classification tree and if the target feature is continuous, it constructs a regression tree. CART uses GINI Index to construct a decision tree. GINI Index is defined as the number of data instances for a class or it is the proportion of instances. It constructs the tree as a binary tree by recursively splitting a node into two nodes. Therefore, even if an attribute has more than two possible values, GINI Index is calculated for all subsets of the attributes and the subset which has maximum value is selected as the best split subset. For example, if an attribute A has three distinct values say $\{a_1, a_2, a_3\}$, the possible subsets are $\{\}$, $\{a_1\}$, $\{a_2\}$, $\{a_3\}$, $\{a_1, a_2\}$, $\{a_1, a_3\}$, $\{a_2, a_3\}$, and $\{a_1, a_2, a_3\}$. So, if an attribute has 3 distinct values, the number of possible subsets is 2^3, which means 8. Excluding the empty set $\{\}$ and the full set $\{a_1, a_2, a_3\}$, we have 6 subsets. With 6 subsets, we can form three possible combinations such as:

$\{a_1\}$ with $\{a_2, a_3\}$

$\{a_2\}$ with $\{a_1, a_3\}$

$\{a_3\}$ with $\{a_1, a_2\}$

Hence, in this CART algorithm, we need to compute the best splitting attribute and the best split subset i in the chosen attribute.

Higher the GINI value, higher is the homogeneity of the data instances.

Gini_Index(T) is computed as given in Eq. (6.13).

$$\text{Gini\_Index}(T) = 1 - \sum_{i=1}^{m} P_i^2 \tag{6.13}$$

where,

P_i be the probability that a data instance or a tuple 'd' belongs to class C_i. It is computed as:

P_i = |No. of data instances belonging to class i|/|Total no of data instances in the training dataset T|

GINI Index assumes a binary split on each attribute, therefore, every attribute is considered as a binary attribute which splits the data instances into two subsets S_1 and S_2.

Gini_Index(T, A) is computed as given in Eq. (6.14).

$$\text{Gini\_Index}(T, A) = \frac{|S_1|}{|T|}\text{Gini}(S_1) + \frac{|S_2|}{|T|}\text{Gini}(S_2) \quad (6.14)$$

The splitting subset with minimum Gini_Index is chosen as the best splitting subset for an attribute. The best splitting attribute is chosen by the minimum Gini_Index which is otherwise maximum ΔGini because it reduces the impurity.

ΔGini is computed as given in Eq. (6.15):

$$\Delta\text{Gini}(A) = \text{Gini}(T) - \text{Gini}(T, A) \quad (6.15)$$

Algorithm 6.4: Procedure to Construct a Decision Tree using CART

1. Compute Gini_Index Eq. (6.13) for the whole training dataset based on the target attribute.
2. Compute Gini_Index for each of the attribute Eq. (6.14) and for the subsets of each attribute in the training dataset.
3. Choose the best splitting subset which has minimum Gini_Index for an attribute.
4. Compute ΔGini Eq. (6.15) for the best splitting subset of that attribute.
5. Choose the best splitting attribute that has maximum ΔGini.
6. The best split attribute with the best split subset is placed as the root node.
7. The root node is branched into two subtrees with each subtree an outcome of the test condition of the root node attribute. Accordingly, the training dataset is also split into two subsets.
8. Recursively apply the same operation for the subset of the training set with the remaining attributes until a leaf node is derived or no more training instances are available in the subset.

Example 6.5: Choose the same training dataset shown in Table 6.3 and construct a decision tree using CART algorithm.

Solution:

Step 1: Calculate the Gini_Index for the dataset shown in Table 6.3, which consists of 10 data instances. The target attribute 'Job Offer' has 7 instances as Yes and 3 instances as No.

$$\text{Gini\_Index}(T) = 1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2$$
$$= 1 - 0.49 - 0.09$$
$$= 1 - 0.58$$
$$\text{Gini\_Index}(T) = 0.42$$

Step 2: Compute Gini_Index for each of the attribute and each of the subset in the attribute.

CGPA has 3 categories, so there are 6 subsets and hence 3 combinations of subsets (as shown in Table 6.15).

Table 6.15: Categories of CGPA

CGPA	Job Offer = Yes	Job Offer = No
≥9	3	1
≥8	4	0
<8	0	2

$$\text{Gini\_Index}(T, \text{CGPA} \in \{\geq 9, \geq 8\}) = 1 - (7/8)^2 - (1/8)^2$$
$$= 1 - 0.7806$$
$$= 0.2194$$
$$\text{Gini\_Index}(T, \text{CGPA} \in \{<8\}) = 1 - (0/2)^2 - (2/2)^2$$
$$= 1 - 1$$
$$= 0$$
$$\text{Gini\_Index}(T, \text{CGPA} \in \{(\geq 9, \geq 8), <8\} = (8/10) \times 0.2194 + (2/10) \times 0$$
$$= 0.17552$$
$$\text{Gini\_Index}(T, \text{CGPA} \in \{\geq 9, <8\}) = 1 - (3/6)^2 - (3/6)^2$$
$$= 1 - 0.5 = 0.5$$
$$\text{Gini\_Index}(T, \text{CGPA} \in \{\geq 8\}) = 1 - (4/4)^2 - (0/4)^2$$
$$= 1 - 1 = 0$$
$$\text{Gini\_Index}(T, \text{CGPA} \in \{(\geq 9, <8), \geq 8\}) = (6/10) \times 0.5 + (4/10) \times 0$$
$$= 0.3$$
$$\text{Gini\_Index}(T, \text{CGPA} \in \{\geq 8, <8\}) = 1 - (4/6)^2 - (2/6)^2$$
$$= 1 - 0.555$$
$$= 0.445$$
$$\text{Gini\_Index}(T, \text{CGPA} \in \{\geq 9\}) = 1 - (3/4)^2 - (1/4)^2$$
$$= 1 - 0.625$$
$$= 0.375$$
$$\text{Gini\_Index}(T, \text{CGPA} \in \{(\geq 8, <8), \geq 9\}) = (6/10) \times 0.445 + (4/10) \times 0.375$$
$$= 0.417$$

Table 6.16 shows the Gini_Index for 3 subsets of CGPA.

Table 6.16: Gini_Index of CGPA

Subsets		Gini_Index
(≥9, ≥8)	<8	0.1755
(≥9, <8)	≥8	0.3
(≥8, <8)	≥9	0.417

Step 3: Choose the best splitting subset which has minimum Gini_Index for an attribute.

The subset CGPA ∈ {(≥9 ≥8), <8} has the lowest Gini_Index value as 0.1755 is chosen as the best splitting subset.

Step 4: Compute ΔGini or the best splitting subset of that attribute.

$$\Delta\text{Gini(CGPA)} = \text{Gini}(T) - \text{Gini}(T, \text{CGPA})$$
$$= 0.42 - 0.1755$$
$$= 0.2445$$

Repeat the same process for the remaining attributes in the dataset such as for Interactiveness shown in Table 6.17, Practical Knowledge in Table 6.18, and Communication Skills in Table 6.20.

Table 6.17: Categories for Interactiveness

Interactiveness	Job Offer = Yes	Job Offer = No
Yes	5	1
No	2	2

$$\text{Gini\_Index}(T, \text{Interactiveness} \in \{\text{Yes}\}) = 1 - \left(\frac{5}{6}\right)^2 - \left(\frac{1}{6}\right)^2$$
$$= 1 - 0.72$$
$$= 0.28$$

$$\text{Gini\_Index}(T, \text{Interactiveness} \in \{\text{No}\}) = 1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2$$
$$= 1 - 0.5$$
$$= 0.5$$

$$\text{Gini\_Index}(T, \text{Interactiveness} \in \{\text{Yes, No}\}) = \frac{6}{10}(0.28) + \frac{4}{10}(0.5)$$
$$= 0.168 + 0.2$$
$$= 0.368$$

$$\Delta\text{Gini(Interactiveness)} = \text{Gini}(T) - \text{Gini}(T, \text{Interactiveness})$$
$$= 0.42 - 0.368$$
$$= 0.052$$

Table 6.18: Categories for Practical Knowledge

Practical Knowledge	Job Offer = Yes	Job Offer = No
Very Good	2	0
Good	4	1
Average	1	2

$$\text{Gini\_Index}(T, \text{Practical Knowledge} \in \{\text{Very Good, Good}\} = \left(\frac{6}{7}\right)^2 - \left(\frac{1}{7}\right)^2$$
$$= 1 - 0.7544$$
$$= 0.2456$$

$$\text{Gini\_Index}(T, \text{Practical Knowledge} \in \{\text{Average}\}) = 1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2$$
$$= 1 - 0.555 = 0.445$$

$$\text{Gini\_Index}(T, \text{Practical Knowledge} \in \{\text{Very Good, Good}\}, \text{Average}) = \left(\frac{7}{10}\right)^2 \times 0.2456 + \left(\frac{3}{10}\right) \times 0.445$$
$$= 0.3054$$

$$\text{Gini\_Index}(T, \text{Practical Knowledge} \in \{\text{Very Good, Average}\}) = 1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2$$
$$= 1 - 0.52$$
$$= 0.48$$

$$\text{Gini\_Index}(T, \text{Practical Knowledge} \in \{\text{Good}\}) = 1 - \left(\frac{4}{5}\right)^2 - \left(\frac{1}{5}\right)^2$$
$$= 1 - 0.68$$
$$= 0.32$$

$$\text{Gini\_Index}(T, \text{Practical Knowledge} \in \{\text{Very Good, Average}\}, \text{Good}) = \left(\frac{5}{10}\right) \times 0.48 + \left(\frac{5}{10}\right) \times 0.32$$
$$= 0.40$$

$$\text{Gini\_Index}(T, \text{Practical Knowledge} \in \{\text{Very Good, Average}\}) = 1 - \left(\frac{5}{8}\right)^2 - \left(\frac{3}{8}\right)^2$$
$$= 1 - 0.5312 = 0.4688$$

$$\text{Gini\_Index}(T, \text{Practical Knowledge} \in \{\text{Very Good}\}) = 1 - \left(\frac{2}{2}\right)^2 - \left(\frac{0}{2}\right)^2$$
$$= 1 - 1 = 0$$

$$\text{Gini\_Index}(T, \text{Practical Knowledge} \in \{\text{Good, Average}\}, \text{Very Good}) = \left(\frac{8}{10}\right) \times 0.4688 + \left(\frac{2}{10}\right) \times 0$$
$$= 0.3750$$

Table 6.19 shows the Gini_Index for various subsets of Practical Knowledge.

Table 6.19: Gini_Index for Practical Knowledge

Subsets		Gini_Index
(Very Good, Good)	Average	0.3054
(Very Good, Average)	Good	0.40
(Good, Average)	Very Good	0.3750

$$\Delta\text{Gini(Practical Knowledge)} = \text{Gini}(T) - \text{Gini}(T, \text{Practical Knowledge})$$
$$= 0.42 - 0.3054 = 0.1146$$

Table 6.20: Categories for Communication Skills

Communication Skills	Job Offer = Yes	Job Offer = No
Good	4	1
Moderate	3	0
Poor	0	2

$$\text{Gini\_Index}(T, \text{Communication Skills} \in \{\text{Good, Moderate}\}) = 1 - \left(\frac{7}{8}\right)^2 - \left(\frac{1}{8}\right)^2$$
$$= 1 - 0.7806$$
$$= 0.2194$$

$$\text{Gini\_Index}(T, \text{Communication Skills} \in \{\text{Poor}\}) = 1 - \left(\frac{2}{2}\right)^2 - \left(\frac{0}{2}\right)^2$$
$$= 1 - 1 = 0$$

$$\text{Gini\_Index}(T, \text{Communication Skills} \in \{\text{Good, Moderate}\}, \text{Poor}) = \left(\frac{8}{10}\right) \times 0.2194 + \left(\frac{2}{10}\right) \times 0$$
$$= 0.1755$$

$$\text{Gini\_Index}(T, \text{Communication Skills} \in \{\text{Good, Poor}\}) = 1 - \left(\frac{4}{7}\right)^2 - \left(\frac{3}{7}\right)^2$$
$$= 1 - 0.5101$$
$$= 0.4899$$

$$\text{Gini\_Index}(T, \text{Communication Skills} \in \{\text{Moderate}\}) = 1 - \left(\frac{3}{3}\right)^2 - \left(\frac{0}{3}\right)^2$$
$$= 1 - 1 = 0$$

$$\text{Gini\_Index}(T, \text{Communication Skills} \in \{\text{Good, Poor}\}, \text{Moderate}) = \left(\frac{7}{10}\right) \times 0.4899 + \left(\frac{3}{10}\right) \times 0$$
$$= 0.3429$$

$$\text{Gini\_Index}(T, \text{Communication Skills} \in \{\text{Moderate, Poor}\}) = 1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2$$
$$= 1 - 0.52$$
$$= 0.48$$

$$\text{Gini\_Index}(T, \text{Communication Skills} \in \{\text{Good}\}) = 1 - \left(\frac{4}{5}\right)^2 - \left(\frac{1}{5}\right)^2$$
$$= 1 - 0.68$$
$$= 0.32$$

$$\text{Gini\_Index}(T, \text{Communication Skills} \in \{\text{Moderate, Poor}\}, \text{Good}) = \left(\frac{5}{10}\right)^2 \times 048 + \left(\frac{5}{10}\right)^2 \times 0.32$$
$$= 0.40$$

Table 6.21 shows the Gini_Index for various subsets of Communication Skills.

Table 6.21: Gini-Index for Subsets of Communication Skills

Subsets		Gini_Index
(Good, Moderate)	Poor	0.1755
(Good, Poor)	Moderate	0.3429
(Moderate, Poor)	Good	0.40

$$\Delta\text{Gini}(\text{Communication Skills}) = \text{Gini}(T) - \text{Gini}(T, \text{Communication Skills})$$
$$= 0.42 - 0.1755$$
$$= 0.2445$$

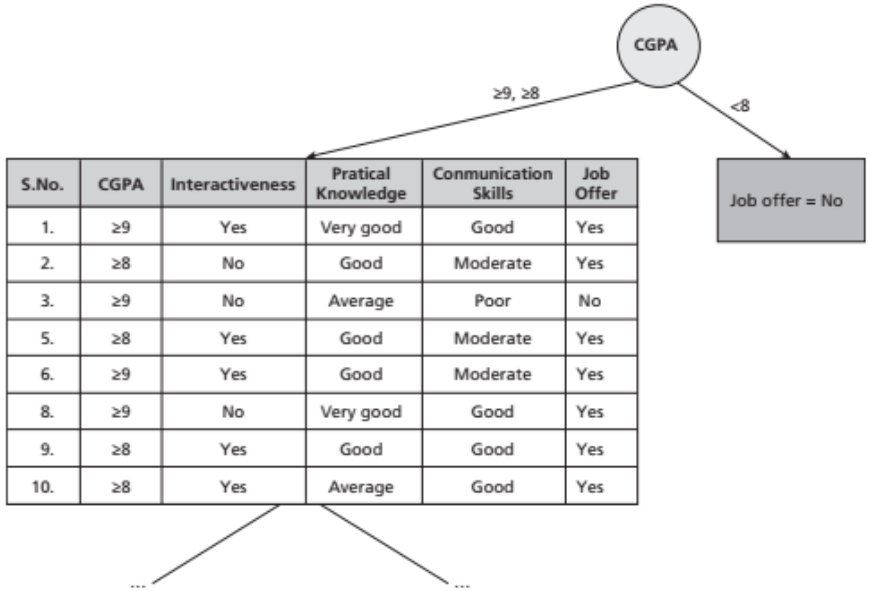
Table 6.22 shows the Gini_Index and ΔGini values calculated for all the attributes.

Table 6.22: Gini_Index and ΔGini for all Attributes

Attribute	Gini_Index	ΔGini
CGPA	0.1755	0.2445
Interactiveness	0.368	0.052
Practical knowledge	0.3054	0.1146
Communication Skills	0.1755	0.2445

Step 5: Choose the best splitting attribute that has maximum ΔGini.

CGPA and Communication Skills have the highest ΔGini value. We can choose CGPA as the root node and split the datasets into two subsets shown in Figure 6.7 since the tree constructed by CART is a binary tree.



S.No.	CGPA	Interactiveness	Pratical Knowledge	Conmunication Skills	Job Offer
1.	≥9	Yes	Very good	Good	Yes
2.	≥8	No	Good	Moderate	Yes
3.	≥9	No	Average	Poor	No
5.	≥8	Yes	Good	Moderate	Yes
6.	≥9	Yes	Good	Moderate	Yes
8.	≥9	No	Very good	Good	Yes
9.	≥8	Yes	Good	Good	Yes
10.	≥8	Yes	Average	Good	Yes

Figure 6.7: Decision Tree after Iteration 1

Iteration 2:

In the second iteration, the dataset has 8 data instances as shown in Table 6.23. Repeat the same process to find the best splitting attribute and the splitting subset for that attribute.

Table 6.23: Subset of the Training Dataset after Iteration 1

S.No.	CGPA	Interactiveness	Practical Knowledge	Communication Skills	Job Offer
1.	≥9	Yes	Very good	Good	Yes
2.	≥8	No	Good	Moderate	Yes
3.	≥9	No	Average	Poor	No
5.	≥8	Yes	Good	Moderate	Yes
6.	≥9	Yes	Good	Moderate	Yes
8.	≥9	No	Very good	Good	Yes
9.	≥8	Yes	Good	Good	Yes
10.	≥8	Yes	Average	Good	Yes

$$\text{Gini\_Index}(T) = 1 - \left(\frac{7}{8}\right)^2 - \left(\frac{1}{8}\right)^2$$
$$= 1 - 0.766 - 0.0156$$
$$= 1 - 0.58$$
$$\text{Gini\_Index}(T) = 0.2184$$

Tables 6.24, 6.25, and 6.27 show the categories for attributes Interactiveness, Practical Knowledge, and Communication Skills, respectively.

Table 6.24: Categories for Interactiveness

Interactiveness	Job Offer = Yes	Job Offer = No
Yes	5	0
No	2	1

$$\text{Gini\_Index}(T, \text{Interactiveness} \in \{\text{Yes}\}) = 1 - \left(\frac{5}{5}\right)^2 - \left(\frac{0}{5}\right)^2$$
$$= 1 - 1 = 0$$
$$\text{Gini\_Index}(T, \text{Interactiveness} \in \{\text{No}\}) = 1 - \left(\frac{2}{3}\right)^2 - \left(\frac{1}{3}\right)^2$$
$$= 1 - 0.44 - 0.111 = 0.449$$
$$\text{Gini\_Index}(T, \text{Interactiveness} \in \{\text{Yes, No}\}) = \left(\frac{7}{8}\right) \times 0 + \left(\frac{1}{8}\right) \times 0.449$$
$$= 0.056$$
$$\Delta\text{Gini(Interactiveness)} = \text{Gini}(T) - \text{Gini}(T, \text{Interactiveness})$$
$$= 0.2184 - 0.056 = 0.1624$$

Table 6.25: Categories for Practical Knowledge

Practical Knowledge	Job Offer = Yes	Job Offer = No
Very Good	2	0
Good	4	0
Average	1	1

$$\text{Gini\_Index}(T, \text{Practical Knowledge} \in \{\text{Very Good, Good}\}) = 1 - \left(\frac{6}{6}\right)^2 - \left(\frac{0}{6}\right)^2$$
$$= 1 - 1 = 0$$
$$\text{Gini\_Index}(T, \text{Practical Knowledge} \in \{\text{Average}\}) = 1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2$$
$$= 1 - 0.25 - 0.25$$
$$= 0.5$$
$$\text{Gini\_Index}(T, \text{Practical Knowledge} \in \{\text{Very Good, Good}\}, \text{Average}) = \left(\frac{6}{8}\right)^2 \times 0 + \left(\frac{2}{8}\right) \times 0.5$$
$$= 0.125$$
$$\text{Gini\_Index}(T, \text{Practical Knowledge} \in \{\text{Very Good, Average}\}) = 1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2$$
$$= 1 - 0.5625 - 0.0625$$
$$= 0.375$$
$$\text{Gini\_Index}(T, \text{Practical Knowledge} \in \{\text{Good}\}) = 1 - \left(\frac{4}{4}\right)^2 - \left(\frac{0}{4}\right)^2$$
$$= 1 - 1 = 0$$
$$\text{Gini\_Index}(T, \text{Practical Knowledge} \in \{\text{Very Good, Average}\}, \text{Good}) = \left(\frac{4}{8}\right) \times 0.375 + \left(\frac{4}{8}\right) \times 0$$
$$= 0.1875$$
$$\text{Gini\_Index}(T, \text{Practical Knowledge} \in \{\text{Good, Average}\}) = 1 - \left(\frac{5}{6}\right)^2 - \left(\frac{1}{6}\right)^2$$
$$= 1 - 0.694 - 0.028$$
$$= 0.278$$
$$\text{Gini\_Index}(T, \text{Practical Knowledge} \in \{\text{Very Good}\}) = 1 - \left(\frac{2}{2}\right)^2 - \left(\frac{0}{2}\right)^2$$
$$= 1 - 1 = 0$$
$$\text{Gini\_Index}(T, \text{Practical Knowledge} \in \{\text{Good, Average}\}, \text{Very Good}) = \left(\frac{6}{8}\right)^2 \times 0.278 + \left(\frac{2}{8}\right)^2 \times 0$$
$$= 0.2085$$

Table 6.26 shows the Gini_Index values for various subsets of Practical Knowledge.

Table 6.26: Gini_Index for Subsets of Practical Knowledge

Subsets		Gini_Index
(Very Good, Good)	Average	0.125
(Very Good, Average)	Good	0.1875
(Good, Average)	Very Good	0.2085

$$\Delta\text{Gini}(\text{Practical Knowledge}) = \text{Gini}(T) - \text{Gini}(T, \text{Practical Knowledge})$$
$$= 0.2184 - 0.125$$
$$= 0.0934$$

Table 6.27: Categories for Communication Skills

Communication Skills	Job Offer = Yes	Job Offer = No
Good	4	0
Moderate	3	0
Poor	0	1

$$\text{Gini\_Index}(T, \text{Communication Skills} \in \{\text{Good, Moderate}\}) = 1 - \left(\frac{7}{7}\right)^2 - \left(\frac{0}{7}\right)^2$$
$$= 1 - 1 = 0$$

$$\text{Gini\_Index}(T, \text{Communication Skills} \in \{\text{Poor}\}) = 1 - \left(\frac{0}{1}\right)^2 - \left(\frac{1}{1}\right)^2$$
$$= 1 - 1 = 0$$

$$\text{Gini\_Index}(T, \text{Communication Skills} \in \{\text{Good, Moderate}\}, \text{Poor}) = \left(\frac{7}{8}\right)^2 \times 0 + \left(\frac{1}{8}\right) \times 0$$
$$= 0$$

$$\text{Gini\_Index}(T, \text{Communication Skills} \in \{\text{Good, Poor}\}) = 1 - \left(\frac{4}{5}\right)^2 - \left(\frac{1}{5}\right)^2$$
$$= 1 - 0.64 - 0.04$$
$$= 0.32$$

$$\text{Gini\_Index}(T, \text{Communication Skills} \in \{\text{Moderate}\}) = 1 - \left(\frac{3}{3}\right)^2 - \left(\frac{0}{3}\right)^2$$
$$= 1 - 1 = 0$$

$$\text{Gini\_Index}(T, \text{Communication Skills} \in \{\text{Good, Poor}\}, \text{Moderate}) = \left(\frac{5}{8}\right) \times 0.32 + \left(\frac{3}{8}\right) \times 0$$
$$= 0.2$$

$$\text{Gini\_Index}(T, \text{Communication Skills} \in \{\text{Moderate, Poor}\}) = 1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2$$
$$= 1 - 0.5625 - 0.0625$$
$$= 0.375$$

$$\text{Gini\_Index}(T, \text{Communication Skills} \in \{\text{Good}\}) = 1 - \left(\frac{4}{4}\right)^2 - \left(\frac{0}{4}\right)^2$$
$$= 1 - 1 = 0$$

$$\text{Gini\_Index}(T, \text{Communication Skills} \in \{\text{Moderate, Poor}\}, \text{Good}) = \left(\frac{4}{8}\right)^2 \times 0.375 + \left(\frac{4}{8}\right)^2 \times 0$$
$$= 0.1875$$

Table 6.28 shows the Gini_Index for subsets of Communication Skills.

Table 6.28: Gini_Index for Subsets of Communication Skills

Subsets		Gini_Index
(Good, Moderate)	Poor	0
(Good, Poor)	Moderate	0.2
(Moderate, Poor)	Good	0.1875

$$\Delta\text{Gini}(\text{Communication Skills}) = \text{Gini}(T) - \text{Gini}(T, \text{Communication Skills})$$
$$= 0.2184 - 0 = 0.2184$$

Table 6.29 shows the Gini_Index and ΔGini values for all attributes.

Table 6.29: Gini_Index and ΔGini Values for All Attributes

Attribute	Gini_Index	ΔGini
Interactiveness	0.056	0.1624
Practical knowledge	0.125	0.0934
Communication Skills	0	0.2184

Communication Skills has the highest ΔGini value. The tree is further branched based on the attribute 'Communication Skills'. Here, we see all branches end up in a leaf node and the process of construction is completed. The final tree is shown in Figure 6.8.

Figure 6.8: Final Tree

6.2.4 Regression Trees

Regression trees are a variant of decision trees where the target feature is a continuous valued variable. These trees can be constructed using an algorithm called reduction in variance which uses standard deviation to choose the best splitting attribute.

Algorithm 6.5: Procedure for Constructing Regression Trees

1. Compute standard deviation for each attribute with respect to target attribute.
2. Compute standard deviation for the number of data instances of each distinct value of an attribute.
3. Compute weighted standard deviation for each attribute.

(*Continued*)

4. Compute standard deviation reduction by subtracting weighted standard deviation for each attribute from standard deviation of each attribute.
5. Choose the attribute with a higher standard deviation reduction as the best split attribute.
6. The best split attribute is placed as the root node.
7. The root node is branched into subtrees with each subtree as an outcome of the test condition of the root node attribute. Accordingly, the training dataset is also split into different subsets.
8. Recursively apply the same operation for the subset of the training set with the remaining attributes until a leaf node is derived or no more training instances are available in the subset.

Example 6.6: Construct a regression tree using the following Table 6.30 which consists of 10 data instances and 3 attributes 'Assessment', 'Assignment' and 'Project'. The target attribute is the 'Result' which is a continuous attribute.

Table 6.30: Training Dataset

S.No.	Assessment	Assignment	Project	Result (%)
1.	Good	Yes	Yes	95
2.	Average	Yes	No	70
3.	Good	No	Yes	75
4.	Poor	No	No	45
5.	Good	Yes	Yes	98
6.	Average	No	Yes	80
7.	Good	No	No	75
8.	Poor	Yes	Yes	65
9.	Average	No	No	58
10.	Good	Yes	Yes	89

Solution:

Step 1: Compute standard deviation for each attribute with respect to the target attribute:

$$\text{Average} = (95 + 70 + 75 + 45 + 98 + 80 + 75 + 65 + 58 + 89) = 75$$

$$\text{Standard Deviation} = \sqrt{\frac{(95-75)^2 + (70-75)^2 + (75-75)^2 + (45-75)^2 + (98-75)^2 + (80-75)^2 + (75-75)^2 + (65-75)^2 + (58-75)^2 + (89-75)^2}{10}}$$

$$= 16.55$$

Assessment = Good (Table 6.31)

Table 6.31: Attribute Assessment = Good

S.No.	Assessment	Assignment	Project	Result (%)
1.	Good	Yes	Yes	95
3.	Good	No	Yes	75
5.	Good	Yes	Yes	98
7.	Good	No	No	75
10.	Good	Yes	Yes	89

$$\text{Average} = (95 + 75 + 98 + 75 + 89) = 86.4$$

$$\text{Standard Deviation} = \sqrt{\frac{(95-86.4)^2 + (75-86.4)^2 + (98-86.4)^2 + (75-86.4)^2 + (89-86.4)^2}{5}}$$

$$= 10.9$$

Assessment = Average (Table 6.32)

Table 6.32: Attribute Assessment = Average

S.No.	Assessment	Assignment	Project	Result (%)
2.	Average	Yes	No	70
6.	Average	No	Yes	80
9.	Average	No	No	58

$$\text{Average} = (70 + 80 + 58) = 69.3$$

$$\text{Standard Deviation} = \sqrt{\frac{(70-69.3)^2 + (80-69.3)^2 + (58-69.3)^2}{3}} = 11.01$$

Assessment = Poor (Table 6.33)

Table 6.33: Attribute Assessment = Poor

S.No.	Assessment	Assignment	Project	Result (%)
4.	Poor	No	No	45
8.	Poor	Yes	Yes	65

$$\text{Average} = (45 + 65) = 55$$

$$\text{Standard Deviation} = \sqrt{\frac{(45-55)^2 + (65-55)^2}{2}} = 14.14$$

Table 6.34 shows the standard deviation and data instances for the attribute-Assessment.

Table 6.34: Standard Deviation for Assessment

Assessment	Standard Deviation	Data Instances
Good	10.9	5
Average	11.01	3
Poor	14.14	2

$$\text{Weighted standard deviation for Assessment} = \left(\frac{5}{10}\right) \times 10.9 + \left(\frac{3}{10}\right) \times 11.01 + \left(\frac{2}{10}\right) \times 14.14$$

$$= 11.58$$

Standard deviation reduction for Assessment = 16.55 – 11.58 = 4.97

Assignment = Yes (Table 6.35)

Table 6.35: Assignment = Yes

S.No.	Assessment	Assignment	Project	Result (%)
1.	Good	Yes	Yes	95
2.	Average	Yes	No	70
5.	Good	Yes	Yes	98
8.	Poor	Yes	Yes	65
10.	Good	Yes	Yes	89

Average = (95 + 70 + 98 + 65 + 89) = 83.4

$$\text{Standard Deviation} = \sqrt{\frac{(95-83.4)^2 + (70-83.4)^2 + (98-83.4)^2 + (65-83.4)^2 + (89-83.4)^2}{5}}$$

$$= 14.98$$

Assignment = No (Table 6.36)

Table 6.36: Assignment = No

S.No.	Assessment	Assignment	Project	Result (%)
3.	Good	No	Yes	75
4.	Poor	No	No	45
6.	Average	No	Yes	80
7.	Good	No	No	75
9.	Average	No	No	58

Average = (75 + 45 + 80 + 75 + 58) = 66.6

$$\text{Standard Deviation} = \sqrt{\frac{(75-66.6)^2 + (45-66.6)^2 + (80-66.6)^2 + (75-66.6)^2 + (58-66.6)^2}{5}}$$

$$= 14.7$$

Table 6.37 shows the Standard Deviation and Data Instances for attribute, Assignment.

Table 6.37: Standard Deviation for Assignment

Assessment	Standard Deviation	Data Instances
Yes	14.98	5
No	14.7	5

Weighted standard deviation for Assignment = $\left(\frac{5}{10}\right) \times 14.98 + \left(\frac{5}{10}\right) \times 14.7 = 14.84$

Standard deviation reduction for Assignment = 16.55 – 14.84 = 1.71

Project = Yes (Table 6.38)

Table 6.38: Project = Yes

S.No.	Assessment	Assignment	Project	Result (%)
1.	Good	Yes	Yes	95
3.	Good	No	Yes	75
5.	Good	Yes	Yes	98
6.	Average	No	Yes	80
8.	Poor	Yes	Yes	65
10.	Good	Yes	Yes	89

Average = (95 + 75 + 98 + 80 + 65 + 89) = 83.7

$$\text{Standard Deviation} = \sqrt{\frac{(95-83.7)^2 + (75-83.7)^2 + (98-83.7)^2 + (80-83.7)^2 + (65-83.7)^2 + (89-83.7)^2}{6}}$$

$$= 12.6$$

Project = No (Table 6.39)

Table 6.39: Project = No

S.No.	Assessment	Assignment	Project	Result (%)
2.	Average	Yes	No	70
4.	Poor	No	No	45
7.	Good	No	No	75
9.	Average	No	No	58

Average = (70 + 45 + 75 + 58) = 62

$$\text{Standard Deviation} = \sqrt{\frac{(70-75)^2 + (45-75)^2 + (75-75)^2 + (58-75)^2}{4}}$$

$$= 13.39$$

Table 6.40 shows the Standard Deviation and Data Instances for attribute, Project.

Table 6.40: Standard Deviation for Project

Project	Standard Deviation	Data Instances
Yes	12.6	6
No	13.39	4

Weighted standard deviation for Assessment = $\left(\frac{6}{10}\right) \times 2.6 + \left(\frac{4}{10}\right) \times 13.39 = 12.92$

Standard deviation reduction for Assessment = 16.55 – 12.92 = 3.63

Table 6.41 shows the standard deviation reduction for each attribute in the training dataset.

Table 6.41: Standard Deviation Reduction for Each Attribute

Attributes	Standard Deviation Reduction
Assessment	4.97
Assignment	1.71
Project	3.63

The attribute 'Assessment' has the maximum Standard Deviation Reduction and hence it is chosen as the best splitting attribute.

The training dataset is split into subsets based on the attribute 'Assessment' and this process is continued until the entire tree is constructed. Figure 6.9 shows the regression tree with 'Assessment' as the root node and the subsets in each branch.

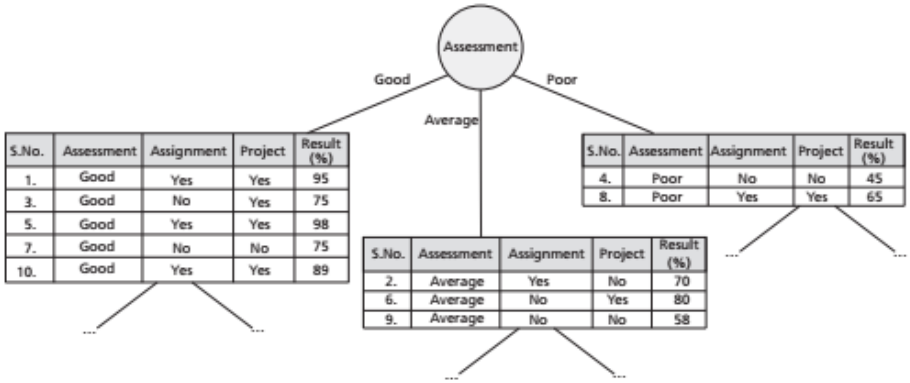


Figure 6.9: Regression Tree with Assessment as Root Node

The rest of regression tree construction can be done as an exercise.

6.3 VALIDATING AND PRUNING OF DECISION TREES

Inductive bias refers to a set of assumptions about the domain knowledge added to the training data to perform induction that is to construct a general model out of the training data. A bias is generally required as without it induction is not possible, since the training data can normally be generalized to a larger hypothesis space. Inductive bias in ID3 algorithm is the one that prefers the first acceptable shorter trees over larger trees, and when selecting the best split attribute during construction, attributes with high information gain are chosen. Thus, even though *ID*3 searches a large space of decision trees, it constructs only a single decision tree when there may exist many alternate decision trees for the same training data. It applies a hill-climbing search that does not backtrack and may finally converge to a locally optimal solution that is not globally optimal. The shorter tree is preferred using Occam's razor principle which states that the simplest solution is the best solution.

Overfitting is also a general problem with decision trees. Once the decision tree is constructed, it must be validated for better accuracy and to avoid over-fitting and under-fitting. There is always a tradeoff between accuracy and complexity of the tree. The tree must be simple and accurate. If the tree is more complex, it can classify the data instances accurately for the training set but when test data is given, the tree constructed may perform poorly which means misclassifications are higher and accuracy is reduced. This problem is called as over-fitting.

To avoid overfitting of the tree, we need to prune the trees and construct an optimal decision tree. Trees can be pre-pruned or post-pruned. If tree nodes are pruned during construction or the construction is stopped earlier without exploring the nodes' branches, then it is called as pre-pruning whereas if tree nodes are pruned after the construction is over then it is called as post-pruning. Basically, the dataset is split into three sets called training dataset, validation dataset and test dataset. Generally, 40% of the dataset is used for training the decision tree and the remaining 60% is used for validation and testing. Once the decision tree is constructed, it is validated with the validation dataset and the misclassifications are identified. Using the number of

instances correctly classified and number of instances wrongly classified, Average Squared Error (ASE) is computed. The tree nodes are pruned based on these computations and the resulting tree is validated until we get a tree that performs better. Cross validation is another way to construct an optimal decision tree. Here, the dataset is split into k-folds, among which $k-1$ folds are used for training the decision tree and the k^{th} fold is used for validation and errors are computed. The process is repeated for randomly $k-1$ folds and the mean of the errors is computed for different trees. The tree with the lowest error is chosen with which the performance of the tree is improved. This tree can now be tested with the test dataset and predictions are made.

Another approach is that after the tree is constructed using the training set, statistical tests like error estimation and Chi-square test are used to estimate whether pruning or splitting is required for a particular node to find a better accurate tree.

The third approach is using a principle called Minimum Description Length which uses a complexity measure for encoding the training set and the growth of the decision tree is stopped when the encoding size (i.e., (size(tree)) + size(misclassifications(tree)) is minimized. CART and C4.5 perform post-pruning, that is, pruning the tree to a smaller size after construction in order to minimize the misclassification error. CART makes use of 10-fold cross validation method to validate and prune the trees, whereas C4.5 uses heuristic formula to estimate misclassification error rates.

Some of the tree pruning methods are listed below:

1. Reduced Error Pruning
2. Minimum Error Pruning (MEP)
3. Pessimistic Pruning
4. Error–based Pruning (EBP)
5. Optimal Pruning
6. Minimum Description Length (MDL) Pruning
7. Minimum Message Length Pruning
8. Critical Value Pruning

Chapter 8

Bayesian Learning

"In science, progress is possible. In fact, if one believes in Bayes' theorem, scientific progress is inevitable as predictions are made and as beliefs are tested and refined."

— **Nate Silver**

Bayesian Learning is a learning method that describes and represents knowledge in an uncertain domain and provides a way to reason about this knowledge using probability measure. It uses Bayes theorem to infer the unknown parameters of a model. Bayesian inference is useful in many applications which involve reasoning and diagnosis such as game theory, medicine, etc. Bayesian inference is much more powerful in handling missing data and for estimating any uncertainty in predictions.

Learning Objectives

- Understand the basics of probability-based learning and probability theory
- Learn the fundamentals of Bayes theorem
- Introduce Bayes Classification models such as Brute Force Bayes learning algorithm, Bayes Optimal classifier, and Gibbs algorithm
- Introduce Naïve Bayes Classification models that work on the principle of Bayes theorem
- Explore the Naïve Bayes classification algorithm
- Study about Naïve Bayes Algorithm for continuous attributes using Gaussian distribution
- Introduce other popular types of Naive Bayes classifiers such as Bernoulli Naive Bayes classifier, Multinomial Naive Bayes classifier, and Multi-class Naïve Bayes classifier

8.1 INTRODUCTION TO PROBABILITY-BASED LEARNING

Probability-based learning is one of the most important practical learning methods which combines prior knowledge or prior probabilities with observed data. Probabilistic learning uses the concept of probability theory that describes how to model randomness, uncertainty, and noise to predict future events. It is a tool for modelling large datasets and uses Bayes rule to infer unknown quantities, predict and learn from data. In a *probabilistic model*, randomness plays a major role which gives probability distribution a solution, while in a *deterministic model* there is no randomness and

hence it exhibits the same initial conditions every time the model is run and is likely to get a single possible outcome as the solution.

Bayesian learning differs from probabilistic learning as it uses subjective probabilities (i.e., probability that is based on an individual's belief or interpretation about the outcome of an event and it can change over time) to infer parameters of a model. Two practical learning algorithms called Naïve Bayes learning and Bayesian Belief Network (BBN) form the major part of Bayesian learning. These algorithms use prior probabilities and apply Bayes rule to infer useful information. Bayesian Belief Networks (BBN) is explained in detail in Chapter 9.

Scan for information on *'Probability Theory'* and for *'Additional Examples'*

8.2 FUNDAMENTALS OF BAYES THEOREM

Naïve Bayes Model relies on Bayes theorem that works on the principle of three kinds of probabilities called prior probability, likelihood probability, and posterior probability.

Prior Probability

It is the general probability of an uncertain event before an observation is seen or some evidence is collected. It is the initial probability that is believed before any new information is collected.

Likelihood Probability

Likelihood probability is the relative probability of the observation occurring for each class or the sampling density for the evidence given the hypothesis. It is stated as P (Evidence | Hypothesis), which denotes the likeliness of the occurrence of the evidence given the parameters.

Posterior Probability

It is the updated or revised probability of an event taking into account the observations from the training data. P (Hypothesis | Evidence) is the posterior distribution representing the belief about the hypothesis, given the evidence from the training data. Therefore,

Posterior probability = prior probability + new evidence

8.3 CLASSIFICATION USING BAYES MODEL

Naïve Bayes Classification models work on the principle of Bayes theorem. Bayes' rule is a mathematical formula used to determine the posterior probability, given prior probabilities of events. Generally, Bayes theorem is used to select the most probable hypothesis from data, considering both prior knowledge and posterior distributions. It is based on the calculation of the posterior probability and is stated as:

P (Hypothesis h | Evidence E)

where, Hypothesis h is the target class to be classified and Evidence E is the given test instance.

P (Hypothesis h | Evidence E) is calculated from the prior probability P (Hypothesis h), the likelihood probability P (Evidence E | Hypothesis h) and the marginal probability P (Evidence E). It can be written as:

$$P(\text{Hypothesis } h \mid \text{Evidence } E) = \frac{P(\text{Evidence } E \mid \text{Hypothesis } h)\, P(\text{Hypothesis } h)}{P(\text{Evidence } E)} \quad (8.1)$$

where, P (Hypothesis h) is the prior probability of the hypothesis h without observing the training data or considering any evidence. It denotes the prior belief or the initial probability that the hypothesis h is correct. P (Evidence E) is the prior probability of the evidence E from the training dataset without any knowledge of which hypothesis holds. It is also called the marginal probability.

P (Evidence E | Hypothesis h) is the prior probability of Evidence E given Hypothesis h. It is the likelihood probability of the Evidence E after observing the training data that the hypothesis h is correct. P (Hypothesis h | Evidence E) is the posterior probability of Hypothesis h given Evidence E. It is the probability of the hypothesis h after observing the training data that the evidence E is correct. In other words, by the equation of Bayes Eq. (8.1), one can observe that:

Posterior Probability α Prior Probability × Likelihood Probability

Bayes theorem helps in calculating the posterior probability for a number of hypotheses, from which the hypothesis with the highest probability can be selected.

This selection of the most probable hypothesis from a set of hypotheses is formally defined as Maximum A Posteriori (MAP) Hypothesis.

Maximum A Posteriori (MAP) Hypothesis, h_{MAP}

Given a set of candidate hypotheses, the hypothesis which has the maximum value is considered as the *maximum probable hypothesis* or *most probable hypothesis*. This most probable hypothesis is called the Maximum A Posteriori Hypothesis h_{MAP}. Bayes theorem Eq. (8.1) can be used to find the h_{MAP}.

$$\begin{aligned} h_{MAP} &= \max_{h \in H} P(Hypothesish \mid Evidence\ E) \\ &= \max_{h \in H} \frac{P(Evidence\ E \mid Hypothesis\ h) P(Hypothesis\ h)}{P(Evidence\ E)} \\ &= \max_{h \in H} P(Evidence\ E \mid Hypothesis\ h) P(Hypothesis\ h) \end{aligned} \quad (8.2)$$

Maximum Likelihood (ML) Hypothesis, h_{ML}

Given a set of candidate hypotheses, if every hypothesis is equally probable, only P (E | h) is used to find the *most probable hypothesis*. The hypothesis that gives the maximum likelihood for P (E | h) is called the Maximum Likelihood (ML) Hypothesis, h_{ML}.

$$h_{ML} = \max_{h \in H} P(Evidence\ E \mid Hypothesis\ h) \quad (8.3)$$

Correctness of Bayes Theorem

Consider two events A and B in a sample space S.

A T F T T F T T F

B F T T F T F T F

P (A) = 5/8

P (B) = 4/8

P (A | B) = 2/4

P (B | A) = 2/5

P (A | B) = P (B | A) P (A) / P (B) = = 2/4

P (B | A) = P (A | B) P (B) / P (A) = = 2/5

Let us consider a numerical example to illustrate the use of Bayes theorem now:

Example 8.1: Consider a boy who has a volleyball tournament on the next day, but today he feels sick. It is unusual that there is only a 40% chance he would fall sick since he is a healthy boy. Now, Find the probability of the boy participating in the tournament. The boy is very much interested in volley ball, so there is a 90% probability that he would participate in tournaments and 20% that he will fall sick given that he participates in the tournament.

Solution: P (Boy participating in the tournament) = 90%

P (He is sick | Boy participating in the tournament) = 20%

P (He is Sick) = 40%

The probability of the boy participating in the tournament given that he is sick is:

P (Boy participating in the tournament | He is sick) = P (Boy participating in the tournament) × P (He is sick | Boy participating in the tournament)/P (He is Sick)

P (Boy participating in the tournament | He is sick) = (0.9 × 0.2)/0.4

= 0.45

Hence, 45% is the probability that the boy will participate in the tournament given that he is sick.

One related concept of Bayes theorem is the principle of Minimum Description Length (MDL). The minimum description length (MDL) principle is yet another powerful method like Occam's razor principle to perform inductive inference. It states that the best and most probable hypothesis is chosen for a set of observed data or the one with the minimum description. Recall from Eq. (8.2) Maximum A Posteriori (MAP) Hypothesis, h_{MAP} which says that given a set of candidate hypotheses, the hypothesis which has the maximum value is considered as the *maximum probable hypothesis* or *most probable hypothesis*. Naïve Bayes algorithm uses the Bayes theorem and applies this MDL principle to find the best hypothesis for a given problem. Let us clearly understand how this algorithm works in the following Section 8.3.1.

8.3.1 NAÏVE BAYES ALGORITHM

It is a supervised binary class or multi class classification algorithm that works on the principle of Bayes theorem. There is a family of Naïve Bayes classifiers based on a common principle. These algorithms classify for datasets whose features are independent and each feature is assumed to be given equal weightage. It particularly works for a large dataset and is very fast. It is one of the most effective and simple classification algorithms. This algorithm considers all features to be independent of each other even though they are individually dependent on the classified object. Each of the features contributes a probability value independently during classification and hence this algorithm is called as Naïve algorithm.

Some important applications of these algorithms are text classification, recommendation system and face recognition.

Algorithm 8.1: Naïve Bayes

1. Compute the prior probability for the target class.
2. Compute Frequency matrix and likelihood Probability for each of the feature.
3. Use Bayes theorem Eq. (8.1) to calculate the probability of all hypotheses.
4. Use Maximum A Posteriori (MAP) Hypothesis, h_{MAP} Eq. (8.2) to classify the test object to the hypothesis with the highest probability.

Example 8.2: Assess a student's performance using Naïve Bayes algorithm with the dataset provided in Table 8.1. Predict whether a student gets a job offer or not in his final year of the course.

Table 8.1: Training Dataset

S.No.	CGPA	Interactiveness	Practical Knowledge	Communication Skills	Job Offer
1.	≥9	Yes	Very good	Good	Yes
2.	≥8	No	Good	Moderate	Yes
3.	≥9	No	Average	Poor	No
4.	<8	No	Average	Good	No
5.	≥8	Yes	Good	Moderate	Yes
6.	≥9	Yes	Good	Moderate	Yes
7.	<8	Yes	Good	Poor	No
8.	≥9	No	Very good	Good	Yes
9.	≥8	Yes	Good	Good	Yes
10.	≥8	Yes	Average	Good	Yes

Solution: The training dataset T consists of 10 data instances with attributes such as 'CGPA', 'Interactiveness', 'Practical Knowledge' and 'Communication Skills' as shown in Table 8.1. The target variable is Job Offer which is classified as Yes or No for a candidate student.

Step 1: Compute the prior probability for the target feature 'Job Offer'. The target feature 'Job Offer' has two classes, 'Yes' and 'No'. It is a binary classification problem. Given a student instance, we need to classify whether 'Job Offer = Yes' or 'Job Offer = No'.

From the training dataset, we observe that the frequency or the number of instances with 'Job Offer = Yes' is 7 and 'Job Offer = No' is 3.

The prior probability for the target feature is calculated by dividing the number of instances belonging to a particular target class by the total number of instances.

Hence, the prior probability for 'Job Offer = Yes' is 7/10 and 'Job Offer = No' is 3/10 as shown in Table 8.2.

Table 8.2: Frequency Matrix and Prior Probability of Job Offer

Job Offer Classes	No. of Instances	Probability Value
Yes	7	*P* (Job Offer = Yes) = 7/10
No	3	*P* (Job Offer = No) = 3/10

Step 2: Compute Frequency matrix and Likelihood Probability for each of the feature.

Step 2(a): Feature – CGPA

Table 8.3 shows the frequency matrix for the feature CGPA.

Table 8.3: Frequency Matrix of CGPA

CGPA	Job Offer = Yes	Job Offer = No
≥9	3	1
≥8	4	0
<8	0	2
Total	7	3

Table 8.4 shows how the likelihood probability is calculated for CGPA using conditional probability.

Table 8.4: Likelihood Probability of CGPA

CGPA	*P* (Job Offer = Yes)	*P* (Job Offer = No)
≥9	*P* (CGPA ≥9 | Job Offer = Yes) = 3/7	*P* (CGPA ≥9 | Job Offer = No) = 1/3
≥8	*P* (CGPA ≥8 | Job Offer = Yes) = 4/7	*P* (CGPA ≥8 | Job Offer = No) = 0/3
<8	*P* (CGPA <8 | Job Offer = Yes) = 0/7	*P* (CGPA <8 | Job Offer = No) = 2/3

As explained earlier the Likelihood probability is stated as the sampling density for the evidence given the hypothesis. It is denoted as *P* (Evidence | Hypothesis), which says how likely is the occurrence of the evidence given the parameters.

It is calculated as the number of instances of each attribute value and for a given class value divided by the number of instances with that class value.

For example *P* (CGPA ≥9 | Job Offer = Yes) denotes the number of instances with 'CGPA ≥9' and 'Job Offer = Yes' divided by the total number of instances with 'Job Offer = Yes'.

From the Table 8.3 Frequency Matrix of CGPA, number of instances with 'CGPA ≥9' and 'Job Offer = Yes' is 3. The total number of instances with 'Job Offer = Yes' is 7. Hence, *P* (CGPA ≥9 | Job Offer = Yes) = 3/7.

Similarly, the Likelihood probability is calculated for all attribute values of feature CGPA.

Step 2(b): Feature – Interactiveness

Table 8.5 shows the frequency matrix for the feature Interactiveness.

Table 8.5: Frequency Matrix of Interactiveness

Interactiveness	Job Offer = Yes	Job Offer = No
YES	5	1
NO	2	2
Total	7	3

Table 8.6 shows how the likelihood probability is calculated for Interactiveness using conditional probability.

Table 8.6: Likelihood Probability of Interactiveness

Interactiveness	P (Job Offer = Yes)	P (Job Offer = No)
YES	*P* (Interactiveness = Yes ⏐ Job Offer = Yes) = 5/7	*P* (Interactiveness = Yes ⏐ Job Offer = No) = 1/3
NO	*P* (Interactiveness = No ⏐ Job Offer = Yes) = 2/7	*P* (Interactiveness = No ⏐ Job Offer = No) = 2/3

Step 2(c): Feature – Practical Knowledge

Table 8.7 shows the frequency matrix for the feature Practical Knowledge.

Table 8.7: Frequency Matrix of Practical Knowledge

Practical Knowledge	Job Offer = Yes	Job Offer = No
Very Good	2	0
Average	1	2
Good	4	1
Total	7	3

Table 8.8 shows how the likelihood probability is calculated for Practical Knowledge using conditional probability.

Table 8.8: Likelihood Probability of Practical Knowledge

Practical Knowledge	*P* (Job Offer = Yes)	*P* (Job Offer = No)
Very Good	*P* (Practical Knowledge = Very Good ⏐ Job Offer = Yes) = 2/7	*P* (Practical Knowledge = Very Good ⏐ Job Offer = No) = 0/3
Average	*P* (Practical Knowledge = Average ⏐ Job Offer = Yes) = 1/7	*P* (Practical Knowledge = Average ⏐ Job Offer = No) = 2/3
Good	*P* (Practical Knowledge = Good ⏐ Job Offer = Yes) = 4/7	*P* (Practical Knowledge = Good ⏐ Job Offer = No) = 1/3

Step 2(d): Feature – Communication Skills

Table 8.9 shows the frequency matrix for the feature Communication Skills.

Table 8.9: Frequency Matrix of Communication Skills

Communication Skills	Job Offer = Yes	Job Offer = No
Good	4	1
Moderate	3	0
Poor	0	2
Total	7	3

Table 8.10 shows how the likelihood probability is calculated for Communication Skills using conditional probability.

Table 8.10: Likelihood Probability of Communication Skills

Communication Skills	*P* (Job Offer = Yes)	*P* (Job Offer = No)
Good	*P* (Communication Skills = Good ⏐ Job Offer = Yes) = 4/7	*P* (Communication Skills = Good ⏐ Job Offer = No) = 1/3
Moderate	*P* (Communication Skills = Moderate ⏐ Job Offer = Yes) = 3/7	*P* (Communication Skills = Moderate ⏐ Job Offer = No) = 0/3
Poor	*P* (Communication Skills = Poor ⏐ Job Offer = Yes) = 0/7	*P* (Communication Skills = Poor ⏐ Job Offer = No) = 2/3

Step 3: Use Bayes theorem Eq. (8.1) to calculate the probability of all hypotheses.

Given the test data = (CGPA ≥9, Interactiveness = Yes, Practical knowledge = Average, Communication Skills = Good), apply the Bayes theorem to classify whether the given student gets a Job offer or not.

P (Job Offer = Yes ⏐ Test data) = (*P*(CGPA ≥9 ⏐ Job Offer = Yes) *P* (Interactiveness = Yes ⏐ Job Offer = Yes) *P* (Practical knowledge = Average ⏐ Job Offer = Yes) *P* (Communication Skills = Good ⏐ Job Offer = Yes) *P* (Job Offer = Yes)))/(*P* (Test Data))

We can ignore *P* (Test Data) in the denominator since it is common for all cases to be considered.

Hence, *P* (Job Offer = Yes ⏐ Test data) = (*P*(CGPA ≥9 ⏐Job Offer = Yes) *P* (Interactiveness = Yes ⏐ Job Offer = Yes) *P* (Practical knowledge = Average ⏐ Job Offer = Yes) *P* (Communication Skills = Good ⏐ Job Offer = Yes) *P* (Job Offer = Yes)

= 3/7 × 5/7 × 1/7 × 4/7 × 7/10

= 0.0175

Similarly, for the other case 'Job Offer = No',

We compute the probability,

P (Job Offer = No⏐ Test data) = (*P*(CGPA ≥9 ⏐Job Offer =No) *P* (Interactiveness = Yes ⏐ Job Offer = No) *P* (Practical knowledge = Average ⏐ Job Offer = No) *P* (Communication Skills = Good ⏐ Job Offer =No) *P* (Job Offer = No))/(*P*(Test Data)).

P (CGPA ≥9 ⏐Job Offer = No) *P* (Interactiveness = Yes ⏐ Job Offer = No) *P* (Practical knowledge = Average ⏐ Job Offer = No) *P* (Communication Skills = Good ⏐ Job Offer = No) *P* (Job Offer = No)

= 1/3 × 1/3 × 2/3 × 1/3 × 3/10

= 0.0074

Step 4: Use Maximum A Posteriori (MAP) Hypothesis, h_{MAP} Eq. (8.2) to classify the test object to the hypothesis with the highest probability.

Since *P* (Job Offer = Yes ⏐ Test data) has the highest probability value, the test data is classified as **'Job Offer = Yes'**.

Zero Probability Error

In Example 8.1, consider the test data to be (CGPA ≥8, Interactiveness = Yes, Practical knowledge = Average, Communication Skills = Good)

When computing the posterior probability,

P (Job Offer = Yes ⏐ Test data) = (*P*(CGPA ≥8 ⏐Job Offer = Yes) *P* (Interactiveness = Yes ⏐ Job Offer = Yes) *P* (Practical knowledge = Average ⏐ Job Offer = Yes) *P* (Communication Skills = Good ⏐ Job Offer = Yes) *P* (Job Offer = Yes)))/((P(Test Data))

P (Job Offer = Yes | Test data) = (P(CGPA ≥8 | Job Offer = Yes) P(Interactiveness = Yes | Job Offer = Yes) P (Practical knowledge = Average | Job Offer = Yes) P (Communication Skills = Good | Job Offer = Yes) P (Job Offer = Yes)

= 4/7 × 5/7 × 1/7 × 4/7 × 7/10

= **0.0233**

Similarly, for the other case 'Job Offer = No',

When we compute the probability:

P (Job Offer = No | Test data) = (P(CGPA ≥8 | Job Offer = No) P (Interactiveness = Yes | Job Offer = No) P (Practical knowledge = Average | Job Offer = No) P (Communication Skills = Good | Job Offer = No) P (Job Offer = No))/(P(Test Data))

= P (CGPA ≥8 | Job Offer =No) P (Interactiveness = Yes | Job Offer = No) P (Practical knowledge = Average | Job Offer = No) P (Communication Skills = Good | Job Offer =No) P (Job Offer = No)

= 0/3 × 1/3 × 2/3 × 1/3 × 3/10

= **0**

Since the probability value is zero, the model fails to predict, and this is called as Zero-Probability error. This problem arises because there are no instances in the given Table 8.1 for the attribute value CGPA ≥8 and Job Offer = No and hence the probability value of this case is zero. This zero-probability error can be solved by applying a smoothing technique called Laplace correction which means given 1000 data instances in the training dataset, if there are zero instances for a particular value of a feature we can add 1 instance for each attribute value pair of that feature which will not make much difference for 1000 data instances and the overall probability does not become zero.

Now, let us scale the values given in Table 8.1 for 1000 data instances. The scaled values without Laplace correction are shown in Table 8.11.

Table 8.11: Scaled Values to 1000 without Laplace Correction

CGPA	P (Job Offer = Yes)	P (Job Offer = No)
≥9	P (CGPA ≥9 | Job Offer = Yes) = 300/700	P (CGPA ≥9 | Job Offer = No) = 100/300
≥8	P (CGPA ≥8 | Job Offer = Yes) = 400/700	P (CGPA ≥8 | Job Offer = No) = 0/300
<8	P (CGPA <8 | Job Offer = Yes) = 0/700	P (CGPA <8 | Job Offer = No) = 200/300

Now, add 1 instance for each CGPA-value pair for 'Job Offer = No'. Then,

P (CGPA ≥9 | Job Offer = No) = 101/303 = 0.333

P (CGPA ≥8 | Job Offer = No) = 1/303 = 0.0033

P (CGPA <8 | Job Offer = No) = 201/303 = 0.6634

With scaled values to 1003 data instances, we get

P (Job Offer = Yes | Test data) = (P(CGPA ≥8 | Job Offer = Yes) P (Interactiveness = Yes | Job Offer = Yes) P (Practical knowledge = Average | Job Offer = Yes) P (Communication Skills = Good | Job Offer = Yes) P (Job Offer = Yes)

= 400/700 × 500/700 × 100/700 × 400/700 × 700/1003

= **0.02325**

P (Job Offer = No | Test data) = P(CGPA ≥8 | Job Offer = No) P (Interactiveness = Yes | Job Offer = No) P (Practical knowledge = Average | Job Offer = No) P (Communication Skills = Good | Job Offer = No) P (Job Offer = No)

= 1/303 × 100/300 × 200/300 × 100/300 × 303/1003

= **0.00007385**

Thus, using Laplace Correction, Zero Probability error can be solved with Naïve Bayes classifier.

8.3.2 Brute Force Bayes Algorithm

Applying Bayes theorem, Brute Force Bayes algorithm relies on the idea of concept learning wherein given a hypothesis space H for the training dataset T, the algorithm computes the posterior probabilities for all the hypothesis $h_i \in H$. Then, Maximum A Posteriori (MAP) Hypothesis, h_{MAP}, is used to output the hypothesis with maximum posterior probability. The algorithm is quite expensive since it requires computations for all the hypotheses. Although computing posterior probabilities is inefficient, this idea is applied in various other algorithms which is also quite interesting.

8.3.3 Bayes Optimal Classifier

Bayes optimal classifier is a probabilistic model, which in fact, uses the Bayes theorem to find the most probable classification for a new instance given the training data by combining the predictions of all posterior hypotheses, This is different from Maximum A Posteriori (MAP) Hypothesis, h_{MAP} which chooses the maximum probable hypothesis or the most probable hypothesis.

Here, a new instance can be classified to a possible classification value C_i by the following Eq. (8.4).

$$= \max_{C_i} \sum_{h_i \in H} P(C_i \mid h_i)\, P(h_i \mid T) \tag{8.4}$$

Example 8.3: Given the hypothesis space with 4 hypothesis h_1, h_2, h_3 and h_4. Determine if the patient is diagnosed as COVID positive or COVID negative using Bayes Optimal classifier.

Solution: From the training dataset T, the posterior probabilities of the four different hypotheses for a new instance are given in Table 8.12.

Table 8.12: Posterior Probability Values

$P(h_i \mid T)$	P (COVID Positive | h_i)	P (COVID Negative | h_i)
0.3	0	1
0.1	1	0
0.2	1	0
0.1	1	0

h_{MAP} chooses h_1 which has the maximum probability value 0.3 as the solution and gives the result that the patient is COVID negative. But Bayes Optimal classifier combines the predictions of h_2, h_3 and h_4 which is 0.4 and gives the result that the patient is COVID positive.

$$\sum_{h_i \in H} P(\text{COVID Negative} \mid h_i) P(h_i \mid T) = 0.3 \times 1 = 0.3$$

$$\sum_{h_i \in H} P(\text{COVID Positive} \mid h_i) P(h_i \mid T) = 0.1 \times 1 + 0.2 \times 1 + 0.1 \times 1 = 0.4$$

Therefore, $\max_{C_i \in \{\text{COVID Positive, COVID Negative}\}} \sum_{h_i \in H} P(C_i|h_i)P(h_i \mid T)$ = COVID Positive.

Thus, this algorithm, diagnoses the new instance to be COVID positive.

8.3.4 Gibbs Algorithm

The main drawback of Bayes optimal classifier is that it computes the posterior probability for all hypotheses in the hypothesis space and then combines the predictions to classify a new instance.

Gibbs algorithm is a sampling technique which randomly selects a hypothesis from the hypothesis space according to the posterior probability distribution and classifies a new instance. It is found that the prediction error occurs twice with the Gibbs algorithm when compared to Bayes Optimal classifier.

8.4 NAÏVE BAYES ALGORITHM FOR CONTINUOUS ATTRIBUTES

There are two ways to predict with Naive Bayes algorithm for continuous attributes:

1. Discretize continuous feature to discrete feature.
2. Apply Normal or Gaussian distribution for continuous feature.

Gaussian Naive Bayes Algorithm

In Gaussian Naive Bayes, the values of continuous features are assumed to be sampled from a Gaussian distribution.

Example 8.4: Assess a student's performance using Naïve Bayes algorithm for the continuous attribute. Predict whether a student gets a job offer or not in his final year of the course. The training dataset T consists of 10 data instances with attributes such as 'CGPA' and 'Interactiveness' as shown in Table 8.13. The target variable is Job Offer which is classified as Yes or No for a candidate student.

Table 8.13: Training Dataset with Continuous Attribute

S.No.	CGPA	Interactiveness	Job Offer
1.	9.5	Yes	Yes
2.	8.2	No	Yes
3.	9.3	No	No
4.	7.6	No	No
5.	8.4	Yes	Yes
6.	9.1	Yes	Yes
7.	7.5	Yes	No
8.	9.6	No	Yes
9.	8.6	Yes	Yes
10.	8.3	Yes	Yes

Solution:

Step 1: Compute the prior probability for the target feature 'Job Offer'.

Prior probabilities of both the classes are calculated using the same formula (refer to Table 8.14).

Table 8.14: Prior Probability of Target Class

Job Offer Classes	No. of Instances	Probability Value
Yes	7	P (Job Offer = Yes) = 7/10
No	3	P (Job Offer = No) = 3/10

Step 2: Compute Frequency matrix and Likelihood Probability for each of the feature.

Likelihood probabilities for a continuous attribute is obtained from Gaussian (Normal) Distribution. In the above data set, CGPA is a continuous attribute for which we need to apply Gaussian distribution to calculate the likelihood probability.

Gaussian distribution for continuous feature is calculated using the given formula,

$$P(X_i = x_k \mid C_j) = g(x_k, \mu_{ij}, \sigma_{ij}) \tag{8.5}$$

where,

X_i is the i^{th} continuous attribute in the given dataset and x_k is a value of the attribute.

C_j denotes the j^{th} class of the target feature.

μ_{ij} denotes the mean of the values of that continuous attribute X_i with respect to the class j of the target feature.

σ_{ij} denotes the standard deviation of the values of that continuous attribute X_i with respect to the class j of the target feature.

Hence, the normal distribution formula is given as:

$$P(X_i = x_k \mid C_j) = \frac{1}{\sigma_{ij}\sqrt{2\pi}} e^{\frac{-(x_k - \mu_{ij})^2}{2\sigma_{ij}^2}} \tag{8.6}$$

Step 2(a): Consider the feature CGPA

In this example CGPA is a continuous attribute,

To calculate the likelihood probability for this continuous attribute, first compute the mean and standard deviation for CGPA with respect to the target class 'Job Offer'.

Here, X_i = CGPA

C_j = 'Job Offer = Yes'

Mean and Standard Deviation for class 'Job Offer = Yes' are given as:

$\mu_{ij} = \mu_{CGPA-YES} = 8.814286$

$\sigma_{ij} = \sigma_{CGPA-YES} = 0.58146$

Mean and Standard Deviation for class 'Job Offer = No' are given as:

C_j = 'Job Offer = No'

$\mu_{ij} = \mu_{CGPA-NO} = 8.133333$

$\sigma_{ij} = \sigma_{CGPA-NO} = 1.011599$

Once Mean and Standard Deviation are computed, the likelihood probability for any test value using Gaussian distribution formula can be calculated.

Step 2(b): Consider the feature Interactiveness

Interactiveness is a discrete feature whose probability is calculated as earlier.

Table 8.15 shows the frequency matrix for the feature Interactiveness.

Table 8.15: Frequency Matrix of Interactiveness

Interactiveness	Job Offer = Yes	Job Offer = No
YES	5	1
NO	2	2
Total	7	3

Table 8.16 shows how the likelihood probability is calculated for Interactiveness using conditional probability.

Table 8.16: Likelihood Probability of Interactiveness

Interactiveness	*P* (Job Offer = Yes)	*P* (Job Offer = No)
YES	*P* (Interactiveness = Yes | Job Offer = Yes) = 5/7	*P* (Interactiveness = Yes | Job Offer = No) = 1/3
NO	*P* (Interactiveness = No | Job Offer = Yes) = 2/7	*P* (Interactiveness = No | Job Offer = No) = 2/3

Step 3: Use Bayes theorem to calculate the probability of all hypotheses.

Consider the test data to be (CGPA = 8.5, Interactiveness = Yes).

For the hypothesis 'Job Offer = Yes':

P (Job Offer = Yes | Test data) = (*P*(CGPA = 8.5 | Job Offer = Yes) × *P* (Interactiveness = Yes | Job Offer = Yes) × *P* (Job Offer = Yes)

To compute *P* (CGPA = 8.5 | Job Offer = Yes) use Gaussian distribution formula:

$P(X_i = x_k \mid C_j) = g(x_k, \mu_{ij}, \sigma_{ij})$

$$P(X_{CGPA} = 8.5 \mid C_{\text{Job Offer = Yes}}) = \frac{1}{\sigma_{ij}\sqrt{2\pi}} e^{\frac{(x_k - \mu_{ij})^2}{2\sigma_{ij}^2}}$$

$$P(X_{CGPA} = 8.5 \mid C_{\text{Job Offer = Yes}}) = \frac{1}{\sigma_{CGPA-YES}\sqrt{2\pi}} e^{\frac{(8.5 - \mu_{CGPA-YES})^2}{2\sigma_{CGPA-YES}^2}}$$

$P(\text{CGPA} = 8.5 \mid \text{Job Offer = Yes}) = g(x_k = 8.5, \mu_{ij} = 8.814, \sigma_{ij} = 0.581)$

$$= \frac{1}{0.581\sqrt{2\pi}} e^{\frac{(8.5 - 8.814)^2}{2 \times 0.581^2}} = 0.594$$

P (Interactiveness = Yes | Job Offer = Yes) = 5/7

P (Job Offer = Yes) = 7/10

Hence:

P (Job Offer = Yes | Test data) = (*P*(CGPA = 8.5 | Job Offer = Yes) × *P* (Interactiveness = Yes | Job Offer = Yes) × *P* (Job Offer = Yes)

= 0.594 × 5/7 × 7/10

= **0.297**

Similarly, for the hypothesis 'Job Offer = No':

P (Job Offer = No | Test data) = *P* (CGPA = 8.5 | Job Offer = No) × *P* (Interactiveness = Yes | Job Offer = No) × *P* (Job Offer = No)

P (CGPA = 8.5 | Job Offer = No) = $g(x_k = 8.5, \mu_{ij} = 8.133\ \sigma_{ij} = 1.0116)$

$$P(X_{CGPA} = 8.5 \mid C_{\text{Job Offer = Yes}}) = \frac{1}{\sigma_{CGPA-NO}\sqrt{2\pi}} e^{\frac{(8.5 - \mu_{CGPA-NO})^2}{2\sigma_{CGPA-NO}^2}}$$

$$= \frac{1}{1.0116\sqrt{2\pi}} e^{\frac{(8.5 - 8.133)^2}{2 \times 1.0116^2}} = 0.369$$

P (Interactiveness = Yes | Job Offer = No) = 1/3

= *P* (Job Offer = No) = 0.369

Hence,

P (Job Offer = No | Test data) = *P* (CGPA = 8.5 | Job Offer = No) *P* (Interactiveness = Yes | Job Offer = No) × *P* (Job Offer = No)

= 0.369 × 1/3 × 3/10

= **0.0369**

Step 4: Use Maximum A Posteriori (MAP) Hypothesis, h_{MAP} to classify the test object to the hypothesis with the highest probability.

Since *P* (Job Offer = Yes | Test data) has the highest probability value of 0.297,the test data is classified as '**Job Offer = Yes**'.

8.5 OTHER POPULAR TYPES OF NAIVE BAYES CLASSIFIERS

Some of the popular variants of Bayesian classifier are listed below:

Bernoulli Naive Bayes Classifier

Bernoulli Naive Bayes works with discrete features. In this algorithm, the features used for making predictions are Boolean variables that take only two values either 'yes' or 'no'. This is particularly useful for text classification where all features are binary with each feature containing two values whether the word occurs or not.

Multinomial Naive Bayes Classifier

This algorithm is a generalization of the Bernoulli Naive Bayes model that works for categorical data or particularly integer features. This classifier is useful for text classification where each feature will have an integer value that represents the frequency of occurrence of words.

Multi-class Naïve Bayes Classifier

This algorithm is useful for classification problems with more than two classes where the target feature contains multiple classes and test instance has to be predicted with the class it belongs to.

Chapter 10

Artificial Neural Networks

"I think the brain is essentially a computer and consciousness is like a computer program. It will cease to run when the computer is turned off. Theoretically, it could be re-created on a neural network, but that would be very difficult, as it would require all of one's memories."

— Stephen Hawking

Artificial Neural Networks (ANNs) imitate human brain behaviour and the way in which learning happens in a human. The human brain constitutes a mass of neurons that are all connected as a network, which is actually a directed graph. These neurons are the processing units which receive information, process it and then transmit this data to other neurons that allows humans to learn almost any task. ANN is a learning mechanism that models a human brain to solve any non-linear and complex problem. Each neuron is modelled as a computing unit, or simply called as a node in ANN, that is capable of doing complex calculations. ANN is a system that consists of many such computing units operating in parallel that can learn from observations. They are widely used in developing artificial learning systems and have inspired researchers and industry in Machine Learning nowadays. Some typical applications of ANN in the field of computer science are Natural Language Processing (NLP), pattern recognition, face recognition, speech recognition, character recognition, text processing, stock prediction, computer vision, etc. ANNs also have been considerably used in other engineering fields such as Chemical industry, Medicine, Robotics, Communications, Banking, and Marketing. This chapter aims to introduce the concepts necessary to understand the working of Artificial Neural Networks.

Learning Objectives

- Understand the basics of human nervous system and biological neurons
- Learn about McCulloch & Pitts Neuron Mathematical Model of an artificial neuron
- Know about the structure of Artificial Neural Network
- Introduce different types of activation functions
- Explore about the first neural network model called 'Perceptron' which is a linear binary classifier used for supervised learning
- Introduce different types of Artificial Neural Networks
- Understand the concepts of learning in Multi-Layer Perceptron (MLP) using back propagation
- Study about Radial Basis Function Neural Network (RBFNN) which is a type of multi-layer perceptron particularly useful for interpolation, function approximation, time series prediction, classification and system control
- Explore Self Organizing Feature Map (SOFM) which is a competitive learning network for unsupervised learning

10.1 INTRODUCTION

The human nervous system has billions of neurons that are the processing units which make humans to perceive things, to hear, to see and to smell. The human nervous system works beautifully, making us understand who we are, what we do, where we are and everything in our surrounding. It makes us to remember, recognize and correlate things around us. It is a learning system that consists of functional units called nerve cells, typically called as neurons. The human nervous system is divided into two sections called the Central Nervous System (CNS) and the Peripheral Nervous System (PNS). The brain and the spinal cord constitute the CNS and the neurons inside and outside the CNS constitute the PNS. The neurons are basically classified into three types called sensory neurons, motor neurons and interneurons. Sensory neurons get information from different parts of the body and bring it into the CNS, whereas motor neurons receive information from other neurons and transmit commands to the body parts. The CNS consists of only interneurons which connect one neuron to another neuron by receiving information from one neuron and transmitting it to another. The basic functionality of a neuron is to receive information, process it and then transmit it to another neuron or to a body part.

Scan for information on *'Convolution Neural Network', Modular Neural Network', and 'Recurrent Neural Network'*

10.2 BIOLOGICAL NEURONS

A typical biological neuron has four parts called dendrites, soma, axon and synapse. The body of the neuron is called as soma. Dendrites accept the input information and process it in the cell body called soma. A single neuron is connected by axons to around 10,000 neurons and through these axons the processed information is passed from one neuron to another neuron. A neuron gets fired if the input information crosses a threshold value and transmits signals to another neuron through a synapse. A synapse gets fired with an electrical impulse called spikes which are transmitted to another neuron. A single neuron can receive synaptic inputs from one neuron or multiple neurons. These neurons form a network structure which processes input information and gives out a response. The simple structure of a biological neuron is shown in Figure 10.1.

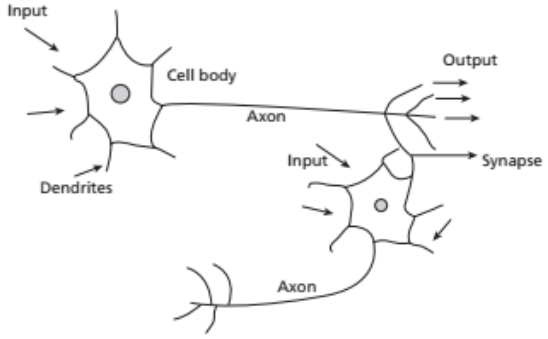


Figure 10.1: A Biological Neuron

10.3 ARTIFICIAL NEURONS

Artificial neurons are like biological neurons which are called as nodes. A node or a neuron can receive one or more input information and process it. Artificial neurons or nodes are connected by connection links to one another. Each connection link is associated with a synaptic weight. The structure of a single neuron is shown in Figure 10.2.

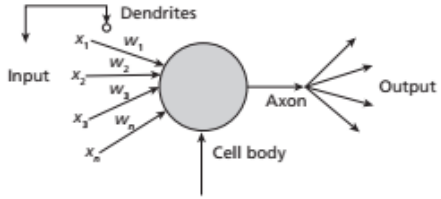


Figure 10.2: An Artificial Neuron

10.3.1 Simple Model of an Artificial Neuron

The first mathematical model of a biological neuron was designed by McCulloch & Pitts in 1943. It includes two steps:

1. It receives weighted inputs from other neurons
2. It operates with a threshold function or activation function

The received inputs are computed as a weighted sum which is given to the activation function and if the sum exceeds the threshold value the neuron gets fired. The mathematical model of a neuron is shown in Figure 10.3.

The neuron is the basic processing unit that receives a set of inputs $x_1, x_2, \ldots, x_n$ and their associated weights $w_1, w_2, \ldots, w_n$. The Summation function 'Net-sum' Eq. (10.1) computes the weighted sum of the inputs received by the neuron.

$$\text{Net-sum} = \sum_{i=1}^{n} x_i w_i \tag{10.1}$$

The activation function is a binary step function which outputs a value 1 if the Net-sum is above the threshold value θ, and a 0 if the Net-sum is below the threshold value θ. Therefore, the activation function is applied to Net-sum as shown in Eq. (10.2).

$$f(x) = \text{Activation function } (Net - sum) \tag{10.2}$$

$$\text{Then, output of a neuron } Y = \begin{cases} 1 & if\ f(x) \geq \theta \\ 0 & if\ f(x) < \theta \end{cases} \tag{10.3}$$

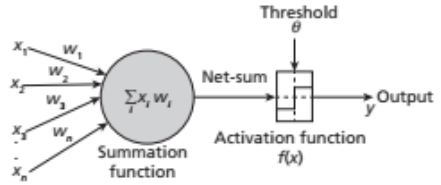


Figure 10.3: McCulloch & Pitts Neuron Mathematical Model

McCulloch & Pitts Neuron model can represent only a few Boolean functions. A Boolean function has binary inputs and provides a binary output. For example, an AND Boolean function neuron would fire when all the inputs are 1, whereas an OR Boolean function neuron would fire even when one input is 1. Moreover, the weight and threshold values are fixed in this mathematical model.

10.3.2 Artificial Neural Network Structure

Artificial Neural Network (ANN) imitates a human brain which inhibits some intelligence. It has a network structure represented as a directed graph with a set of neuron nodes and connection links or edges connecting the nodes as shown in Figure 10.4. The nodes in the graph are arrayed in a layered manner and can process information in parallel. The network given in the figure has three layers called input layer, hidden layer and output layer. The input layer receives the input information ($x_1, x_2, \ldots, x_n$) and passes it to the nodes in the hidden layer. The edges connecting the nodes from the input layer to the hidden layer are associated with synaptic weights called as connection weights. These computing nodes or neurons perform some computations based on the input information ($x_1, x_2, \ldots, x_n$) received and if the weighted sum of the inputs to a neuron is above the threshold or the activation level of the neuron, then the neuron fires. Each neuron employs an activation function that determines the output of the neuron. The neuron transforms linearly the input signals by computing the sum of the product of input signals and weights and adds biases to it. Then, the activation function maps the weighted input sum to a non-linear output value. The node in the output layer gives the output as a single value.

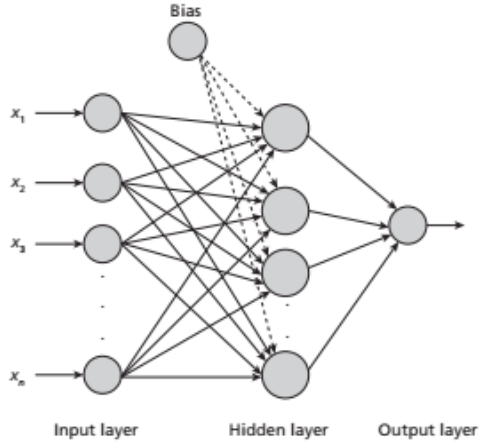


Figure 10.4: Artificial Neural Network Structure

10.3.3 Activation Functions

Activation functions are mathematical functions associated with each neuron in the neural network that map input signals to output signals. It decides whether to fire a neuron or not based on the input signals the neuron receives. These functions normalize the output value of each neuron either between 0 and 1 or between –1 and +1. Typical activation functions can be linear or non-linear.

Linear functions are useful when the input values can be classified into any one of the two groups and are generally used in binary perceptrons. Non-linear functions, on the other hand, are continuous functions that map the input in the range of (0, 1) or (−1, 1), etc. These functions are useful in learning high-dimensional data or complex data such as audio, video and images.

Below are some of the activation functions used in ANNs:

1. Identity Function or Linear Function

$$f(x) = x \ \forall x \tag{10.4}$$

The value of $f(x)$ increases linearly or proportionally with the value of x. This function is useful when we do not want to apply any threshold. The output would be just the weighted sum of input values. The output value ranges between $-\infty$ and $+\infty$.

2. Binary Step Function

$$f(x) = \begin{cases} 1 & if\ f(x) \geq \theta \\ 0 & if\ f(x) < \theta \end{cases} \tag{10.5}$$

The output value is binary, i.e., 0 or 1 based on the threshold value θ. If value of $f(x)$ is greater than or equal to θ, it outputs 1 or else it outputs 0.

3. Bipolar Step Function

$$f(x) = \begin{cases} 1 & if\ f(x) \geq \theta \\ -1 & if\ f(x) < \theta \end{cases} \tag{10.6}$$

The output value is bipolar, i.e., +1 or −1 based on the threshold value θ. If value of $f(x)$ is greater than or equal to θ, it outputs +1 or else it outputs −1.

4. Sigmoidal Function or Logistic Function

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{10.7}$$

It is a widely used non-linear activation function which produces an *S*-shaped curve and the output values are in the range of 0 and 1. It has a vanishing gradient problem, i.e., no change in the prediction for very low input values and very high input values.

5. Bipolar Sigmoid Function

$$\sigma(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \tag{10.8}$$

It outputs values between −1 and +1.

6. Ramp Functions

$$f(x) = \begin{cases} 1 & if\ x > 1 \\ x & if\ 0 \leq x \leq 1 \\ 0 & if\ x < 0 \end{cases} \tag{10.9}$$

It is a linear function whose upper and lower limits are fixed.

7. Tanh – Hyperbolic Tangent Function

The Tanh function is a scaled version of the sigmoid function which is also non-linear. It also suffers from the vanishing gradient problem. The output values range between −1 and 1.

$$\tan h(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{10.10}$$

8. ReLu – Rectified Linear Unit Function

This activation function is a typical function generally used in deep learning neural network models in the hidden layers. It avoids or reduces the vanishing gradient problem. This function outputs a value of 0 for negative input values and works like a linear function if the input values are positive.

$$r(x) = \max(0, x) = \begin{cases} x & if\ x \geq 0 \\ 0 & if\ x < 0 \end{cases} \tag{10.11}$$

9. Softmax Function

This is a non-linear function used in the output layer that can handle multiple classes. It calculates the probability of each target class which ranges between 0 and 1. The probability of the input belonging to a particular class is computed by dividing the exponential of the given input value by the sum of the exponential values of all the inputs.

$$s(x_i) = \frac{e^{x_i}}{\sum_{j=0}^{k} e^{x_j}} \text{ where } i = 0 \ldots k \tag{10.12}$$

10.4 PERCEPTRON AND LEARNING THEORY

The first neural network model 'Perceptron', designed by Frank Rosenblatt in 1958, is a linear binary classifier used for supervised learning. He modified the McCulloch & Pitts Neuron model by combining two concepts, McCulloch-Pitts model of an artificial neuron and Hebbian learning rule of adjusting weights. He introduced variable weight values and an extra input that represents *bias* to this model. He proposed that artificial neurons could actually learn weights and thresholds from data and came up with a supervised learning algorithm that enabled the artificial neurons to learn the correct weights from training data by itself. The perceptron model (shown in Figure 10.5) consists of 4 steps:

1. Inputs from other neurons
2. Weights and bias
3. Net sum
4. Activation function

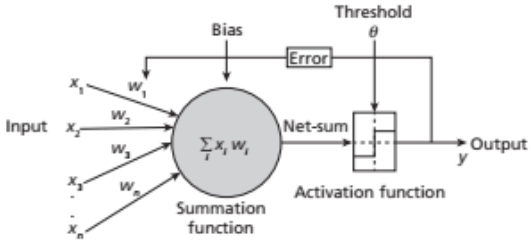


Figure 10.5: Perceptron Model

Thus, the modified neuron model receives a set of inputs $x_1, x_2, \ldots, x_n$, their associated weights $w_1, w_2, \ldots, w_n$ and a bias. The summation function 'Net-sum' Eq. (10.13) computes the weighted sum of the inputs received by the neuron.

$$\text{Net-sum} = \sum_{i=1}^{n} x_i w_i \tag{10.13}$$

After computing the 'Net-sum', bias value is added to it and inserted in the activation function as shown below:

$$f(x) = \text{Activation function } (Net\text{-}sum + bias) \tag{10.14}$$

The activation function is a binary step function which outputs a value 1 if $f(x)$ is above the threshold value θ, and a 0 if $f(x)$ is below the threshold value θ. Then, output of a neuron:

$$Y = \begin{cases} 1 & if\ f(x) \geq \theta \\ 0 & if\ f(x) < \theta \end{cases} \tag{10.15}$$

Before learning how a neural network works, let us learn about how a perceptron model works.

Algorithm 10.1: Perceptron Algorithm

Set initial weights $w_1, w_2, \ldots, w_n$ and bias θ to a random value in the range [–0.5, 0.5].

For each Epoch,

1. Compute the weighted sum by multiplying the inputs with the weights and add the products.
2. Apply the activation function on the weighted sum:
$$Y = \text{Step } ((x_1 w_1 + x_2 w_2) - \theta)$$
3. If the sum is above the threshold value, output the value as positive else output the value as negative.
4. Calculate the error by subtracting the estimated output $Y_{estimated}$ from the desired output $Y_{desired}$:
$$\text{error } e(t) = Y_{desired} - Y_{estimated}$$
[If error $e(t)$ is positive, increase the perceptron output Y and if it is negative, decrease the perceptron output Y.]
5. Update the weights if there is an error:
$$\Delta w_i = \propto \times e(t) \times x_i$$
$$w_i = w_i + \Delta w_i$$
where, x_i is the input value, $e(t)$ is the error at step t, $\propto$ is the learning rate and Δw_i is the difference in weight that has to be added to w_i.

Example 10.1: Consider a perceptron to represent the Boolean function AND with the initial weights $w_1 = 0.3$, $w_2 = -0.2$, learning rate $\propto = 0.2$ and bias $\theta = 0.4$ as shown in Figure 10.6. The activation function used here is the Step function $f(x)$ which gives the output value as binary, i.e., 0 or 1. If value of $f(x)$ is greater than or equal to 0, it outputs 1 or else it outputs 0. Design a perceptron that performs the Boolean function AND and update the weights until the Boolean function gives the desired output.

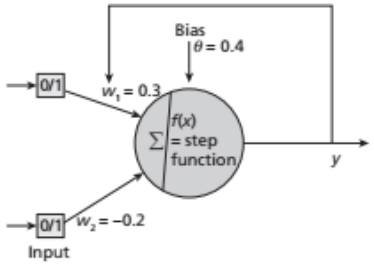


Figure 10.6: Perceptron for Boolean Function AND

Solution: Desired output for Boolean function AND is shown in Table 10.1.

Table 10.1: AND Truth Table

x_1	x_2	Y_{des}
0	0	0
0	1	0
1	0	0
1	1	1

For each Epoch, weighted sum is calculated and the activation function is applied to compute the estimated output Y_{est}. Then, Y_{est} is compared with Y_{des} to find the error. If there is an error, the weights are updated.

Tables 10.2 to 10.5 show how the weights are updated in the four Epochs.

Table 10.2: Epoch 1

Epoch	x_1	x_2	Y_{des}	Y_{est}	Error	w_1	w_2	Status
1	0	0	0	Step $((0 \times 0.3 + 0 \times -0.2) - 0.4) = 0$	0	0.3	–0.2	No change
	0	1	0	Step $((0 \times 0.3 + 1 \times -0.2) - 0.4) = 0$	0	0.3	–0.2	No change
	1	0	0	Step $((1 \times 0.3 + 0 \times -0.2) - 0.4) = 0$	0	0.3	–0.2	No change
	1	1	1	Step $((1 \times 0.3 + 1 \times -0.2) - 0.4) = 0$	1	0.5	0	Change

For input (1, 1) the weights are updated as follows:

$\Delta w_1 = \propto \times e(t) \times x_1 = 0.2 \times 1 \times 1 = 0.2$

$w_1 = w_1 + \Delta w_1 = 0.3 + \Delta w_1 = 0.3 + 0.2 = 0.5$

$\Delta w_2 = \propto \times e(t) \times x_2 = 0.2 \times 1 \times 1 = 0.2$

$w_2 = w_2 + \Delta w_1 = -0.2 + \Delta w_2 = -0.2 + 0.2 = 0$

Table 10.3: Epoch 2

Epoch	x_1	x_2	Y_{des}	Y_{est}	Error	w_1	w_2	Status
2	0	0	0	Step $((0 \times 0.5 + 0 \times 0) - 0.4) = 0$	0	0.5	0	No change
	0	1	0	Step $((0 \times 0.5 + 1 \times 0) - 0.4) = 0$	0	0.5	0	No change
	1	0	0	Step $((1 \times 0.5 + 0 \times 0) - 0.4) = 1$	–1	0.3	0	Change
	1	1	1	Step $((1 \times 0.3 + 1 \times 0) - 0.4) = 0$	1	0.5	0.2	Change

For input (1, 0) the weights are updated as follows:

$\Delta w_1 = \propto \times e(t) \times x_1 = 0.2 \times -1 \times 1 = -0.2$

$w_1 = w_1 + \Delta w_1 = 0.5 + \Delta w_1 = 0.5 - 0.2 = 0.3$

$\Delta w_2 = \propto \times e(t) \times x_2 = 0.2 \times -1 \times 0 = 0$

$w_2 = w_2 + \Delta w_2 = 0 + \Delta w_2 = 0 + 0 = 0$

For input (1, 1), the weights are updated as follows:

$\Delta w_1 = \propto \times e(t) \times x_1 = 0.2 \times 1 \times 1 = 0.2$

$w_1 = w_1 + \Delta w_1 = 0.3 + \Delta w_1 = 0.3 + 0.2 = 0.5$

$\Delta w_2 = \propto \times e(t) \times x_2 = 0.2 \times 1 \times 1 = 0.2$

$w_2 = w_2 + \Delta w_2 = 0 + \Delta w_2 = 0 + 0.2 = 0.2$

Table 10.4: Epoch 3

Epoch	x_1	x_2	Y_{des}	Y_{est}	Error	w_1	w_2	Status
3	0	0	0	Step $((0 \times 0.5 + 0 \times 0.2) - 0.4) = 0$	0	0.5	0.2	No change
	0	1	0	Step $((0 \times 0.5 + 1 \times 0.2) - 0.4) = 0$	0	0.5	0.2	No change
	1	0	0	Step $((1 \times 0.5 + 0 \times 0.2) - 0.4) = 1$	–1	0.3	0.2	Change
	1	1	1	Step $((1 \times 0.3 + 1 \times 0.2) - 0.4) = 1$	0	0.3	0.2	No change

For input (1, 0) the weights are updated as follows:

$\Delta w_1 = \propto \times e(t) \times x_1 = 0.2 \times -1 \times 1 = -0.2$

$w_1 = w_1 + \Delta w_1 = 0.5 + \Delta w_1 = 0.5 - 0.2 = 0.3$

$\Delta w_2 = \propto \times e(t) \times x_2 = 0.2 \times -1 \times 0 = 0$

$w_2 = w_2 + \Delta w_2 = 0 + \Delta w_2 = 0.2 + 0 = 0.2$

Table 10.5: Epoch 4

Epoch	x_1	x_2	Y_{des}	Y_{est}	Error	w_1	w_2	Status
4	0	0	0	Step $((0 \times 0.3 + 0 \times 0.2) - 0.4) = 0$	0	0.3	0.2	No change
	0	1	0	Step $((0 \times 0.3 + 1 \times 0.2) - 0.4) = 0$	0	0.3	0.2	No change
	1	0	0	Step $((1 \times 0.3 + 0 \times 0.2) - 0.4) = 0$	0	0.3	0.2	No change
	1	1	1	Step $((1 \times 0.3 + 1 \times 0.2) - 0.4) = 1$	0	0.3	0.2	No change

It is observed that with 4 Epochs, the perceptron learns and the weights are updated to 0.3 and 0.2 with which the perceptron gives the desired output of a Boolean AND function.

10.4.1 XOR Problem

A perceptron model can solve all Boolean functions which are linearly separable. However, the XOR problem was identified in 1969 by Minsky and Papert. An XOR function returns a 1, if the two inputs are not equal and a 0 if they are equal. Following is the truth table of an XOR function shown in Table 10.6.

Table 10.6: XOR Truth Table

x_1	x_2	Y
0	0	1
0	1	0
1	0	0
1	1	1

Since XOR is not linearly separable, the single layer perceptron failed to classify and hence this problem led to the evolution of a multi-layer perceptron. Initially, the Multi-Layer Perceptron (MLP) was not a success due to the lack of an appropriate learning algorithm. In 1974, Werbos introduced a back propagation algorithm for the three-layered perceptron network and in 1986, a general back propagation algorithm for a multi-layered perceptron was introduced by Rummelhart and Mclelland. Then, again ANN and Deep Neural Networks became a success, solving many complex problems in the current era. Thus, MLP that emerged could solve any non-linear separable problem which is explained later in this chapter.

10.4.2 Delta Learning Rule and Gradient Descent

Generally, learning in neural networks is performed by adjusting the network weights in order to minimize the difference between the desired and estimated outputs. This delta difference is measured as an error function or also called as cost function. The cost function, being linear and continuous, is differentiable. This way of learning called as delta rule (also known as Widrow-Hoff" rule or Adaline rule) is a type of back propagation applied for training the network. The training error of a hypothesis is half the squared difference between the desired target output and actual output and is given as follows:

$$\text{Training Error} = \frac{1}{2}\sum_{d \in T}(O_{Desired} - O_{Estimated})^2 \tag{10.16}$$

where, T is the training dataset, $O_{Desired}$ and $O_{Estimated}$ are the desired target output and estimated actual output, respectively, for a training instance d.

The principle of gradient descent is an optimization approach which is used to minimize the cost function by converging to a local minimal point moving in the negative direction of the gradient and each step size during movement is determined by the learning rate and the slope of the gradient.

Gradient descent learning is the foundation of back propagation algorithm used in MLP. Before we study about an MLP, let us first understand the different types of neural networks that differ in their structure, activation function and learning mechanism.

10.5 TYPES OF ARTIFICIAL NEURAL NETWORKS

ANNs consist of multiple neurons arranged in layers. There are different types of ANNs that differ by the network structure, activation function involved and the learning rules used. In an ANN, there are three layers called input layer, hidden layer and output layer. Any general ANN would consist of one input layer, one output layer and zero or more hidden layers.

10.5.1 Feed Forward Neural Network

This is the simplest neural network that consists of neurons which are arranged in layers and the information is propagated only in the forward direction. This model may or may not contain a hidden layer and there is no back propagation. Based on the number of hidden layers they are further classified into single-layered and multi-layered feed forward networks. These ANNs are simple to design and easy to maintain. They are fast but cannot be used for complex learning. They are used for simple classification and simple image processing, etc. The model of a Feed Forward Neural Network is shown in Figure 10.7.

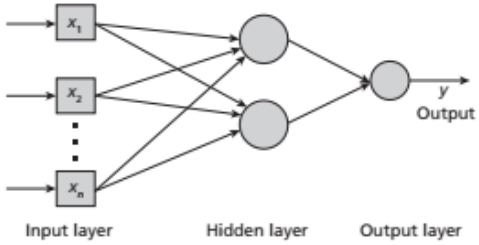


Figure 10.7: Model of a Feed Forward Neural Network

10.5.2 Fully Connected Neural Network

Fully connected neural networks are the ones in which all the neurons in a layer are connected to all other neurons in the next layer. The model of a fully connected neural network is shown in Figure 10.8.

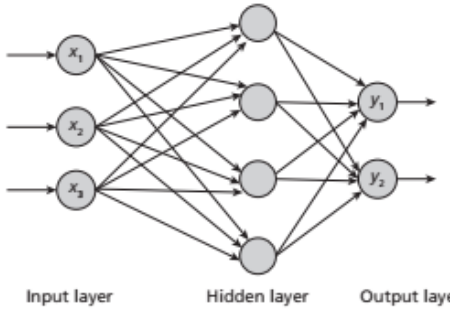


Figure 10.8: Model of a Fully Connected Neural Network

10.5.3 Multi-Layer Perceptron (MLP)

This ANN consists of multiple layers with one input layer, one output layer and one or more hidden layers. Every neuron in a layer is connected to all neurons in the next layer and thus they are fully connected. The information flows in both the directions. In the forward direction, the inputs are multiplied by weights of neurons and forwarded to the activation function of the

neuron and output is passed to the next layer. If the output is incorrect, then in the backward direction, error is back propagated to adjust the weights and biases to get correct output. Thus, the network learns with the training data. This type of ANN is used in deep learning for complex classification, speech recognition, medical diagnosis, forecasting, etc. They are comparatively complex and slow. The model of an MLP is shown in Figure 10.9.

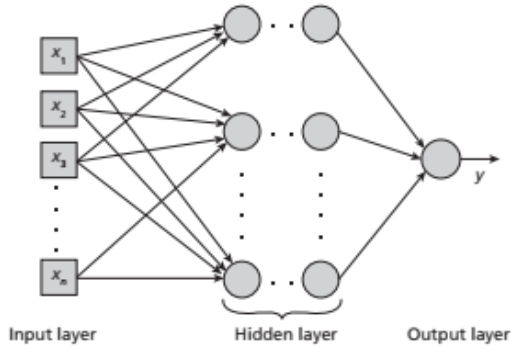


Figure 10.9: Model of a Multi-Layer Perceptron

10.5.4 Feedback Neural Network

Feedback neural networks have feedback connections between neurons that allow information flow in both directions in the network. The output signals can be sent back to the neurons in the same layer or to the neurons in the preceding layers. Hence, this network is more dynamic during training. The model of a feedback neural network is shown in Figure 10.10.

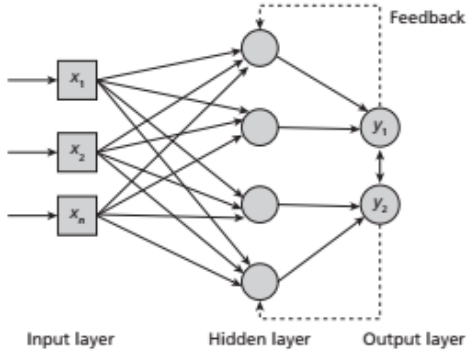


Figure 10.10: Model of a Feedback Neural Network

10.6 LEARNING IN A MULTI-LAYER PERCEPTRON

A multi-layer perceptron is a type of Feed Forward Neural Network with multiple neurons arranged in layers. All the neurons in a layer are fully connected to the neurons in the next layer. The network has atleast three layers with an input layer, one or more hidden layers and

an output layer. The input layer is the visible layer. It just passes the input to the next layer. The layers following the input layer are the hidden layers. The hidden layers neither directly receive inputs nor send outputs to the external environment. The final layer is the output layer which outputs a single value or a vector of values. An MLP has atleast one hidden layer and as the number of hidden layers is increased, the learning becomes more complex and they form a deep neural network. It uses back propagation for supervised learning of the network. The activation functions used in the layers can be linear or non-linear depending on the type of the problem modelled. Typically, a sigmoid activation function is used if the problem is a binary classification problem and a softmax activation function is used in a multi-class classification problem.

The typical non-linear separable problem is the XOR problem which could be solved with an MLP. A single neuron can only classify the values into two categories by drawing a single straight line. However, to classify the XOR inputs, we need two straight lines. Hence, the problem could be solved if we add a hidden layer having two neurons working in parallel in the same layer and combine the outputs to get a single output. One neuron should work as an OR gate and the other neuron should work as a NAND gate. The neuron in the output layer should work as an AND gate.

The MLP network learns with two phases called the forward phase and the backward phase. In the forward phase, an input vector from the training dataset is taken and given to the network which outputs a value called the estimated value $O_{Estimated}$. This value is compared with the desired output value $O_{Desired}$ and the error is calculated. The calculated error is back propagated in the backward phase to update the weights and biases of the network. This process is repeated for the entire training dataset.

The algorithm can be terminated after a pre-defined number of epochs or if the training error is reduced below a threshold value. This way of minimizing the training error by updating the weights and biases may lead to overfitting problem in neural networks. Several techniques are there to overcome this overfitting problem. One successful solution is to provide a set of validation data along with the training data and use a cross validation approach to estimate the number of iterations or weight updates to perform that produces the lowest error with the validation set and stop when the validation error begins to increase.

Algorithm 10.2: Learning in an MLP

Input: Input vector $(x_1, x_2, ..., x_n)$

Output: Y_n

Learning rate: α

Assign random weights and biases for every connection in the network in the range [–0.5, +0.5].

Step 1: Forward Propagation

1. **Calculate Input and Output in the *Input Layer*:**

   (Input layer is a direct transfer function, where the output of the node equals the input).

   Input at Node j 'I_j' in the Input Layer is

$$I_j = x_j$$

(*Continued*)

where,

x_j is the input received at Node j

Output at Node j 'O_j' in the Input Layer is

$$O_j = I_j$$

2. **Calculate Net Input and Output in the *Hidden Layer* and *Output Layer*:**

*Net Input at Node j in the **Hidden Layer** is*

$$I_j = \sum_{i=1}^{n} x_i w_{ij} + x_0 \times \theta_j$$

where,

x_i is the input from Node i

w_{ij} is the weight in the link from Node i to Node j

x_0 is the input to bias node '0' which is always assumed as 1

θ_j is the weight in the link from the bias node '0' to Node j

*Net Input at Node j in the **Output Layer** is*

$$I_j = \sum_{i=1}^{n} O_i w_{ij} + x_0 \times \theta_j$$

where,

O_i is the output from Node i

w_{ij} is the weight in the link from Node i to Node j

x_0 is the input to bias node '0' which is always assumed as 1

θ_j is the weight in the link from the bias node '0' to Node j

Output at Node j

$$O_j = \frac{1}{1 + e^{-I_j}}$$

where,

I_j is the input received at Node j

3. **Estimate error at the node in the *Output Layer*:**

$$\text{Error} = O_{Desired} - O_{Estimated}$$

where,

$O_{Desired}$ is the desired output value of the Node in the Output Layer

$O_{Estimated}$ is the estimated output value of the Node in the Output Layer

Step 2: Backward Propagation

1. **Calculate Error at each node:**

*For each Unit k in the **Output Layer***

$$\text{Error}_k = O_k (1 - O_k) (O_{Desired} - O_k)$$

(*Continued*)

where,

O_k is the output value at Node k in the Output Layer.

$O_{Desired}$ is the desired output value of the Node in the Output Layer.

For each unit j in the ***Hidden Layer***

$$\text{Error}_j = O_j(1 - O_j)\sum_k \text{Error}_k\, w_{jk}$$

where,

O_j is the output value at Node j in the Hidden Layer.

Error_k is the error at Node k in the Output Layer.

w_{jk} is the weight in the link from Node j to Node k.

2. Update all weights and biases:

Update weights

$$\Delta w_{ij} = \propto \times \text{Error}_j \times O_i$$
$$w_{ij} = w_{ij} + \Delta w_{ij}$$

where,

O_i is the output value at Node i.

Error_j is the error at Node j.

$\propto$ is the learning rate.

w_{ij} is the weight in the link from Node i to Node j.

Δw_{ij} is the difference in weight that has to be added to w_{ij}.

Update Biases

$$\Delta\theta_j = \propto \times \text{Error}_j$$
$$\theta_j = \theta_j + \Delta\theta_j$$

where,

Error_j is the error at Node j.

$\propto$ is the learning rate.

θ_j is the bias value from Bias Node 0 to Node j.

$\Delta\theta_j$ is the difference in bias that has to be added to θ_j.

Example 10.2: Consider learning in a Multi-Layer Perceptron. The given MLP consists of an Input layer, one Hidden layer and an Output layer. The input layer has 4 neurons, the hidden layer has 2 neurons and the output layer has a single neuron. Train the MLP by updating the weights and biases in the network.

x_1	x_2	x_3	x_4	$O_{Desired}$
1	1	0	1	1

Learning rate: = 0.8.

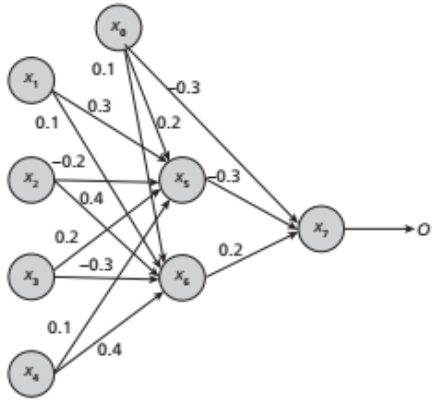


Figure 10.11: Example MLP

Solution: From the Figure 10.11, the weights and biases are tabulated in Table 10.7.

Table 10.7: Weights and Biases

x_1	x_2	x_3	x_4	w_{15}	w_{16}	w_{25}	w_{26}	w_{35}	w_{36}	w_{45}	w_{46}	w_{57}	w_{67}	θ_5	θ_6	θ_7
1	1	0	1	0.3	0.1	−0.2	0.4	0.2	−0.3	0.1	0.4	−0.3	0.2	0.2	0.1	−0.3

Step 1: Forward Propagation

1. Calculate Input and Output in the Input Layer shown in Table 10.8.

Table 10.8: Net Input and Output Calculation

Input layer	I_j	O_j
x_1	1	1
x_2	1	1
x_3	0	0
x_4	1	1

2. Calculate Net Input and Output in the Hidden Layer and Output Layer as shown in Table 10.9.

Table 10.9: Unit j at Hidden Layer and Output Layer – Net Input and Output Calculation

Unit_j	Net Input I_j	Output O_j
x_5	$I_5 = x_1 \times w_{15} + x_2 \times w_{25} + x_3 \times w_{35} + x_4 \times w_{45} + x_0 \times \theta_5$ $I_5 = 1 \times 0.3 + 1 \times -0.2 + 0 \times 0.2 + 1 \times 0.1 + 1 \times 0.2 = 0.4$	$O_5 = \frac{1}{1+e^{-I_5}}\ \frac{1}{1+e^{-0.4}} = 0.599$
x_6	$I_6 = x_1 \times w_{15} + x_2 \times w_{26} + x_3 \times w_{36} + x_4 \times w_{46} + x_0 \times \theta_6$ $I_5 = 1 \times 0.3 + 1 \times 0.4 + 0 \times -0.3 + 1 \times 0.4 + 1 \times 0.1 = 1.2$	$O_6 = \frac{1}{1+e^{-I_6}}\ \frac{1}{1+e^{-1.2}} = 0.769$
x_7	$I_7 = O_5 \times w_{57} + O_6 \times w_{67} + x_0 \times \theta_7$ $I_7 = 0.599 \times -0.3 + 0.769 \times 0.2 + 1 \times -0.3 = -0.326$	$O_7 = \frac{1}{1+e^{-I_7}}\ \frac{1}{1+e^{0.326}} = 0.419$

3. Calculate Error = $O_{desired} - O_{Estimated}$

   So, error for this network is:

   Error = $O_{desired} - O_7 = 1 - 0.419 = 0.581$

   So, we need to back propagate to reduce the error.

Step 2: Backward Propagation

1. Calculate Error at each node as shown in Table 10.10.

   For each unit k in the output layer, calculate:

   $\text{Error}_k = O_k(1 - O_k)(O_{desired} - O_k)$

   For each unit j in the hidden layer, calculate:

   $\text{Error}_j = O_j(1 - O_j)\sum_k \text{Error}_k w_{jk}$

Table 10.10: Error Calculation for Each Unit in the Output Layer and Hidden Layer

For Output Layer Unit_k	Error_k
x_7	$\text{Error}_7 = O_7(1 - O_7)(Y_a - O_7)$ $= 0.419 \times (1 - 0.419) \times (1 - 0.419) = 0.141$
For Hidden Layer Unit_j	**Error_j**
x_6	$\text{Error}_6 = O_6(1 - O_6)\sum_k \text{Error}_k w_{jk} = O_6(1 - O_6)\,\text{Error}_7 w_{67}$ $= 0.769\,(1 - 0.769) \times 0.2 \times 0.141 = 0.005$
x_5	$\text{Error}_5 = O_5(1 - O_5)\sum_k \text{Error}_k w_{jk} = O_5(1 - O_5)\,\text{Error}_7 w_{57}$ $= 0.599(1 - 0.599) \times 0.141 \times -0.3 = -0.0101$

2. Update weight using the below formula:

   Learning rate $\alpha = 0.8$.

   $\Delta w_{ij} = \propto \times \text{Error}_j \times O_i$

   $w_{ij} = w_{ij} + \Delta w_{ij}$

   The updated weights and bias are shown in Tables 10.11 and 10.12, respectively.

Table 10.11: Weight Updation

w_{ij}	$w_{ij} = w_{ij} + \propto \times \text{Error}_j \times O_i$	New Weight
w_{15}	$w_{15} = w_{15} + 0.8 \times \text{Error}_5 \times O_1$ $= 0.3 + 0.8 \times -0.0101 \times 1$	0.292
w_{16}	$w_{16} = w_{16} + 0.8 \times \text{Error}_6 \times O_1$ $= 0.1 + 0.8 \times 0.005 \times 1$	0.104
w_{25}	$w_{25} = w_{25} + 0.8 \times \text{Error}_5 \times O_2$ $= -0.2 + 0.8 \times -0.0101 \times 1$	−0.208
w_{26}	$w_{26} = w_{26} + 0.8 \times \text{Error}_6 \times O_2$ $= 0.4 + 0.8 \times 0.005 \times 1$	0.404

(*Continued*)

w_{ij}	$w_{ij} = w_{ij} + \propto \times \text{Error}_j \times O_i$	New Weight
w_{35}	$w_{35} = w_{35} + 0.8 \times \text{Error}_5 \times O_3$ $= 0.2 + 0.8 \times -0.0101 \times 0$	0.2
w_{36}	$w_{36} = w_{36} + 0.8 \times \text{Error}_6 \times O_3$ $= -0.3 + 0.8 \times 0.005 \times 0$	−0.3
w_{45}	$w_{45} = w_{45} + 0.8 \times \text{Error}_5 \times O_4$ $= 0.1 + 0.8 \times -0.0101 \times 1$	0.092
w_{46}	$w_{46} = w_{46} + 0.8 \times \text{Error}_6 \times O_4$ $= 0.4 + 0.8 \times 0.005 \times 1$	0.404
w_{57}	$w_{57} = w_{57} + 0.8 \times \text{Error}_7 \times O_5$ $= -0.3 + 0.8 \times 0.141 \times 0.599$	−0.232
w_{67}	$w_{67} = w_{67} + 0.8 \times \text{Error}_7 \times O_6$ $= 0.2 + 0.8 \times 0.141 \times 0.769$	0.287

Update bias using the below formula:

$$\Delta\theta_j = \propto \times \text{Error}_j$$

$$\theta_j = \theta_j + \Delta\theta_j$$

Table 10.12: Bias Updation

θ_j	$\theta_j = \theta_j + \propto \times \text{Error}_j$	New Bias
θ_5	$\theta_5 = \theta_5 + \propto \times \text{Error}_5$ $= 0.2 + 0.8 \times -0.0101$	0.192
θ_6	$\theta_6 = \theta_6 + \propto \times \text{Error}_6$ $= 0.1 + 0.8 \times 0.005$	0.104
θ_7	$\theta_7 = \theta_7 + \propto \times \text{Error}_7$ $= -0.3 + 0.8 \times 0.141$	−0.187

Iteration 2

Now, with the updated weights and biases:

1. Calculate Input and Output in the Input Layer as shown in Table 10.13.

Table 10.13: Net Input and Output Calculation

Input Layer	I_j	O_j
x_1	1	1
x_2	1	1
x_3	0	0
x_4	1	1

2. Calculate Net Input and Output in the Hidden Layer and Output Layer as shown in Table 10.14.

Table 10.14: Net Input and Output Calculation in the Hidden Layer and Output Layer

Unit j	Net Input I_j	Output O_j
x_5	$I_5 = x_1 \times w_{15} + x_2 \times w_{25} + x_3 \times w_{35} + x_4 \times w_{45} + x_0 \times \theta_5$ $I_5 = 1 \times 0.292 + 1 \times -0.208 + 0 \times 0.2 + 1 \times 0.092 + 1 \times 0.192 = 0.368$	$O_5 = \frac{1}{1+e^{-I_5}} = \frac{1}{1+e^{-0.368}} = 0.591$
x_6	$I_6 = x_1 \times w_{16} + x_2 \times w_{26} + x_3 \times w_{36} + x_4 \times w_{46} + x_0 \times \theta_6$ $I_6 = 1 \times 0.292 + 1 \times 0.404 + 0 \times -0.3 + 1 \times 0.404 + 1 \times 0.104 = 1.204$	$O_6 = \frac{1}{1+e^{-I_6}} = \frac{1}{1+e^{-1.204}} = 0.7692$
x_7	$I_7 = O_5 \times w_{57} + O_6 \times w_{67} + x_0 \times \theta_7$ $I_7 = 0.591 \times -0.232 + 0.7692 \times 0.287 + 1 \times -0.187 = -0.326$	$O_7 = \frac{1}{1+e^{-I_7}} = \frac{1}{1+e^{0.1034}} = 0.474$

The output we receive in the network at node 7 is 0.474.

Error = 1 – 0.474 = 0.526

Now, when we compare the error we get in the previous iteration and in the current iteration, it is visible that the network has learnt and reduced the error by 0.055.

Error is reduced by 0.055: 0.581 – 0.526.

Thus, the training is continued for a predefined number of epochs or until the training error is reduced below a threshold value.

10.7 RADIAL BASIS FUNCTION NEURAL NETWORK

Radial Basis Function Neural Network (RBFNN) was introduced by Broomhead and Lowe in 1988. It is a type of Multi Layer Perceptron which has one input layer, one output layer and with strictly one hidden layer. The hidden layer uses a non-linear radial basis function as the activation function, which converts the input parameters into high dimension space which is then fed into the network to linearly separate the problem. An XOR function is not linearly separable and requires at least one hidden layer to classify it. The RBFNN uses the hidden layer to derive the feature vector whose dimension is increased in space. This neural network is useful for interpolation, function approximation, time series prediction, classification and system control.

Typical Radial Basis Functions (RBF) are:

The Gaussian RBF which monotonically decreases with distance from the centre.

$$H(x) = e^{\frac{-(x-c)^2}{r^2}} \tag{10.17}$$

where, c is the centre and r is the radius.

A Multiquadric RBF which monotonically increases with distance from the centre.

$$H(x) = \sqrt{\frac{r^2 + (x-c)^2}{r}} \tag{10.18}$$

RBFNN architecture includes:

1. An input layer that feeds the input vector of n-dimension to the network ($x_1, x_2, \ldots, x_n$).
2. A hidden layer that comprises 'm' non-linear radial basis function neurons where $m \geq n$. The hidden layer implements the Radial Basis Function called Gaussian function. The output of a hidden layer neuron for an input vector x is given as in Eq. (10.17):

$$H(x) = e^{\frac{-(x-c)^2}{r^2}}$$

where, x is the input vector, c is the centre and r is the radius.

Each RBF neuron in the hidden layer compares the input vector with the centre of the neuron which is a bell curve and outputs a similarity value between 0 and 1. If the input is equal to the neuron centre, then the output is 1 but as the difference increases, the activation value or the output of the neuron falls off exponentially towards 0.

3. An output layer that computes the linear weighted sum of the output of each neuron from the hidden layer neurons:

$$F(x) = \sum_{i=1}^{m} w_i H_i(x) \tag{10.19}$$

where,

w_i is the weight in the link from the Hidden Layer neuron i to the Output Layer.

$H_i(x)$ is the output of a Hidden Layer neuron i for an input vector x.

The architecture of a Radial Basis Function Neural Network is shown in Figure 10.12.

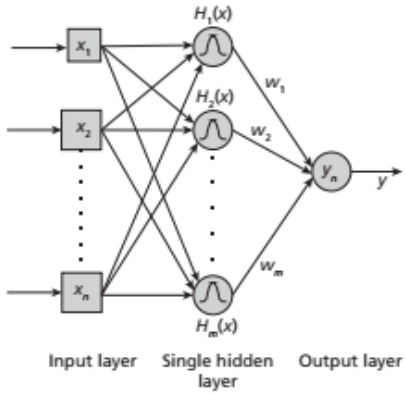


Figure 10.12: Architecture of RBFNN

Training or learning with RBFNN is very fast and the neural network is very good at interpolation.

Algorithm 10.3: Radial Basis Function Neural Network

Input: Input vector ($x_1, x_2, \ldots, x_n$)

Output: Y_n

Assign random weights for every connection from the Hidden layer to the Output layer in the network in the range [–1, +1].

Forward Phase:

Step 1: Calculate Input and Output in the Input Layer:

(Input layer is a direct transfer function, where the output of the node equals the input).

Input at Node i 'I$_i$' in the Input Layer is

$$\boxed{I_i = x_i}$$

where,

x_i is the input received at Node i.

(*Continued*)

Output at Node i 'O$_i$' in the Input Layer is

$$O_i = I_i$$

Step 2: For each node j in the Hidden Layer, find the centre/receptor c and the variance r.

Define hidden layer neurons with Gaussian RBF whose output is:

$$H_j(x) = e^{\frac{-(x-c_j)^2}{r^2}}$$

where, x is the input, c_j is the centre and r is the radius.

Compute $(x - c_j)^2$ applying Euclidean distance measure between x and c_j.

Step 3: For each node k in the Output Layer, compute linear weighted sum of the output of each neuron k from the hidden layer neurons j.

$$F_k(x) = \sum_{j=1}^{m} w_{jk} H_j(x)$$

where,

w_{jk} is the weight in the link from the Hidden Layer neuron j to the Output Layer neuron k.

$H_j(x)$ is the output of a Hidden Layer neuron j for an input vector x.

Backward Phase:

Step 1: Train the Hidden layer using Back propagation.

Step 2: Update the weights between the Hidden layer and Output layer.

Example 10.3: Consider the XOR Boolean function that has 4 patterns (0, 0) (0, 1) (1, 0) and (1, 1) in a 2-dimensional input space.

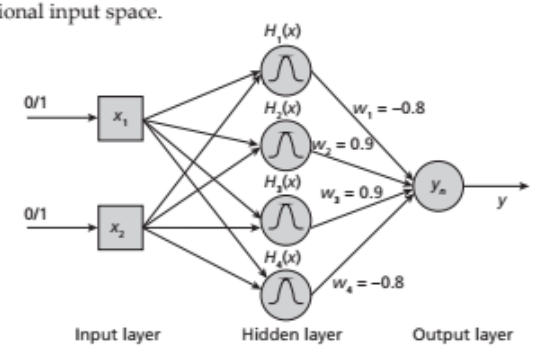


Figure 10.13: Example RBFNN

Construct a RBFNN as shown in Figure 10.13 that classifies the input pattern:

(0, 0) → 0

(0, 1) → 1

(1, 0) → 1

(1, 1) → 0

Solution: Define 4 hidden layer neurons with Gaussian RBF:

$H_1(x) = e^{\frac{-(x-c1)^2}{r^2}}$; $c1 = (0, 0)$

$H_2(x) = e^{\frac{-(x-c2)^2}{r^2}}$; $c2 = (0, 1)$

$H_3(x) = e^{\frac{-(x-c3)^2}{r^2}}$; $c3 = (1, 0)$

$H_4(x) = e^{\frac{-(x-c4)^2}{r^2}}$; $c4 = (1, 1)$

For input pattern (0, 0)	**For input pattern (0, 1)**
Distance squared of x from $c1 = (0, 0)$ $= (0-0)^2 + (0-0)^2 = 0$ $H_1(x) = e^{\frac{-(x-c1)^2}{r^2}} = e^{-\frac{0}{2}} = 1.0$ Distance squared of x from $c2 = (0, 1)$ $= (0-0)^2 + (0-1)^2 = 1$ $H_2(x) = e^{\frac{-(x-c2)^2}{r^2}} = e^{-\frac{1}{2}} = 0.6$ Distance squared of x from $c3 = (1, 0)$ $= (0-1)^2 + (0-0)^2 = 1$ $H_3(x) = e^{\frac{-(x-c3)^2}{r^2}} = e^{-\frac{1}{2}} = 0.6$ Distance squared of x from $c4 = (1, 1)$ $= (0-1)^2 + (0-1)^2 = 2$ $H_4(x) = e^{\frac{-(x-c4)^2}{r^2}} = e^{-\frac{2}{2}} = 0.4$ $\sum_{j=1}^{m} w_j H_j(x) = -0.8 \times 1.0 + 0.9 \times 0.6 + 0.9 \times 0.6 + -0.8 \times 0.4 = -0.04$	Distance squared of x from $c1 = (0, 0)$ $= (0-0)^2 + (1-0)^2 = 1$ $H_1(x) = e^{\frac{-(x-c2)^2}{r^2}} = e^{-\frac{1}{2}} = 0.6$ Distance squared of x from $c2 = (0, 1)$ $= (0-0)^2 + (1-1)^2 = 0$ $H_2(x) = e^{\frac{-(x-c2)^2}{r^2}} = e^{-\frac{1}{2}} = 1.0$ Distance squared of x from $c3 = (1, 0)$ $= (0-1)^2 + (1-0)^2 = 2$ $H_3(x) = e^{\frac{-(x-c3)^2}{r^2}} = e^{-\frac{2}{2}} = 0.4$ Distance squared of x from $c4 = (1, 1)$ $= (0-1)^2 + (1-1)^2 = 1$ $H_4(x) = e^{\frac{-(x-c4)^2}{r^2}} = e^{-\frac{1}{2}} = 0.6$ $\sum_{j=1}^{m} w_j H_j(x) = -0.8 \times 0.6 + 0.9 \times 1.0 + 0.9 \times 0.4 + -0.8 \times 0.6 = 0.3$
For input pattern (1, 0)	**For input pattern (1, 1)**
Distance squared of x from $c1 = (0, 0)$ $= (1-0)^2 + (0-0)^2 = 1$ $H_1(x) = e^{\frac{-(x-c1)^2}{r^2}} = e^{-\frac{1}{2}} = 0.6$ Distance squared of x from $c2 = (0, 1)$ $= (1-0)^2 + (0-1)^2 = 2$ $H_2(x) = e^{\frac{-(x-c2)^2}{r^2}} = e^{-\frac{2}{2}} = 0.4$ Distance squared of x from $c3 = (1, 0)$ $= (1-1)^2 + (0-0)^2 = 0$ $H_3(x) = e^{\frac{-(x-c3)^2}{r^2}} = e^{-\frac{1}{2}} = 1.0$ Distance squared of x from $c4 = (1, 1)$ $= (1-1)^2 + (0-1)^2 = 1$ $H_4(x) = e^{\frac{-(x-c4)^2}{r^2}} = e^{-\frac{1}{2}} = 0.6$ $\sum_{j=1}^{m} w_j H_j(x) = -0.8 \times 0.6 + 0.9 \times 0.4 + 0.9 \times 1.0 + -0.8 \times 0.6 = 0.3$	Distance squared of x from $c1 = (0, 0)$ $= (1-0)^2 + (1-0)^2 = 2$ $H_1(x) = e^{\frac{-(x-c2)^2}{r^2}} = e^{-\frac{2}{2}} = 0.4$ Distance squared of x from $c2 = (0, 1)$ $= (1-0)^2 + (1-1)^2 = 1$ $H_2(x) = e^{\frac{-(x-c2)^2}{r^2}} = e^{-\frac{1}{2}} = 0.6$ Distance squared of x from $c3 = (1, 0)$ $= (1-1)^2 + (1-0)^2 = 1$ $H_3(x) = e^{\frac{-(x-c3)^2}{r^2}} = e^{-\frac{1}{2}} = 0.6$ Distance squared of x from $c4 = (1, 1)$ $= (1-1)^2 + (1-1)^2 = 0$ $H_4(x) = e^{\frac{-(x-c4)^2}{r^2}} = e^{-\frac{0}{2}} = 1.0$ $\sum_{j=1}^{m} w_j H_j(x) = -0.8 \times 0.4 + 0.9 \times 0.6 + 0.9 \times 0.6 + -0.8 \times 1.0 = -0.04$

Table 10.15 shows the obtained values during the calculation done in the forward phase.

Table 10.15: Forward Phase Calculation

Input		$H_1(x)$	$H_2(x)$	$H_3(x)$	$H_4(x)$	$\sum_{j=1}^{m} w_j H_j(x)$	Output
0	0	1.0	0.6	0.6	0.4	–0.04	0
0	1	0.6	1.0	0.4	0.6	0.3	1
1	0	0.6	0.4	1.0	0.6	0.3	1
1	1	0.4	0.6	0.6	1.0	–0.04	0
		$w_1 = -0.8$	$w_2 = 0.9$	$w_3 = 0.9$	$w_4 = -0.8$		

RBF Networks are generally trained to determine the following parameters:

1. The number of neurons in the hidden layer
2. The center of each hidden-layer RBF neuron
3. The radius or variance of each RBF function
4. The weights assigned from the Hidden layer to the Output layer for the summation function

Different approaches are followed to determine the centres for the Hidden layer RBF neurons, comprising:

1. Random selection of fixed cluster centres
2. Self-organized selection of centres using *K*-means clustering
3. Supervised selection of centres

10.8 SELF-ORGANIZING FEATURE MAP

Self-Organizing Feature Map (SOFM) is a special type of Feed Forward Artificial Neural Network developed by Dr Teuvo Kohonen in 1982. Kohonen network is a competitive learning network or also called as adaptive learning network. SOFM is an unsupervised learning model that clusters data by mapping a high-dimensional data into a two-dimensional map (neurons) or plane. The model learns to cluster or self organize a high-dimensional data without knowing the class membership of the input data, and hence the name self-organizing nodes. These self-organizing nodes are also called as feature maps. The mapping is based on the relative distance or similarity between the points and the points that are near to each other in the input space are mapped to nearby output map units in the SOFM.

Network Architecture and Operations

The network architecture consists of only two layers called the Input layer and the Output layer, and there are no Hidden layers. The number of units in the Input layer is based on the length of the input samples which is a vector of length '*n*'. Each connection from the Input units in the Input layer to the output units in the Output layer is assigned with random weights. There is one weight vector of length '*n*' associated with each output unit. Output units have intra layer connections with no weights assigned between these connections but used for updating the weights. The network architecture of SOFM is shown in Figure 10.14.

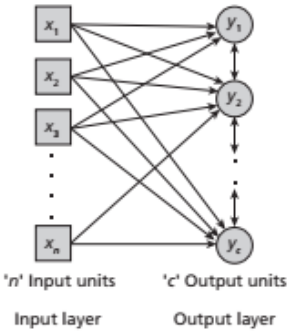


Figure 10.14: Network Architecture of Self Organizing Feature Map

The SOFM network operates in two phases, the training phase and the mapping phase. During the training phase, the input layer is fed with input samples randomly from the training data. The units or neurons in the output layer are initially assigned with some weights. As the input sample is fed, each output unit computes a similarity score by Euclidean distance measure and compete with each other. The output unit, which is close to the input sample by similarity, is chosen as the winning unit and its connection weights are adjusted by a learning factor. Thus, the best matching output unit whose weights are adjusted are moved close to the input sample and a topological feature map is formed. This process is repeated until the map does not change. During the mapping phase, the test samples are just classified.

Algorithm 10.4: Self-Organizing Feature Map

Input: '*m*' Training Vectors ($x_1, x_2, \ldots, x_m$), each of length '*n*':

$$\begin{bmatrix} x_{1,1} & x_{1,2} \ldots x_{1,n} \\ x_{2,1} & x_{2,2} \ldots x_{2,n} \\ \\ \\ x_{m,1} & x_{m,2} \ldots x_{m,n} \end{bmatrix}$$

Output: '*c*' categories

Input Layer: '*n*' Input units (Length of the training vector)

Output Layer: '*c*' Output units (Number of categories)

Step 1: Initialize random weights w_j between 0 and 1, for each Output unit *j*.
(A Weight vector of length '*n*' is associated with each Output unit)

Step 2: For each training sample x_s:

(a) Compute Euclidean distance score between the training sample x_s and the weight vector w_j of each output unit:

(*Continued*)

$$d^2 = \text{(Euclidean distance)}^2 = \sum_{k=1}^{n}(x_{s,k} - w_{j,k}(t))^2$$

(b) Choose the winning output unit which has smaller distance to the input sample as the best matching unit.

(c) Update the weights for the winning unit:

$$w_j(t+1) = w_j(t) + \eta(t)(x_s - w_j(t))$$

where,

$\eta(t)$ is the Learning rate.

$w_j(t)$ is the weight of the winning unit at step t.

$w_j(t+1)$ is the updated weight of the winning unit at step $(t+1)$.

Step 3: Repeat until the feature map doesn't change.

Example 10.4: Consider the example shown in Figure 10.15 which considers four training samples each vector of length 4 and two output units. Train the SOFM network by determining the class memberships of the input data.

Training Samples:

x_1: (1, 0, 1, 0)

x_2: (1, 0, 0, 0)

x_3: (1, 1, 1, 1)

x_4: (0, 1, 1, 0)

Output Units: Unit 1, Unit 2

Learning rate $\eta(t) = 0.6$

Initial Weight matrix

$$\begin{bmatrix}\text{Unit 1}\\ \text{Unit 2}\end{bmatrix}:\begin{bmatrix}0.3 & 0.5 & 0.7 & 0.2\\ 0.6 & 0.5 & 0.4 & 0.2\end{bmatrix}$$

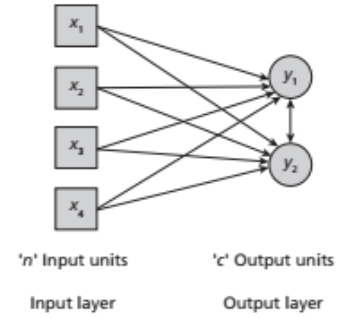


Figure 10.15: Example SOFM

Solution:

Iteration 1:

Training Sample x_1: (1, 0, 1, 0)

Weight matrix:

$$\begin{bmatrix}\text{Unit 1}\\ \text{Unit 2}\end{bmatrix}:\begin{bmatrix}0.3 & 0.5 & 0.7 & 0.2\\ 0.6 & 0.7 & 0.4 & 0.3\end{bmatrix}$$

Compute Euclidean distance between x_1: (1, 0, 1, 0) and Unit 1 weights.

$$d^2 = (0.3-1)^2 + (0.5-0)^2 + (0.7-1)^2 + (0.2-0)^2 = 0.87$$

Compute Euclidean distance between x_1: (1, 0, 1, 0) and Unit 2 weights.

$$d^2 = (0.6-1)^2 + (0.7-0)^2 + (0.4-1)^2 + (0.3-0)^2 = 1.1$$

Unit 1 wins

Update the weights of the winning unit.

New Unit 1 weights = [0.3 0.5 0.7 0.2] + 0.6 ([1 0 1 0] – [0.3 0.5 0.7 0.2])

= [0.3 0.5 0.7 0.2] + 0.6 [0.7 –0.5 0.3 –0.2]

= [0.3 0.5 0.7 0.2] + [0.42 –0.30 0.18 –0.12]

= [0.72 0.2 0.88 0.08]

$$\begin{bmatrix}\text{Unit 1}\\ \text{Unit 2}\end{bmatrix}:\begin{bmatrix}0.72 & 0.2 & 0.88 & 0.08\\ 0.6 & 0.7 & 0.4 & 0.3\end{bmatrix}$$

Iteration 2:

Training Sample x_2: (1, 0, 0, 0)

Weight matrix:

$$\begin{bmatrix}\text{Unit 1}\\ \text{Unit 2}\end{bmatrix}:\begin{bmatrix}0.72 & 0.2 & 0.88 & 0.08\\ 0.6 & 0.7 & 0.4 & 0.3\end{bmatrix}$$

Compute Euclidean distance between x_2: (1, 0, 0, 0) and Unit 1 weights.

$$d^2 = (0.72-1)^2 + (0.2-0)^2 + (0.88-0)^2 + (0.08-0)^2 = 0.74$$

Compute Euclidean distance between x_2: (1, 0, 0, 0) and Unit 2 weights.

$$d^2 = (0.6-1)^2 + (0.7-0)^2 + (0.4-0)^2 + (0.3-0)^2 = 0.9$$

Unit 1 wins

Update the weights of the winning unit:

New Unit 1 weights = [0.72 0.2 0.88 0.08] + 0.6 ([1 0 0 0] – [0.72 0.2 0.88 0.08])

= [0.72 0.2 0.88 0.08] + 0.6 [0.28 –0.2 –0.88 –0.08]

= [0.72 0.2 0.88 0.08] + [0.17 –0.12 –0.53 –0.05]

= [0.89 0.08 0.35 0.03]

$$\begin{bmatrix}\text{Unit 1}\\ \text{Unit 2}\end{bmatrix}:\begin{bmatrix}0.89 & 0.08 & 0.35 & 0.03\\ 0.6 & 0.7 & 0.4 & 0.3\end{bmatrix}$$

Iteration 3:

Training Sample x_3: (1, 1, 1, 1)

Weight matrix:

$$\begin{bmatrix}\text{Unit 1}\\ \text{Unit 2}\end{bmatrix}:\begin{bmatrix}0.89 & 0.08 & 0.35 & 0.03\\ 0.6 & 0.7 & 0.4 & 0.3\end{bmatrix}$$

Compute Euclidean distance between x_3: (1, 1, 1, 1) and Unit 1 weights.

$d^2 = (0.89 - 1)^2 + (0.08 - 1)^2 + (0.35 - 1)^2 + (0.03 - 1)^2$

$= 2.2$

Compute Euclidean distance between x_3: (1, 1, 1, 1) and Unit 2 weights.

$d^2 = (0.6 - 1)^2 + (0.7 - 1)^2 + (0.4 - 1)^2 + (0.3 - 1)^2$

$= 1.1$

Unit 2 wins

Update the weights of the winning unit:

New Unit 2 weights = [0.6 0.7 0.4 0.3] + 0.6 ([1 1 1 1] – [0.6 0.7 0.4 0.3])

= [0.6 0.7 0.4 0.3] + 0.6 [0.4 0.3 0.6 0.7]

= [0.6 0.7 0.4 0.3] + [0.24 0.18 0.36 0.42] = [0.84 0.88 0.76 0.72]

$$\begin{bmatrix}\text{Unit 1}\\ \text{Unit 2}\end{bmatrix}:\begin{bmatrix}0.89 & 0.08 & 0.35 & 0.03\\ 0.84 & 0.88 & 0.76 & 0.72\end{bmatrix}$$

Iteration 4:

Training Sample x_4: (0, 1, 1, 0)

Weight matrix:

$$\begin{bmatrix}\text{Unit 1}\\ \text{Unit 2}\end{bmatrix}:\begin{bmatrix}0.89 & 0.08 & 0.35 & 0.03\\ 0.84 & 0.88 & 0.76 & 0.72\end{bmatrix}$$

Compute Euclidean distance between x_4: (0, 1, 1, 0) and Unit 1 weights.

$d^2 = (0.89 - 0)^2 + (0.08 - 1)^2 + (0.35 - 1)^2 + (0.03 - 0)^2$

$= 2.06$

Compute Euclidean distance between x_4: (0, 1, 1, 0) and Unit 2 weights.

$d^2 = (0.84 - 0)^2 + (0.88 - 1)^2 + (0.76 - 1)^2 + (0.72 - 0)^2$

$= 1.3$

Unit 2 wins

Update the weights of the winning unit:

New Unit 2 weights = [0.84 0.88 0.76 0.72] + 0.6 ([0 1 1 0] – [0.84 0.88 0.76 0.72])

= [0.84 0.88 0.76 0.72] + 0.6 [–0.84 0.12 0.24 –0.72]

= [0.84 0.88 0.76 0.72] + [–0.5 0.07 0.14 –0.43] = [0.34 0.95 0.9 0.29]

$$\begin{bmatrix}\text{Unit 1}\\ \text{Unit 2}\end{bmatrix}:\begin{bmatrix}0.89 & 0.08 & 0.35 & 0.03\\ 0.34 & 0.95 & 0.9 & 0.29\end{bmatrix}$$

Best mapping units for each of the sample taken are:

x_1: (1, 0, 1, 0) → Unit 1

x_2: (1, 0, 0, 0) → Unit 1

x_3: (1, 1, 1, 1) → Unit 2

x_4: (0, 1, 1, 0) → Unit 2

This process is continued for many epochs until the feature map does not change.

10.9 POPULAR APPLICATIONS OF ARTIFICIAL NEURAL NETWORKS

ANN learning mechanisms are used in many complex applications that involve modelling of non-linear processes. ANN is a useful model that can handle even noisy and incomplete data. They are used to model complex patterns, recognize patterns and solve prediction problems like humans in many areas such as:

1. Real-time applications: Face recognition, emotion detection, self-driving cars, navigation systems, routing systems, target tracking, vehicle scheduling, etc.
2. Business applications: Stock trading, sales forecasting, customer behaviour modelling, Market research and analysis, etc.
3. Banking and Finance: Credit and loan forecasting, fraud and risk evaluation, currency price prediction, real-estate appraisal, etc.
4. Education: Adaptive learning software, student performance modelling, etc.
5. Healthcare: Medical diagnosis or mapping symptoms to a medical case, image interpretation and pattern recognition, drug discovery, etc.
6. Other Engineering Applications: Robotics, aerospace, electronics, manufacturing, communications, chemical analysis, food research, etc.

10.10 ADVANTAGES AND DISADVANTAGES OF ANN

Advantages of ANN

1. ANN can solve complex problems involving non-linear processes.
2. ANNs can learn and recognize complex patterns and solve problems as humans solve a problem.
3. ANNs have a parallel processing capability and can predict in less time.
4. They have an ability to work with inadequate knowledge. It can even handle incomplete and noisy data.
5. They can scale well to larger data sets and outperforms other learning mechanisms.

Limitations of ANN

1. An ANN requires processors with parallel processing capability to train the network running for many epochs. The function of each node requires a CPU capability which is difficult for very large networks with a large amount of data.
2. They work like a 'black box' and it is exceedingly difficult to understand their working in inner layers. Moreover, it is hard to understand the relationship between the representations learned at each layer.

3. The modelling with ANN is also extremely complicated and the development takes a much longer time.
4. Generally, neural networks require more data than traditional machine learning algorithms, and they do not perform well on small datasets.
5. They are also more computationally expensive than traditional learning techniques.

10.11 CHALLENGES OF ARTIFICIAL NEURAL NETWORKS

The major challenges while modelling a real-time application with ANNs are:

1. Training a neural network is the most challenging part of using this technique. Overfitting or underfitting issues may arise if datasets used for training are not correct. It is also hard to generalize to the real-world data when trained with some simulated data. Moreover, neural network models normally need a lot of training data to be robust and are usable for a real-time application.
2. Finding the weight and bias parameters for neural networks is also hard and it is difficult to calculate an optimal model.

Chapter 13

Clustering Algorithms

"Wherever you see a successful business, someone once made a courageous decision."

— Peter Drucker

Cluster analysis is a technique of partitioning a collection of unlabelled objects, with many attributes, into meaningful disjoint groups or clusters. This chapter aims to provide the basic concepts of clustering algorithms.

Learning Objectives

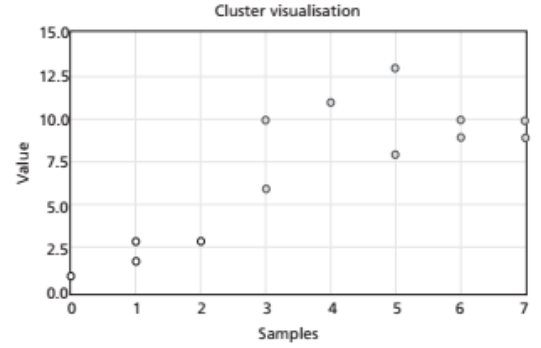
- Introduce the concepts of clustering
- Highlight the role of distance measures in clustering process
- Provide a taxonomy of clustering algorithms
- Explain hierarchical clustering algorithms
- Explain partitional clustering algorithms
- Briefly explain density-based, grid-based, and probabilistic model-based clustering techniques
- Discuss the validation techniques for clustering algorithms

13.1 INTRODUCTION TO CLUSTERING APPROACHES

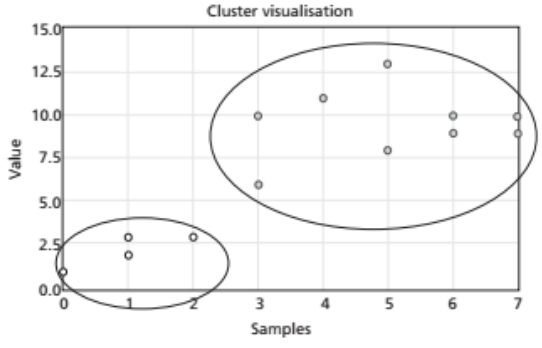
Cluster analysis is the fundamental task of unsupervised learning. Unsupervised learning involves exploring the given dataset. Cluster analysis is a technique of partitioning a collection of unlabelled objects that have many attributes into meaningful disjoint groups or clusters. This is done using a trial and error approach as there are no supervisors available as in classification. The characteristic of clustering is that the objects in the clusters or groups are similar to each other within the clusters while differ from the objects in other clusters significantly.

The input for cluster analysis is examples or samples. These are known as objects, data points or data instances. All these terms are same and used interchangeably in this chapter. All the samples or objects with no labels associated with them are called unlabelled. The output is

the set of clusters (or groups) of similar data if it exists in the input. For example, the following Figure 13.1(a) shows data points or samples with two features shown in different shaded samples and Figure 13.1(b) shows the manually drawn ellipse to indicate the clusters formed.



(a)



(b)

Figure 13.1: (a) Data Samples (b) Clusters' Description

Visual identification of clusters in this case is easy as the examples have only two features. But, when examples have more features, say 100, then clustering cannot be done manually and automatic clustering algorithms are required. Also, automating the clustering process is desirable as these tasks are considered difficult by humans and almost impossible. All clusters are represented by centroids. For example, if the input examples or data is (3, 3), (2, 6) and (7, 9), then the centroid is given as $\left(\frac{3+2+7, 3+6+9}{3}\right) = (4, 6)$. The clusters should not overlap and every cluster should represent only one class. Therefore, clustering algorithms use trial and error method to form clusters that can be converted to labels. Thus, the important differences between classification and clustering are given in Table 13.1.

Table 13.1: Differences between Classification and Clustering

S.No.	Clustering	Classification
1.	Unsupervised learning and cluster formation are done by trial and error as there is no supervisor	Supervised learning with the presence of a supervisor to provide training and testing data
2.	Unlabelled data	Labelled data
3.	No prior knowledge in clustering	Knowledge of the domain is must to label the samples of the dataset
4.	Cluster results are dynamic	Once a label is assigned, it does not change

Applications of Clustering

1. Grouping based on customer buying patterns
2. Profiling of customers based on lifestyle
3. In information retrieval applications (like retrieval of a document from a collection of documents)
4. Identifying the groups of genes that influence a disease
5. Identification of organs that are similar in physiology functions
6. Taxonomy of animals, plants in Biology
7. Clustering based on purchasing behaviour and demography
8. Document indexing
9. Data compression by grouping similar objects and finding duplicate objects

Challenges of Clustering Algorithms

A huge collection of data with higher dimensions (i.e., features or attributes) can pose a problem for clustering algorithms. With the arrival of the internet, billions of data are available for clustering algorithms. This is a difficult task, as scaling is always an issue with clustering algorithms. Scaling is an issue where some algorithms work with lower dimension data but do not perform well for higher dimension data. Also, units of data can post a problem, like some weights in kg and some in pounds can pose a problem in clustering. Designing a proximity measure is also a big challenge.

The advantages and disadvantages of the cluster analysis algorithms are given in Table 13.2.

Table 13.2: Advantages and Disadvantages of Clustering Algorithms

S.No.	Advantages	Disadvantages
1.	Cluster analysis algorithms can handle missing data and outliers.	Cluster analysis algorithms are sensitive to initialization and order of the input data.
2.	Can help classifiers in labelling the unlabelled data. Semi-supervised algorithms use cluster analysis algorithms to label the unlabelled data and then use classifiers to classify them.	Often, the number of clusters present in the data have to be specified by the user.

(*Continued*)

S.No.	Advantages	Disadvantages
3.	It is easy to explain the cluster analysis algorithms and to implement them.	Scaling is a problem.
4.	Clustering is the oldest technique in statistics and it is easy to explain. It is also relatively easy to implement.	Designing a proximity measure for the given data is an issue.

13.2 PROXIMITY MEASURES

Scan for *'Additional Information on Proximity Measures'*

Clustering algorithms need a measure to find the similarity or dissimilarity among the objects to group them. Similarity and dissimilarity are collectively known as proximity measures. Often, the distance measures are used to find similarity between two objects, say i and j.

Distance measures are known as dissimilarity measures, as these indicate how one object is different from another. Measures like cosine similarity indicate the similarity among objects. Distance measures and similarity measures are two sides of the same coin, as more distance indicates more similarity and vice versa. Distance between two objects, say i and j, is denoted by the symbol D_{ij}.

The properties of the distance measures are:

1. D_{ij} is always positive or zero.
2. $D_{ij} = 0$, i.e., the distance between the object to itself is 0.
3. $D_{ij} = D_{ji}$. This property is called symmetry.
4. $D_{ij} \leq D_{ik} + D_{kj}$. This property is called triangular inequality.

If all these conditions are satisfied, then the distance measure is called a metric.

Based on the data types of the attributes of an object, distance measures vary. The concept of data types is discussed in Chapter 2. It can be recollected that the data types are divided into categorical and quantitative variables. Quantitative variables are numbers and are of two types – nominal and ordinal. For example, gender is a nominal variable as gender can be enumerated as Gender = {male, female}. Ordinal variables look like nominal variables but have an inherent order present in the enumeration. For example, temperature is a nominal variable as temperature can be enumerated as Temperature = {low, medium, high}. One can observe the inherent order present, that is, medium > low and low < medium. Quantitative variables are real or Integer numbers or binary data. In binary data, the attributes of the object can take a Boolean value. Objects whose attributes take binary data are called binary objects.

Let us review some of the proximity measures.

Quantitative Variables

Some of the qualitative variables are discussed below.

Euclidean Distance It is one of the most important and common distance measures. It is also called as L_2 norm. It can be defined as the square root of squared differences between the coordinates of a pair of objects.

The Euclidean distance between objects x_i and x_j with k features is given as follows:

$$\text{Distance}(x_i, x_j) = \sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2} \tag{13.1}$$

The advantage of Euclidean distance is that the distance does not change with the addition of new objects. But the disadvantage is that if the units change, the resulting Euclidean or squared Euclidean changes drastically. Another disadvantage is that as the Euclidean distance involves a square root and a square, the computational complexity is high for implementing the distance for millions or billions of operations involved.

City Block Distance City block distance is known as Manhattan distance. This is also known as boxcar, absolute value distance, Manhattan distance, Taxicab or L_1 norm. The formula for finding the distance is given as follows:

$$\text{Distance}(x_i, x_j) = \sum_{k=1}^{n}\left|x_{ik} - x_{jk}\right| \tag{13.2}$$

Chebyshev Distance Chebyshev distance is known as maximum value distance. This is the absolute magnitude of the differences between the coordinates of a pair of objects. This distance is called supremum distance or L_{max} or L_∞ norm. The formula for computing Chebyshev distance is given as follows:

$$\text{Distance}(x_i, x_j) = \max_k\left|x_{ik} - x_{jk}\right| \tag{13.3}$$

Example 13.1: Suppose, if the coordinates of the objects are (0, 3) and (5, 8), then what is the Chebyshev distance?

Solution: The Euclidean distance using Eq. (13.1) is given as follows:

$$\text{Distance}(x_i, x_j) = \sqrt{(0-5)^2 + (3-8)^2}$$
$$= \sqrt{50} = 7.07$$

The Manhattan distance using Eq. (13.2) is given as follows:

$$\text{Distance}(x_i, x_j) = \left|(0-5) + (3-8)\right| = 10$$

The Chebyshev distance using Eq. (13.3) is given as follows:

$$\text{Max}\left\{\left|0-5\right|, \left|3-8\right|\right\} = \text{Max}\{5, 5\} = 5$$

Minkowski Distance In general, all the above distance measures can be generalized as:

$$\text{Distance}(x_i, x_j) = \left(\sum\left|x_{ik} - x_{jk}\right|^r\right)^{\frac{1}{r}} \tag{13.4}$$

This is called Minkowski distance. Here, r is a parameter. When the value of r is 1, the distance measure is called city block distance. When the value of r is 2, the distance measure is called Euclidean distance. When, r is ∞, then this is Chebyshev distance.

Binary Attributes

Binary attributes have only two values. Distance measures discussed above cannot be applied to find distance between objects that have binary attributes. For finding the distance among objects with binary objects, the contingency Table 13.3 can be used. Let x and y be the objects consisting of N-binary objects. Then, the contingency table can be constructed by counting the number of matching of transitions, 0–0, 0–1, 1–0 and 1–1.

Table 13.3: Contingency Table

Attributes Matching	0	1
0	a	b
1	c	d

In other words, 'a' is the number of attributes where x attribute is 0 and y attribute is 0. 'b' is the number of attributes where x attribute is 0 and y attribute is 1, 'c' is the number of attributes where x attribute is 1 and y attribute is 0 and 'd' is the number of attributes where x attribute is 1 and y attribute is 1.

Simple Matching Coefficient (SMC) SMC is a simple distance measure and is defined as the ratio of number of matching attributes and the number of attributes. The formula is given as:

$$\frac{a+d}{a+b+c+d} \tag{13.5}$$

The values of a, b, c, and d can be observed from the Table 13.4.

Jaccard Coefficient Jaccard coefficient is another useful measure for and is given as follows:

$$J = \frac{d}{b+c+d} \tag{13.6}$$

Example 13.2: If the given vectors are $x = (1, 0, 0)$ and $y = (1, 1, 1)$ then find the SMC and Jaccard coefficient?

Solution: It can be seen from Table 13.2 that, $a = 0$, $b = 2$, $c = 0$ and $d = 1$.

The SMC using Eq. (13.5) is given as $\frac{a+d}{a+b+c+d} = 0+1/3 = 0.33$

Jaccard coefficient using Eq. (13.6) is given as $J = \frac{d}{b+c+d} = 1/3 = 0.33$

Hamming Distance Hamming distance is another useful measure that can be used for knowing the sequence of characters or binary values. It indicates the number of positions at which the characters or binary bits are different.

For example, the hamming distance between $x = (1\ 0\ 1)$ and $y = (1\ 1\ 0)$ is 2 as x and y differ in two positions. The distance between two words, say wood and hood is 1, as they differ in only one character. Sometimes, more complex distance measures like edit distance can also be used.

Categorical Variables

In many cases, categorical values are used. It is just a code or symbol to represent the values. For example, for the attribute Gender, a code 1 can be given to female and 0 can be given to male.

To calculate the distance between two objects represented by variables, we need to find only whether they are equal or not. This is given as:

$$\text{Distance}(x, y) = \begin{cases} 1 & \text{if } x \neq y \\ 0 & \text{if } x = y \end{cases} \tag{13.7}$$

Ordinal Variables

Ordinal variables are like categorical values but with an inherent order. For example, designation is an ordinal variable. If job designation is 1 or 2 or 3, it means code 1 is higher than 2 and code 2 is higher than 3. It is ranked as $1 > 2 > 3$.

Let us assume the designations of office employees are clerk, supervisor, manager and general manager. These can be designated as numbers as clerk = 1, supervisor = 2, manager = 3 and general manager = 4. Then, the distance between employee X who is a clerk and Y who is a manager can be obtained as:

$$\text{Distance}(X, Y) = \frac{|position(X) - position(Y)|}{n-1} \tag{13.8}$$

Here, position (X) and position(Y) indicate the designated numerical value. Thus, the distance between X (Clerk = 1) and Y (Manager = 3) using Eq. (13.8) is given as:

$$\text{Distance}(X, Y) = \frac{|position(X) - position(Y)|}{n-1} = \frac{|1-3|}{4-1} = \frac{2}{3} \approx 0.66$$

Vector Type Distance Measures

For text classification, vectors are normally used. Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Cosine similarity measures the cosine of the angle between two vectors projected in a multi-dimensional space. The similarity function for vector objects can be defined as:

$$sim(X, Y) = \frac{X \bullet Y}{\|X\| \times \|Y\|} = \frac{\sum_{i=1}^{n} X_i \times Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2} \times \sqrt{\sum_{i=1}^{n} Y_i^2}} \tag{13.9}$$

The numeration is the dot product of the vectors A and B. The denominator is the product of the norm of vectors A and B.

Example 13.3: If the given vectors are $A = \{1, 1, 0\}$ and $B = \{0, 1, 1\}$, then what is the cosine similarity?

Solution: The dot product of the vector is $1 \times 0 + 1 \times 1 + 0 \times 1 = 1$. The norm of the vectors A and B is $\sqrt{2}$.

So, the cosine similarity using Eq. (13.9) is given as $\frac{1}{\sqrt{2}\sqrt{2}} = \frac{1}{2} = 0.5$

Now, let us discuss the types of clustering algorithms, which include hierarchical, partitional, density-based and grid-based algorithms.

Scan for information on *'Taxonomy of Clustering Algorithms'*

13.3 HIERARCHICAL CLUSTERING ALGORITHMS

Hierarchical methods produce a nested partition of objects with hierarchical relationships among objects. Often, the hierarchy relationship is shown in the form of a dendrogram.

Hierarchical methods include categories, agglomerative methods and divisive methods. In agglomerative methods, initially all individual samples are considered as a cluster, that is, a cluster with a single element. Then, they are merged and the process is continued to get a single cluster. Divisive methods use another kind of philosophy, where a single cluster of all samples of the dataset taken initially is chosen and then partitioned. This partition process is continued until the cluster is split into smaller clusters.

Agglomerative methods merge clusters to reduce the number of clusters. This is repeated each time while merging two closest clusters to get a single cluster. The procedure of agglomerative clustering is given as follows:

Algorithm 13.1: Agglomerative Clustering

1. Place each N sample or data instance into a separate cluster. So, initially N clusters are available.
2. Repeat the following steps until a single cluster is formed:
   (a) Determine two most similar clusters.
   (b) Merge the two clusters into a single cluster reducing the number of clusters as N–1.
3. Choose resultant clusters of step 2 as result.

All the clusters that are produced by hierarchical algorithms have equal diameters. The main disadvantage of this approach is that once the cluster is formed, it is an irreversible decision.

13.3.1 Single Linkage or MIN Algorithm

Hierarchical clustering algorithms produce nested clusters, which can be visualized as a hierarchical tree or dendrogram. The idea behind this approach is proximity among clusters. In single linkage algorithm, the smallest distance (x, y), where x is from one cluster and y is from another cluster, is the distance between all possible pairs of the two groups or clusters (or simply the smallest distance of two points where points are in different clusters) and is used for merging the clusters. This corresponds to finding of minimum spanning tree (MST) of a graph.

The distance measures between individual samples or data points is already demonstrated in the previous section 13.2. To understand the single linkage algorithm, go through the following numerical problem that involves finding the distance between clusters.

Example 13.4: Consider the array of points as shown in the following Table. 13.4.

Table 13.4: Sample Data

Objects	X	Y
0	1	4
1	2	8
2	5	10
3	12	18
4	14	28

Apply single linkage algorithm.

Solution: The similarity table among the variables is computed as shown in Table 13.4. Euclidean distance is computed as shown in the following Table 13.5.

Table 13.5: Proximity Matrix

Objects	0	1	2	3	4
0	–	4.123	7.211	17.804	27.295
1		–	3.606	14.142	23.324
2			–	10.630	20.124
3				–	10.198
4					–

The minimum distance is 3.606. Therefore, the items 1 and 2 are clustered together. The resultant is shown in Table 13.6.

Table 13.6: After Iteration 1

Clusters	{1, 2}	0	3	4
{1, 2}	–	4.123	10.630	20.124
0		–	17.804	27.295
3			–	10.198
4				–

The distance between the group {1, 2} and items {0}, {3} and {4} is computed using the formula:

$$D_{SL}(C_i, C_j) = \text{minimum}_{a \in C_i, b \in C_j} d(a, b) \quad (13.10)$$

Here, D_{SL} is the single linkage distance, C_i, C_j are clusters and $d(a, b)$ is the distance between the elements a and b.

Thus, the distance between {1, 2} and {0} is:

Minimum {{1, 0}, {2, 0}} = Minimum {4.123, 7.211} = 4.123

The distance between {1, 2} and {3} is given as:

Minimum {{1, 3}, {2, 3}} = Minimum {14.412, 10.630} = 10.630

The distance between {1, 2} and {4} is given as:

Minimum {{1, 4}, {2, 4}} = Minimum {23.324, 20.124} = 20.124

This is shown in Table 13.6. The minimum distance of Table 13.6 is 4.123.

Therefore, {0, 1, 2} is clustered together. This result is shown in Table 13.7.

Table 13.7: After Iteration 2

Clusters	{0, 1, 2}	3	4
{0, 1, 2}	–	10.630	20.124
3		–	10.198
4			–

Thus, the distance between {0, 1, 2} and {3} using Eq. (13.10) is given as

Minimum {{0, 3}, {1, 3}, {2, 3}} = Minimum {17.804, 14.142, 10.630} = 10.630.

Thus, the distance between {0, 1, 2} and {4} is:

Minimum {{0, 4}, {1, 4}, {2, 4}} = Minimum {27.295, 23.324, 20.124} = 20.124.

This is shown in Table 13.7. The minimum is 10.198.

Therefore, the items {3, 4} are merged. And, the items {1, 2, 3} and {4, 5} are merged. The resultant is shown in Table 13.8.

Table 13.8: After Iteration 3

Clusters	{0, 1, 2}	{3, 4}
{0, 1, 2}	–	?
{3, 4}		–

The computation of ? in Table 13.8 is not needed as no other items are left. Therefore, the clusters {0, 1, 2} and {3, 4} are merged.

Dendrograms are used to plot the hierarchy of clusters. Dendrogram for the above clustering is shown in the following Figure 13.2.

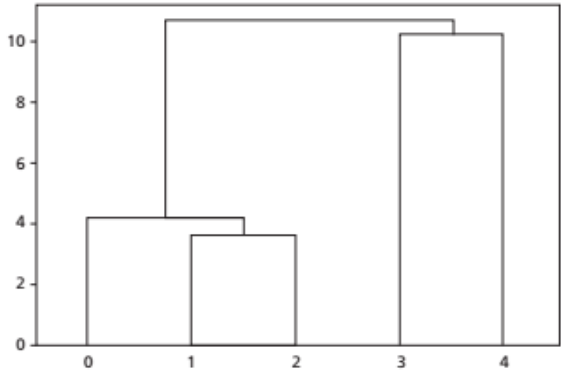


Figure 13.2: Dendrogram for Table 13.4

13.3.2 Complete Linkage or MAX or Clique

In complete linkage algorithm, the distance (x, y), (where x is from one cluster and y is from another cluster), is the largest distance between all possible pairs of the two groups or clusters (or simply the largest distance of two points where points are in different clusters) as given below. It is used for merging the clusters.

$$D_{CL}(C_i, C_j) = \text{maximum}_{a \in C_i,\ b \in C_j} d(a, b) \tag{13.11}$$

Dendrogram for the above clustering is shown in Figure 13.3.

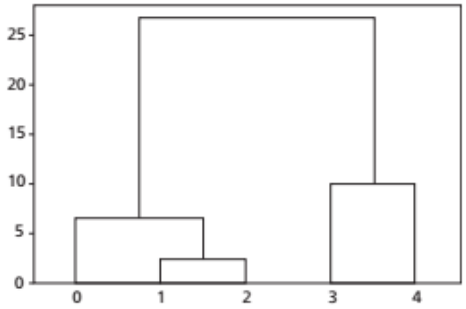


Figure 13.3: Dendrogram for Table 13.4

13.3.3 Average Linkage

In case of an average linkage algorithm, the average distance of all pairs of points across the clusters is used to form clusters. The average value computed between clusters c_i, c_j is given as follows:

$$D_{AL}(C_i, C_j) = \frac{1}{m_i m_j} \sum_{a \in C_i,\ b \in C_j} d(a, b) \tag{13.12}$$

Here, m_i and m_j are sizes of the clusters.

The dendrogram for Table 13.4 is given in Figure 13.4.

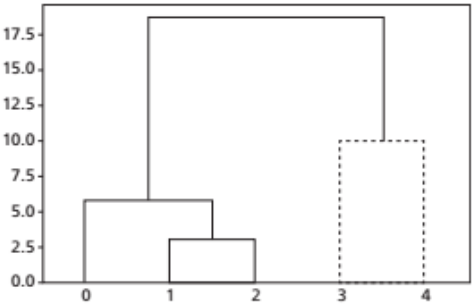


Figure 13.4: Dendrogram for Table 13.4

13.3.4 Mean-Shift Clustering Algorithm

Mean-shift is a non-parametric and hierarchical clustering algorithm. This algorithm is also known as mode seeking algorithm or a sliding window algorithm. It has many applications in image processing and computer vision.

There is no need for any prior knowledge of clusters or shape of the clusters present in the dataset. The algorithm slowly moves from its initial position towards the dense regions.

The algorithm uses a window, which is basically a weighting function. Gaussian window is a good example of a window. The radius of the kernel is called bandwidth. The entire window is called a kernel. The window is based on the concept of kernel density function and its aim is to find the underlying data distribution. The method of calculation of mean is dependent on the choice of windows. If a Gaussian window is chosen, then every point is assigned a weight that decreases as the distance from the kernel center increases. The algorithm is given below.

Algorithm 13.2: Mean-Shift Clustering

Step 1: Design a window.

Step 2: Place the window on a set of data points.

Step 3: Compute the mean for all the points that come under the window.

Step 4: Move the center of the window to the mean computed in step 3. Thus, the window moves towards the dense regions. The movement to the dense region is controlled by a mean shift vector. The mean shift vector is given as:

$$v_s = \frac{1}{K} \sum_{x_i \in S_k} (x_i - x) \tag{13.13}$$

Here, K is the number of points and S_k is the data points where the distance from data points x_i and centroid of the kernel x is within the radius of the sphere. Then, the centroid is updated as $x = x + v_s$.

Step 5: Repeat the steps 3–4 for convergence. Once convergence is achieved, no further points can be accommodated.

Advantages

1. No model assumptions
2. Suitable for all non-convex shapes
3. Only one parameter of the window, that is, bandwidth is required
4. Robust to noise
5. No issues of local minima or premature termination

Disadvantages

1. Selecting the bandwidth is a challenging task. If it is larger, then many clusters are missed. If it is small, then many points are missed and convergence occurs as the problem.
2. The number of clusters cannot be specified and user has no control over this parameter.

Scan for *'Additional Examples'*

13.4 PARTITIONAL CLUSTERING ALGORITHM

'*k*-means' algorithm is a straightforward iterative partitional algorithm. Here, *k* stands for the user specified requested clusters as users are not aware of the clusters that are present in the dataset. The *k*-means algorithm assumes that the clusters do not overlap. Therefore, a sample or data point can belong to only one cluster in the end. Also, this algorithm can detect clusters of shapes like circular or spherical.

Initially, the algorithm needs to be initialized. The algorithm can select *k* data points randomly or use the prior knowledge of the data. In most cases, in *k*-means algorithm setup, prior knowledge is absent. The composition of the cluster is based on the initial condition, therefore, initialization is an important task. The sample or data points need to be normalized for better performance. The concepts of normalization are covered in Chapter 3.

The core process of the *k*-mean algorithm is assigning a sample to a cluster, that is, assigning each sample or data point to the *k* cluster centers based on its distance and the centroid of the clusters. This distance should be minimum. As a new sample is added, new computation of mean vectors of the points for that cluster to which sample is assigned is required. Therefore, this iterative process is continued until no change of instances to clusters is noticed. This algorithm then terminates and the termination is guaranteed.

Algorithm 13.3: *k*-means Algorithm

Step 1: Determine the number of clusters before the algorithm is started. This is called *k*.

Step 2: Choose *k* instances randomly. These are initial cluster centers.

Step 3: Compute the mean of the initial clusters and assign the remaining sample to the closest cluster based on Euclidean distance or any other distance measure between the instances and the centroid of the clusters.

Step 4: Compute new centroid again considering the newly added samples.

Step 5: Perform the steps 3–4 till the algorithm becomes stable with no more changes in assignment of instances and clusters.

k-means can also be viewed as greedy algorithm as it involves partitioning *n* samples to *k* clusters to minimize Sum of Squared Error (SSE). SSE is a metric that is a measure of error that gives the sum of the squared Euclidean distances of each data to its closest centroid. It is given as:

$$SSE = \sum_{i=1}^{k} dist(c_i, x)^2 \tag{13.14}$$

Here, c_i is the centroid of the i^{th} cluster, x is the sample or data point and dist is the Euclidean distance. The aim of the *k*-means algorithm is to minimize SSE.

Advantages

1. Simple
2. Easy to implement

Disadvantages

1. It is sensitive to initialization process as change of initial points leads to different clusters.
2. If the samples are large, then the algorithm takes a lot of time.

How to Choose the Value of k?

It is obvious that k is the user specified value specifying the number of clusters that are present. Obviously, there are no standard rules available to pick the value of k. Normally, the k-means algorithm is run with multiple values of k and within group variance (sum of squares of samples with its centroid) and plotted as a line graph. This plot is called Elbow curve. The optimal or best value of k can be determined from the graph. The optimal value of k is identified by the flat or horizontal part of the Elbow curve.

Complexity

The complexity of k-means algorithm is dependent on the parameters like n, the number of samples, k, the number of clusters, $\Theta(nkId)$. I is the number of iterations and d is the number of attributes. The complexity of k-means algorithm is $O(n^2)$.

Example 13.5: Consider the following set of data given in Table 13.9. Cluster it using k-means algorithm with the initial value of objects 2 and 5 with the coordinate values (4, 6) and (12, 4) as initial seeds.

Table 13.9: Sample Data

Objects	X-coordinate	Y-coordinate
1	2	4
2	4	6
3	6	8
4	10	4
5	12	4

Solution: As per the problem, choose the objects 2 and 5 with the coordinate values. Hereafter, the objects' id is not important. The samples or data points (4, 6) and (12, 4) are started as two clusters as shown in Table 13.10.

Initially, centroid and data points are same as only one sample is involved.

Table 13.10: Initial Cluster Table

Cluster 1	Cluster 2
(4, 6)	(12, 4)
Centroid 1 (4, 6)	Centroid 2 (12, 4)

Iteration 1: Compare all the data points or samples with the centroid and assign to the nearest sample. Take the sample object 1 (2, 4) from Table 13.9 and compare with the centroid of the clusters in Table 13.10. The distance is 0. Therefore, it remains in the same cluster. Similarly, consider the remaining samples. For the object 1 (2, 4), the Euclidean distance between it and the centroid is given as:

$$\text{Dist (1, centroid 1)} = \sqrt{(2-4)^2 + (4-6)^2} = \sqrt{8}$$

$$\text{Dist (1, centroid 2)} = \sqrt{(2-12)^2 + (4-4)^2} = \sqrt{100} = 10$$

Object 1 is closer to the centroid of cluster 1 and hence assign it to cluster 1. This is shown in Table 13.11. Object 2 is taken as centroid point.

For the object 3 (6, 8), the Euclidean distance between it and the centroid points is given as:

$$\text{Dist (3, centroid 1)} = \sqrt{(6-4)^2 + (8-6)^2} = \sqrt{8}$$

$$\text{Dist (3, centroid 2)} = \sqrt{(6-12)^2 + (8-4)^2} = \sqrt{52}$$

Object 3 is closer to the centroid of cluster 1 and hence remains in the same cluster 1.

Proceed with the next point object 4(10, 4) and again compare it with the centroids in Table 13.10.

$$\text{Dist (4, centroid 1)} = \sqrt{(10-4)^2 + (4-6)^2} = \sqrt{40}$$

$$\text{Dist (4, centroid 2)} = \sqrt{(10-12)^2 + (4-4)^2} = \sqrt{4} = 2$$

Object 4 is closer to the centroid of cluster 2 and hence assign it to the cluster table. Object 4 is in the same cluster. The final cluster table is shown in Table 13.11.

Obviously, Object 5 is in Cluster 3. Recompute the new centroids of cluster 1 and cluster 2. They are (4, 6) and (11, 4), respectively.

Table 13.11: Cluster Table After Iteration 1

Cluster 1	Cluster 2
(4, 6)	(10, 4)
(2, 4)	(12, 4)
(6, 8)	
Centroid 1 (4, 6)	Centroid 2 (11, 4)

The second iteration is started again with the Table 13.11.

Obviously, the point (4, 6) remains in cluster 1, as the distance of it with itself is 0. The remaining objects can be checked. Take the sample object 1 (2, 4) and compare with the centroid of the clusters in Table 13.12.

$$\text{Dist (1, centroid 1)} = \sqrt{(2-4)^2 + (4-6)^2} = \sqrt{8}$$

$$\text{Dist (1, centroid 2)} = \sqrt{(2-11)^2 + (4-4)^2} = \sqrt{81} = 9$$

Object 1 is closer to centroid of cluster 1 and hence remains in the same cluster. Take the sample object 3 (6, 8) and compare with the centroid values of clusters 1 (4, 6) and cluster 2 (11, 4) of the Table 13.12.

$$\text{Dist (3, centroid 1)} = \sqrt{(6-4)^2 + (8-6)^2} = \sqrt{8}$$

$$\text{Dist (3, centroid 2)} = \sqrt{(6-11)^2 + (8-4)^2} = \sqrt{41}$$

Object 3 is closer to centroid of cluster 1 and hence remains in the same cluster. Take the sample object 4 (10, 4) and compare with the centroid values of clusters 1 (4, 6) and cluster 2 (11, 4) of the Table 13.12:

Dist (4, centroid 1) = $\sqrt{(10-4)^2 + (4-6)^2} = \sqrt{40}$

Dist (3, centroid 2) = $\sqrt{(10-11)^2 + (4-4)^2} = \sqrt{1} = 1$

Object 3 is closer to centroid of cluster 2 and hence remains in the same cluster. Obviously, the sample (12, 4) is closer to its centroid as shown below:

Dist (5, centroid 1) = $\sqrt{(12-4)^2 + (4-6)^2} = \sqrt{68}$

Dist (5, centroid 2) = $\sqrt{(12-11)^2 + (4-4)^2} = \sqrt{1} = 1$. Therefore, it remains in the same cluster. Object 5 is taken as centroid point.

The final cluster Table 13.12 is given below:

Table 13.12: Cluster Table After Iteration 2

Cluster 1	Cluster 2
(4, 6)	(10, 4)
(2, 4)	(12, 4)
(6, 8)	
Centroid (4, 6)	Centroid (11, 4)

There is no change in the cluster Table 13.12. It is exactly the same; therefore, the *k*-means algorithm terminates with two clusters with data points as shown in the Table 13.12.

Scan for *'Additional Examples'*

13.5 DENSITY-BASED METHODS

Density-based spatial clustering of applications with noise (DBSCAN) is one of the density-based algorithms. Density of a region represents the region where many points above the specified threshold are present. In a density-based approach, the clusters are regarded as dense regions of objects that are separated by regions of low density such as noise. This is same as a human's intuitive way of observing clusters.

The concept of density and connectivity is based on the local distance of neighbours. The functioning of this algorithm is based on two parameters, the size of the neighbourhood (ε) and the minimum number of points (m).

1. Core point – A point is called a core point if it has more than specified number of points (m) within ε-neighbourhood.
2. Border point – A point is called a border point if it has fewer than 'm' points but is a neighbour of a core point.
3. Noise point – A point that is neither a core point nor border point.

The main idea is that every data point or sample should have atleast a minimum number of neighbours in a neighbourhood. The neighbourhood of radius ε should have atleast m points. The notion of density connectedness determines the quality of the algorithm.

The following connectedness measures are used for this algorithm.

1. Direct density reachable – The point X is directly reachable from Y, if:
   (a) X is the ε-neighborhood of Y
   (b) Y is a core point
2. Densely reachable – The point X is densely reachable from Y, if there is a set of core points that leads from Y to X.
3. Density connected – X and Y are densely connected if Z is a core point and thus points X and Y are densely reachable from Z.

Algorithm 13.4: DBSCAN

Step 1: Randomly select a point p. Compute distance between p and all other points.

Step 2: Find all points from p with respect to its neighbourhood and check whether it has minimum number of points m. If so, it is marked as a core point.

Step 3: If it is a core point, then a new cluster is formed, or existing cluster is enlarged.

Step 4: If it is a border point, then the algorithm moves to the next point and marks it as visited.

Step 5: If it is a noise point, they are removed.

Step 6: Merge the clusters if it is mergeable, $dist\ (c_i, c_j) < \varepsilon$.

Step 7: Repeat the process 3–6 till all points are processed.

Advantages

1. No need for specifying the number of clusters beforehand
2. The algorithm can detect clusters of any shapes
3. Robust to noise
4. Few parameters are needed

The complexity of this algorithm is $O(n \log n)$.

13.6 GRID-BASED APPROACH

Grid-based approach is a space-based approach. It partitions space into cells, the given data is fitted on the cells for cluster formation.

There are three important concepts that need to be mastered for understanding the grid-based schemes. They are:

1. Subspace clustering
2. Concept of dense cells
3. Monotonicity property

Let us discuss about them.

Subspace Clustering

Grid-based algorithms are useful for clustering high-dimensional data, that is, data with many attributes. Some data like gene data may have millions of attributes. Every attribute is called a dimension. But all the attributes are not needed, as in many applications one may not require all the attributes. For example, an employee's address may not be required for profiling his diseases. Age may be required in that case. So, one can conclude that only a subset of features is required. For example, one may be interested in grouping gene data with similar characteristics or organs that have similar functions.

Finding subspaces is difficult. For example, N dimensions may have 2^{N-1} subspaces. Exploring all the subspaces is a difficult task. Here, only the CLIQUE algorithms are useful for exploring the subspaces. CLIQUE (Clustering in Quest) is a grid-based method for finding clustering in subspaces. CLIQUE uses a multiresolution grid data structure.

Concept of Dense Cells

CLIQUE partitions each dimension into several overlapping intervals and intervals it into cells. Then, the algorithm determines whether the cell is dense or sparse. The cell is considered dense if it exceeds a threshold value, say τ. Density is defined as the ratio of number of points and volume of the region. In one pass, the algorithm finds the number of cells, number of points, etc. and then combines the dense cells. For that, the algorithm uses the contiguous intervals and a set of dense cells.

Algorithm 13.5: Dense Cells

Step 1: Define a set of grid points and assign the given data points on the grid.

Step 2: Determine the dense and sparse cells. If the number of points in a cell exceeds the threshold value τ, the cell is categorized as dense cell. Sparse cells are removed from the list.

Step 3: Merge the dense cells if they are adjacent.

Step 4: Form a list of grid cells for every subspace as output.

Monotonicity Property

CLIQUE uses anti-monotonicity property or apriori property of the famous apriori algorithm. It means that all the subsets of a frequent item should be frequent. Similarly, if the subset is infrequent, then all its supersets are infrequent as well. Based on the apriori property, one can conclude that a k-dimensional cell has r points if and only if every $(k-1)$ dimensional projections of this cell have atleast r points. So like association rule mining that uses apriori rule, the candidate dense cells are generated for higher dimensions. The algorithm works in two stages as shown below.

Algorithm 13.6: CLIQUE

Stage 1:

Step 1: Identify the dense cells.

Step 2: Merge dense cells c_1 and c_2 if they share the same interval.

(*Continued*)

Step 3: Generate Apriori rule to generate $(k+1)^{th}$ cell for higher dimension. Then, check whether the number of points cross the threshold. This is repeated till there are no dense cells or new generation of dense cells.

Stage 2:

Step 1: Merging of dense cells into a cluster is carried out in each subspace using maximal regions to cover dense cells. The maximal region is an hyperrectangle where all cells fall into.

Step 2: Maximal region tries to cover all dense cells to form clusters.

In stage two, CLIQUE starts from dimension 2 and starts merging. This process is continued till the n-dimension.

Advantages of CLIQUE

1. Insensitive to input order of objects
2. No assumptions of underlying data distributions
3. Finds subspace of higher dimensions such that high-density clusters exist in those subspaces

Disadvantage

The disadvantage of CLIQUE is that tuning of grid parameters, such as grid size, and finding optimal threshold for finding whether the cell is dense or not is a challenge.

13.7 PROBABILITY MODEL-BASED METHODS

In the earlier clustering algorithms, the samples were assigned to the clusters permanently. Also, the samples were not allowed to be present in two clusters. In short, the clusters were non-overlapping. In model-based schemes, the sample is associated with a probability for membership. This assignment based on probability is called soft assignment. Soft assignments are dynamic as compared to hard assignments which are static. Also, the sample can belong to more than one clusters. This is acceptable as person X can be a father, a manager as well as a member of a prestigious club. In short, person X has different roles in life.

A 'Model' means a statistic method like probability distributions associated with its parameters. In the EM algorithm, the model assumes data is generated by a process and the focus is to find the distribution that observes that data. There are two probability based soft assignment schemes that are discussed here. One is fuzzy-C means (FCM) clustering algorithm and another is EM algorithm. EM algorithm is discussed in detail in Chapter 2.

13.7.1 Fuzzy Clustering

Fuzzy C-Means is one of the most widely used algorithms for implementing the fuzzy clustering concept. In fuzzy clustering, an object can belong to more than one cluster. Let us assume two clusters c_i and c_j, then an element, say x, can belong to both the clusters. The strength of association of an object with the cluster is given as w_{ij}. The value of w_{ij} lies between zero and one. The sum of weights of an object, if added, gives 1.

Like in *k*-means algorithm, the centroid of the cluster, c_j is computed. The membership weight is inversely proportional to the distance between object and centroid computed in earlier pass.

Algorithm 13.7: Fuzzy C-Means

Step 1: Choose the clusters, *k*, randomly.

Step 2: Assign weights w_{ij} of objects to clusters randomly.

Step 3: Compute the centroid:

$$c_j = \frac{\sum_{i=1}^{m} w_{ij}^p x_i}{\sum_{i=1}^{m} w_{ij}^p} \quad (13.15)$$

Step 4: The Sum of Squared Error (SSE) is computed as:

$$SSE = \sum_{j=1}^{k} \sum_{i=1}^{m} w_{ij}^p dist\,(x_i,\ c_j)^2 \quad (13.16)$$

Step 5: Minimize SSE to update the membership weights. Here, p is a fuzzifier whose value ranges from 1 to ∞. This parameter determines the influence of the weights. If p is 1, then fuzzy-c acts like *k*-means algorithm. A large weight results in a smaller value of the membership and hence more fuzziness. Typically, p value is 2.

Step 6: Repeat steps 3–5 till convergence is reached, which mean when there is no change in weights exceeding the threshold value.

Advantages and Disadvantages of FCM

The advantages and disadvantages of the FCM algorithm are given in Table 13.13.

Table 13.13: Advantages and Disadvantages of FCM Algorithm

S.No.	Advantages	Disadvantages
1.	Minimum intra class variance	The quality depends on the initial choice of weights
2.	Robust to noise	Local minima rather than global minimum

13.7.2 Expectation-Maximization (EM) Algorithm

Like FCM algorithm, in EM algorithm too there are no hard assignments but there are overlapping clusters.

In this scheme, clustering is done by statistical models. What is a model? A statistical model is described in terms of a distribution and a set of parameters. The data is assumed to be generated by a process and the focus is to describe the data by finding a model that fits the data. In fact, data is assumed to be generated by multiple distributions – a mixture model. As mostly gaussian distributions are used, it also called as GMM (Gaussian mixture model).

Given a mix of distributions, data can be generated by randomly picking a distribution and generating the point. The basics of Gaussian distribution are given in Chapter 3. One can recollect

that Gaussian distribution is a bell-shaped curve. The function of gaussian distribution is given as follows:

$$N(x : \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)}{2\sigma^2}}$$

Two parameters characterize this function – mean and standard deviation. Sometimes, variance can also be used as it is the square of standard deviation. When the mean is zero, the peak of the bell-shaped curve occurs. Standard deviation is the spread of the shape. The above function is called probability distribution function that tells how to find the probability of the observed point x.

The same gaussian function can be extended for multivariate too. In 2D, the mean is also a vector and variance takes the form of covariance matrix. Chapter 3 discusses these important concepts.

Let us assume that:

k = Number of distributions

n = Number of samples

$\theta = \{\theta_1, \theta_2, \theta_3, \cdots, \theta_k\}$, a set of parameters that are associated with the distributions.

θ_j is the parameter of the j^{th} probability distribution.

Then, $p(x_i \mid \theta_j)$ is the probability of i^{th} object coming from the j^{th} distribution. The probability that j^{th} distribution to be chosen is given by the weight w_j, $1 \le j \le k$. Then,

$$p(x_i \mid \theta) = \sum_{j=1}^{k} w_j p_j (x \mid \theta_j)$$

If all the points are generated randomly, then the entire set of objects can be denoted as:

$$p(\chi \mid \theta) = \prod_{i=1}^{n} p(x_i \mid \theta) = \prod_{i=1}^{n} \sum_{j=1}^{k} w_j p_j (x \mid \theta_j)$$

Every data is assumed to be generated by a distribution. To describe the data point, the corresponding distribution along with its parameters should be known. So, the parameters should be learnt. Since Gaussians are assumed, the probability that data belongs to that distribution should be learnt. This is given as:

$$p(\chi \mid \theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

The parameters μ and σ should be chosen such that the above equation is maximized. This is known as maximum likelihood principle. This kind of estimation is called maximum likelihood estimation. Often, log of likelihood function is used and maximized.

Since there are no multiple distributions as only Gaussians are assumed, the problem is reduced to choosing parameters only. The objective of the EM algorithm is to maximize the likelihood of observation by selecting proper parameters. So, there is a need to find three parameters weights of the gaussian, mean and variance.

The EM algorithm works in two stages:

1. Expectation step – In this step, probability of each data point generated by *K*-gaussian function is computed. This is just a soft assignment.
2. Maximization step – In this step, the parameters are updated.

Algorithm 13.8: Expectation-Maximization

Step 1: Select the parameters randomly.

Step 2: In expectation stage, for each point the condition probability is computed.

Step 3: In maximization stage, the new parameters are computed.

Step 4: Repeat the steps 2–3 till change is minimal within the threshold value or parameters do not change at all.

13.8 CLUSTER EVALUATION METHODS

Scan for information on 'Purity', 'Evaluation based on Ground Truth', and 'Similarity-based Measures'

Evaluation of clustering algorithms is a difficult task, as often, no benchmark data is available as in classification. Also, in clustering algorithms, domain knowledge is absent most of the times. So, clustering algorithms' validation is difficult as compared to the validation of classification algorithms. There are three types of measures that can be used for cluster validation:

1. Internal
2. External
3. Relative

Internal metrics quantify the quality of clustering without the use of any external information or knowledge. External metrics use the ground truth or externally supplied labels to quantify the quality of the validation. In relative measure, different cluster algorithms are compared, or the algorithm is run with multiple parameter values. This measure helps in finding optimal clusters.

Basically, two measures of information measures, that is cohesion and separation, are based on the idea that the objects in the cluster should be same and objects across clusters should be distinct. Alternatively, the average distance within the cluster should be small and average distance across the clusters should be large.

Cohesion and Separation

Cohesion (or compact) measures how close the samples are inside the cluster. This ensures that the clusters are homogeneous. Cohesion is measured as sum of squared errors between the samples and the centroid. The within cluster sum is given as:

$$\sum_{k=1}^{N} \sum_{x_i \in C} (x_i - m_j)^2 \tag{13.17}$$

Here, N is the number of clusters, C is the set of centroids, x_i is the centroid and m_j is the samples. A lower within cluster variation is a necessary condition for greater compactness and high cohesion.

Separation indicates how well a sample differs from other clusters. This is measured as the weighted sum of the differences of the centroid of the dataset and the centroid of the generated clusters. This is given as:

$$\sum_{x_i \in C} C_i (x - x_i)^2 \tag{13.18}$$

Here, x is the centroid of the entire dataset, x_i is the centroid of the clusters and C_i is the size of the clusters. A larger distance is required for well-separated clusters so that the clusters are perfectly distinct. Sometimes, the connectivity between a sample and other member of that cluster may be important, indicating the sort of samples that can be put into the clusters. The connectivity value ranges from 0 to infinity. Dunn index can be computed as:

$$Index = \frac{\alpha \times separation}{\beta \times compactness} \tag{13.19}$$

Here, α and β are parameters. Dunn index is a useful measure that can combine both cohesion and separation.

Silhouette Co-efficient

Silhouette coefficient combines both cohesion and separation. The Silhouette coefficient measures the average distance between clusters. It is given as follows:

$$s_i = \frac{b_i - a_i}{\max\{b_i, a_i\}} \tag{13.20}$$

Here, a_i is the distance between the sample and centroid of the same cluster and b_i is the distance between the sample and the nearest centroid. The silhouette coefficients of the individual objects can be summed to get for the entire cluster as S, given as:

$$S = \frac{1}{N} \sum_{i=1}^{n} s_i \tag{13.21}$$

The value of the silhouette coefficient s_i is between –1 and +1. When it is closer to 1, the clusters are well formed. The value is zero when the data points are between two clusters and negative when the clusters are not formed correctly.