

**GURU JAMBHESHWAR UNIVERSITY OF
SCIENCE & TECHNOLOGY, HISAR**

**DIRECTORATE OF DISTANCE
EDUCATION**

MACHINE LEARNING

(MCA-31)

MCA, 3rd Semester

Authored By:

Dr. Nitin Goyal

Assistant Professor

Department of Computer Science and Engineering,

School of Engineering and Technology,

Central University of Haryana, Mahendragarh, Haryana

ABSTRACT

Machine learning is the paradigm of computer science that deals with the incorporation of adding capabilities into the machine to learn. The learning takes place to make decision, predictions, categorical classifications etc. It is an interesting fact that how a machine learns is similar to how humans learn. The humans learn by the examples or by the experiences. The same way machine learns. For example, a toddler is taught by showing the *apple*; that this round-shaped red fruit is *apple*. Next time, when the toddler is asked to name the fruit, he recalls the example and classifies the fruit as *apple*. Similarly, machine learns, this form of machine learning with supervisory signal of outcome label is known as **supervised machine learning**. The **unsupervised machine learning** categorizes data as per the similarity among the instances; just like a new batch of students in the first year; divides into clusters till final year as per similarity in their behaviors. A toddler will repeat his good behavior if he gets rewards for his idealistic behavior; he will never touch a hot pan if he had experienced the heat earlier. This is the form of reinforcement learning. The **reinforcement machine learning** aims to increase the cumulative reward of successful steps also.

The book is written to give in detail insights to the basic concepts and techniques of machine learning to support futuristic job profiles. Unit I begin with the basic concepts of machine learning along with its types and applications. It includes the description of supervised and unsupervised machine learning with examples. Later on, the **dimensionality reduction** methods such as PCA, LDA and GDA are described in the unit. Regression and classification are the form of supervised machine learning. Regression is prediction of continuous value while classification is categorization into discrete values. This concept is elaborated in Unit II. Decision tree is explained as a classification model in Unit II. **Artificial Neural Network (ANN)** is the computational network similar to human brain with interconnected biological neurons. Unit III encompasses the detailed description of ANN with perceptron network and multilayer neural networks. **Naïve Bayes, Support Vector Machines, k-nearest neighbors** are frequently used classifiers. The basic and mathematical introduction to these classifiers is given in Unit IV.

Preface

Machine learning has emerged as one of the most fascinating and dynamic fields in computer science. With the advent of Big Data and advancements in computing technologies, machine learning has gained tremendous popularity in recent years. This book aims to provide a comprehensive introduction to machine learning, covering both the fundamental concepts and practical applications.

In this book, we will begin with the basics of machine learning, including supervised and unsupervised learning, and move on to more advanced topics such as deep learning and dimensionality reduction. The book is designed for readers with a strong foundation in mathematics and programming, although we will start with the basics of linear algebra and statistics.

We will explore various machine learning techniques such as regression, decision trees, support vector machines, bayes learning, and instance-based learning. We will also delve into the exciting world of artificial neural networks, including the perceptron and multilayer neural networks.

Through practical examples and detailed explanations, we will demonstrate how to apply these concepts to real-world problems. We hope that this book will be a valuable resource for students, researchers, and practitioners who wish to learn about the latest developments in machine learning and its applications.

Organization of the book

In Chapter 1, It serves as a foundation for the rest of the book, introducing the concept of machine learning, its types, and applications. Readers will learn about the different types of machine learning, including supervised, unsupervised, semi-supervised, and reinforcement learning. Additionally, the chapter provides examples of how machine learning is used in various fields, such as healthcare, finance, and marketing.

In Chapter 2, readers will learn about the two fundamental types of machine learning: supervised and unsupervised learning. The chapter will begin with an in-depth look at supervised learning, which involves the use of labeled data to train a machine learning model. The different types of supervised learning, including regression and classification, will be explored in detail. Additionally, readers will learn about the key differences between regression and classification and how to determine which method to use for a specific task.

In Chapter 3, readers will learn about the important concept of Dimensional Reduction, which is crucial for dealing with high-dimensional data. The chapter will cover various methods of Dimensionality Reduction, including Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and Generalized Discriminant Analysis (GDA). The readers will learn about the advantages and limitations of each method, and how they can be applied to different types of data. The chapter will also explore the applications and usage of Dimensional Reduction in various fields such as image and speech recognition, and bioinformatics.

In Chapter 4, readers will learn about the concept of regression, its need, and applications. The chapter covers various types of regression, with a focus on linear regression, including its definition, assumptions, and the method of fitting a line to the data. The chapter also provides examples of linear regression, including the interpretation of coefficients, goodness of fit, and residuals. Finally, the chapter discusses multilinear regression and provides examples to illustrate the concept.

In Chapter 5, readers will learn about decision tree, its representation, and when to use the decision tree learning algorithm. They will understand the basic decision tree learning algorithm, entropy measures, and information gain measures. The chapter includes example problems to illustrate the ID3 algorithm, providing readers with a better understanding of how decision tree learning works and its applications in machine learning. By the end of this chapter, readers will have a strong grasp of decision trees and be able to apply this knowledge to solve practical problems.

In Chapter 6 is an important part of the book as it introduces readers to the basics of Artificial Neural Network (ANN), which is a fundamental concept in machine learning and deep learning. Readers will learn about the biological motivation behind ANN and its representation, which includes the architecture of the neural network and its components. Additionally, readers will gain insight into the types of problems that can be solved using ANN, which will help them in deciding when to use ANN for a given task. Overall, this chapter provides a strong foundation for readers to further explore the world of ANN and deep learning.

In Chapter 7, It provides valuable insights into the world of artificial neural networks and is an important read for anyone interested in the field of machine learning. The chapter starts with an introduction to Perceptron, followed by its representation power and the training rule. The chapter then dives into gradient descent and delta rule, which are key components of Perceptron training. By going through this chapter, readers will gain a thorough understanding of Perceptron and its role in the development of more advanced

artificial neural networks. They will also learn about the different training rules used in Perceptron and the techniques used to optimize the training process

In Chapter 8, readers will learn about multilayer neural networks, which are an extension of the single-layer perceptron. The chapter explains how these networks can be used to solve more complex problems, and how they can be trained using the backpropagation algorithm. Readers will also gain an understanding of the different types of units used in multilayer neural networks, such as differentiable threshold units. Additionally, the chapter covers topics such as convergence and local minima, which are important concepts in neural network training. Finally, readers will be introduced to deep learning, which is a type of multilayer neural network used for tasks such as image recognition and natural language processing.

In Chapter 9, introduces readers to Bayesian Learning, a powerful technique for modeling complex systems and making predictions. Readers will first be introduced to Bayes' theorem and how it can be used to make predictions. Then, readers will learn about Naïve Bayes Classifiers, a popular algorithm used for text classification and other applications. By the end of this chapter, readers will have a solid understanding of Bayesian Learning and how it can be applied to real-world problems. This chapter will be particularly useful for readers interested in natural language processing, image recognition, and other fields where classification is important.

In Chapter 10, readers will learn about instance-based learning, which is a type of machine learning that uses the closest available examples to make predictions or decisions. The chapter specifically covers the nearest neighbor learning algorithm and provides insights on the K-Nearest Neighbor algorithm. Readers will gain an understanding of how instance-based learning algorithms can be used to classify and predict data based on similarity with previously observed instances. This chapter contributes to the book by providing readers with an overview of another type of machine learning that can be applied to various problem domains.

In Chapter 11, "Support Vector Machines," is the final chapter of this book on machine learning. In this chapter, readers will learn about support vector machines, which are widely used for classification and regression tasks. The chapter starts with an introduction to support vector machines, followed by a discussion of optimal separation and kernels. The chapter concludes with an exploration of extensions to the support vector machine. By reading this chapter, readers will gain a comprehensive understanding of support vector machines and their applications in machine learning.

Table of Contents

Abstract	i
Preface	ii

	Unit 1	Page No
Chapter 1	Introduction to Machine Learning	1
1.1	What Is Machine Learning?	1
1.1.1	The Foundations of Machine Learning	
1.1.2	Stages of Machine Learning	4
1.2	Types of Machine Learning	
1.2.1	Supervised Learning	
1.2.2	Unsupervised Learning	
1.2.3	Semi-Supervised Learning	
1.2.4	Reinforcement Learning	
1.3	Examples of Machine Learning	8
1.4	Applications of Machine Learning	9
1.5	Learning Associations	11
1.5.1	Classical Example of Learning Associations	
1.5.2	Methods of learning associations	
Chapter 2	Supervised and Unsupervised Learning	13
2.1	Supervised Learning	14
2.1.1	Regression	
2.1.2	Classification	
2.1.3	Regression vs Classification	
2.2	Unsupervised Learning	21
2.2.1	Clustering	
2.2.2	K-Means and Hierarchical Clustering	
2.2.3	Self-Organizing Feature Map (SOM) and Algorithm	

Chapter 3	Dimensional Reduction	25
	3.1 Dimensional Reduction	25
	3.1.1 Introduction	
	3.1.2 Importance of Dimensionality Reduction.	
	3.1.3 Advantages and Disadvantage of Dimensionality Reduction	
	3.1.4 Methods for Dimensionality Reduction	
	3.2 Methods of Dimensionality Reduction	29
	3.2.1 Principal Component Analysis (PCA)	
	3.2.2 Linear Discriminant Analysis (LDA)	
	3.2.3 Generalized Discriminant Analysis (GDA)	
	3.3 Applications and Usage of Dimensional Reduction	35

Unit 2

Chapter 4	Regression	41
	4.1 Need and Application of Regression	42
	4.2 Linear Regression	44
	4.2.1 Various Types of Regression	
	4.2.2 Simple Linear Regression	
	4.2.3 Principal advantages of linear regression	
	4.3 Linear Regression Examples	48
	4.4 Multi-Linear Regression and Examples	49
	4.4.1 Significance and Usage	
Chapter 5	Decision Tree	53
	5.1 Introduction to Decision Tree	53
	5.1.1 Decision Tree Benefits	
	5.2 Decision Tree Representation	56
	5.2.1 Example of Decision Tree	
	5.2.2 Advantages and Disadvantages of CART	
	5.3 Appropriate Problem for Decision Tree Learning Algorithm	60
	5.4 Basic Decision Tree Learning Algorithm	61

5.4.1 Decision Tree Training	
5.4.2 Predicting using Decision Tree:	
5.4.3 Assumptions	
5.5 Entropy Measures	62
5.6 Information Gain Measures	62
5.7 Example Problems for Illustrating ID3	64
5.7.1 Metrics used in ID3: Entropy and Information gain	
5.7.2 Application of Information Gain and Entropy on COVID dataset to identify best feature	

Unit 3

Chapter 6	Artificial Neural Network (ANN)	73
6.1	Introduction of ANN	73
6.1.1	How do they act?	
6.2	Biological Motivation	74
6.3	Neural Network Representation	77
6.3.1	Artificial Neuron	
6.3.2	Activation Functions	
6.3.2.1	Sigmoid Activation Function	
6.3.2.2	Hyperbolic Tangent Activation Function	
6.3.2.3	Rectified Linear Units Activation Function	
6.3.3	How does Neural Networks Work?	
6.3.4	How does the Neural Networks Learn?	
6.4	Appropriate Problems for Neural Network Learning	85
Chapter 7	Perceptron	87
7.1	Perceptron	87
7.1.1	Perceptron Types	
7.1.2	Advantages and Disadvantage of Perceptrons	
7.1.3	Characteristics of Perceptron Model	
7.2	Representation Power of Perceptron	89

7.3	Perceptron Training Rule	91
7.3.1	Perceptron Function	
7.3.2	Input of Perceptron	
7.3.3	Perceptron Activation Functions	
7.3.4	Output of Perceptron	
7.3.5	Error in Perceptron	
7.3.6	Perceptron: Decision Function	
7.3.7	Bias Unit	
7.3.8	Output	
7.4	Gradient Descent and Delta Rule	95
7.4.1	Introduction and Basic	
7.4.2	The Gradient Descent Procedure	
7.4.3	Convexity and Local Minima	
7.4.4	Stochastic Gradient Descent	
7.4.5	Delta Rule	
Chapter 8	Multi-layer Neural Network	103
8.1	Multilayer Neural Network	103
8.1.1	Advantages and Disadvantages of Multilayer Neural Network	
8.2	A Differentiable Threshold Unit	105
8.3	Convergence and Local Minima	106
8.3.1	Convergence	
8.3.2	Local Minima	
8.4	Backpropagation Algorithm	107
8.4.1	Why do we need Backpropagation algorithm?	
8.4.2	Categories of Backpropagation	
8.4.3	Working of Backpropagation Algorithm	
8.5	Deep Learning	113
8.5.1	Deep learning's workings	
8.5.2	Deep learning application	
8.5.3	Difference between Machine Learning and Deep Learning.	
8.5.4	Future of Deep Learning.	

Unit 4

Chapter 9	Bayesian Learning	119
9.1	Introduction	119
9.1.1	Features of Bayesian learning methods	
9.2	Bayes Theorem	121
9.2.1	Bayes Theorem Applications	
9.3	Naïve Bayes Classifiers	125
Chapter 10	Instance-Based Learning	131
10.1	Nearest Neighbour Learning	131
10.2	Remarks on K- Nearest Neighbour Algorithm	132
10.2.1	Example of KNN classifier:	
10.2.2	Why KNN Algorithm is required?	
10.2.3	How does KNN function?	
10.2.4	How is k in the K-NN Algorithm to be chosen?	
10.2.5	The KNN Algorithms advantages include	
10.2.6	Disadvantages of KNN Algorithm	
Chapter 11	Support Vector Machines	143
11.1	Introduction	143
11.1.1	Types of SVM	
11.1.2	Hyperplane and Support Vectors in the SVM algorithm	
11.1.3	Support Vectors	
11.1.4	How does SVM works?	
11.2	Optimal Separation	149
11.3	Kernels	149
11.4	Extensions to the Support Vector Machine	149

Chapter 1

Introduction to Machine Learning

“Computers are able to see, hear and learn. Welcome to the future.”

—Dave Waters

The evolution of computers has been very impressive and astonishing throughout the entire development span. They developed into tiny, hand-held intelligent gadgets from extremely large, cumbersome computers that needed enormous installation spaces. We are living in an era, that is actually the future generation of computer systems that we used to listen only. Your smartphone looks at you, identifies and unlocks the applications for you only. Alexa, Siri, Google Assistant, etc. hears to your voice and personally assist you. The online banking and financial systems learn from the past transactions' data and protect the services from fraudulent activities. These examples from different domains back the intelligent statement given by Dave Waters, a famous computer scientist and entrepreneur in Artificial Intelligence.

Intelligent tools and services are all around us, and they have improved and streamlined our daily lives. The same way that human intelligence develops via learning from past experiences, machine intelligence develops through the use of machine learning tools and techniques utilizing past data. This chapter gives the introduction to the machine learning, types of machine learning along with some examples and applications.

1.1 What is Machine Learning?

Machine Learning means providing a machine an ability to learn. This learning does not involve any specific or explicit programming to do predictions or forecasts. It only involves the training of algorithmic models with the previous data, known as training data and these trained models output the decisions or predictions with respect to new instances. The machine learning not only involves the training of existing models but focuses on the development of novel models also. Therefore,

Machine learning is defined as the subset of computer science, that aims to build and train models based on previous data/sample data to forecast or predict and decide without any functionality of explicit programming to do so.

1.1.1 The Foundations of Machine Learning

This training and building of models require tools/techniques from other disciplines also as shown in figure 1.1. These different disciplines provide a foundation to the machine learning developments. The description of tools and techniques from different fields is listed in table 1.1.

Table 1.1 Foundations of Machine Learning

Discipline	Description
Statistics	The machine learning model needs to analyse dataset before it can be used for the learning purpose. The statistical tools and methods from statistics knowledge serve the purpose. It provides techniques for descriptive, diagnostic, predictive and prescriptive analytics of the data; required by machine learning models for the learning purposes. The descriptive statistics describes the dataset using measures such as mean, median, mode, etc. The diagnostic analytics tend to find the root cause of any occurred event. The predictive analytics predicts the future outcomes. The prescriptive analytics is to provide feasible and possible solutions by analysing the solutions.
Data and Pattern Mining	It is the process of mining useful insights and inferences from the dataset available. It is also capable of finding associations among features of the training dataset; known as pattern mining. For example, in a grocery store generally bread and butter purchased together. In this case, the different items may be considered as features associated together and a pattern is generated. The store may use the generated patterns to place the food items on same shelf.
Database Management	Database management is the set of tools, techniques and operations required to store, manage and access the large volumes of training data. The data base management technology must include big data handling methods. Big data needs special techniques as it generates with high velocity in larger volume from multiple-heterogeneous sources.
Data Visualization	It is the set of tools and techniques to effectively represent the data in the form of graphs or tables. This makes the data more intelligible and comprehensible. Sometimes, simplistic data visualization techniques provide very useful insights. The tools generally used for data visualization are ggplot Python, ggplot2 R, excel, etc. The advanced excel tools are one of the easiest forms of visualization tools with user-friendly interface.
Programming Language	It is required to implement designed algorithms for the model training, testing and prediction applications. The most common programming languages used are Python, R, Java, JavaScript, Shell, etc.
Domain Specific Knowledge	The machine learning model can be trained and tested for application in different domains such as Healthcare, Education, Banking and Finance, etc. Therefore, an expertise domain specific-knowledge is required to understand the problem and the past data for the model training.

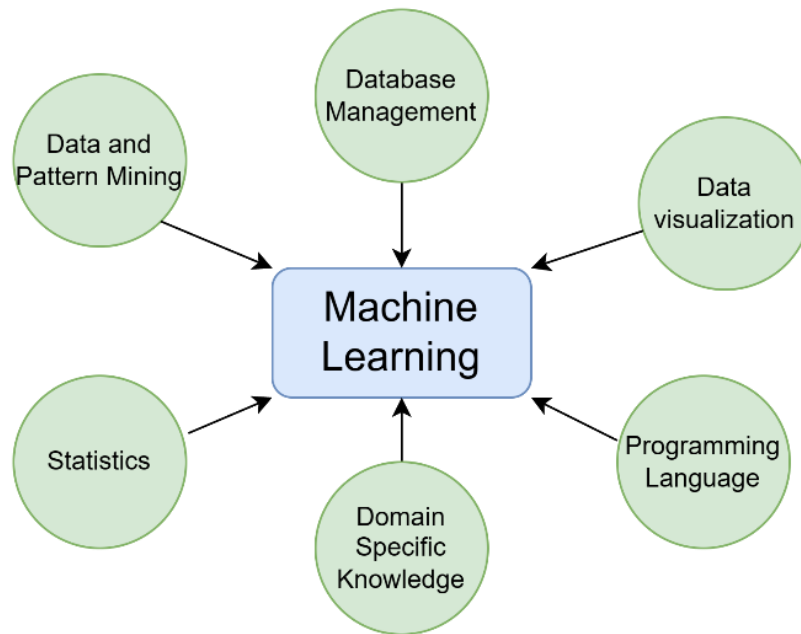


Fig. 1.1 Foundations of Machine Learning

1.1.2 Stages of Machine Learning

The final deployment of machine learning model is achieved through a sequence of multiple stages as shown in figure 1.2.

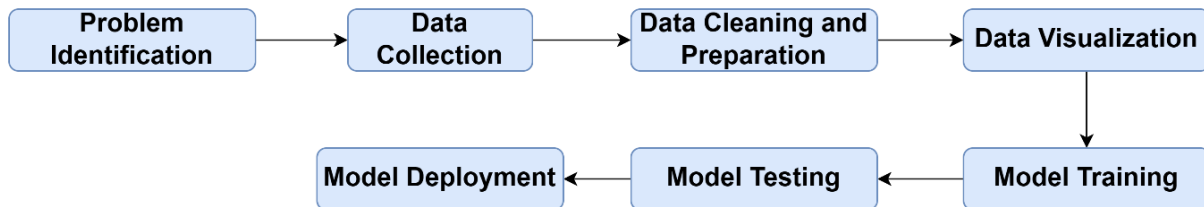


Fig. 1.2 The Flow Chart of Machine Learning Model Deployment

The stages are described as follows:

- a. *Problem Identification:* At this stage, a well thought questionnaire is prepared to understand the problem to be solved. It is similar to the requirement analysis of software development life cycle. To formulate a problem precisely, a good amount of domain-specific knowledge is required for which problem is to be solved.
- b. *Data collection:* The machine learning models learn from the previous data; known as training data. All the primary and secondary data sources of data collection are identified. The primary sources involve the processes of collecting data directly from the participants. On other hand, secondary data sources are the previously published datasets that may be re-utilized to train models for specific purposes. The feature set is also identified in this stage to define the instances

or training examples. The selection of feature set affects the accuracy and decision making of the machine learning model.

- c. *Data Cleaning and Preparation:* The raw data collected in the previous stage may not be suitable for the training purposes. It may contain noise, null-values, outliers, etc. The data cleaning is done to handle such issues to improve the accuracy of the machine learning models. After the data collection, feature selection and data cleaning operations; finally, a database or data warehouse is created for further or future use; this process is known as data preparation.
- d. *Data Visualization:* Sometimes plotting the data into graph or representing it into the tabular format gives more useful insights. These techniques describe the data properties visually with beneficial inferences, sometimes assist the developer to select the appropriate model for the training.
- e. *Model Training:* The model is trained with the previous dataset. The dataset consists of feature set and associated values. Each tuple represents a single instance of the dataset. The model training may be related to the learning of the rules for decision making, finding associations, pattern mining, etc.
- f. *Model Testing:* After training of the model, the trained model is being tested for its performance in terms accuracy, standard error, mean square error, etc. Generally, multiple models are tested against the application desired to find out the most suitable model for the problem identified.
- g. *Model Deployment:* The model deployment relates to the actual implementation of the model for application purpose. This phase generally requires the knowledge of physical resource requirements, platforms and programming language.

1.2 Types of Machine Learning

As stated earlier, machine learning means incorporating the learning capabilities to the machine for the decision making, prediction or forecasting, etc. This learning takes place with the help of training data that constitutes of the feature set with or without the class labels or outcome variables. The training dataset with class labels is referred to as labelled-training-data, otherwise, unlabelled-training-data. The machine learning models that make use of labelled training data fall under the first

category of machine learning known as *Supervised Machine Learning*; models trained with unlabelled training data fall under the second category of machine learning known as *Unsupervised Machine Learning*. The third category of machine learning is *Semi-supervised machine learning* that makes use of mostly unlabelled data i.e., typically a small amount of labelled data with a larger portion of labelled data. The fourth category of machine learning is Reinforcement learning that studies about the behaviour and actions of the intelligent agents to maximize the concept of cumulative reward and minimize the penalties. It does not require labelled training data.

1.2.1 Supervised Machine Learning

As the name suggests, supervision is inclusive in this approach of machine learning. The labelled training data is employed in this approach that acts as the supervisor and supervises the model to learn from the feature set with output labels using statistical techniques. The statistical techniques are used to analyse the training data and produce some inferred functions or rules. These inferred functions/rules are used to predict or forecast the output label for the new/future instances. Some of the techniques used in supervised learning include Neural Networks, Linear Regression, Random Forest, Naive Bayes, Logistic Regression, etc. The table 1.2 represents a labelled training dataset, downloaded from the *Kaggle* (*Kaggle,2023*) website. For the prediction problem, “If house to be rented has furnishing status as *semi-furnished* or *unfurnished* “. The furnishing status will be considered as output label and other attributes will compose feature set.

Table 1.2 The House Prediction Dataset

BHK	Rent	Size	Floor	Area Type	Area Locality	Furnishing Status
2	10000	1100	Ground out of 2	Super Area	Bandel	Unfurnished
2	20000	800	1 out of 3	Super Area	Phool Bagan, Kankurgachi	Semi-Furnished
2	17000	1000	1 out of 3	Super Area	Salt Lake City Sector 2	Semi-Furnished
2	10000	800	1 out of 2	Super Area	Dumdum Park	Unfurnished
2	7500	850	1 out of 2	Carpet Area	South Dum Dum	Unfurnished
2	7000	600	Ground out of 1	Super Area	Thakurpukur	Unfurnished
2	10000	700	Ground out of 4	Super Area	Malancha	Unfurnished
1	5000	250	1 out of 2	Super Area	Malancha	Unfurnished

2	26000	800	1 out of 2	Carpet Area	Palm Avenue Kolkata, Ballygunge	Unfurnished
2	10000	1000	1 out of 3	Carpet Area	Natunhat	Semi- Furnished
3	25000	1200	1 out of 4	Carpet Area	Action Area 1, Rajarhat Newtown	Semi- Furnished
1	5000	400	1 out of 1	Carpet Area	Keshtopur	Unfurnished

The Supervised Machine Learning is categorized as: Classification and Regression. The Classification refers to classify the dataset into multiple discrete data labels. Regression refers to predict the value of independent continuous variable.

- a. *Classification*: Classification is the supervised machine learning technique that targets to identify the correct category of the upcoming or new instance. It tries to set the possible accurate boundaries to maximize the difference between the available categories in the dataset. It includes classifiers such as decision trees, naïve bayes, rule-based classifiers, case-based reasoning, support vector machines etc.
- b. *Regression*: Regression means to comprehend the relationship between dependent and independent variables among the datasets. The most common use of regression is to generate prediction that gives the estimates, such as for a company's sales revenue, where the independent variable may be price of the product and company's sales revenue will be dependent variable. The simple linear equation is represented by the equation (1.1), where a and b are regression coefficients that establishes the relationship between dependent variable y and independent variable x .

$$y = a + bx \quad (1.1)$$

In real life scenarios, the dataset used for training and testing is generally noisy and erroneous; therefore, if permissible error, ϵ is added to the equation (1.1) it takes the form of equation (1.2).

$$y = a + bx + \epsilon \quad (1.2)$$

The prediction of continuous dependent variable y is measured such that root mean squared error between observed and actual value is minimum.

1.2.2 Unsupervised Machine Learning

Unsupervised learning is the process of training a model to use the structure of a dataset to make decisions successfully without using labels. For example, if the same problem of house prediction is considered for the unsupervised learning, the dataset will not contain the *furnishing status* in the training data. Unsupervised learning with unlabelled is used to build a model that is fully input-based. When there is an increase in disproportionally rising disorganized unlabelled data, unsupervised learning is used in addition to labelled data. Framework draws conclusions and finds patterns from the unlabelled data instead than telling the model what it needs to learn. Unsupervised learning makes use of several different algorithms. The most often used methods are those for learning logistic regression models, clustering, neural networks, and anomaly detection.

1.2.3 Semi-supervised Machine Learning

Supervised machine learning needs a large amount of labelled data to produce models with improved predictive performance. Semi-supervised learning is a branch of machine learning that focuses on using both labelled and unlabelled data to implement particular learning algorithms. Conceptually positioned halfway within unsupervised and supervised learning, this permits the usage of the considerable amounts of unlabelled data present from several use cases in addition to the more prevalent smaller clusters of annotated data.

When there is a lack of labelled data, semi-supervised classification approaches are quite helpful. Developing a trustworthy supervised classification method under those circumstances can be difficult. Applications where labelled data is expensive or difficult to obtain include computer-aided diagnostics, part-of-speech labelling, and biological research. It might be possible to create a more effective algorithm using unlabelled data if they include supplementary information that is important for predictions.

1.2.4 Reinforcement Learning

Reinforcement learning means learning from experiences. It is concerned with intelligent agents and their actions in the environment to maximize collective reward as depicted in figure 1.3. The online chess game is based on the reinforcement learning; for each successful step or unsuccessful step, it learns to take actions and change the state with respect to environmental conditions. For each

successful step agent gets a reward; therefore; reinforcement learning tends to maximize the reward gained. It is widely used in self-driving car, fraud detection, etc.

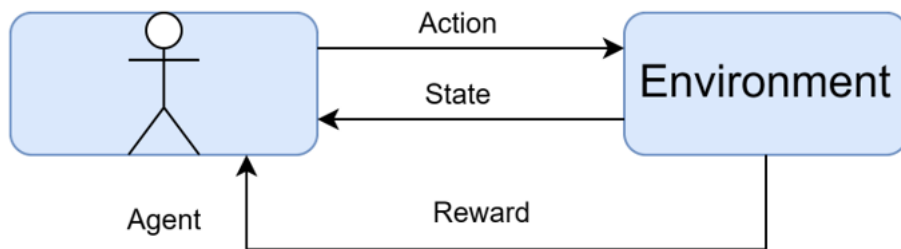


Fig. 1.3 Elements of Reinforcement Learning

1.2 Examples of Machine Learning

The machine learning based technologies have been radically integrated to our daily lives. This integration has made the life easier. Machine learning not only helps in the automation of the industries for automated decision processes but every one of us is also using the machine learning technology knowingly or unknowingly in our day-to-day life. Covid-testing kit from the healthcare to spam detection in the e-mails, machine learning finds applications in many domains. Table 1.3 provides a list of a few illustrations of various machine learning techniques.

Table 1.3 Few real-world examples of Machine Learning

Example	Basic function
Google Home	Gadgets, services, and favourite brands etc. can be connected via Google Home. This helps to create a custom smart home that facilitates every domestic chore.
Amazon Alexa	It plays songs, narrates kindle book, finds nearby locations, checks for entertainments, keeps track of Amazon packages, etc.
Google Maps	Google Maps marks the available parking or traffic rates on the roadways etc. To achieve these services, it utilizes cutting-edge machine learning methods and historical knowledge.
Microsoft Cortana	Microsoft created Cortana, a virtual assistant that leverages the Bing web search engine to create reminders and automated responses to user enquiries.
IBM Watson	The question answering application developed by IBM; that can answer questions posed in natural language.

Spotify and Other Recommendation Systems	Using machine learning and data science at its foundation, Spotify creates customized playlists and recommendations for its users.
Tesla and other self-driving cars	Tesla crowdsources its data using the internal or external sensors and later uses it refining its system.
ChatBot	A chatbot or chatterbot is a software application that sets up an online chat conversation between user and system using text or text-to-speech.
Project InnerEye	This is a research-based project, authored by Microsoft Research used for planning radiotherapy.

1.4 Applications of Machine Learning

The machine learning techniques find its application in many domains and solves many problems making the life easier and smarter.

- a. *Healthcare*: Machine learning techniques are widely used in the medical field. Popular uses include, but are not restricted to clinical diagnosis, drug development, new drug discovery, examining and interpreting the medical records, etc.
- b. *Recommendation Systems*: Machine learning finds its application in the recommendation domain also. It processes and filters the information to suggest the options to the customers or users as per their choice. These suggested options are called recommendations. For example, products to be listed in user's profile for online shopping; setting up the playlist of songs that may suit the taste of listener or online news the reader is interested in.
- c. *Education and Research*: Teaching, learning, and research are all being amazingly upgraded by machine learning to revolutionize education. Now, teachers are able to frame teaching practices as per intelligence level of students. They can monitor and supervise their capabilities and achievements. Localization, transcription, text-to-speech, and personalization of online learning content are increasing its impact and audience.
- d. *Face Recognition*: A facial recognition is a technology that compares an image or video frame of a human face to a database of faces. Such a type of applications learns to detect and identify faces for the authentication purposes. The vary basic example is unlocking of mobile phones by facial inputs from the user.

- e. *Spam Detection*: The mail service puts in a lot of effort to analyse through the emails and populate the spam folder with junk mails. Spam classifiers always work in back end without any explicit programming or setting.
- f. *Augmentation*: Machine learning helps people with their everyday chores, whether for personal or professional reasons. Other applications of this machine learning include software, data analysis, and virtual assistants. The main objective is to lessen bias-related errors.
- g. *Automation*: Machine learning is used at industry to automate the everyday tasks and increase the throughput efficiency of the industry. Machine learning operates completely on its own in any field without requiring any human input. Robots, for instance, carrying out the crucial process stages in manufacturing facilities
- h. *Cyber Security*: Machine learning assists the cryptographers and security professionals to detect threats in early stages, uncover network vulnerabilities, reduce IT workload and cost, synthesize and analyse large volumes of data automatically.
- i. *Web Search Engines*: The web search uses sophisticated machine learning algorithms to rank the pages. These web search engines use the machine learning algorithms for the search engine optimizations. It improves the accuracy of search engines.

1.5 Learning Associations

Associations refer to the relationship between different variables or features that define the data. Learning of these associations is done via rule-based machine learning algorithms and data mining techniques. The process of finding significant relationship among attributes is known as association learning. This involves evaluating degree of similarity, identification of hidden relationships in the databases.

1.5.1 Classical Example of Learning Associations

In a million of transactions per year, 10,000 sales in a store might be for new born baby diapers and 100,000 sales might be for razor blades. Razors and baby diapers initially appear to be statistically

independent, with no discernible association. However, rule mining would go more deeply into the frequency of transactions and discover that 5,000 sales involve both items.

Hence, the association system develops a new rule stating 50% of all customers purchasing new born diapers will also buy razor blades, which might be useful information for marketing campaigns, rather than simply learning or describing data as 1% sales relate to shoppers of diapers and 10% sales relate to shoppers of razor blades. Such type of analysis is generally known as **Market Basket Analysis**. The other applications of learning associations are medical diagnosis, Protein Sequence, Catalog Design, Loss-leader Analysis, etc.

1.5.2 Methods of learning associations

Learning Associations can be classified into three categories:

- a. *Apriori Algorithm*: The algorithm works better on the transaction-based data. It produces the association rules by leveraging the common datasets. This approach effectively calculates the frequent item sets taken together by using a breadth-first search or a hash tree-based approach. To compute item-sets, algorithm computes support and confidence variable from the dataset. The market basket analysis is appropriate example for this. For example, item set $\{bread, butter, jam\}$ may be a frequent item set; that depicts that these items are bought together.
- b. *Eclat algorithm*: Eclat stands for the Equivalence Class Transformation. It was developed by Schmidt *et.al* (Schmidt-Thieme, L. ,2004). This algorithm uses the Depth-First Search (DFS) to search a transaction database for frequent item-sets. The performance of Eclat algorithm is better than Apriori Algorithm in terms of time.
- c. *F-P Growth Algorithm*: The Apriori Algorithm is upgraded to Frequent-Pattern growth algorithm (F-P growth algorithm) (Borgelt, C. 2005). In apriori algorithm, at each step candidate set is generated and for this purpose algorithm has to scan the entire database repeatedly that makes Apriori slower. On other hand, F-P growth algorithm organizes the database as a common pattern or tree. This frequent tree is used to identify the most frequent patterns.

References

- Borgelt, C. (2005, August). An Implementation of the FP-growth Algorithm. In Proceedings of the 1st international workshop on open-source data mining: frequent pattern mining implementations (pp. 1-5).
- Kaggle: Your Machine Learning and Data Science Community, <https://www.kaggle.com>, *last accessed: 10-03-2023*.
- Schmidt-Thieme, L. (2004, November). Algorithmic Features of Eclat. In FIMI.

Chapter 2

Supervised and Unsupervised Machine Learning

The machine learning algorithms need to be trained before they can be used for any type of prediction, classification, decision-making or forecasting. The data is generally in the form of numbers, images, texts, etc. This data serves as the foundation of machine learning models. The data is gathered and divided into two categories: Training Data and Testing Data. Generally, the proportion of 70% and 30% is maintained for the training set and testing set respectively. The training data and testing data are used to train model for learning inferences and the model evaluations such as accuracy, error rate, etc. respectively. To evaluate these measures; it is required to compute an accuracy matrix consists of:

- a. *True Positive (TP)*: The instances that have been observed into a specific category and do actually belong to that category also; such instances are known as true positives.
- b. *True Negative (TN)*: The instances that have been observed to be different from the specific category and in real also they were different; such instances are known as true negatives.
- c. *False Positives (FP)*: The instances that have been observed into a specific category but actually they don't belong to it; such instances are known as false positives.
- d. *False Negatives (FN)*: The left-out instances that belonged to the specific category, but machine learning model discriminates them from the category; such instances are known as false negatives.

True Positives and True Negatives are the count of accurate predictions of the classification model while False Negatives and False Positive are erroneous predictions or inaccurate predictions.

The most common k –fold cross-validation technique is used to compute these evaluation measures of the machine learning model. It iteratively (k –times) uses different permutations of training and testing sets and computes an average evaluation measure.

The training and testing dataset include many training examples known as tuples/instances. The tuple is defined by the feature set consisting of values for various attributes. The tuples may or may not contain instance label with the feature set. As per the availability of the instance label along with feature set; the machine learning is divided into two categories: Supervised Machine Learning and Unsupervised Machine Learning.

2.1 Supervised Machine Learning

The feature set and the instance-label i.e., object class; when taken together creates a supervisory signal for machine learning model to be trained. This supervisory signal is used to map the input data with the output data i.e., categorical labels. During the training phase, the statistical techniques apply computational models on the training dataset to generate some mathematical rules. These mathematical rules are used for prediction or separating the data into categorical classes. This form of machine learning is referred to as supervised machine learning. For example, an organization may develop an application for the spam detection. The training of machine learning model will require a dataset that contains a feature set defining the mail characteristic (such as date of e-mail, sender e-mail id, keywords, number of receivers, etc.) and output label i.e., *spam* or *not spam*. The computational models will map the output labels to the feature set and infer some mathematical rules. These inferred rules will take upcoming new e-mail features as input and output the status as *spam* or *not spam*.

2.1.1 Regression

This is the form of supervised machine learning where the statistical computational models generate a mapping rule-based function. This function maps the input to the output. In the perspective of machine learning model, the input is the independent variable and output is the value of independent continuous variable.

a) *Simple vs Multi-variate Regression*

When the relationship is computed between one dependent and one independent variable; it is simple regression; when the relationship is computed between one dependent and multiple independent variable; for such cases it is multi-variate regression. For example, when happiness index is calculated in terms of income only; it is simple regression. The relationship by regression analysis may be

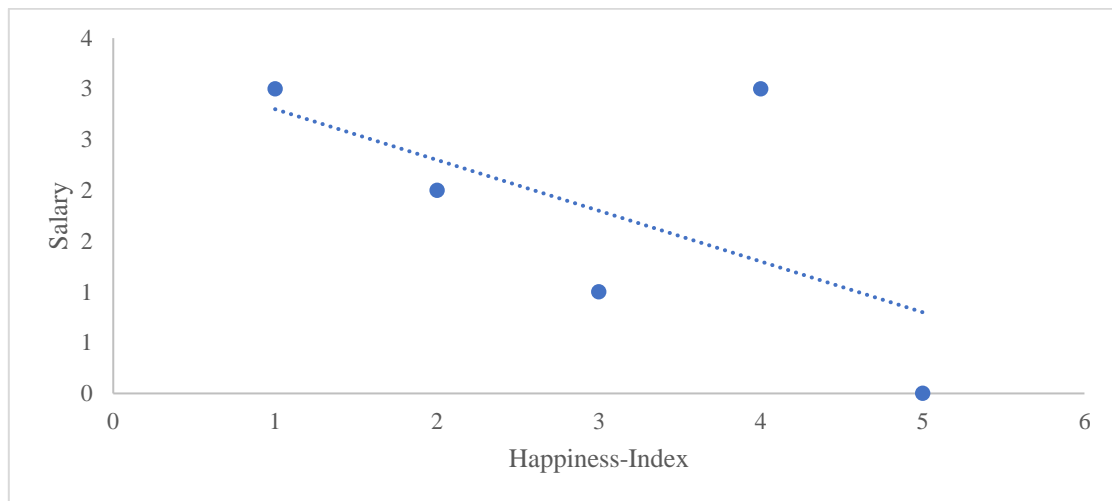
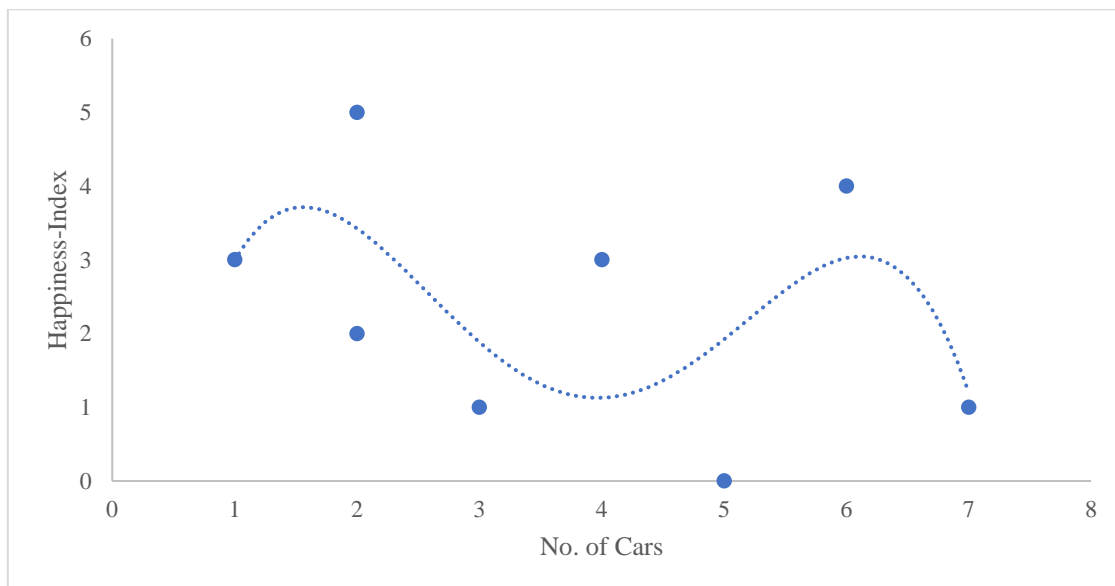
$$happiness = 0.5 * income + 5 \quad (2.1)$$

In another example, if happiness index is calculated in terms of income, time-spent-with-family, health-status, etc., it is multi-variate regression. The relationship may be

$$happiness = 0.5 * income + 0.3 * health + 0.2 * time + 5 \quad (2.2)$$

b) Linear Vs Non-linear Regression

The regression can be either linear or non-linear; if the relationship between dependent and independent variable is either positive or negative i.e., a straight-line relation; it is referred to as linear regression as in figure 2.1. If the curve between dependent and independent variable is not a straight line, then it is non-linear regression as in figure 2.2. For prediction and forecasting, generally, linear regression model is used. Therefore, in the further sections, the linear regression is discussed.

**Fig. 2.1** Positive Linear Regression**Fig. 2.2** Non-Linear Regression

Definition 2.1: The Linear Regression is the set of statistical procedures that computes the relationship between dependent (class label/ categorical label/ output label) and one or more independent variables (features/ attributes). The relationship may be either positive or negative.

Mathematically, the basic regression model is represented in equation (2.3) as,

$$y = bX + a \quad (2.3)$$

Where,

y : The output label i.e., the category label.

X : The feature set, defined as $X = \{x_0, x_1, x_2, \dots, x_n\}$ with attributes $x_0, x_1, x_2, \dots, x_n$.

a and b : Regression coefficients. b defines the slope that infers about the relative change in the dependent variable value with respect to independent variables' value. b is known as intercept; it represents the value of dependent variable in the absence of independent variables.

Since, the data gathered may have clerical errors, noise or incorrect values etc. This may affect the accuracy of the regression estimate. Therefore, the equation (2.3) can be re-written with ε , the permissible error of the estimate (random error component) as given in equation (2.4).

$$y = bX + a + \varepsilon \quad (2.4)$$

The values of a and b regression coefficients are computed by training the model with historical data. This formulates a complete linear regression model; defining a statistical relationship between the dependent and independent variables. The methods used to formulate the regression equations are:

- a. *Ordinary Least Square Method*: This is an estimator method that computes the values of regression coefficients such that sum of squared residuals i.e., error in prediction is minimum. The sum of squared residuals is the difference between the observed dependent variable value and predicted dependent variable value.
- b. *Logistic Regression*: This method builds a probabilistic regression model that computes the probability to predict that an instance belongs to a specific category. The linear regression

assumes that data follows the linear function while logistic regression uses the sigmoid function given in equation (2.5).

$$g(z) = \frac{1}{1+e^{-z}} \quad (2.5)$$

- c. *Lasso and Ridge Regression*: The linear regression model assumes that there are enough number of features available in the dataset. In the real scenarios, the dataset may have small or large number features than required. When the feature set is too weak or small; the model behaviour is poor for both training and testing dataset; this is referred to as underfitting. On the other hand, if the feature set is too large or strong; the score of testing dataset is poor as compared to training dataset; this is referred to as overfitting. Lasso and Ridge regression techniques are used to regularize the linear regression model to prevent the problem of overfitting. The lasso and ridge regression aim to shrink the coefficients to reduce the model complexity and multi-collinearity. This not only prevents the over-generalization due to overfitting but helps in feature selection also.

2.1.2 Classification

In machine learning, classification is process of identifying categories of the instances based on the feature set. It is the supervised form of machine learning that utilizes the labelled training and testing dataset. The category identification is done within the total number of unique categories given as output label in the training dataset. For example, in the dataset shown in table 2.1, for the problem is to *classify the role of the player*; the new tuple or instance will be either classified as *Bowler* or *Batsman*.

Table 2.1 Sample dataset for the classification problem

PLAYER NAME	TEAM	T-RUNS	T-WKTS	ODI-RUNS	ODI-SR-B	ODI-WKTS	Role
Balaji	TNB	0	0	0	0	0	Batsman
Abdul	RET	214	18	657	71.41	185	Bowler
Atgar	UAS	571	58	1269	80.62	288	Bowler
Ashwini	TNB	284	31	241	84.56	51	Bowler
Badal	TNB	63	0	79	45.93	0	Batsman
Bailey	UAS	0	0	172	72.26	0	Batsman
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
Bollint	RET	54	50	50	92.59	62	Bowler

Many classification techniques are available including Decision Tree, Naïve Bayes, Support Vector Machines, k -Nearest Neighbors, etc. The classification algorithms are categorized into two categories namely: Lazy Learner and Eager Learners. The lazy learner holds the training data until test data appears. When it appears; it begins the classification process based on most related stored data. On other hand Eager Learners build the classification model based on the stored training data. Eager learners spend more time in training the model while lazy learners spend more time on prediction. The various classification models are represented in figure 2.3.

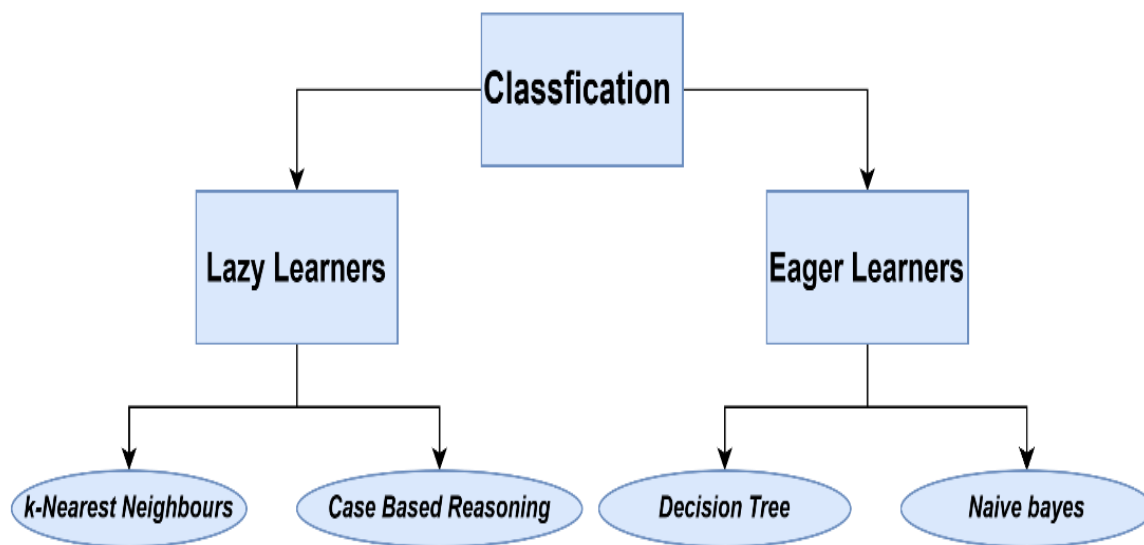


Fig. 2.3 Different examples of classification models

k- Nearest Neighbors:

k - nearest neighbors (k -NN) is the most basic classification model. It is a distance-based, non-parametric classifier as it does not assume any specific dataset distribution. It is a type of lazy learner; when test data appears, it computes the k -nearest neighbors i.e., most similar instances from the training dataset. Each of the k neighbors have a class label defined in the dataset. The label of test dataset will be predicted on the basis of the majority of labels among k - nearest neighbors. The similarity between new instance and training instances is calculated by computing the distance between them. The distance measures (Abeel, T., Van de Peer, Y., & Saeys, Y., 2009) that qualifies the similarity are given in table 2.2.

Table 2.2 Distance measures used in k -NN

Distance Measure	Formula	Description
Manhattan Distance	$D(x, y) = \sum_{i=1}^n x_i - y_i $	Distance between point x and point y in the n -dimensional space.
Euclidean Distance	$D(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$	
Minkowski Distance	$D(x, y) = \left(\sum_{i=1}^n x_i - y_i ^c \right)^{1/c}$	This distance measure is a generalized distance of Manhattan distance and Euclidean distance over a vector space.

Case Based Reasoning:

Case Based Reasoning (CBR) classifier stores the data as cases. A case is a complex symbolic description. As the test data appears to CBR, it checks for the identical dataset. If the any identical case is found then the corresponding label is returned as the output label. Otherwise, CBR searches for the training cases that have similar components to the new case. These training cases are known as neighbors of the new case.

Decision Tree:

Decision tree is the hierarchy-based classifier that predicts the class labels attribute wise, this is referred to as attribute test or split test to partition the dataset. The branches are selectively removed from the tree after partitioning to improve tree structure. This is known as decision tree pruning. During the training phase, the decision tree is trained to organize the different attributes at multiple levels. The attributes are arranged in the hierarchy as per their information giving capability such that attribute at highest level best discriminates dataset. The root node contains the highest information giving attribute; other attributes are arranged as internal nodes. Each node is the attribute test with branches as a result of attribute test. Leaf nodes represent the class labels as represented in figure 2.4. The decision tree in figure 2.4 may be seen as an organizational decision tree to decide the seniority on the basis of date of joining, age, salary, etc.

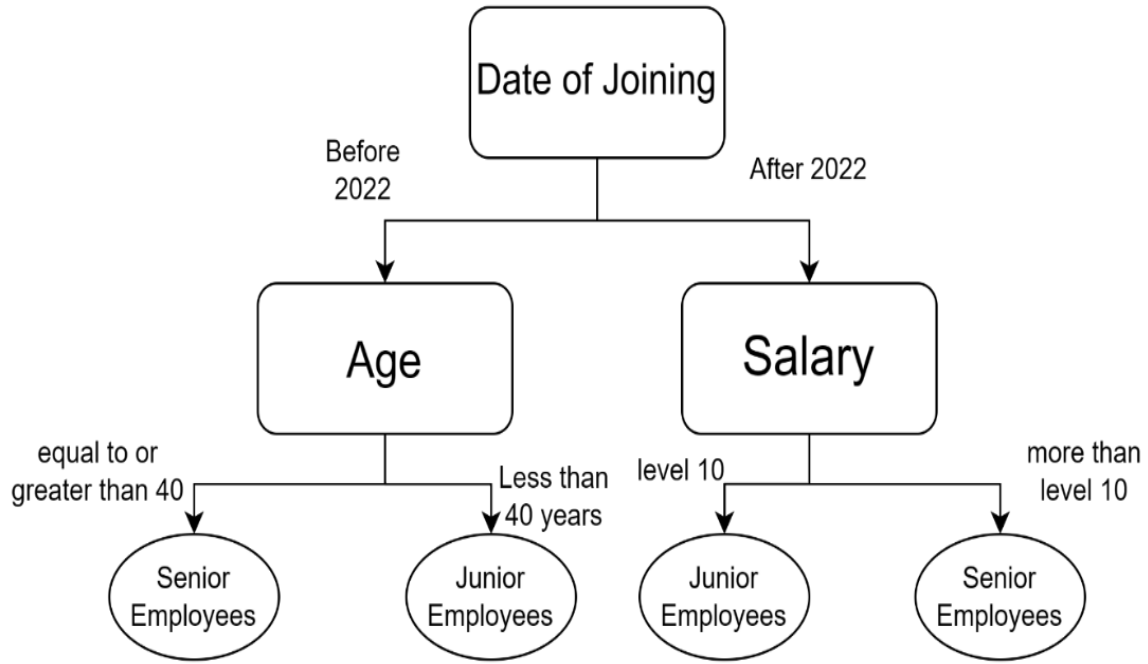


Fig. 2.4 Decision tree to decide the seniority among employees

Attribute selection measures are the heuristics to choose the attributes for the split test. Gini-index, Information gain, gain ratio are commonly used attribute selection measures.

Naïve Bayes:

Unlike decision tree classifier, it assumes that every feature or attribute is independent from one another and contributes equally in predicting outcome class labels. Generally, these assumptions are not correct in real world but works well in practice and predict results with good accuracy.

Naïve Bayes is probabilistic classifier, that is based on Bayes' theorem. Mathematically, the bayes theorem is given by the equation (2.5).

$$P\left(\frac{A}{B}\right) = \frac{P\left(\frac{B}{A}\right)P(A)}{P(B)} \quad (2.5)$$

Where, $P(A)$ and $P\left(\frac{A}{B}\right)$ are the priori probability and posteriori probability, with regard to the dataset in the training phase, the equation (2.5) can be utilized as in equation (2.6).

$$P\left(\frac{y}{x}\right) = \frac{P\left(\frac{x}{y}\right)P(y)}{P(x)} \quad (2.6)$$

Where, y is the class label and X is the feature set. $P\left(\frac{y}{X}\right)$ gives the probability of class label y for if given feature vector is X . The Naïve assumption is added to this Bayes theorem that states about the independence among feature variables.

During the prediction, posteriori probability is calculated for each class label with respect to feature vector of new instance. The posteriori probability with largest value reflects the class label.

2.1.3 Regression vs classification

Regression and classification both are supervised machine learning techniques with some overlaps and differences. These are listed below.

- a) The goal of classification is to predict the discrete variable i.e., target variables are the discrete classes. For example, classifying a mail is *spam* or *not spam*. It classifies upcoming mail out of these two discrete labels. The goal of regression is to predict value of continuous variable. For example, if sales revenue depends on the discount rates, then, regression will predict the sales revenue for future if discount rate is increased by 10%.
- b) A classification algorithm may predict a continuous value, but the continuous value is in the form of a probability for a class label and A regression algorithm may predict a discrete value, but the discrete value in the form of an integer quantity.
- c) The classification model evaluation is done using accuracy, precision, recall, etc. measures while regression model evaluation is done using root mean square error, R2-score, MAPE etc.
- d) In classification, it is targeted to find the best possible decision boundary that separates the two classes with maximum possible separation. While regression aims to find a trendline that represent an overall behaviour of data.
- e) Classification problems are categorized as binary classification problem or multi-class problems. Regression is taken as either linear regression or non-linear regression models and simple regression or multi-variate regression.

2.2 Unsupervised Machine Learning

Unsupervised machine learning is the self-organized form of machine learning that does not require any kind of supervision. The training dataset does not include the output labels. It learns to predict or decide on the basis of similarity among the dataset instances.

2.2.1 Clustering

Clustering is an unsupervised form of machine learning that tends to find the clusters among the datasets. These clusters are the collections of similar data instances. The similarity is calculated on the basis of distance measures given in table 2.2. The clustering algorithms may form clusters by partitioning approach or tree-based approach. The depiction of clustering is shown in figure 2.5. (Nagpal, 2017)

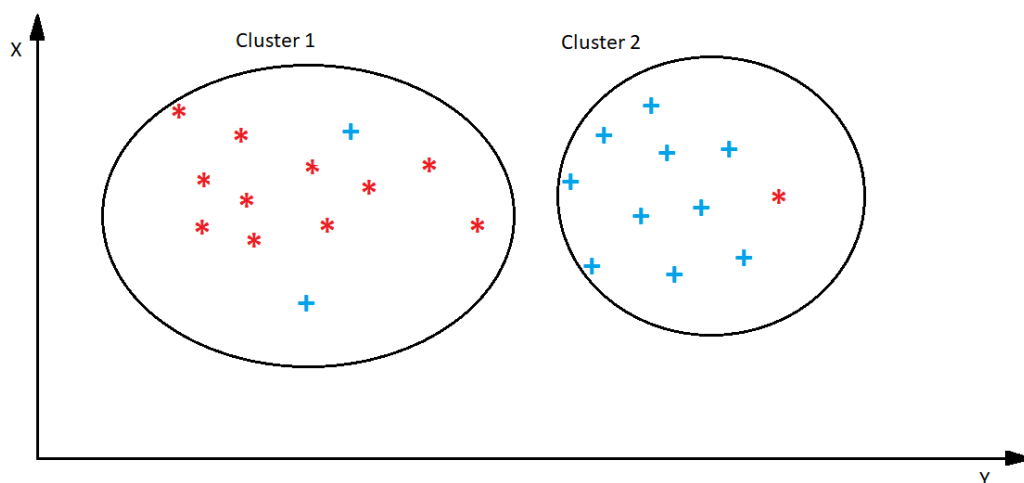


Fig. 2.5 Dataset partitioned into two clusters.

2.2.2 k –Means or Hierarchical Clustering

***k*-means:**

k -means is a partitioning clustering technique that divides the dataset into k clusters. The k –means algorithm proceeds as follows:

Step I. The clustering begins with randomly selecting k centroids. Each centroid is represented by the feature vector. The centroid is assumed to be the centre of cluster.

Step II. The distance is computed between each centroid and each data point in the dataset using either of the distance measures in table 2.2. The selected distance measure is applied on each attribute/feature of the feature vector.

Step III. The data points are grouped with the nearest centroids forming k clusters.

Step IV. The new centroids are computed by re-computing corresponding feature vector values. The updated feature vector of each centroid is the mean value of features values of corresponding cluster members. For example, in a two-dimensional feature space, if $C(2,3)$ is the centroid and $A(1,0)$; $B(3,4)$ and $D(2,1)$ are its members after step III. Then, the new centroid for this cluster will be

$$C' = \left(\frac{2 + 1 + 3 + 2}{4}, \frac{3 + 0 + 4 + 1}{4} \right) = (2,2)$$

Step V. The steps II to IV is iterated up to decided number of rounds or until the same cluster or centroids are achieved.

Hierarchical Clustering:

It uses the tree-based approach. New clusters are formed according to the previous clusters formed. It may follow the top-down approach or bottom-up approach; based on this fact hierarchical clustering is divided into two categories namely: Agglomerative (bottom-up) and Divisive (top-down).

- a) **Divisive:** In this approach, initially all data points belong to the same cluster. These clusters are partitioned into each level of the tree based on the distance from one another as shown in figure 2.6.

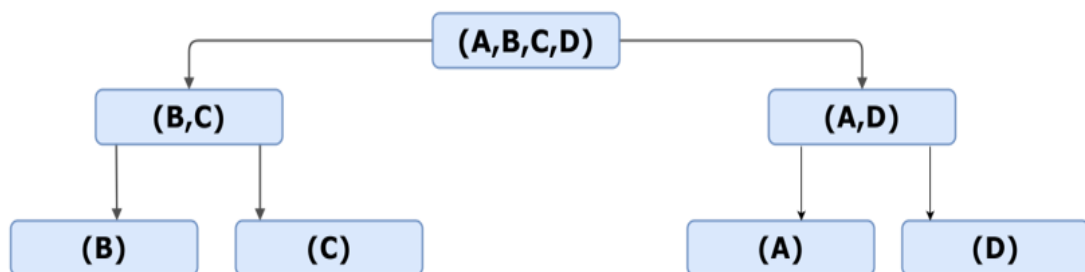


Fig. 2.6 Divisive Clustering.

- b) **Agglomerative:** Initially, each data point is considered as a separate cluster and based on the distance between clusters in previous phase, the nearest data points are combined to form clusters until all data points are clustered into a single cluster. The basic concept of agglomerative clustering may be visualized by the reverse arrows shown in figure 2.6.

2.2.3 Self-Organizing Feature Map (SOM) and Algorithm

Self-organizing Feature Map is an unsupervised machine learning technique that creates a low-dimensional (usually two-dimensional) representation of a higher-dimensional data collection while

maintaining the topological structure of the data. It helps to easily visualize higher dimensional data. These were developed by Kohonen et al., therefore also known as Kohonen Maps (Oja, E., & Kaski, S. (Eds.). 1999).

For instance, clusters of observations with comparable variable values could be used to represent a data set with p variables observed in n observations. Therefore, that observations in proximal clusters have more values in common than observations in distal clusters, these clusters might be represented as a two-dimensional "map." Self-organizing maps are related to artificial neural networks that operates in two modes: training and mapping. Training uses the input space to generate a map with lower dimensional space and mapping classifies upcoming dataset using the generated map.

References

- Abeel, T., Van de Peer, Y., & Saeys, Y. (2009). Java-ml: A machine learning library. *Journal of Machine Learning Research*, 10, 931-934.
- Nagpal, A. (2017) Clustering - unsupervised learning, Medium. Towards Data Science. Available at: <https://towardsdatascience.com/clustering-unsupervised-learning-788b215b074b> (Accessed: April 2, 2023).
- Oja, E., & Kaski, S. (Eds.). (1999). *Kohonen maps*. Elsevier.

Chapter 3

Dimensional Reduction

3.1 Dimensional Reduction

In this chapter, we will learn what is dimensional reduction, usage of dimensional reduction, why we use dimensional reduction.

3.1.1 Introduction

We must first comprehend dimensionality in order to define dimensionality reduction clearly. The performance of a machine learning algorithm may suffer from having too many input variables or features. Let's say you represent your Machine Learning (ML) data using rows and columns, much like you would use a spreadsheet. In that situation, the columns serve as features or input variables for a model that predicts the target variable and each row will be a training example i.e. tuple. When this data is plotted, these data rows are points situated on an n –dimensional feature space and the data column as the n dimensions on the space. The process of representing dataset (in rows-columns) into a feature space is known as Geometric Data Interpretation.

Unfortunately, a large amount of space is created if the feature space has a lot of dimensions. As a result, the data rows or points in the space might only be a very small generating an unrepresentative sample or plot. The performance of machine learning algorithms may also be adversely impacted by this imbalance. The phrase "the curse of dimensionality" applies to this circumstance. Collecting data with a large number of input features complicates the process of predictive modelling, that puts accuracy and performance of the machine learning model on risk.

Here is an example that will help you understand the issue. Let's say you went 50 yards in a straight line and at some point, along the way, you drop a quarter. Most likely, you'll locate it quickly by retracing the line. Next, let's imagine, though, that your search area is a 50-yard square now. Your search is now going to take days! We're not done yet, though. Create a cube that is 50 x 50 x 50 yards for the search area now. You might wish to bid that quarter goodbye! The search becomes more difficult and time-consuming as there are more dimensions involved.

How can we escape the dimensionality curse?

By reducing the number of input features, and subsequently the number of feature space dimensions. "Dimensionality reduction" (Simplilearn (2023) *What is dimensionality reduction?*) is the result. i.e. Dimensionality reduction refers to lowering the dimensions of your feature set that defines the dataset.

3.1.2 Importance of Dimensionality Reduction.

The Machine learning dataset will benefit from dimensionality reduction in a number of ways, including:

- a. Less complexity results from fewer features.
- b. Because there is less data, you will require fewer storage space. Less features also demand less computing time.
- c. Less misleading data results in improved model accuracy.
- d. Less data means faster algorithm training.
- e. Reducing the feature dimensions of the data set makes data visualization quicker.
- f. It eliminates noise & unnecessary features.

3.1.3 Advantages and Disadvantage of Dimensionality Reduction

a) Dimensionality Reduction's Advantages:

Dimensionality reduction is beneficial for AI engineers, data, professionals who work with large datasets, perform data visualization and analyse complex data.

1. Less storage space is needed as a result of its assistance in data compression.
2. It accelerates the computation.
3. It also helps to get rid of any unnecessary features

b) Dimensionality reduction's disadvantages:

1. We may experience some data loss during dimensionality reduction procedure, that may have negative effect on the performance of upcoming training methods.
2. Perhaps a lot of computing power is required.
3. It may be difficult to interpret traits that have been altered.
4. As a result, it becomes more challenging to understand the independent variables.

3.1.4 Methods for Dimensionality Reduction

Below are some methods used by experts in machine learning for dimensionality reduction.

1. Principal Component Analysis (PCA):

PCA is a technique for reducing the number of dimensions in large data sets by combining a large number of variables into a smaller set while retaining the bulk of the information in the larger set. While machine learning algorithms can examine the data much more rapidly and effectively with fewer sets of information since there are less unnecessary elements to evaluate, accuracy must eventually drop as dataset variables are reduced. Nevertheless, dimensionality reduction can be accomplished by simplifying calculations at the expense of some accuracy. In conclusion, PCA seeks to keep as much data as that is practical while reducing the number of variables in the data collection.

2. Removal of the Backward Feature:

By beginning with all the attributes and deleting the least significant one at a time, backward elimination improves the performance of the model. We continue doing this, unless we notice no improvements when we eliminate functionality. Initially, all input variables should be used.

3. Forward Feature Selection:

The forward selection is a method of iterative process that starts without features in the dataset. In each iteration, features are added to increase the model's capability. If performance is improved, the functionality is retained. Features that don't improve the results are eliminated. The process is continued up until the model's growth plateaus.

4. Missing Value Ratio:

Think about getting a dataset. Which occurs first? Presumably, before creating a model, you might want to look at the data. When you study the data, you find that particular dataset has missing values in some feature or dimension. Next, what? Before attempting to impute them or fully remove the variable with the missing values, you will look for the reason for these missing values. What if, feature value is missing for too many data points? Let's say there are more than 50%. Should the missing data be imputed or might the variable be deleted? We should get rid of the variable (feature/dimension) since it won't hold a lot of data. If the percentage of missing data for any variable (known as missing ratio) rises above a certain threshold, we may decide to exclude that variable (feature/dimension).

5. Low Variance Filter:

The Low Variance Filter utilizes a threshold, similar to how the Missing Values Ratio method does. Instead of computing missing value ratio, it tests data columns' variance in this instance. The technique figures out each variable's variance. As low variance characteristics have no effect on the target variable, all data columns with variances below threshold are eliminated.

6. High Correlation Factor:

This approach is used when there are two variables that provide the same information i.e. they are highly correlated, which could make the model less accurate. The correlation is computed between the two feature variables if it crosses a threshold, one of the feature variables is dropped. The Variance Inflation Factor (VIF) is used in this method to select one of the variables with highest association. Variables with a greater value ($VIF > 5$) can be removed.

7. Decision Trees:

A common supervised learning approach that divides data into homogenous groupings depending on input variables is Decision Tree. This method can also address issues including missing values, data outliers, and determining significant variables.

8. Random Forest:

This approach is comparable to the decision tree approach. Yet in this instance, we create a sizable number of trees—hence the name "forest"—against the intended variable. The usage data for each property are then used to identify feature subsets.

9. Factor Analysis:

Say we get two variables: *income* and *education*. There may be a considerable correlation between these parameters given that people with higher education levels also tend to earn significantly more money. As a result, all variables in one category will have a significant correlation among themselves but have a weak relationship in another category. In the Factor Analysis technique, the variables are grouped depending on their correlation(s). Each and every group is referred to as an important factor in this context. Compared to the initial dimensions of the data, these variables are minimal. But it's challenging to identify all important factors.

3.2 Methods of Dimensionality Reduction

This section details about the various methods of dimensional reduction as follows:

3.2.1 Principal Component Analysis (PCA)

Principal component analysis (PCA) is one of the applied linear algebra's most beneficial results. PCA is widely used in many different sorts of research, from computer graphics to neuroscience, because it provides a straightforward, non-parametric method of removing relevant information from complex data sets. PCA offers a simple, step-by-step process for reducing a complicated data set to a lower dimension in order to show the frequently hidden, reduced dynamics underlying it.

The objective is to determine the most significant basis for re-expressing a noisy, jumbled data set using principal component analysis. This new foundation is intended to reduce noise and uncover hidden dynamics

PCA is used to find data patterns, and it also identifies patterns' similarities and differences. As high dimension data without the luxury of graphical representation can be difficult to find patterns in, PCA is a useful technique for data analysis. The data can be compressed (or the number of dimensions reduced) when these patterns in the data have been found, without considerably compromising information, which is the other primary advantage of PCA.

Methodology

Step 1: Get information.

Principle component analysis should begin with some data, at the very least. Thus, get the data on which you wish to run PCA or use online resources.

Step 2: Take the mean away and standardize the dataset.

Every data value in each dimension must have a standardized value in order PCA correctly. For this purpose, mean and standard deviation is computed for each feature variable. The arithmetic average value (mean) for each dimension is deducted from every data value in the dimension and divide the new data value with standard dimension of the dimension. As a result, a dataset is created with an average value of 0 (\bar{x} , for feature x) and standard deviation 1.

Step 3: Compute the covariance matrix.

A suitable metric for measuring the relationship between dimensions in a data set is covariance, formula given in equation (3.1) with n number of instances or training examples. Covariance provides a measurement of the relationship between two dimensions, and by calculating the covariance of a dimension with itself, we can obtain its variance. For instance, we can find the covariance between the x and y , x with z , and z and y directions in a set of 3-dimensional data. By computing the covariance among both x and x , y and y , or z and z , we may also determine the variance of something like the x , y , or z dimensions. There are parallels between the variance formula and the covariance formula, and they can be used interchangeable terms.

$$cov(x, y) = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{(n-1)} \quad (3.1)$$

Step 4: Determine the covariance matrix's eigenvalues and eigenvectors.

In order to determine the principal components of the data, we need to compute the eigen vectors and eigen values of the covariance matrix computed in step 4. As principal components are the linear combination of existing feature variables that represent the direction of the data. In simple language, principal components are the new axes to represent the data. Eigen vectors and eigen values in the covariance matrix serve the purpose to identify principal components. In the case of a 3×3 square matrix, there will be three eigen values and three eigen vectors. Eigenvectors can be scaled by a certain amount without affecting their direction. This is so because a vector's direction is unaffected by scaling; only its length is altered.

Step 5: Selecting components and creating a feature vector

Dimensionality reduction and data compression become important at this stage. Generally, the eigenvalues obtained from the eigenvectors in the previous section are not equal. The significance of eigenvectors is displayed in order of importance according to the size of their eigenvalues.

You can choose to disregard the less important elements. If the eigenvalues are low, some information can be lost, but it is not important. The final data collection will have less variables than the source one if you decide to eliminate specific elements. For instance, if you opt to only use the first k eigenvectors from a data set that originally has d dimensions, your ultimate data set will only have k dimensions. The next step is to create a feature vector, that's merely a vector matrix. To do this, choose the eigenvectors from in the list in equation (3.2) you'd like to maintain and place them in a matrix's columns.

$$\text{Feature Vector} = (eig_1 eig_2 eig_3 \dots eig_n) \quad (3.2)$$

Step 6: Creating the new data collection

The last stage of PCA is the easiest one. To create a feature vector, we select the desired eigenvectors (data components), transpose them, and then perform a left multiplication on the transposed original data set.

$$\text{FinalData} = \text{RowFeatueVector} \times \text{RowDataAdjust} \quad (3.3)$$

The matrix, Row Feature Vector, contains the eigenvectors in its rows, where each eigenvector corresponds to a dimension of the original data set, and the most significant eigenvector is at the top. On the other hand, the matrix, Row Data Adjust, contains the mean-adjusted data items in its columns, where each column represents a data item, and each row represents a dimension. Although transposing the data may seem confusing, it is more convenient for solving equations. In the final data set, Final Data, the dimensions are represented by rows, and the data items are represented by columns.

3.2.2 Linear Discriminant Analysis (LDA)

In order to minimize dimensionality, LDA is frequently employed as a pre-processing method for machine learning and classification and prediction applications. However, despite its frequent use, LDA can be considered a "black box" and not fully understood by users.

LDA aims to project the original data matrix onto a lower dimensional space, which requires three steps. First, the separability between different classes is identified, which is often referred to as the between-class variance or matrix. This entails calculating the separation among the means of several classes. Second, the distance among the average and samples for each class is used to determine of within variance or matrix. Finally, to increase the variance between classes and reduce the variance within classes, a lower dimensional space is created.

Step 1: Doing the Between-Class Variance calculation (S_B):

The displacement between different classes, denoted by (m_i, m) , would be determined as follows to determine the inter-class variation (S_B):

$$(m_i - m)^2 = (W^T \mu_i - W^T \mu)^2 = W^T (\mu_i - \mu) (\mu_i - \mu)^T W \quad (3.4)$$

Where, m_i is the projected mean for i^{th} class and it is determined by equation (3.5).

$$m_i = W^T \mu_i \quad (3.5)$$

Where, m is the projected overall mean across all classes, which is determined as in equation (3.6).

$$m = W^T \mu \quad (3.6)$$

W denotes the LDA¹ transformation matrix, $\mu_i(1 \times M)$ is the mean of the i^{th} class, and $\mu(1 \times M)$ denotes the sum of all class means, which can be calculated using equation (3.7) and equation (3.8).

$$\mu_j = \frac{1}{n_j} \sum_{x_i \in \omega_j} x_i \quad (3.7)$$

$$\mu = \frac{1}{N} \sum_{i=1}^N x_i = \sum_{i=1}^c \frac{n_i}{N} \mu_i \quad (3.8)$$

where c represents the total number of classes (in our example $c = 3$).

The variable $(\mu_i - \mu)(\mu_i - \mu)^T$ in equation (3.4) indicates either the inter-class variance of the i^{th} class or the separation distance between the mean of the i^{th} class (μ_i) and the total mean (μ).

(S_{B_i}) Equation (3.4) should now read as follows in equation (3.9): S_{B_i}

$$(m_i - m)^2 = W^T S_{B_i} W \quad (3.9)$$

The formula for calculating the overall between-class variance is ($S_B = \sum_{i=1}^c n_i S_{B_i}$). The sum of the between-class matrices for all classes is then used to determine the overall for within matrix S_B .

Step 2: The Within-Class Variance (S_w) is Calculation:

The difference of that class's mean and samples can be seen in the within-class variance of the i^{th} class S_{W_i} . The LDA technique looks for a lower-dimensional space to reduce within-class variance, or the gap between its projected mean (m_i) and the predicted samples of each category ($W^T x_i$). Using the equation (3.10), the within-class variance (S_{W_i}) for every category is determined.

$$S_w = \sum_{i=1}^3 S_{W_i} = \sum_{x_i \in \omega_1} (x_i - \mu_1)(x_i - \mu_1)^T + \sum_{x_i \in \omega_2} (x_i - \mu_2)(x_i - \mu_2)^T + \sum_{x_i \in \omega_3} (x_i - \mu_3)(x_i - \mu_3)^T \quad (3.10)$$

Step 3: Constructing the Lower Dimensional Space

Equation (3.11), commonly known as Fisher's criterion, can be used to determine the transformation matrix (W) of the LDA approach after getting for between variance (S_B) and within-class variance (S_W). Equation is able to serve its place as in equation (3.12).

$$\arg \max \frac{W^T S_B W}{W^T S_W W} \quad (3.11)$$

$$S_W W = \lambda S_B W \quad (3.12)$$

where the symbol λ denotes the eigenvalues of the transformation matrix (W). The answer to this issue can be discovered if S_W is non-singular by computing the eigenvalues ($\lambda = \{\lambda_1, \lambda_2, \dots, \lambda_M\}$) and eigenvectors ($V = \{v_1, v_2, \dots, v_M\}$) of $W = S_W^{-1} S_B$.

Since Equation (3.10) is satisfied and the LDA space is revealed, the eigenvalues and eigenvectors are scalar values and non-zero vectors, respectively. The related eigenvectors characterize the orientations of the larger space, whilst the connected eigenvalues indicate the eigenvectors' length, amplitude, or scaling factor. Because of this, each eigenvector identifies a specific axis of an LDA space, as well as its corresponding eigenvalue describes how robust the eigenvector is. In order to achieve the LDA goal, the robustness of the eigenvector demonstrates that it can discriminate between a wide range of classes by raising variance between classes and reducing variance within classes. To create a reduced dimensional space (V_k), the eigenvectors with the k highest eigenvalues are substituted for the other eigenvectors $\{v_{k+1}, v_{k+2}, v_M\}$.

Projecting the data onto the initial eigenvector (v_1) results in a substantially greater separation distance among different classes than reflecting the data onto to the second eigenvector, according to a comparison of the two lesser subspaces (v_2). This means that the three classes can be effectively separated when projected onto v_1 . Additionally, the difference between the means of the first and second classes ($m_1 - m_2$) is much greater when the original data is projected onto v_1 , indicating that the first eigenvector is better at discriminating the three classes than the second eigenvector. Furthermore, the within-class variation is much lower when the data is projected onto v_1 compared to v_2 . Specifically, when projected, S_{W_1} is considerably smaller on v_1 than it is on v_2 . This means that projecting the data onto v_1 significantly reduces the within-class variation as compared to projecting it onto v_2 . These observations lead us to the conclusion that the initial eigenvector is favoured for producing a lower-dimensional space because it is more suited than the second eigenvector to attaining the lower-dimensional space goal of the LDA technique (*University of Salford, M, 1970*).

3.2.3 Generalized Discriminant Analysis (GDA)

Linear discriminant analysis (LDA) is a popular statistical technique used for classification problems. This method is based on resolving eigenvalues and provides an accurate response for maximum inertia. However, it is not suitable for solving nonlinear problems. In this paper, we introduce a Generalized Discriminant Analysis (GDA) to extend LDA to nonlinear scenarios. To do this, the input space is converted into a tall subspace with linear qualities, which enables the application of standard techniques like LDA. This method's main objective is to convert the input space into a useful subspace where variables are connected to the input space nonlinearly.

A popular classification tool is LDA. The transformation of the input space into a new one serves as its basis. The new coordinate values of the data are combined linearly to form the primary components, also referred to as the principal components of the discriminant axis. The conventional criteria for class separability for the common LDA are defined as the ratio between the intra classes inertia and the inter classes inertia. This criterion should be stricter when the intra-class inertia is smaller, and the inter-class inertia is bigger. This maximizing method's equivalence to eigenvalue resolution was shown. Through using Mahala Nobis distance and the assumption that the categories have quite a multivariate Gaussian distribution, each observation can be classified into the class that has the highest posterior probability.

We broaden LDA using kernel functions when the primary components of the transformed space are not linearly connected to the input variables. The kernel operator K allows for the generation of a non-linear distinguishing function inside the input space that is comparable to a linear distinguishing functional in the feature set F . While maximizing inter-class inertia for the LDA, the GDA method aims to minimize intra-class inertia. This maximization is equivalent to finding the following eigenvalues and eigenvectors, which are the solutions of the equation (3.13) (*Generalized discriminant analysis using a kernel approach, 2023*).

$$\lambda Vv = Bv \quad (3.13)$$

The biggest eigenvalue of (3.13) provides the highest inertia quotient for the following equation (3.14).

$$\lambda = \frac{v^t B v}{v^t V v} \quad (3.14)$$

As the eigenvectors are linear combinations of F elements, there exist coefficients α_{pq} ($p = 1, \dots, N; q = 1, \dots, n_p$) equation (3.15) is satisfied.

$$v = \sum_{p=1}^N \sum_{q=1}^{n_p} \alpha_{pq} \phi(x_{pq}) \quad (3.15)$$

All solutions v lie in the span of $\phi(x_y)$.

Let us consider the coefficient vector $\alpha = (\alpha_{pq})_{p=1, \dots, N; q=1, \dots, n_p}$; it can be written in a condensed way as $\alpha = (\alpha_p)_{p=1, \dots, N}$, where $\alpha_p = (\alpha_{pq})_{q=1, \dots, n_p}$, α_p is the coefficient of the vector v in the class p . It is equivalent to the following quotient in equation (3.16).

$$\lambda = \frac{\alpha^t K W K \alpha}{\alpha^t K K \alpha} \quad (3.16)$$

3.3 Applications and Usage of Dimensional Reduction

Reduced dimensionality is ultimately used to obtain fewer feature variables. The other applications of dimensionality reduction are indirectly impacted by this. Let's go through each of them individually with examples. (Pramoditha, R. (2022))

1. Dimensionality reduction can be used to visualize highly dimensional data.

Exploratory Data Analysis (EDA) relies heavily on data visualization. However, visualizing high-dimensional data can be challenging since our visual experience is limited to 2D or 3D charts that are only suitable for 2D or 3D data, respectively. Higher-dimensional designs, such as 4D or 5D plots, are difficult to comprehend. To address this issue, dimensionality reduction techniques, such as PCA, LDA, etc. can be utilized to transform high-dimensional data into 2 or 3-dimensional data that can be easily plotted in 2D or 3D visualizations.

For instance, consider the 30-variable breast cancer dataset. The data has 30 dimensions, making it impossible to plot in EDA to identify significant trends. The original data must first be transformed into 2- or 3-dimensional data through techniques such as PCA to create 2D and 3D plots, as shown in the figure 3.1.

Only 63.24% of variability within original information was captured when it was converted to 2-dimensional data. Only 72.64% of variability within original data had been captured when it was converted into 3-dimensional data. Yet, you are able to create plots and identify certain patterns in the data that roughly represent the original data.

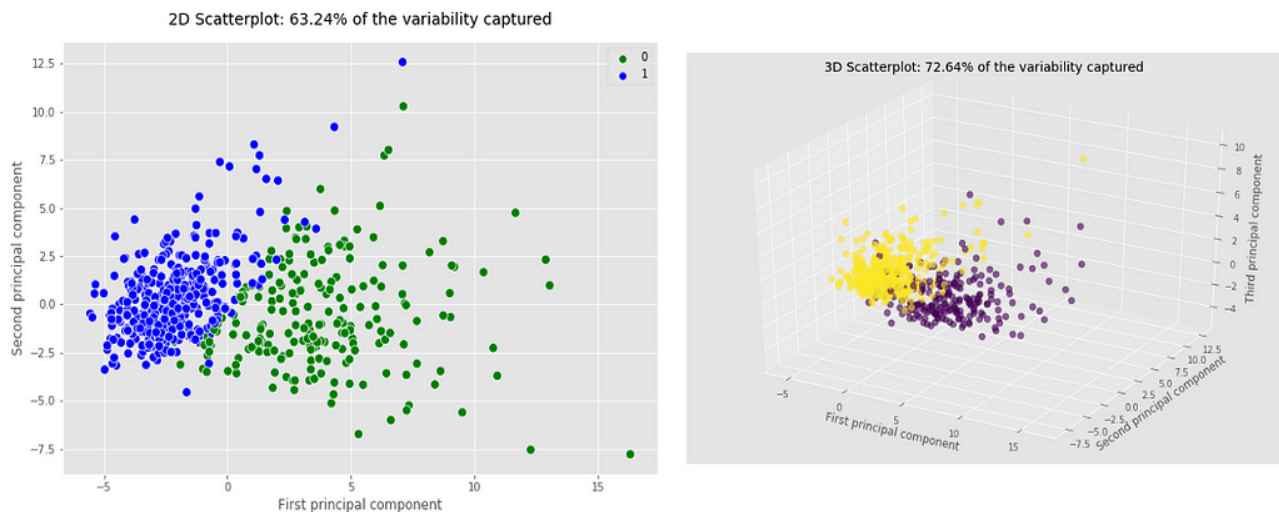


Fig. 3.1 Scatterplot visualization (Pramoditha, R., 2022)

2. It is possible to use dimension reduction to lessen the issue of overfitting.

The worst issue data scientists encounter when creating models is overfitting. One method for reducing overfitting within machine learning models is dimensionality reduction. You may now consider how it functions. Dimensionality reduction reduces the number of variables in the model or eliminates the least significant variables. This will minimize the complexity of the model and decrease some of the data noise. Dimensionality reduction assists in reducing overfitting in this manner.

3. Multicollinearity is automatically eliminated by dimension reduction.

Your data contains several non-independent variables. Certain input variables might be correlated with the dataset's other input variables. Multicollinearity is what this is, and it can harm the effectiveness of both classification and regression models. Dimensionality reduction makes use of the multicollinearity that already exists between the variables. For instance, PCA creates a new collection of uncorrelated variables from a set of highly linked data. PCA can therefore automatically eliminate multicollinearity from your data as shown in figure (3.2).

The breast cancer dataset comprises 30 variables, and the heatmap on the left shows the correlation coefficients of those variables. The dataset contains several highly correlated variables, as can be shown in figure 3.2. We have decreased the number of variables to 6 while maintaining the original dataset's 88.76% variability after applying PCA on the data. The heatmap on the right displays the correlation coefficients among those six factors. It is clear that these variables are no longer connected. So, it means that by using PCA, a dimensionality reduction technique, we have effectively eliminated multicollinearity.

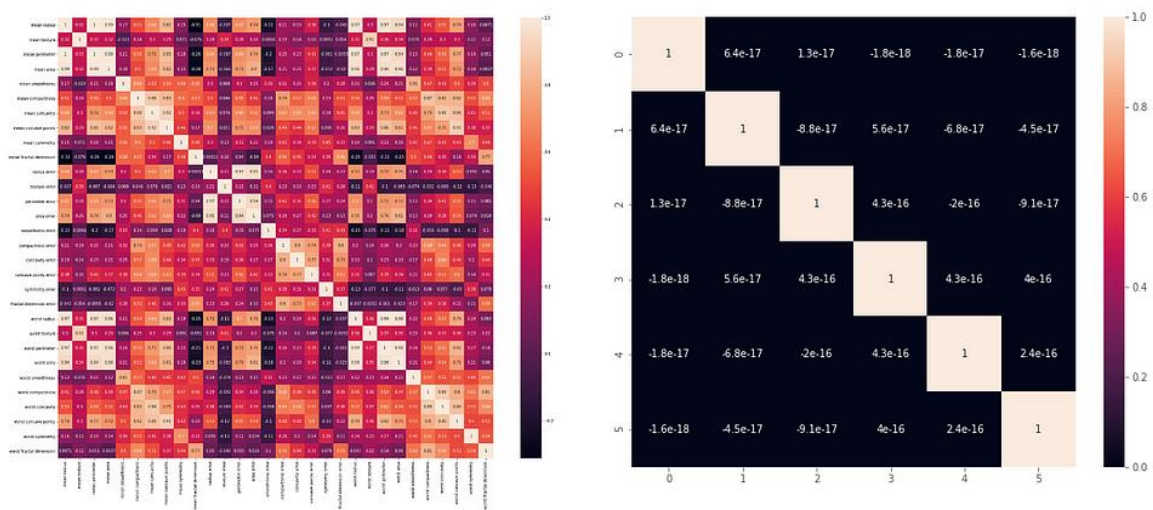


Fig. 3.2 Heatmap (Pramoditha, R., 2022)

4. In factor analysis, dimensionality reduction is employed.

Factor analysis is a technique for decreasing dimensionality. This uses the principal components method and is essentially identical to PCA, with one notable exception. A lot of important variables or factors that weren't present in the initial dataset are discovered using factor analysis. The factors are obtained by analysing these principal components.

5. Image compression can be achieved through dimensional reduction.

By reducing the number of dimensions in your dataset, you can preserve as much of the original data's variability as you can. Compression of images can be accomplished using a similar strategy. In order to minimize the number of pixels in an image while maintaining the highest level of quality feasible, image compression is used.

With dimensionality reduction, RGB colour images can be compressed as shown in figure 3.3.

- Top-left: 400-pixel-wide version of the original flower image.

- Top-right: A flower image that is compressed down to only 40 dimensions. The original image's quality has been preserved but the dimensions have been shrunk by a factor of ten! The image's crucial components can still be seen.
- Image of a compressed flower only with 25 dimensions is shown in the bottom left. 16 times less dimensions have been used while maintaining an image quality of 83.41%! The image's crucial components can still be seen.
- Bottom-right: Flower image produced without feature scaling following application of PCA. The altered data is indistinguishable from the original data. This is provided to demonstrate the value of feature scaling before using PCA.

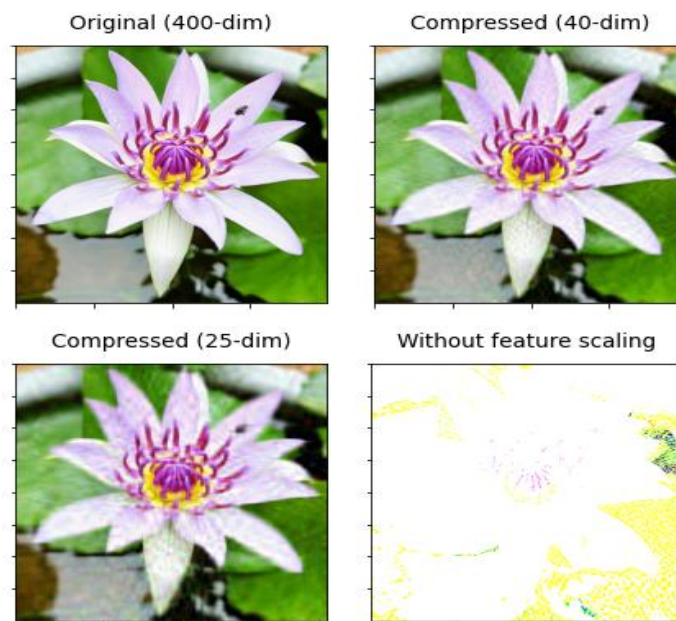


Fig. 3.3 Comparison the Two Flower Images (Pramoditha, R., 2022)

6. Models require less time to train when dimensions are reduced.

Higher-dimensional spaces have a higher probability of having far-off data points. Algorithms perform computations on these pieces of data during training. The calculations are going to be slow if there are a lot of dimensions as the data points were spread out. As a result, methods for deep learning and machine learning won't be able to train model on higher-dimensional data effectively with good performance. Thus, the data should be reduced in dimension before being fed into the algorithms.

7. While training models, dimensionality reduction significantly reduces the computational load.

The requirement of computational resources for training such models will be very low because dimensionality reduction speeds up model training time by streamlining calculations. Hence, when training models, dimensionality reduction instantly saves a significant amount of computational resources.

As long as basic models are capable of producing precise predictions, they are preferable to complex models. Furthermore, interpreting them is very simple. Dimensionality reduction lessens the complexity of models by retaining only the most crucial variables within the data or by reducing the total number of variables.

8. Dimensionality decrease enhances model correctness.

Due to the elimination of correlated and least-important variables, dimensionality reduction allows for the noise removal of data. This will increase the models' precision even more. Dimensionality reduction converts nonlinear data into a shape that can be separated linearly.

The majority of real-world material cannot be separated linearly. Then, the linear hyperplane (the straight line in two-dimensional space or a suitable plane in three dimensions or higher) is unable to identify or classify the non-linear data. We must convert non-linear data into a linear shape so that linear hyperplane can classify the data.

Kernel PCA, also referred to as the nonlinear form of normal PCA, is a dimensionality reduction technique that can be used to convert non-linear data into a linearly-separable shape. There are two stages in the Kernel PCA process.

- The incoming data are fed into the algorithm's "rbf" kernel function, which projects them into a space with higher dimensions where non-linear data can be separated linearly.
- The higher-dimensional data derived from the preceding step is then subjected to standard PCA in order to return the data to its original lower-dimensional location.

The image on the left shows the raw data plotted in figure 3.4. It is evident that the initial data is nonlinear and a linear hyperplane cannot be used to separate it. The picture to the right shows the data after Kernel PCA was applied. This converted data into a form that can be separated linearly.

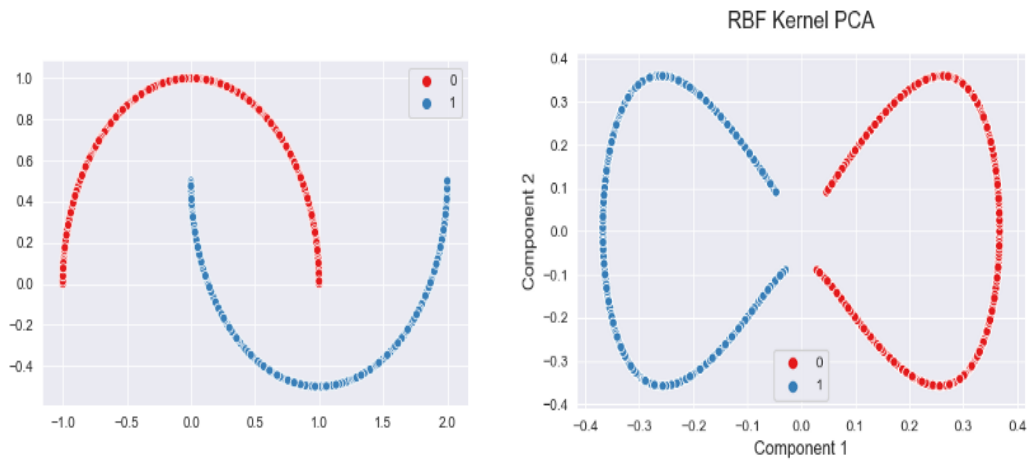


Fig. 3.4 PCA plot (Pramoditha, R., 2022)

References:

Pramoditha, R. (2022) *11 different uses of dimensionality reduction*, *Medium*. Towards Data Science. Available at: <https://towardsdatascience.com/11-different-uses-of-dimensionality-reduction-4325d62b4fa6> (Accessed: March 22, 2023).

Simplilearn (2023) *What is dimensionality reduction? overview, and popular techniques*, *Simplilearn.com*. Simplilearn. Available at: <https://www.simplilearn.com/what-is-dimensionality-reduction-article> (Accessed: March 22, 2023).

Chapter 4

Regression

Regression assignments are machine learning tasks where the primary objective is establishing an estimate. Regression strategies are modelled using input information (independent variables) that produce results as continuous numerical values for dependent variables, as opposed to classification, which results in discrete categories or classes. In order to remember specific links and affiliations between inputs and their corresponding outputs, relapse models use the features information (also known as informative or autonomous factors) as input and their corresponding numerical yield values as output and generates a relationship between input and output. With this knowledge, it is able to predict yield results for new examples. It is similar to classification but with numerical yields.

Regression is a guarantee that there is a quantifiable connection between two or more elements. Basic relapse involves two components, one is an autonomous variable that drives the behaviour of the other (characterized as a subordinate variable). The variable who derives the behaviour is known as independent feature variable and variable whose behaviour is derived is dependent variable. Relapse is a type of physical translation, hence there needs to be a physical mechanism by which the independent variable X can affect the independent variable y . The fundamental link between X and y is depicted by equation (4.1)

$$y = a + bX \quad (4.1)$$

Where, y denotes the observed value for any value of X . The equation (4.1) known as relapse condition of y on X (sometimes referred to as the relapse line of y on X when drawn on a chart) with a and b regression co-efficient, denoting intercept and slope of the regression line/ trendline/relapse line.

Among the most common applications of machine learning models, particularly in supervised machine learning, is handling regression problems. To comprehend the connection between an outcome or dependent variable and independent variables, algorithms are imparted. After that, it can be applied to analyse the consequences of novel, unforeseen enter data or to adjacent data gaps.

It is a method of predictive modelling that is based on machine learning and uses a computer programme to predict ongoing outcomes. Any predicting or predictive model must include regression analysis, which is an established technique in machine learning-powered statistical analysis.

Regression is used in many different machine-learning applications since it is an essential part of predictive modelling. Regression analysis may provide companies with crucial information for decision-making, whether it is applied to predict healthcare patterns or support financial forecasts. It has been utilized in many different industries to predict housing values, map wage changes, and map changes in stock and share prices. Machine learning forecasting algorithms rely on the regression statistical research area. Since it is a technique for predicting continuous outcomes in predictive modelling, it aids in planning and result prediction using data. A line of best fit is commonly drawn through the data values in predictive regression. To produce the best-fit line, the separation among each spot and the line is reduced.

Further, supervised machine learning models are typically applied to regression in addition to categorization. The both the input and the output training data for this approach of training models should be labelled. Since machine learning regression models must be able to understand the relationship between features and outcome components, appropriately labelled training data is crucial.

As with all supervised machine learning, the labelled training data should be demonstrative of the entire population. The prediction model will overfit data that if the training data is not representative for the current and unadulterated data. After the model is implemented, this will cause predictions to be incorrect. Regression analysis examines relationships between variables and results, so it is important to carefully choose the traits to include.

Regression in machine learning (Alpaydin, E.,2020) is covered in this guide, along with its definition, applications, and various varieties.

4.1 Need and application of linear regression

Predictive analytics heavily relies on machine learning regression models to anticipate outcomes and identify patterns. Regression models will receive training to comprehend how various independent factors relate to the result. As a result, the model is able to comprehend the vast array of variables

that could result in the desired forecasting for new and unobserved data. The developed models can be applied in various contexts and procedures such as controlling for market swings, and evaluating campaigns by altering a number of possible independent variables. Regression machine learning can be used in the sales setting to forecast sales for the forthcoming month based on a variety of traits. In a medical setting, a group could predict long-term health trends for the general public.

In practice, labelled data will be used to build a regression model to identify the relationship between data characteristics and the dependent variable. The algorithm can forecast the outcome of fresh, unforeseen data by calculating through derived relation using regression. This could be applied to forecast future outcomes and estimate the absence of prior data. Therefore, regression is supervised form of machine learning. The other problem solved by supervised machine learning is classification. So, how do they differ from one another? Classification is the process of teaching a model to categorise an object according to its characteristics. Such as, to discover phishing emails, the process may involve installing a firewall or using facial recognition software. On the basis of labelled input and output data, an algorithm will be created to determine the precise traits that distinguish marked items. On the other hand, a regression problem appears when a model is employed to predict continuous results or values. This model might be able to forecast shifts in retail sales, housing costs, or salaries. The model undergoes training using tagged data from the input and output to ascertain the strength of connections between data features and output.

Regression is used to find patterns and relationships in a dataset so that they can be applied to fresh, unknowable data. Regression is therefore a crucial part of machine learning in accounting and is commonly applied to forecast stock prices, market dynamics, or asset success. Models can be taught to understand how various features connect to the desired outcome. Regression using machine learning typically gives businesses insight into particular outcomes. Yet, machine learning's capacity for explanation is a key consideration due to how this tactic may impact an organization's decision-making process. (Kothari, C. ,2017)

To summarize, the following are some typical applications for machine learning regression models:

- Forecasting a continuous result, such as stock prices, sales, or property values.
- Estimating the success of future retail or marketing initiatives in order to ensure that resources are used efficiently.
- Predict user or customer behaviour on webpages that sell goods or offer streaming services, etc.

- Evaluating data sets to discover the connections between factors and outcomes.
- Calculating interest rates or stocks value based on a diversity of factors.
- Data visualisation for time series.

4.2 Linear Regression

Simple linear regression draws an imaginary line inside the data points such that the variance between the line and the data points is least. It is one of the most fundamental and straightforward kinds of machine learning regression. In this instance, the dependent and independent variables are thought to have a linear relationship that is either positive or negative. This method is the easiest form of regression due to the fact that it only analyses the factor that is dependent and one other independent variable. Outliers are common in basic linear regression because the straight line of best fit is used. (Kanade, V., 2022). Linear regression is a fundamental and widely used predictive analytic technique.

The two primary goals of the regression technique are:

- i. Does a set of predictor factors accurately predict a dependent variable (outcomes)?
- ii. Which individual factors, as evidenced by the magnitude as well as the trajectory of the beta values, are highly important indicators of the output variable, and how do they affect it?

These regression approximations are utilized to articulate the connection among a variable that is dependent and a number of independent variables. The regression dependent variable goes by several different names such as standard variable, regression variable, inherent variable, or outcome variable. The independent factors are also known as external variables, variables that predict, and a regression line.

There are three primary uses for regression analysis:

- i. Measuring the predictive power of variables.
- ii. Predicting a result.
- iii. Prediction of trends.

Finding the degree to which an independent variable has an effect on a dependent variable is the first stage in the regression analysis. Common examples include the intensity of the dose-effect on recovery, marketing expenses effect on sales, or age effect on money spent.

It can be used to predict how alterations will impact various aspects of life. In other words, regression analysis enables us to understand the extent to which the dependent variable alters when one or even more independent factors alter. The question "How much more in sales income do I receive for every additional \$1,000 I spend on marketing?" is one that is frequently asked question to increase marketing expenses such that it positively effects sales. Regression analysis also predicts future trends and values. Regression analysis can be used to get continuous values. A common question is what the gold price is going to be in six months.

4.2.1 Various Types of Regression

- i. *Simple linear regression*: It uses a single independent variable along with a variable that is dependent (period or percentage).
- ii. *Multivariate linear regression*: a ratio or interval that has more than two independent variables and one dependent variable (interval or ratio or dichotomous).
- iii. *Rational regression*: Two or more independent variables, one dependent variable that is dichotomous (interval or ratio) that means regression equation is the ratio of two linear equations.
- iv. Regular regression; ordinal dependent variable, one or even more independent factors (nominal or dichotomous) One nominal dependent variable, multiple independent variables in multiple regression (interval or ratio or dichotomous)
- v. Multiple independent variables are compared to one notional dependent variable (interval or ratio).

4.2.2 Simple Linear Regression

The application of linear regression models demonstrates or predicts the connection among two separate variables or factors. The dependent variable is the expected variable. (the variable the equation solves for). The independent factors are those that are used to forecast the value of the dependent variable. Each observation in a linear regression has two values. There is only one value for each independent and dependent variable. In this fundamental model, the link between the dependent and independent variables is typically shown by a straight line. There are various variables that reflect the demographic being studied. These model parameters are represented by 0 and 1. The basic linear regression equation appears as a straight line on a grid.

A regression line illustrates a positive, negative, or irrelevant linear connection. The regression line is said to slopes upward, that begins from the points closer to x –axis and raises itself far from the x -

intercept. This signals a productive or positive relationship (axis). The relationship between the two variables is characterized by positive linearity, which means that when one variable's value increases, so does the other. When the regression line's bottom end stretches into the graph field in the direction of the x -intercept from the graph's y -intercept (axis) represents the regression line's upper side (axis). The two variables are said to be negatively correlated, which means that if one variable's value increases, the other decreases, this is referred to as negative regression. If the population's characteristics were known, the mean value of y for a given value of x could be calculated using the basic linear regression equation. When there is no slope in the regression line, it represents no-relation between the variables. However, in real-world situations, parameter values are typically unknown, thus they must be approximated using information from a sample of the population.

Sample statistics are used to estimate the population parameters. The numbers 0 and 1 represent the sample statistics. By substituting the sample data for the population parameters in the computed regression equation.

According to the estimated regression equation (4.2).

$$(y) = \beta_0 + \beta_1 x + \varepsilon \quad (4.2)$$

Where,

β_0 : y -intercept of line, that represents the value of y when the mean value of x is 0.

β_1 : slope/gradient, that defines the rate of change of y with respect to x .

y : The predicted value of dependent variable.

ε : it represents the error estimates of real life.

These variables may be represented by the other notations also. For example, in figure 4.1, the regression equation (4.3) is displayed.

$$y = mx + c \quad (4.3)$$

Where, y is dependent variable, x is independent variable and regression coefficients are m and c .

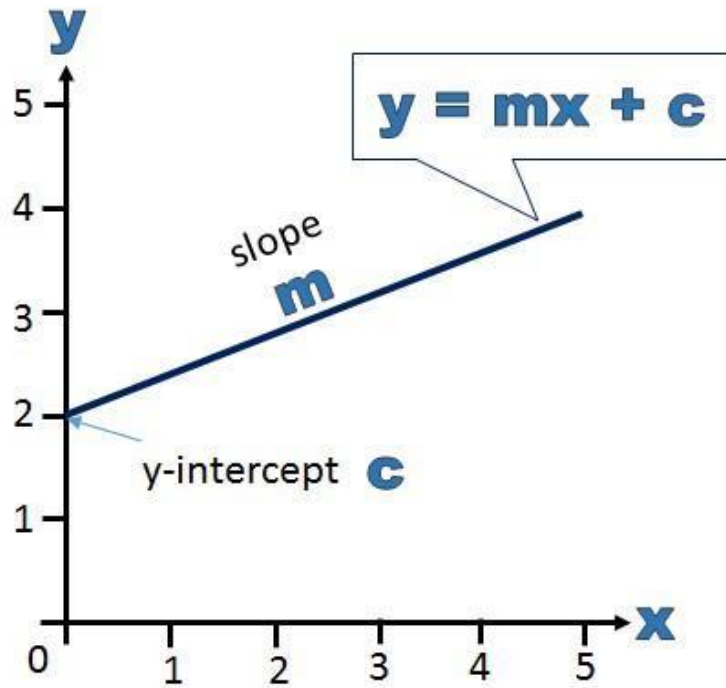


Fig. 4.1 Demonstration of slop and intercept using formula $y = mx + c$ (Kanade, V. ,2022)

4.2.3 Principal advantages of linear regression

Given the many advantages it provides, linear regression is a widely used statistical tool in data science. These advantages include:

1) Simple application

The linear regression model is simple to build computationally since there aren't many engineering overheads needed, either before or after the model/machine is started.

2) Interpretability

In comparison to other deep learning models, the linear regression model is easier (neural networks). This method performs better than black-box models. Black box models are those models which cannot show how an input variable affects an output variable. Regression model works different to black box models.

3) Scalability

Scaling is frequently required, and since linear regression requires little computational effort, it works well in these circumstances. For instance, the model scales effectively as the volume of data grows.

4) Ideal settings for internet use

Due to their simplicity in computation, these techniques can be used in online environments. The models may be trained and trained up with each new instance to produce predictions in real-time, as opposed to neural networks or support vector machines, which must be computationally complex, resource-heavy, and wait a long time to retrain on a new dataset. Due to all of these problems, such compute-intensive models are expensive and inappropriate for real-time applications.

The aforementioned qualities clearly show why using linear regression as a solution to practical machine learning issues is a common practice.

4.3 Linear Regression Examples

1. A hospital would be curious to see how a patient's body weight affects the entire cost of their treatment.
2. Insurance firms are interested in knowing how healthcare expenses and ageing are related.
3. A company can be interested in learning how variables like pricing, marketing expenses, rival prices, and promotion charges affect the sales of a product.
4. Restaurants want to discover the correlation between revenue and the amount of time customers wait after placing an order.
5. E-commerce businesses like Amazon, BigBasket, and Flipkart would like to know how income and features like
 - (a) The number of users who have visited their portal.
 - (b) The number of product clicks.
 - (c) The number of goods being sold.
 - (d) The typical discount rates.
6. Banks and other financial institutions are interested in knowing how different factors, such as the unemployment rate, marital status, account balance, rainfall, etc., affect the proportion of non-performing assets (NPA). (Kanade, V., 2022)

4.4 Multi-Linear Regression and examples

To predict the outcome of a parameter according to the values of two or more additional components, a statistical technique called multiple linear regression is utilised. Multiple regression is the name given to this progression of linear regression. The variables themselves are referred to as the dependent variable, while the variables we utilise to forecast the value of the dependent variable are referred to as independent or explanatory variables. Researchers can use this method to determine the variation of the model as well as the relative contributions of each independent variable to the total variance. Multiple regression can be divided into two categories: linear regression and non-linear regression. (Pradhan, M., & Kumar, U. D, 2019)

The formula for multiple linear regression is given in equation (4.4):

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_i + \varepsilon \quad (4.4)$$

Where:

y : dependent or predicted variable

$\{x_1, x_2, x_3, \dots, x_i\}$ are the independent variables, they all together form feature set represented by X .

β_i : regression coefficient that defines the rate of change of y with respect to feature x_i .

ε : model's random error (residual) that defines the permissible error rate.

y -intercept is β_0 i.e., the value of y , when both x_i and ε are 0.

Example 1:

With the help of an example in table 4.1, let's try to learn the significance of multiple regression analysis. Let's start to evaluate the relationship between an UBER driver's travelled distance, age (years), and number of years of experience.

Table 4.1 Dataset for the Regression Example.

Experience	Age	Distance (Kilometres)
5	18	32513
7	20	27897
8	22	29929
6	23	30159
7	23	21554
5	25	28466
8	2	27842
6	28	33671

5	29	32214
7	32	34550
9	37	20920
6	41	33714
7	46	26998
8	49	34294
6	53	21912

Performing Regression in Excel

Firstly, you have to install the plugin tool pack for data analysis. In Excel, choose the "Data" tab and choose the "Data Analysis" option to perform multiple regression. You will get the outcome similar to figure 4.2.

D	E	F	G	H	I	J	K	L	M	N
	SUMMARY OUTPUT									
	Regression Statistics									
	Multiple R	0.332038084								
	R Square	0.110249289								
	Adjusted R Square	-0.038042496								
	Standard Error	4758.496205								
	Observations	15								
	ANOVA									
		df	SS	MS	F	Significance F				
	Regression	2	33668840.11	16834420	0.743462	0.496146648				
	Residual	12	271719433.6	22643286						
	Total	14	305388273.7							
		Coefficients	Standard Error	t Stat	P-value	Lower 95%	Upper 95%	Lower 95.0%	Upper 95.0%	
	Intercept	37908.64599	7447.755504	5.089942	0.000266	21681.38074	54135.91123	21681.3807	54135.91123	
	Experience (Years)	-1230.274132	1030.756319	-1.19356	0.255711	-3476.09922	1015.550961	-3476.0992	1015.550961	
	Age (years)	-20.02070678	94.46943243	-0.21193	0.83572	-225.851918	185.8105046	-225.85192	185.8105046	
		-47396984.43								

Fig. 4.2 Multiple regression calculation in Excel for example 1.

Calculate the regression by using the above-discussed formula in equation (4.4).

$$\begin{aligned}
 y_i &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 \\
 &= 0 + 37908.6 * -1230.2 + 37908.6 * -20.0 \\
 &= -47396984.43
 \end{aligned}$$

In this particular scenario, we'll examine the variable that is the dependent variable (distance travelled) whereas the other variables are the independent variables (age and experience) i.e. the journey that the UBER driver has travelled is the depends on the person's age and the extent of his driving experience.

Example 2:

Let's now investigate the relationship amongst a staff team's wage, their age, and the number of years they have worked for the company. The dataset is given in table 4.2.

Table 4.2 Dataset for regression example

Salary (Rupees)	Experience (Years)	Age (years)
32513	2	33
27897	4	20
29929	5	28
30159	3	29
21554	4	32
28466	5	37
27842	5	41
33671	6	46
32214	5	49
34550	5	32
20920	6	37
33714	6	41
26998	2	46
34294	2	28
21912	1	50

Select the "Data Analysis" choice for the computation under the "Data" tab in Excel. The outcome is given in figure 4.3.

Calculate the regression equation (4.4):

$$y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

$$= -1576979520$$

In this specific case, we will examine which is the independent variable and which is the dependent variable. In this regression equation, skills and worker's age are independent factors, and salary is the dependent variable.

Sheet 1 (+)

Chapter 5

Decision Tree

5.1 Introduction to Decision tree

A decision tree is the simplest and extremely understandable categorization or classification approach. It is a powerful supervised machine learning algorithm that has a hierarchy based-tree like structure, whose leaf nodes contains the class labels; root node and internal node represents the attribute test or split test. The results of attribute test i.e., decisions are represented by the branches, therefore it referred to as decision tree. Moving through the multiple attribute test and respective decisions; leaf node is arrived that defines the outcome label of the test case. The figure 5.1 represents the sample decision tree to decide whether to play tennis or not. (*Decision tree (2023) GeeksforGeeks*)

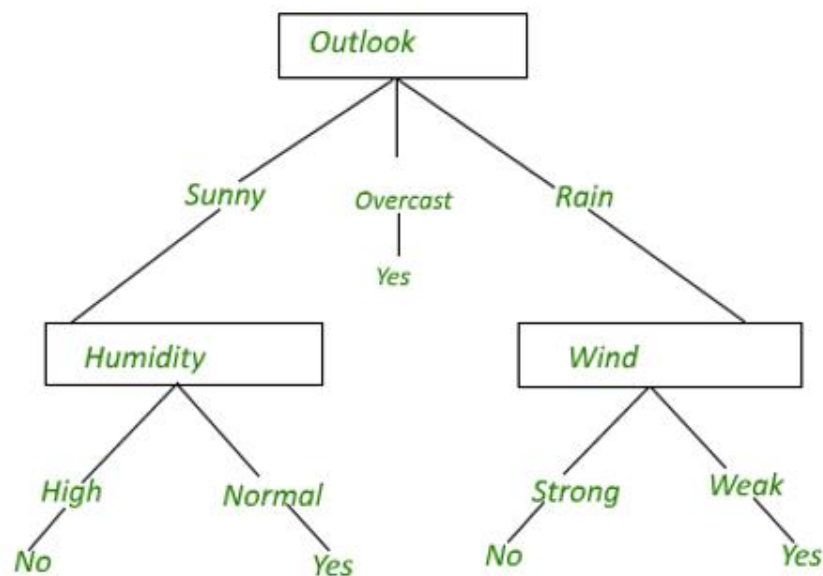


Fig. 5.1 Sample Decision Tree (*Decision tree (2023) GeeksforGeeks*)

Steps to create Decision Tree:

It is essential to comprehend the creation of a decision tree because of the real intricacy of the approach and the tight associations of hyperparameters. A greedy iterative division of the initial dataset forms the basis of the decision tree approach. It employs a structure for decision-based tree for making verdict over the implementation of the partitions. Learning involves the succeeding steps:

Step 1. Start by identifying the feature or attribute that will best distinguish between the classes using the complete dataset. Metrics like computing information gain or the Gini index are used to identify the finest trait.

Step 2. After identifying the finest characteristic, split the data collection into two (or more) parts of the attribute value based on split test results.

Step 3. If a part of the dataset only contains labels for a class, the process stops here and designate it as the leaf node i.e., class label.

Step 4. This procedure is repeated until we have all leaf nodes, that contain information from the same class. The finalized decision tree algorithm's model is shown in a flowchart.

The decision tree is simple to learn it just involves the tracing of path from root node to leaf node through multiple levels using attribute tests. The expected class for the data point is present in the leaf node you end up with. The modelled test attribute tests are the characteristics that defines the instance. For example, to decide the mortality the characteristics of human such as sex, age, etc. may be used in the same order. After partitioning the dataset on these attributes, the following inferences may be observed such that a woman would have most likely survived and the baby boys are more likely to die before 9 months and 6 years of age.

An instance may have numerous characteristics or attributes to define. Partitioning data on every attribute may not give useful inferences or may generate a tree with many attributes arranged at multiple levels of the tree. Such tree may generate the decisions that may lead leaf nodes that have only one sample under it. This will cause the problem of overfitting and model will not be able to learn anything. This problem may be solved by the defining a hyperparameter, called *min samples leaf*, this manages to split the data until a threshold is arrived. As dataset partitioning meets threshold criteria i.e., minimum number of samples under a decision or leaf node, the splitting process halts. For instance, we want to halt the dividing process and mark a leaf node with the majority class if it makes up 3%-4% of the entire dataset. This type of methods is called splitting criterion. Other criteria are tree maximum depth, the tree's maximum depth (max depth), numeral features (max features), etc.

Such type of criterion not only included in the decision tree but is included in other methods also such as logistic regression, loss rates in neural networks, and kernels in SVMs. In general, hyperparameters and learning algorithms have a strong connection that creates a noticeable impact on the overall quality of the model. Therefore, it is mandatory to understand the technique to decide and evaluate the change in the value of specific variables or hyperparameters.

5.1.1 Decision Tree Benefits:

Decision trees, among other machine learning techniques, (Shalev-Shwartz, S., & Ben-David, S., 2014) have the succeeding benefits:

- 1) **Easy to comprehend and interpret.** Decision trees are easily understood and traced by not only the programmers but also by the non-technical persons. It is a comprehensive representation of the decisions. Data standardization, the creation of fake variables, and the elimination of null values are common requirements for other techniques.
- 2) **Capable of working with both categorical and numerical data.** Other approaches are frequently concentrated on the analysis of datasets with just one type of variable.
- 3) **It is reliable model** as it uses white box rendering. Boolean logic can be used to easily describe a condition if it is possible to witness it in a model. It is feasible to validate a model using statistical testing.
- 4) **Works well with big datasets**, even when the base model that the data were derived from slightly violates some of its principles. With standard computing capabilities, large amounts of data can be evaluated in a reasonable length of time.

Decision Tree Limitations:

Decision trees, among other machine learning techniques, have the succeeding limitations:

- 1) Decision-tree learners could create overly complex trees from the training data that are difficult to generalize. With a few exceptions, such as the Conditional Inference method, most algorithms require mechanisms like pruning to avoid this issue. Because decision trees don't adequately illustrate some issues, including XOR, parity, or multiplexer challenges, they can be difficult to comprehend. The decision tree can get too large in some circumstances. The difficulty may be overcome by employing learning approaches based on more expressive




representations or by propositionalizing, or changing how the problem domain is represented (such as deductive logic programming or statistical relationship learning).

- 2) Decision tree with categorical variables when using information gain criteria for splitting it may favour attributes with higher levels of data having categorical attributes though, the Conditional Inference method avoids the problem of biased predictor selection.

5.2 Decision Tree Representation

Decision trees are used to formally and graphically describe choices and decision making in decision analysis. As the term suggests, it uses a tree-like hierarchy-based approach. It is often used in machine learning based applications to generate an action plan to achieve a specific goal. The tree uses a specific representation language given in table 5.1.

Table 5.1. Representations used in decision tree.

Notation	Description
Rectangle, 	Represent the attribute test
Leaf node, 	Represents the class label
Branches, 	Represents the result of the attribute test

5.2.1 Example of Decision Tree:

The representation of the decision tree is illustrated in figure 5.2. It predicts about the planning of the day, the characteristics that may affect the decision are friends' visit, weather and available money.

The flow of the decision tree and attribute tests depict that if a friend is visiting the house, then they will spend day at restaurant. If no one comes at home, day planning depends on weather conditions. For the rainy day, it is better to stay indoors while on a sunny day, person will go out and play. In

case if, weather is windy, and then person may opt to go either on shopping or movies, as per available money.

The decision tree is also known as classification tree as it classifies whether to go to restaurant, stay indoors, go for shopping or go outside to play. This is known as decision tree learning from the dataset.

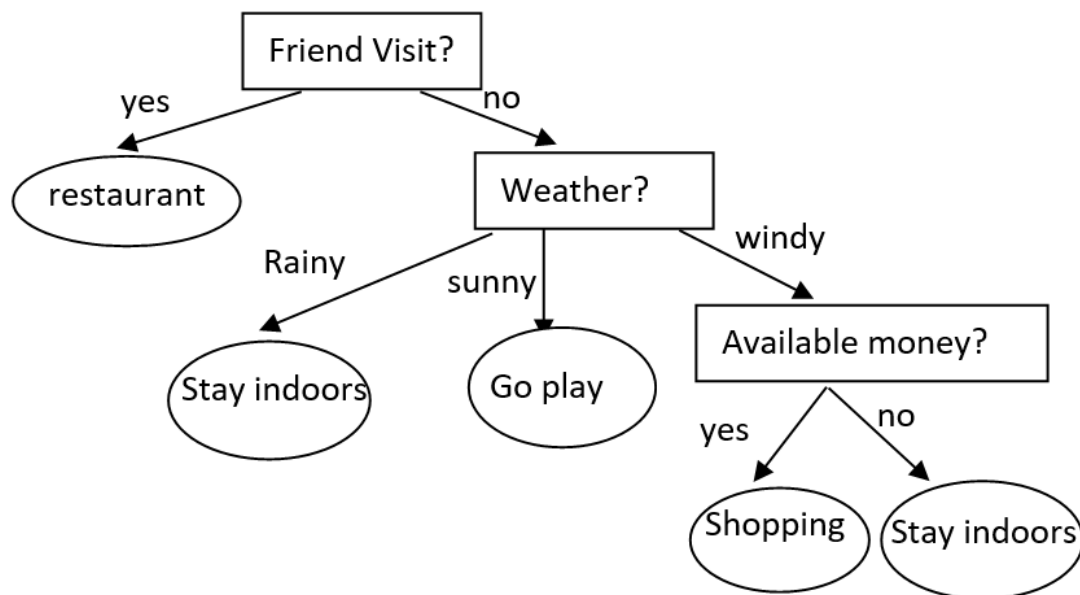


Fig. 5.2 Decision tree for planning a day.

The simplicity of this approach cannot be overstated, although real datasets contain much more functionality as compared to this example of day-planning and this may be the small part of a larger tree. The correlation of attributes is clear and the connection between things is easy to understand.

The example aforementioned is the Classification Tree that gives predictions in terms of categories. Other variant of decision tree is Regression Tree that predicts the continuous value of variable as per attribute tests such as predicting the amount of dosage for patients, forecasting the Covid cases in coming winters, etc. Some specific decision trees serve both purposes i.e., classification and regression. Such trees are known as Classification and Regression Tree (CART).

5.2.2 Advantages and Disadvantages of CART

a) Advantages of CART

- Easy to recognize, analyze, and envision.

- CARTs are capable of processing both category and numerical data and can automatically screen variables or choose characteristics. Additionally, it can manage issues with multiple outcomes.
- Users don't have to put in a lot of effort to prepare their material for decision trees.
- Non-linear interactions between parameters have no impact on a tree's performance.

b) Disadvantages of CART

- Overfitting is a term used to describe how novice decision tree developers frequently produce overly complicated trees that fall short of properly generalizing the data.
- CART can become unreliable because even small changes in the data can produce completely different trees. A greedy algorithm cannot guarantee a globally optimal decision tree; this is called variance, and strategies such as bagging and boosting should be used to reduce the variance. This can be reduced by training multiple trees using replacement sampling for features and samples.
- A decision tree learner will build biased trees if certain categories are dominant. Therefore, it is recommended to balance the dataset before fitting it to a decision tree.

Recursive and greedy Splitting

Every feature is considered during the procedure, and a cost function is used to test a number of split points. It is decided which division has the best (or least expensive) cost.

Consider the tree that was obtained in the example of day planning. The original split, also referred to as the root, which considers all traits and features, is used to categorize the training data into groups. There will be three possible divisions because there are three characteristics. Each split's accuracy cost will now be determined by a function. In this case, the passenger's sex is selected as the split with the lowest expense. Since the groups that are created can be further split using the same method, this approach is recursive in nature. As a result of this procedure, the algorithm has gained the nickname "greedy algorithm" because we have a strong desire to lower the cost. Therefore, the master node is the greatest classifier.

Distribution Costs

The cost function finds the most unified branch, or the branch with similarly behaving groups, in both scenarios. This makes sense given that there is a greater likelihood that the entry of test data will follow a specific course.

Cost function for Regression:

$$C = \text{sum} (y - \text{prediction})^2 \quad (5.1)$$

Assume for the moment that when the home prices are to be forecasted. Now that segmentation has started, the decision tree will consider every feature in the training set. A prediction for a given group is defined using the mean of the responses of the training data sources for that group. All data points are subjected to the aforementioned function in equation (5.1), and the cost is determined for each possible attribute. The attribute with the lowest expense is chosen for split. The decrease in standard deviation is another cost function.

Classification:

$$G = \text{sum} (pk * (1 - pk)) \quad (5.2)$$

Based on the uniform distribution of response categories among the groups formed by the distribution, the Gini score in equation (5.2) describes the quality of the split. The pk notation indicates the number of entries of the same class contained in a given group. Since it has the lowest purity, a node with a 50/50 split between classes within a cluster will have $pk = 0.5$ and $G = 0.5$ for binary classification. Nodes with perfect class purity are created when a group contains all entries belonging to the same class, in this case pk is 1 or 0, and $G = 0$.

When to stop splitting?

The decision tree implementation is iterative and greedy. You may be curious as to when a tree should stop growing. An instance often has a lot of characteristics or attributes, which causes lots of splits, which results in a huge tree. This may cause decision tree run the risk of being overfit. So how do we decide when to stop? For this purpose, it is required to add the splitting criterion in the implementation. One method to do it is to specify a least amount of training input to utilize on every leaf. Another option is to specify a maximum depth for the model. The maximum depth is the distance you can travel along the path from root to leaf.

Pruning

Pruning can enhance a decision tree performance even afterwards. It entails eliminating the subdivide that depends on insignificant qualities. By doing this, we lower the tree's complexity and, by reducing overfitting, raise its predictive value.

Pruning may begin at the leaves or the root. The most basic pruning technique begins at the leaf level and eliminates every node belonging to the class that is most popular there; if this adjustment doesn't

reduce accuracy, it is preserved. Additionally, it is known as low error pruning. A learned parameter called alpha, is used along with more sophisticated pruning techniques like cost complexity pruning to determine if nodes can be removed based on the extent of the subtree. This is called weakest link pruning.

5.3 Appropriate Problem for Decision Tree Learning Algorithm

Decision tree has wide scope capabilities to solve the problems defines with following characteristics.

- 1) Attribute-value pairs serve as the representation of instances.
“Instances are described by a fixed set of attributes (e.g., Temperature) and their values (e.g., Hot). The easiest situation for decision tree learning is when each attribute takes on a small number of disjoint possible values (e.g., Hot, Mild, Cold). However, extensions to the basic algorithm allow handling real-valued attributes as well (e.g., representing Temperature numerically).”
- 2) The output values of the target function are discrete.
“The decision tree is usually used for Boolean classification (e.g., Y or N) kind of example. Decision tree methods easily extend to learning functions with more than two possible output values. A more substantial extension allows learning target functions with real-valued outputs, though the application of decision trees in this setting is less common.”
- 3) Sometimes disjunctive descriptions are necessary.
Disjunctive statements are naturally represented by decision trees.
- 4) There could be mistakes in the given data.
“Decision tree learning methods are robust to errors, both errors in classifications of the training examples and errors in the attribute values that describe these examples.”
- 5) There could be empty attribute values in the given data.
“Decision tree methods can be used even when some training examples have unknown values (e.g., if the Humidity of the day is known for only some of the training examples).”

5.4 Basic Decision Tree Learning Algorithm

The family of supervised learning algorithms includes decision tree algorithms. Unlike other supervised learning methods, decision tree techniques can handle both classification and regression problems. The decision tree learns from the historical data and generates multiple inferences and rules that may be utilized to predict classes (classification tree) or predict value of continuous variable (regression tree).

The decision tree method looks for a solution by modelling the problem as a tree-like structure shown in figure 5.3. The internal nodes of the tree relate to attributes, while the leaf nodes of the tree correspond to class labels.

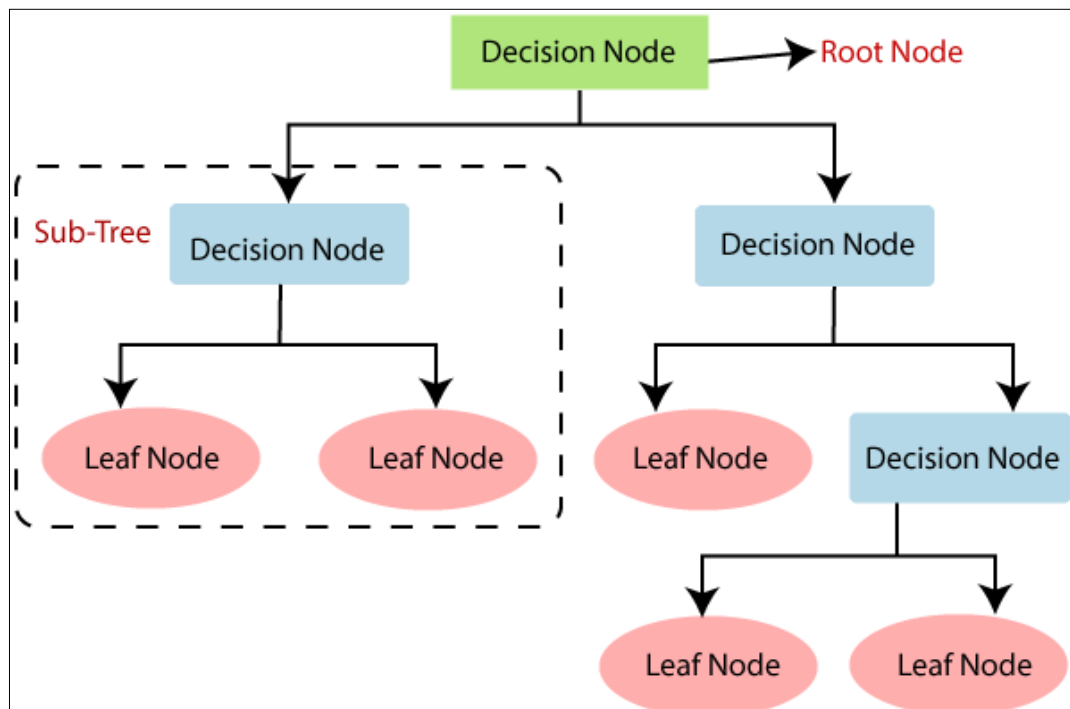


Fig.5.3 Basic representation of decision tree (*Javatpoint*)

5.4.1 Decision Tree Training

The general steps for decision tree training are as follows:

Step 1. Place the best features of the dataset at the top of the tree.

Step 2. Break training set into more manageable units or data subsets. Each data subset must contain the same value for each attribute. This should cause subsets to be generated on the basis of root attribute.

Step 3. Repeat steps 1 and 2 for each subgroup until each branch of the tree has visible leaf nodes as class labels.

5.4.2 Predicting using Decision Tree:

Step 1. When predicting the class label of a new instance, decision trees initiate at the root of the tree.

Step 2. We check the new instance's attribute values with those of the base feature at the root.

Step 3. Based on the previous step, we move to the next child (training subset) along the branch corresponding to this value.

Step 4. Step 3 is repeated until, leaf node is arrived. The leaf node marks the output label of the new instance.

5.4.3 Assumptions

While using a decision tree, we understand the succeeding assumptions, like:

- i. The complete training set is initially considered the root.
- ii. Classification of attribute values takes precedence. When there are continuous values, discretize them before building the model.
- iii. Recursively, entries are assigned on the basis of features' values.
- iv. The properties of the tree are organized in ascending or descending order using a statistical method.
- v. The organization of features in the tree follows the greedy approach.

5.5 Entropy Measures

In classification trees, entropy is well-known impurity measurement physical attribute. It is utilized in decision tree to check the quality of split at an attribute-based node in the decision tree. Imagine there are J classes with the labels $1, 2, \dots, J$. Given in equation (5.3) is the entropy at node k .

$$\text{Entropy} = - \sum_{j=1}^J (p_{jk}) \log_2(p_{jk}) \quad (5.3)$$

Entropy ranges from 0 to 1, larger the value, more is the impurities at the node i.e. misclassification or partitioning of the training subsets.

5.6 Information Gain Measures

Information gain and Kullback-Leibler divergence are synonymous in information theory and machine learning. However, the phrase is sometimes synonymous with mutual information also. The mutual information is used as the correlation co-efficient between two random variables. Its value is

equal to or larger than 0. 0 value means variables are independent. The mutual information can be computed as the Kullback-Leibler (KL) divergence. Formally, it is given in equation (5.4).

$$I(X ; Y) = KL(p(X, Y) || p(X) * p(Y)) \quad (5.4)$$

Where, $||$ is the divergence operator, $p(X, Y)$ is the joint probability of variable X and Y .

To quickly constrain the state of X , this concept can be applied in machine learning to describe the desired sequence of attributes to be considered to develop hierarchy of the decision tree. A decision tree is this list of steps. In general, attributes with high mutual information should be chosen over others (Pradhan, M., & Kumar, U. D. 2019, Machine Learning with Python.).

General definition

The expected information gain is computed as the change in entropy H in equation (5.5) from a previous state to a state that accepts the particular information provided.

$$IG(T, a) = H(T) - H(T|a) \quad (5.5)$$

Formal definition

Assume T be a set of training instances, each of which is defined as $(X, y) = (x_1, x_2, x_3, \dots, x_k, y)$, where x_i the value of the first characteristic of the example is and y is the class label with respect to feature set X . In terms of entropy $H()$, the information gain of an attribute is defined as follows in equation (5.6):

$$IG(T, a) = H(T) - \sum_{v \in vals(a)} |\{x \in T | xa = v\}| / |T| \cdot H(\{x \in T | xa = v\}) \quad (5.6)$$

If it is possible to build a different classification for each of the attribute values for the result attribute, the mutual information is equal to the total entropy of the characteristic. In this instance, the overall entropy is zero after deducting the relative entropies.

Drawbacks

Information gain is not perfect, but it is often a good indicator of how well a characteristic applies or splits the dataset with minimum impurities. There is a clear problem when knowledge gain is used to

describe traits that have a large variety of various values. Think about leveraging client information to build a decision tree, for instance. Information gain is usually used to examine the traits towards the base of the tree in order to decide which trait is most crucial. But sometimes even when the information gain and mutual information favour one attribute may not be included in the decision tree. For example, the customer's credit card number because each customer is identified by the unique credit card; this fact will cause a high mutual information. Even then it will not be applied in decision making. Including such attributes in the decision tree may cause a biasness biased as a result to neglect characteristics that have a wide range of potential values. So, all of a sudden it seemed that traits with extremely low information levels had an unfair advantage.

5.7 Example Problems for Illustrating ID3

ID3 is the method to generate the decision tree from dataset, developed by Ross Quinlan. It stands for Iterative Dichotomizer. It iterates through each unused attribute and identifies the attribute with lowest entropy or highest information gain and partitions the dataset.

The detailed steps of IDS are given as follows:

- Step 1. Determine the information gain of each unused feature.
- Step 2. Select the feature with the highest information gain and split the dataset according to its value.
- Step 3. Build the decision tree by iterating through step 1 and step 2 until the iterations stop if any of the conditions meet. The conditions are
 - 1) No unused attributes are left.
 - 2) Every element of the subset belongs to the same class, in such cases, the node is marked as leaf.
 - 3) No training instances are left in the dataset.

Example: COVID-19 Dataset description

The example included the dataset related to COVID-19 infections. An overview of the dataset is shown in table 5.2. The sample contains 14 training examples, each training example or COVID-19 infection is defined by the feature set {ID, Fever, Cough, Breathing Issues} with infected as class label. Y and N represent the values “yes” or “no” respectively.

Table 5.2 COVID-19 sample dataset.

ID	Fever	Cough	Breathing Issues	Infected
1	N	N	N	Y
2	Y	Y	Y	Y
3	Y	Y	N	N
4	Y	N	Y	Y
5	Y	Y	Y	Y
6	N	Y	N	N
7	Y	N	Y	Y
8	Y	N	Y	Y
9	N	Y	Y	Y
10	Y	Y	N	Y
11	N	Y	N	N
12	N	Y	Y	Y
13	N	Y	Y	N
14	N	Y	N	N

5.7.1 Metrics used in ID3: Entropy and Information Gain

As mentioned earlier, the ID3 algorithm builds a decision tree at each step by selecting the best features. Before asking the question, answer the question: "How does ID3 select the best features?"

Yes, ID3 uses information gain, or simply gain, to determine the best features.

Gain of Information assesses how well a certain characteristic categorizes or separates the target classes and determines the attribute that splits dataset with least entropy.

If data set is denoted by S and the entropy is calculated as in equation (5.7):

$$Entropy(S) = - \sum p_i * \log_2(p_i) ; i = 1 \text{ to } n \quad (5.7)$$

where n is the total number of categories in the target column (i.e. infected, $n = 2$ (Y and N) in COVID sample dataset), and p_i is the probability of category i that is computed as the ratio of "number of rows for the category i in the target column" to the dataset "Total Rows".

The information gain is computed for each attribute or feature in the dataset. To assume for attribute A is calculated as follows in equation (5.8):

$$IG(S, A) = Entropy(S) - \sum((|S_v| / |S|) * Entropy(S_v)) \quad (5.8)$$

where S_v is the set of rows in S with value v in for the attribute under computation, S is the total number of rows in the dataset. In the COVID example, $A \in \{ID, Fever, Cough, Breathing Issues\}$.

5.7.2 Application of Information Gain and Entropy on COVID dataset to identify best feature:

Step 1. Computing Entropy:

Finding the finest feature—that is, the one with the greatest Knowledge Gain—is the first stage, as mentioned in the previous section, it is required to compute information gain for each attribute. To compute information gain entropy measure (including each possible class label) is required. The entropy of S is determined using equation (5.7).

Of the total 14 rows in our data set S , there are 8 rows with target value Y and 6 rows with target value N. Determine the entropy of S is as in equation (5.9):

$$Entropy(S) = - (9/14) * \log_2(9/14) - (5/14) * \log_2(5/14) = 0.32 \quad (5.9)$$

The entropy will be 0 if all the values in our goal column are the same (means it has N or zero randomness).

Step 2. Computing information gain for each attribute.

After computing the entropy, it is required to compute the information gain to identify the best splitting feature among unused feature sets.

Information Gain calculation for fever: This trait (Fever) has 7 rows of Y values and 7 rows of N values. The next 7 rows of Y represent fever with 6 rows of Y target value and 1 row of N target value

(given in table 5.3). Table 5.4, among the 7 rows of N, there are 2 rows with the target value of Y and 5 rows with the target value of N.

Table 5.3 Fever attribute with Y value.

Fever	Cough	Breathing Issues	Infected
Y	Y	Y	Y
Y	Y	N	N
Y	N	Y	Y
Y	Y	Y	Y
Y	N	Y	Y
Y	N	Y	Y
Y	Y	N	Y

Table 5.4 Fever attribute with N value.

Fever	Cough	Breathing Issues	Infected
N	N	N	N
N	Y	N	N
N	Y	Y	Y
N	Y	N	N
N	Y	Y	Y
N	Y	Y	N
N	Y	N	N

Information Gain: Fever attribute.

Number of Total tuples $|S| = 14$

$$\text{For, } v = Y, |S_v| = 7$$

$$\text{Entropy}(S_v) = -(6/7) * \log_2(6/7) - (1/7) * \log_2(1/7) = 0.175$$

$$\text{For, } v = N, |S_v| = 7$$

$$\text{Entropy}(S_v) = -(5/7) * \log_2(5/7) - (2/7) * \log_2(2/7) = 0.259$$

Exceeding the conclusion in the Information Gain (IG) formula:

$$IG(S, \text{Fever}) = \text{Entropy}(S) - (|S_{YES}| / |S|) * \text{Entropy}(S_{YES}) - (|S_{NO}| / |S|) * \text{Entropy}(S_{NO})$$

$$IG(S, \text{Fever}) = 0.32 - (7/14) * 0.17 - (7/14) * 0.25 = .011$$

Similarly, IG may be calculated in a similar way for the characteristic's "cough" and "breathing issues".

$$IG(S, \text{Cough}) = 0.04$$

$$IG(S, \text{Breathing Issues}) = 0.40$$

The root node is created using the feature Breathing issues since it has the largest Information Gain. As a result, after taking this first step, the decision tree looks like in figure 5.4.

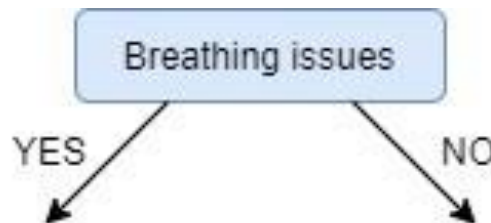


Fig. 5.4 Decision tree with Breathing issue split. (Sakkaf, Y. (2020) *Decision trees for classification*)

The procedure of computing information gain (IG) is iterated for the remaining attributes i.e., fever and cough for the subsets distributed under breathing issues. (Sakkaf, Y. (2020) *Decision trees for classification*)

The left subset for which breathing issues is Y is represented in table 5.5 and the right subset is represented in table 5.6 for which breathing issue is N.

Table 5.5 Left subset after partitioning dataset with respect to breathing issues in figure 5.4

Fever	Cough	Breathing Issues	Infected
Y	Y	Y	Y
Y	N	Y	Y
Y	Y	Y	Y
Y	N	Y	Y
Y	N	Y	Y
N	Y	Y	Y

The GI of the Fever and Cough characteristics is then calculated using the subset S_A in table 5.5. (a set of respiratory problems Y) described above:

$$IG(S_{AY}, \text{Fever}) = 0.20$$

$$IG(S_{AY}, \text{Cough}) = 0.09$$

N	Y	Y	Y
N	Y	Y	N

The fever has the higher information gain therefore the decision tree takes the form of data as shown in figure 5.5.

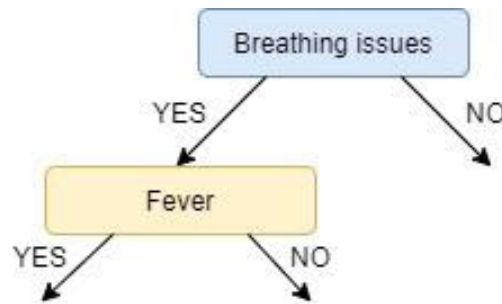


Fig. 5.5 decision tree with fever split. (Sakkaf, Y. (2020) *Decision trees for classification*)

Only one attribute is left i.e., cough attributes. This can be organized for at level second for the left subtree subset (instances for which breathing issue has N class label). The decision tree with all three attribute-test is shown in figure 5.6.

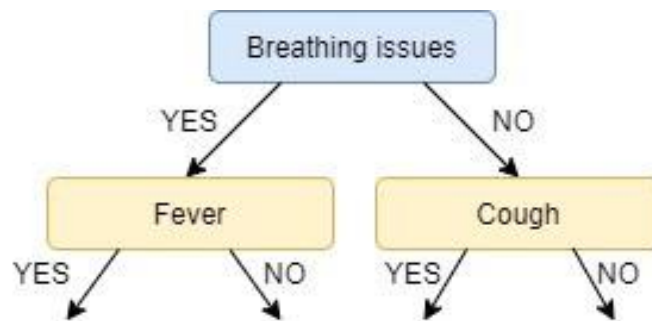


Fig 5.6 Decision tree all attribute tests for split. (Sakkaf, Y., 2020)

No attribute is left to be organized on the decision tree; this marks the last step of building the decision tree i.e., creating leaf nodes. As per the decision tree in figure 5.6, there are four leaf nodes. Reading from most left to the most right, the four-leaf nodes are:

Leaf node1: The left most leaf node for which Fever and Breathing issue both have value Y, shown in table 5.6. all

Leaf node 2: The next leaf for which Fever has value N and Breathing issue has value Y, shown in table 5.7.

Leaf node 3: The leaf node for which Breathing issue has value N and cough has value Y, shown in table 5.8.

Leaf node 4: The right most leaf node for which Breathing issue and Fever both have value N, shown in table 5.9.

The labels of the four-leaf nodes are decided by the majority of labels in the class labels in the dataset. Therefore, for leaf nodes 1 and 4 the class label is infected and for leaf nodes 2 and 3, the class label is not infected as shown in figure 5.7.

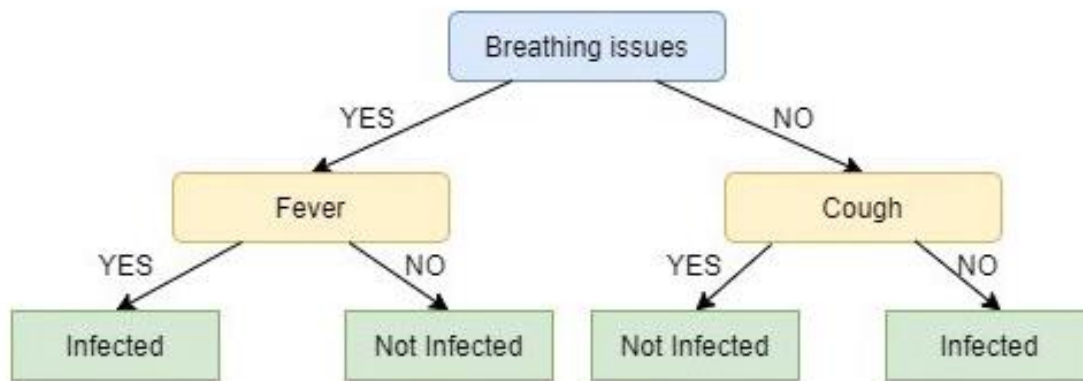


Fig. 5.7 The complete decision tree with leaf nodes labelled by classes.

Table 5.6 Instances for leaf node 1.

Fever	Cough	Breathing Issues	Infected
Y	Y	Y	Y
Y	N	Y	Y
Y	Y	Y	Y
Y	N	Y	Y
Y	N	Y	Y

Table 5.7 Instances for leaf node 2.

Fever	Cough	Breathing Issues	Infected
N	Y	Y	Y
N	Y	Y	N
N	Y	Y	N

Table 5.8 Instances for leaf node 3.

Fever	Cough	Breathing Issues	Infected
Y	Y	N	N
N	Y	N	N
Y	Y	N	Y
N	Y	N	N
N	Y	N	N

Table 5.9 Instances for leaf node 4.

Fever	Cough	Breathing Issues	Infected
N	N	N	Y

In figure 5.7, it may be seen that for leaf node 4 the decision seems to be incorrect in real life, but according to the dataset the decision tree is accurate; it may happen because of the noise present in the dataset. The noise may affect the accuracy of any machine learning model. The disadvantages specific to the ID3 are no-pruning and overfitting.

References

Alpaydin, E. (2020). Introduction to machine learning. MIT press.

Decision tree (2023) *GeeksforGeeks*. GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/decision-tree/> (Accessed: March 27, 2023).

Decision tree algorithm in Machine Learning - Javatpoint (no date) www.javatpoint.com. Available at: <https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm> (Accessed: March 27, 2023).

Introduction.statsmodels. (n.d.). Retrieved February 16, 2023, from <https://www.statsmodels.org/stable/index.html>

Pradhan, M., & Kumar, U. D. (2019). Machine Learning with Python. Wiley.Scikit-learn documentation: <http://scikit-learn.org/>

- Sakkaf, Y. (2020) *Decision trees for classification: Id3 Algorithm explained*, Medium. Towards Data Science. Available at: <https://towardsdatascience.com/decision-trees-for-classification-id3-algorithm-explained-89df76e72df1> (Accessed: March 27, 2023).
- Sarkar, D., Bali, R., & Sharma, T. (2018). Practical machine learning with Python. A Problem-Solvers Guide To Building Real-World Intelligent Systems. Berkely: Apress.
- Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding machine learning: From theory to algorithms. Cambridge university press.

Chapter 6

Artificial Neural Network (ANN)

6.1 Introduction to Artificial Neural Network (ANN)

Human brains understand the context and circumstances of the real world in a manner that computers cannot. An artificial neural network can be used to mimic how the human brain functions so that the computer can learn and make judgments similarly to the human brain.

Machine learning algorithms like ANN are used to solve classification, regression, and clustering issues. It serves as the foundation for deep neural networks. When the dataset is too vast and there are too many characteristics as a result, it is mostly used to learn complex non-linear hypotheses.

A variety of mathematical processing stages are used by ANN. It features several pieces stacked on top of one another. A neuron exists by itself. The input units of the input layer receive a range of inputs from the external domain. Following that, the data is sent to the hidden unit, where it is changed into a format that the output units can use.

Because they are artificial, unlike the neurons in your brain, neural networks are frequently referred to as Artificial Neural Networks (ANN's). They mimic the structure and operation of neural networks in an artificial way. In order to tackle certain issues, ANNs are made up of a significant number of closely interconnected processing components (neurons). (Chauhan, Nagesh Singh, 2019)

6.1.1 How do they act?

Like people and children, ANNs also learn by doing. Through a learning process, an ANN is customized for a particular application, such as data classification or pattern recognition, image recognition, or speech recognition.

A paradigm for information processing that borrows from biological systems is called neural computing. It is made up of many complex interrelated processing elements, or neurons, that collaborate to solve problems. Like humans, ANN understands by imitating other things. Through a learning process, an ANN is customised for a specific task, such as pattern classification or data

classification. Learning causes changes in the synaptic connections between neurons in biological systems. This also applies for ANNs.

The highest-performing artificial intelligence systems over the past 10 years have been created using a technique called "deep learning," such as Google's most recent automatic translator or the speech recognition software on smartphones.

The artificial intelligence technique known as neural networks, which has been used for more than 70 years, is just going by a new name: deep learning. Two researchers from University of Chicago named Warren McCulloch and Walter Pitts, who shifted to MIT in 1952 and went to become the founding members of what is widely considered as the first cognitive science department, proposed neural networks in 1944.

6.2 Biological Motivation

The brain is the most important part of the human body. The signals are inputted into the biological neural network, which then processes and outputs the signals. The fundamental component of the brain is the neuron.

The 200 billion neurons in the brain are made up of four basic components: dendrites, somas, axons, and synapses. When the aggregate of all the signals collected by the neuron's dendrites surpasses a certain threshold, the signal is sent through the axon to the other neurons. The strength of the connections between the neurons was indicated by the synapse.

ANNs are biologically inspired, which means they look at how the brain is structured while taking network configurations and algorithms into consideration. The complexity of the human nervous system, which is made up of neurons, is extraordinary. It has about 1011 million basic neurons. There are approximately ten thousand (104) connections between each of these neurons.

A neural network is made up of a lot of neurons connected to one another. These extensive connections offer a remarkable amount of computational power and memory. Many inputs are delivered to the neuron, and they are all integrated in some way. The neuron would be activated and "fire" if enough active inputs were received all at once; otherwise, the neuron would stay in its inactive, quiet condition. In figure 6.1, a schematic representation of a biological neuron is presented.

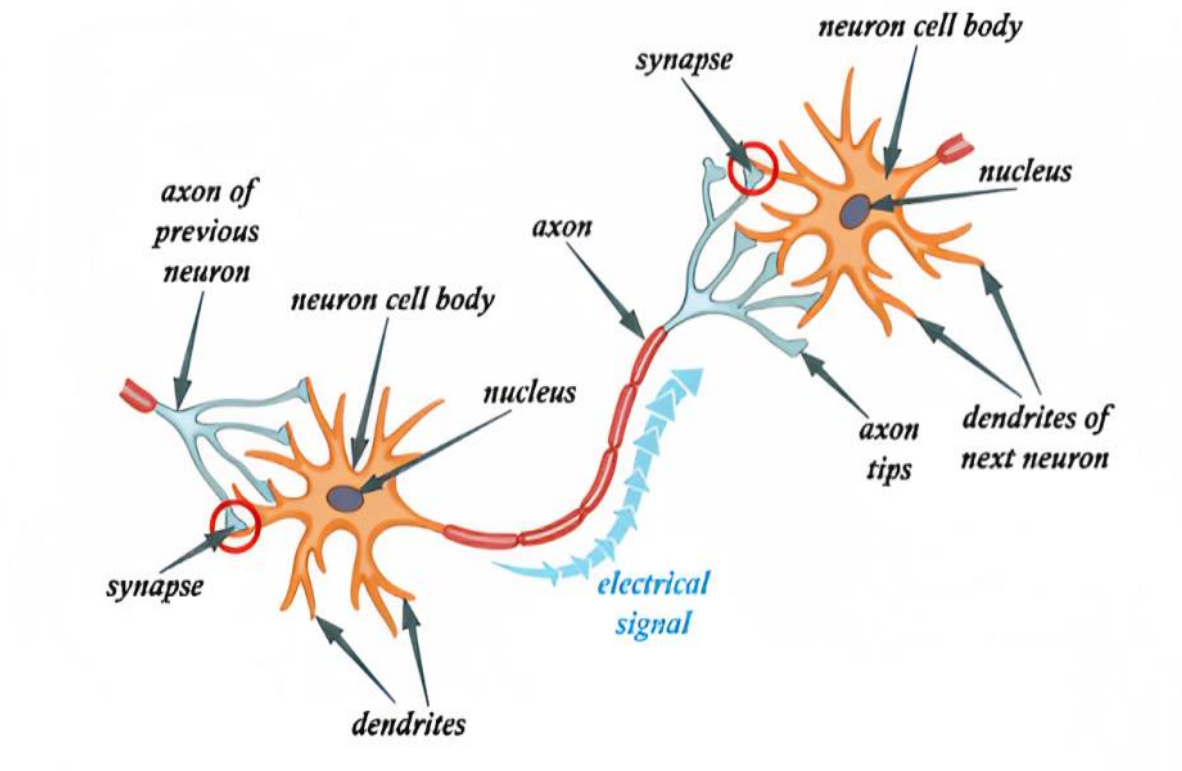


Fig. 6.1 Schematic View of biological neuron (Nilsson, 2023)

According to a theory of systems, the neuron is a multiple-input, single-output (MISO) system, as shown in figure 6.2. Table 6.1 is comparative study of human- biological neuron and artificial neuron.

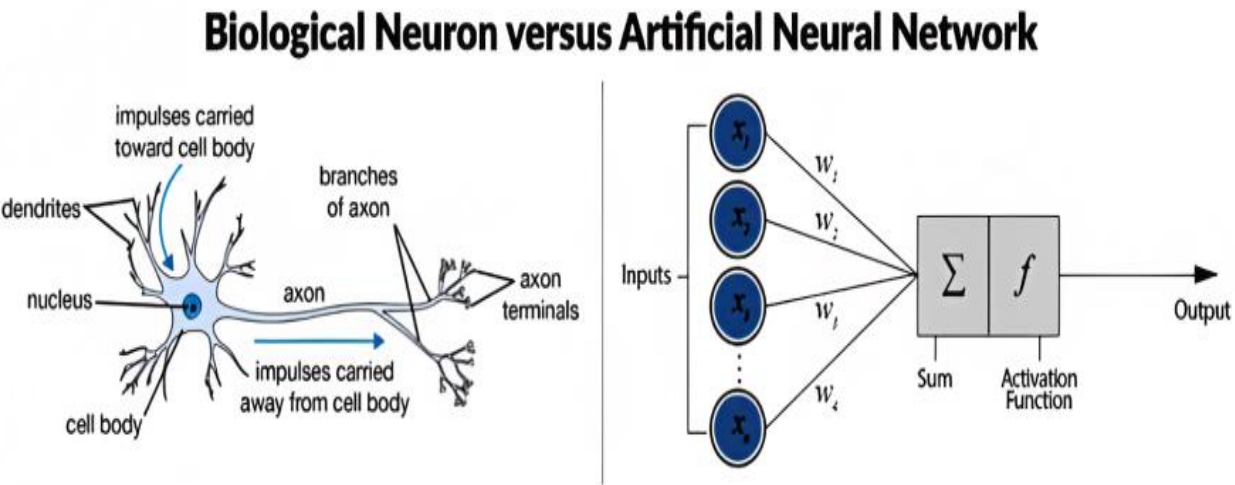


Fig. 6.2 Model representation of biological neuron with multiple inputs (Nilsson, 2023)

Table 6.1 Comparison of Biological Neural Network with Artificial Neural Network

Human	Artificial
Neuron	Processing Element
Dendrites	Combining Function
Cell Body	Transfer Function
Axons	Element Output
Synapses	Weights

The soma is the neuron's body. Dendrites, which are lengthy strands of erratic shape, are attached to the soma. These nerve processes frequently have a diameter of less than a micron and exhibit complex branching patterns. All of the information to the neurons passes through the dendrites, which serve as connectors. Though assuming basic addition is a realistic approximation, these cells are capable of doing more complicated operations on the inputs they acquire than simple addition. An axon is a different class of nerve process that is connected to the soma. This is the neuron's output channel and, unlike the dendrite, is electrically active.

On output cells, axons are always present, whereas interconnections, which include inputs as well as outputs on dendrites, frequently lack axons. The axons are a non-linear threshold device that generates an action potential, a voltage pulse with a duration of roughly 1 millisecond (10⁻³ seconds), when the soma's resting potential increases above a specific critical threshold. The axon joins the dendrite from another cell at a specific junction called a synapse where it terminates. There isn't a direct linkage from across junctions; instead, it's a chemical one that occurs over time.

When the action potential raises the synapse's potential enough, the release of neurotransmitters occurs. Before a synapse is activated, it might require multiple actions for the potential to arrive. The neurotransmitters that the synapse releases diffuse across the gap and chemically open gates on the dendrites that permit the passage of charged ions. This ion movement changes the dendrite potential and causes a voltage pulse to be applied to the dendrite, which then travels into the next neuron's body. Numerous synapses working on each dendrite may enable extremely high levels of interconnectivity.

6.3 Neural Network Representation

Input neurons are the neurons found in the input layer, which is the layer on the left of a neural network. The output layer, which is the topmost layer, is made up of output neurons. One neuron makes up the output layer in the diagram below. The term "hidden layers" also applies to the middle layers. Two layers are concealed in the figure 6.3. Multi-layer perceptrons, or MLPs, are another name for these multi-layered neural networks.

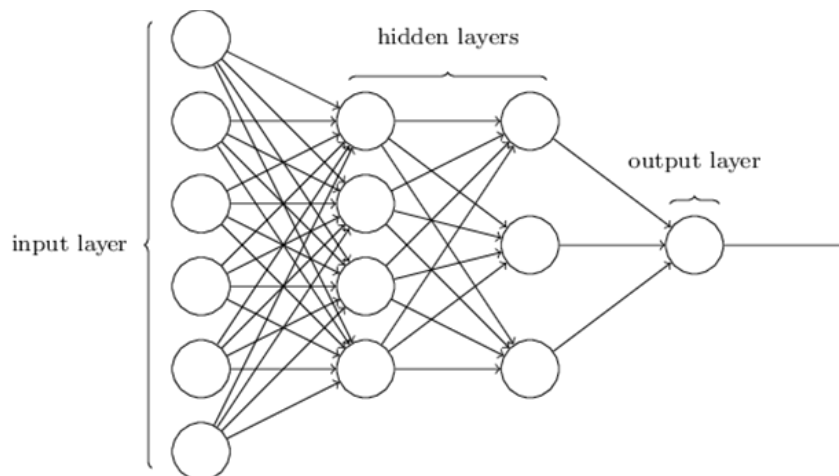


Fig. 6.3 Representation of Artificial Neural Network (ANN) (Nilsson, 2023)

An effective neural network has three key layers:

1. *Input layer*: This layer accepts all inputs from the programmer, as the name would imply.
2. *Hidden Layers*: These are located between the input layer & the output layer. This is the layer where the computations that produce the output are done.
3. *Output Layer*: The output is finally supplied via this layer after the inputs undergo a number of changes via the hidden layer.

6.3.1 Artificial Neuron

The first-order properties of the biological neuron are replicated in the artificial neuron. The artificial neuron receives numerous inputs that correspond to the output of other neurons, much like the biological neuron does. As with synaptic strength, each input is amplified by a matching weight. To determine the neuron input, all of these weighted inputs are then added together and processed through an activation function. The process is given in figure 6.4. There are three fundamental parts of a neuron: weights, thresholds, and a single activation function. (Nilson, 2023).

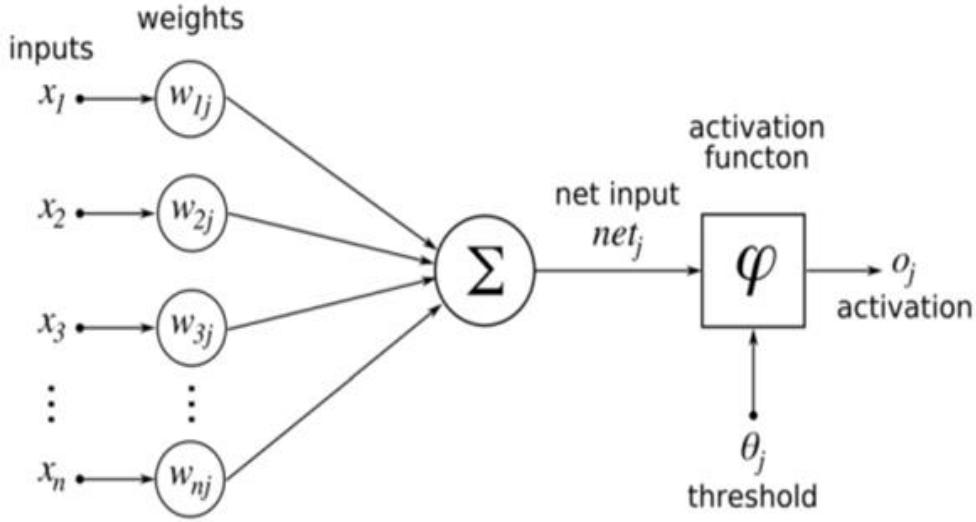


Fig. 6.4 Artificial Neurons (Nilsson, 2023)

Artificial neurons can be modelled mathematically as follows in equation (6.1):

$$u(t) = w_1x_1 + w_2x_2 + w_3x_3 + \dots = \sum_{i=1}^n w_i x_i + \theta = \sum_{i=1}^n w_i x_i \quad (6.1)$$

Assuming $w_0 = \theta$ and $x_0 = 1$,

$$y(t) = f[u(t)] \quad (6.2)$$

In equation (6.2), $f(\dots)$ is an activation function which is used to introduce non-linearity into the neural network. According to equation (6.1) and figure 6.4, $[x_0, x_1, \dots, x_n]$ are inputs, while $[w_0, w_1, \dots, w_n]$ are the respective weights. These inputs and weight can also be represented in vector form as $X = [x_0, x_1, \dots, x_n]^T$, and $W = [w_0, w_1, \dots, w_n]$ respectively.

Equations (6.1) and (6.2) can be represented in vector form as equation (6.3) and equation (6.4) respectively:

$$U = WX \quad (6.3)$$

$$Y = f[U] \quad (6.4)$$

In order to mimic the non-linear behaviour of the conduction current mechanism in biological neurons, the activation function $f(\dots)$ is selected as a non-linear function. Both the weights and the activation function affect how the artificial neuron behaves. In multi-layer static neural networks, sigmoidal functions are frequently utilized as activation functions. Later modules discuss additional categories of activation functions.

6.3.2 Activation Functions

An ANN needs the activation function to learn and comprehend anything really complex. Their primary function is to transform an input signal into an output signal for a node in an ANN. The subsequent layer of the stack receives this output signal as an input. By computing the weighted sum and afterwards adding bias to it, the activation function determines whether or not it should activate a neuron. The goal is to introduce a few non-linearity into the output of a neuron.

The output signal would be a linear function if we did not use an activation function (one-degree polynomial). Despite being simple to solve, linear functions are weaker due to their limited complexity. Our model cannot learn complex patterns without an activation function.

6.3.2.1 Sigmoid Activation Function

A mathematical function known as a "sigmoid function" is employed in models where the output must be a probability prediction because it has a distinctive "S"-shaped curve that varies between 0 and 1 as shown in figure 6.5.

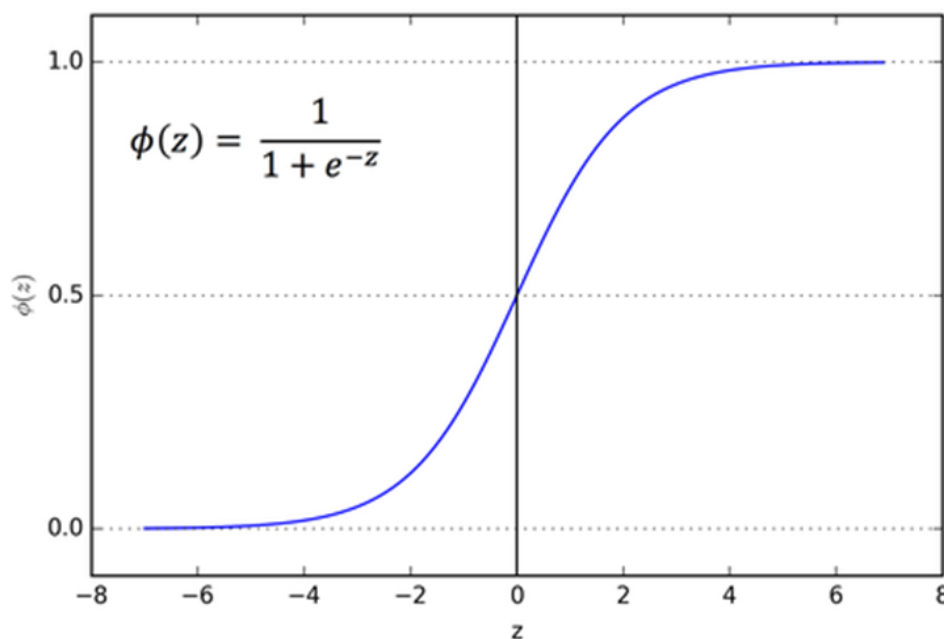


Fig. 6.5 Graph of Sigmoid Function

We can calculate the curve's slope at any two points because the sigmoid function is differentiable. The Sigmoid activation function has the disadvantage is that, if substantial negative input is provided, it may result in the neural network becoming stuck during training.

6.3.2.2 Hyperbolic Tangent Activation Function

It performs better than the Sigmoid yet is equivalent. We can stack layers because it is nonlinear. The task consists of $(-1,1)$ given in figure 6.6.

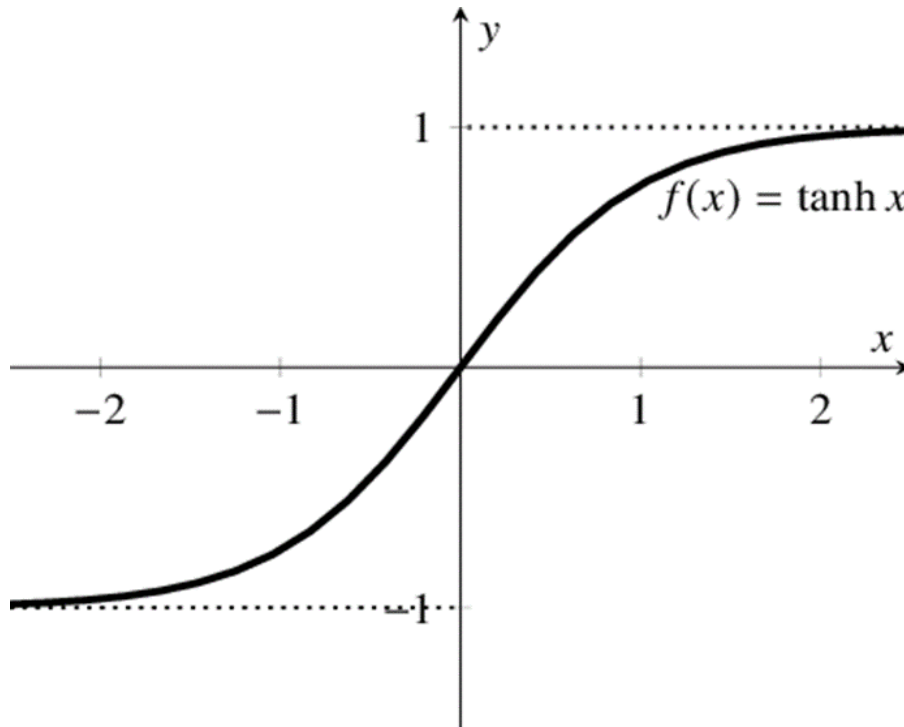


Fig. 6.6 Hyperbolic Tangent Activation Function

The main benefit of this function is that only inputs with values close to zero are mapped to outputs that are close to zero, whereas strongly negative inputs map to outputs that are negative. As a result, training makes it less probable to get trapped.

6.3.2.3 Rectified Linear Units Activation Function (ReLU)

ReLU is by far the most widely used activation function in Convolution Neural Network (CNN) and ANN, and it has a zero to infinity range. $[0, \infty)$ It prints "x" if x is positive, and "0" otherwise. For positive axes and linear functions, it seems to be having the same problem. Both the individual ReLU phenomenon and the combined ReLU phenomenon are non-linear. ReLU may be combined to approximate any function, hence it is actually a good approximator. Figure 6.7 represents the ReLU function and its behaviour.

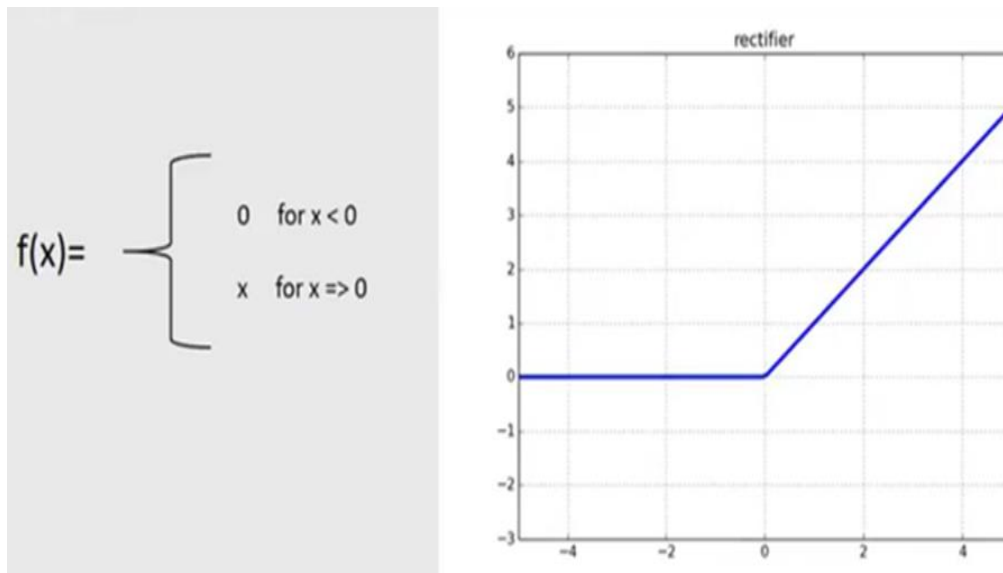


Fig. 6.7 Graph of Rectified Linear Units Activation Function (Concepts to Artificial Neural Networks – Advanced Business Analytics, 2023)

ReLU performs six times better than the hyperbolic tangent function. It would only be applied to the hidden layers of a neural network. Utilize a Softmax function for the output layer of a classification problem and a Linear function for a regression task. One problem with this is that some gradients can fail and stop working during training. It did cause a weight update, that also stops it from working on any subsequent data point. In essence, ReLU may cause neuronal death. Leaky ReLU was created to address the issue of dying neurons. As a result, to maintain the updates coming, Leaky ReLU has a small gradient. Leaky ReLU is found between - and +.

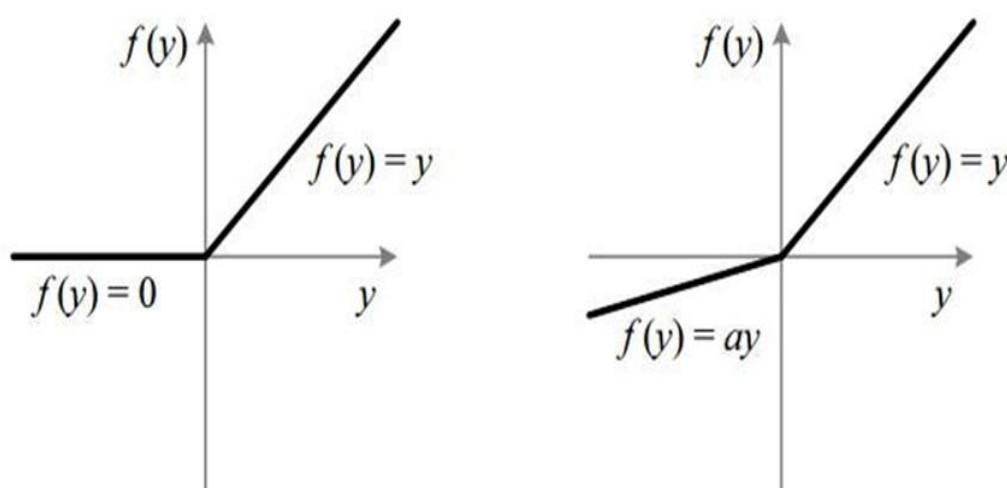


Fig. 6.8 Graph of Leaky ReLU (Concepts to Artificial Neural Networks – Advanced Business Analytics, 2023)

The ReLu function's coverage area is aided by leak. A typical value for is close to 0.01. When it is not 0.01, it is referred to as Randomized ReLu shown in figure 6.8.

6.3.3 How does Neural Networks Work?

Consider the cost of a piece of property. In the first row of data, we have the variables shown in figure 6.9: Age, Location, Bedrooms, and Distance to City.

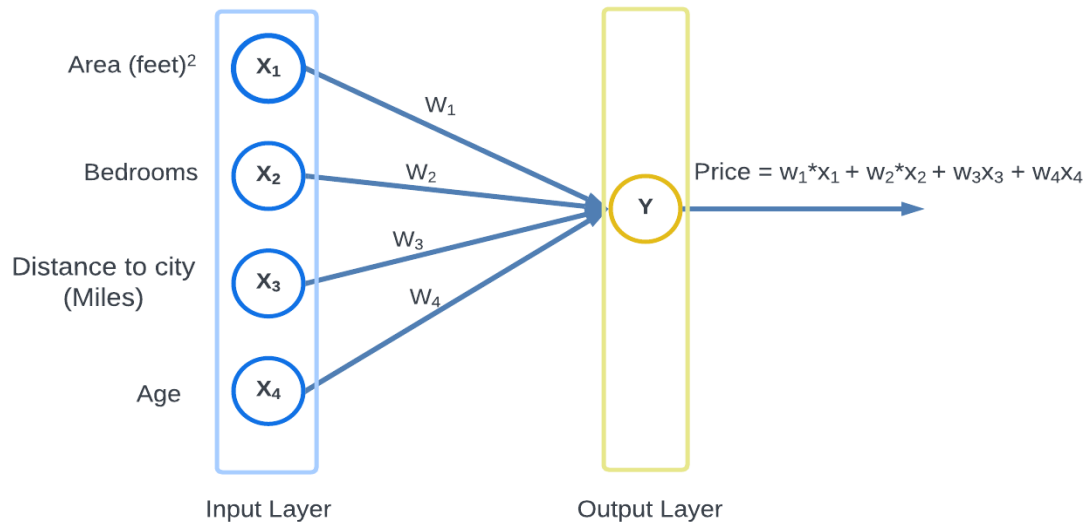


Fig. 6.9 A single neural network with 4 inputs (Concepts to Artificial Neural Networks – Advanced Business Analytics, 2023)

The input values are sent directly to the output layer from the weighted synapses. All four will be examined using an activation function, and conclusions will be drawn. This is fairly simple, but the strength and accuracy of the neural network can be increased by adding a hidden layer between the input and output layers. (Concepts to Artificial Neural Networks – Advanced Business Analytics, 2023)

Now, as shown in figure 6.10, a synapse connects each of the four variables to a neuron. Not every synapse, though, are weighted. They will either be non-0 or have a value of 0. Here, the importance is indicated by the non-0 value. They will be ignored if their value is 0.

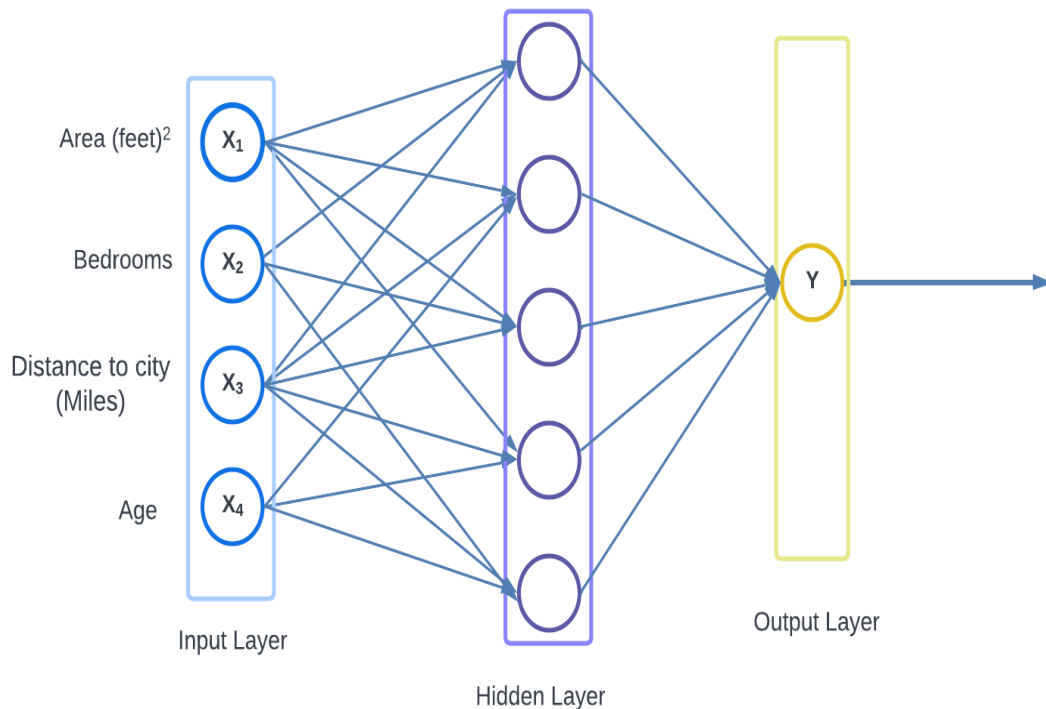


Fig. 6.10 Neural Network with a hidden layer (Concepts to Artificial Neural Networks – Advanced Business Analytics, 2023)

Let's use the first neuron as an example. Distance and area of the city is not zero, indicating that they are weighted and important to the first neuron. The first neuron does not consider the other two variables, Bedrooms and Age, because they are not weighted. You might be curious as to why the first neuron is only considering two of the four variables. In this situation, larger homes tend to cost less on the real estate market the further they are from the city. That is an obvious fact. Thus, it's possible that this neuron is specifically searching for enormous properties close to the city.

Now, this is where neural networks get their power. These neurons are numerous, and they all perform the same calculations using various combinations of these factors. Once this requirement has been satisfied, the neuron activates and does its calculations. The neuron below might contain synapses that are weighted for bedrooms and distance to the city.

In this manner, the neuron's function and interact in a very same flexible manner, enabling it to search for particular items and then conduct an extensive search for whatsoever it is taught for.

6.3.4 How does the Neural Networks Learn?

An analogy might be a way to explain how a neural network works. We learn in our everyday lives and activities when we do something and obtain feedback from a trainer or coach on how to improve. This is very similar to gaining knowledge in a neural network. Similarly, neural networks need a coach to specify what should have occurred in response towards the input. An error value, also known as a cost function, is evaluated and returned by the system according to the variation between the actual and predicted values. (Chauhan, Nagesh Singh)

Cost Function: The square root of the difference between the output value and the actual value.

The cost function analysis is used to modify the threshold and weights for such following input for each network layer. Our goal is to minimise the cost function. The lesser the cost function, the closer the real value is to the expected value. When the network learns to interpret values, the error keeps getting a little bit smaller with each run.

As input, the output is sent to the whole neural network. We only have control over the weighted synapses which connect input variables to neurons. We must update those weights as long as there is a difference between the real and predicted values. After we make a few minor changes and re - run this same neural network, a new Cost function will be generated, ideally one that is less than the previous one. The cost function must be reduced as much as feasible before repeating this approach.

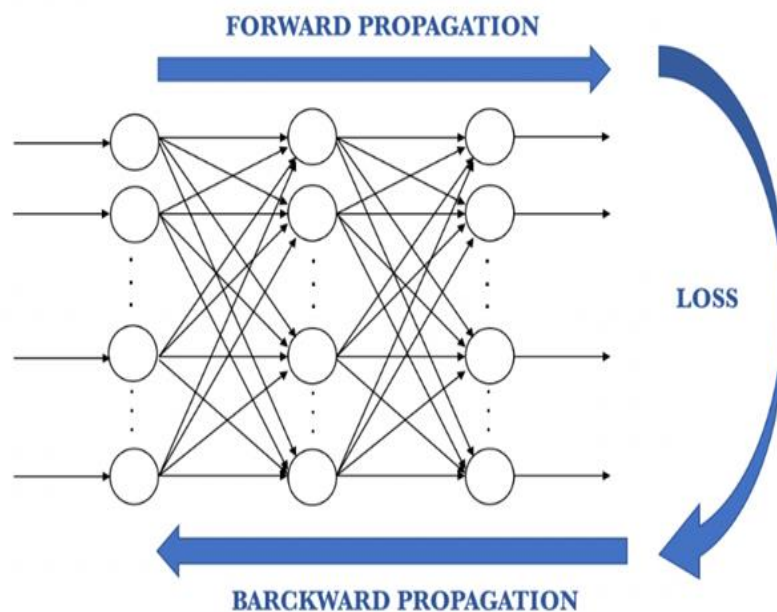


Fig. 6.11 Backpropagation in Neural Network (Chauhan,2019)

Until the value of the error remains to a low level, the previously mentioned technique, also known as the backpropagation (shown in figure 6.11), is constantly applied throughout a network.

6.4 Appropriate Problems for Neural Network Learning

When ANNs are the ideal representation strategy for the task, it is crucial to understand when they are not. A few characteristics of learning activities that make artificial neural networks an accurate representation consist of the following:

1. A real-valued function can be used to describe the idea (target function) that needs to be learned. The output of the function is actual-valued or (if a classification) may be mapped to a collection of real values, so there is a certain translation from the examples used for training to a set of real numbers.

The data that ANNs manipulate are numbers rather than logical statements or other types of data, which is crucial to keep in mind since ANNs are nothing more than enormous mathematical functions. Although it may sound limited, many learning problems can be stated in a way that ANNs can handle them. This is particularly true given that real numbers contain Boolean operators (true and false linked to +1 and -1), integers, and vectors of these kinds of data types.

2. Extended workout sessions are permissible. Compared to decision trees, for instance, neural networks typically require more time for training. The number of training examples, the learning rate value chosen, and the network's design are just a few of the variables that affect how long it takes to train a network. It may take a short amount of time to several hours for training.

3. Humans do not absolutely need to be able to understand how the trained network performs categorizations. ANNs like black boxes from the perspective up, making it challenging for humans to understand what its computations are doing.

4. The evaluation of the target function must be quick when being used for the specific purpose for which it was taught. While training an ANN to determine if a transport is a bus, automobile, or tank, for instance, can take a long time, once trained, utilizing it for categorization tasks is typically very quick. This can be essential: if the network were to be employed in a collision situation, it would be crucial to quickly determine whether the object speeding toward it is a bus, tank, car, or elderly woman.

References:

Chauhan, Nagesh Singh. "Introduction to Artificial Neural Networks." *KDnuggets*, 8 October 2019, <https://www.kdnuggets.com/2019/10/introduction-artificial-neural-networks.html>. Accessed 17 March 2023.

Concepts to Artificial Neural Networks – Advanced Business Analytics. *IBS InteractiveBooks*, <https://ebooks.ibsindia.org/advanced-business-analytics/chapter/session-15-concepts-to-artificial-neural-networks/>. Accessed 17 March 2023.

Nilsson, Nils. "Introduction to ANN." Vardhaman College of Engineering, <https://vardhaman.org/wp-content/uploads/2021/03/Neural-Networks.pdf>. Accessed 14 March 2023.

Chapter 7

Perceptron

7.1 Perceptron

Perceptrons are a particular kind of neural network connection that tracks features by using AI for computations and input data. Simple gates with binary outputs are used by the neural interface to connect to the artificial neurons. The nucleus, dendrites, synapses, and axons of an artificial neuron are analogous to the nucleus, dendrites, synapses, and axons of a real neuron, and it performs the mathematical function.

Frank Rosenblatt first introduced the perceptron in 1957. He proposed a unique MCP-based Perceptron learning method. A supervised learning method for binary classifiers is the perceptron. This technique enables neurons to independently take in and process each component of the training data.

7.1.1 Perceptron Types

Single layer: Single layer perceptron can only learn a single label pattern.

Multilayer: Multilayer perceptrons have the ability to process information from two or more layers and comprehend it. The Perceptron approach determines a linear decision boundary by learning the weights for the input signals.

Supervised learning, a subset of machine learning, is used to build models from labelled training data. It enables the prediction of output for impending or ambiguous input. Next, let's concentrate on the Perceptron Training Rule.

What does machine learning's perceptron model do?

A computer-based approach known as a perceptron is utilised to accomplish supervised learning of various binary sorting jobs. Business intelligence calculations that identify particular input data as Artificial Neurons or Neural Connections depend heavily on perceptrons. The greatest and most

precisely targeted artificial neural networks are categorised using the perceptron model. We can also think of it as a single-layer neural network with four primary parameters: input values, weights and biases, net sum, and also an activation function because it is a supervised learning technique for binary classifiers.

7.1.2 Advantages and Disadvantage of Perceptrons

Advantages:

- Multi-layered perceptron models are capable of resolving complex nonlinear problems.
- For both little and large input data, it functions well.
- After training, help us in making quick predictions.
- Enables the achievement of a comparable accuracy ratio for both large and small data.

Disadvantages:

- The computations of multi-layered perceptron models are difficult and time-consuming.
- It is challenging to predict the extent to which each independent variable is influenced by the dependent variable.
- The effectiveness of the model depends on how well it was trained.

7.1.3 Characteristics of Perceptron Model

The following describe a perceptron model's characteristics:

- 1) It's an algorithm for machine learning that makes use of binary classifiers that are learned under supervision.
- 2) The weighted coefficient is automatically learnt in a perceptron.
- 3) Weights are first compounded by input features before choosing whether to activate a neuron or not.
- 4) To establish if a function is more significant than zero, the activation function employs a step rule.
- 5) In order to distinguish between two linearly separable classes, +1 and -1, linear decision boundaries are drawn.
- 6) They must have output signals if the total of all input values is greater than the threshold value; otherwise, no outputs will be displayed.

There are four components in a perceptron.

- 1) Input layer or input values
- 2) Biases and Weights
- 3) Total
- 4) The Activation Function

7.2 Representation of Perceptrons Power

In the n -dimensional field of instances, the perceptron can be seen as a hyperplane decision surface (i.e., points). According to figure 7.1, the perceptron produces an output of a for examples that are on 1 side of the hyperplane and an output of a-1 for instance that are on the opposite side. This decision hyperplane's equation is 0. Of course, there are some pairs of both positive & negative examples that no hyperplane can divide. Linearly separable collections of instances are those that can be divided.

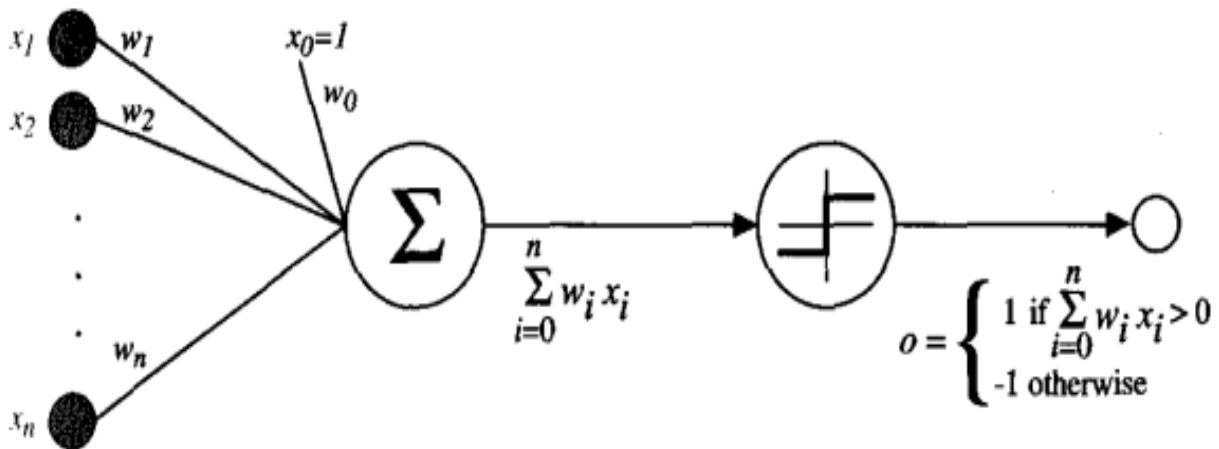


Fig.7.1 A Perceptron (Mitchell, T.M. (1997))

Several Boolean functions can be represented by a single perceptron. For instance, setting the weights $w_0 = -0.8$ and $w_1 = w_2 = 5$ would enable a two-input-perceptron network to implement the AND function if we assumed boolean values that were 1 (True) & -1 (False). By changing the threshold to $w_0 = -3$, this perceptron can be used to symbolize the OR function instead. In fact, the m - of - n functions, which require that at least one of the n input data to the perceptron be true, can be considered as special examples of AND and OR. $m = 1$ corresponds to the OR function, $m = n$ to

the AND function. By adjusting the threshold w_0 and all inputs weight to the same value (for example, 0.5), any $m - of - n$ function can be simply expressed using a perceptron. (Mitchell, T.M.,1997). All of the basic Boolean functions AND, OR, can be represented using perceptrons. NOR, NAND (AND), and (OR). Sadly, some Boolean operations, like the XOR function, whose result is 1 if and only if $x_1 \neq x_2$, cannot be expressed by a single perceptron.

The figure 7.2(a) represents the linearly sepearable data and take note that this XOR function corresponds to the collection of linearly non-separable training instances displayed in figure 7.2(b). Every Boolean function may be represented by some connected network of units based on such primitives, hence the abilities of perceptrons to represents AND, OR, NAND, and NOR is important. In fact, a network of perceptrons just two levels deep can represent every Boolean function.

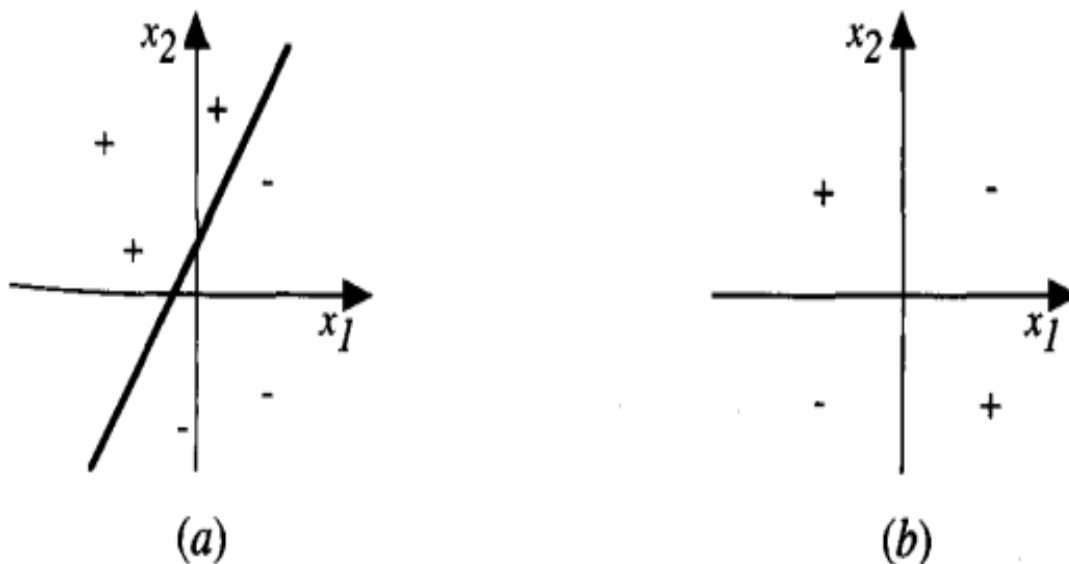


Fig. 7.2 Separation of datapoints using a line (Mitchell, T.M,1997)

A two-input perceptron serves as the decision surface's representation. (a) Examples for training the perceptron, along with its decision surface for accurate classification. (b) A group of practice cases that cannot be separated sequentially (i.e., that cannot be correctly classified by any straight line). The inputs to the perceptron are x_1 and x_2 . The inputs are given to multiple parts, and the outcomes of those devices are then fed to a second, concluding stage. Positive examples are denoted by "+," while negative examples are denoted by "-".

The disjunction (OR) of a group of conjunctions (ANDs) containing the inputs and their negations, or the representation of the Boolean function in disjunctive normal form, is one method. You should

be aware that you can negate any inputs to an AND perceptron by simply changing the sign of the associated input weight. We will typically be concerned in learning networks with multiple layers of threshold units because they can express an extensive range of functions in a way that single units alone cannot.

7.3 Perceptron Training Rule

The disjunction (OR) of a group of conjunctions (ANDs) containing the inputs along with their negations, or the representation of the Boolean function in its disjunctive normal form, is one method.

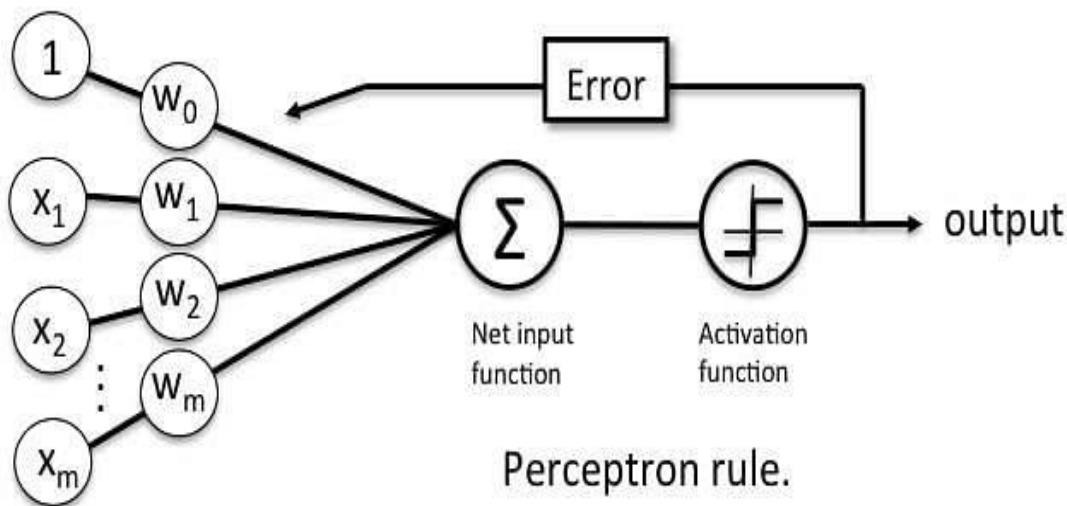


Fig. 7.3 Perceptron rule (Banoula, M. (2023))

You should be aware that you can negate any inputs for an AND perceptron by just altering the symbol of the associated input weight. For example, in figure (7.3), the input x_2 can be negated by adjusting weight w_2 . We will typically be concerned in learning networks with multiple layers of threshold units because they can express an extensive range of functions in a way that single units alone cannot.

7.3.1 Perceptron Function

As mentioned earlier, the perceptron function is what produces the output value, " $f(x)$ ", as it maps the input value " x " which becomes multiplied by the weighting coefficient learned from its learning process shown in figure 7.3 (Banoula, M., 2023). The definition of $f(x)$ is given in equation (7.1).

$$f(x) = \begin{cases} 1, & \text{if } w \cdot x + b > 0 \\ 0, & \text{otherwise} \end{cases} \quad (7.1)$$

The summation weighted input, defined as net input is $\sum_{i=1}^m w_i x_i$

Where, $w = \{w_1, w_2, \dots, w_m\}$ is a vector of actual weights for inputs $x = \{x_1, x_2, \dots, x_m\}$ for m number of inputs with bias b to adjust the boundary without taking input into account.

Either "1" or "0" can be used to symbolize the output. Depending on the triggering function used, it may also be expressed as "1" or "-1". Let us study the inputs that are part of perceptrons in the upcoming section.

7.3.2 Input of Perceptron

After regulating the inputs with predetermined weight values, the transformation function is applied to produce the final output of a Perceptron. This is an illustration of a perceptron that produces a Boolean outcome.

Based on inputs like salary, marital status, age, prior credit history, etc., a Boolean result is produced. It only has two possible values: True and False, or Yes and No. The summation function adds all of the variables of x after multiplying them by weights w such as $w_0 + w_1 x_1 + w_2 x_2 + \dots + w_n x_n$, that defines the net input function.

Let's talk about how perceptrons work as activators in the section after this.

7.3.3 Perceptron Activation Functions

The activation function converts the numerical result into +1 or -1, using a step rule to determine if the output of the function that weights is higher than zero or not.

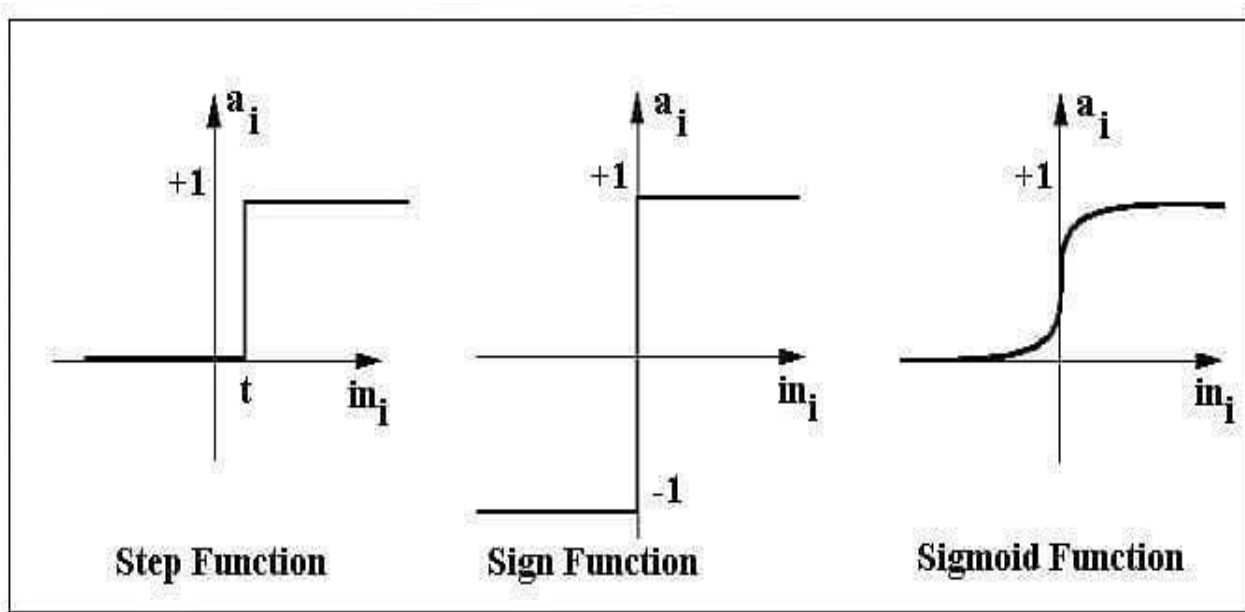


Fig. 7.4 Activation Function (Banoula, M. (2023))

As an illustration,

if $w_i x_i > 0$ and the ultimate output "0" is 1, the bank loan will be issued.

If not, the ultimate output "0" will be -1. (Refuse a Bank Loan)

In figure 7.4, the step function is engaged when a certain neuron output number is reached; otherwise, it outputs zero. The sign of the Function results in +1 or -1 depending on whether the neuron output is greater than zero or not. The S-curve, also known as the sigmoid, yields a number between 0 and 1.

7.3.4 Output of Perceptron

Perceptron with a Boolean output:

Inputs: $x_1 \dots x_n$ and Output: $o(x_1 \dots x_n)$

The input vector is defined using equation (7.2).

$$(x_1, \dots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + w_2 x_2 + \dots + w_n x_n > 0 \\ -1 & \text{otherwise} \end{cases} \quad (7.2)$$

Weights: $w_i \Rightarrow$ Contribution of input x_i to the output of the perceptron

$w_0 \Rightarrow$ bias or threshold value

A +1 output indicates that the neuron has been activated. A value of -1 indicates that the neuron was not activated.

The sign function in equation (7.3), abbreviated as sgn , has an output of +1 or -1.

$$o(\vec{x}) = sgn(\vec{w} \cdot \vec{x})$$

$$sgn(y) = \begin{cases} 1 & \text{if } y > 0 \\ -1 & \text{otherwise} \end{cases} \quad (7.3)$$

7.3.5 Error in Perceptron

The expected output and the known output are contrasted in the Perceptron Learning Rule. The error is communicated backward to enable weight correction if it does not match.

In the following section, let's discuss about the perceptron's capacity for decision-making in the part that follows.

7.3.6 Perceptron: Decision Function

The decision function $\phi(z)$ of a perceptron must be a linear combination of the x and w vectors given in equation(7.4).

$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \quad (7.4)$$

The decision function's value z is determined by equation (7.5) :

$$Z = w_1x_1 + \dots + w_mx_m \quad (7.5)$$

If z is greater than a threshold, the decision function is +1; if not, it is -1 as given in equation (7.6).

$$\phi(z) = \begin{cases} 1 & \text{if } z \geq \theta \\ -1 & \text{otherwise} \end{cases} \quad (7.6)$$

All equation together generates the Perceptron algorithm, where θ is the expression of bias

7.3.7 Bias Unit

In order to keep things simple, the equation (7.5) can be re-written as equation (7.7) with threshold as w_0x_0 , initially set to 0.

$$Z = w_0x_0 + w_1x_1 + \dots + w_mx_m = w^T x \quad (7.7)$$

Now, the choice-making procedure is $\phi(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{otherwise} \end{cases}$

7.3.8 Output

The decision function can be used to distinguish between two classes that are linearly separable since it squashes $(w^T)x$ to either +1 or -1, as seen in the figure 7.5.

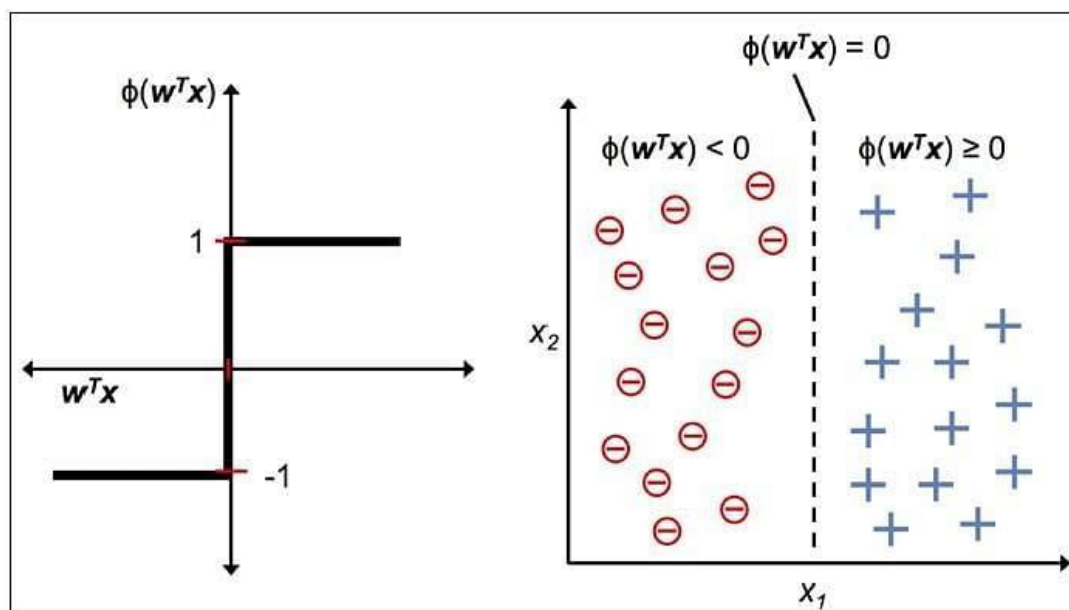


Fig. 7.5 Decision Function (Banoula, M. (2023))

7.4 Gradient Descent and Delta Rule

7.4.1 Introduction and Basic

The aim of optimization is to select the parameters that best optimise (minimise or maximise) some form of objective function, which is a function of a collection of parameters. Assumedly, we aim to reduce the goal function. Training a neural network is a form of optimization; we are trying to find the weights of the network that minimise the loss on the training data.

Gradient descent (Salian, 2022) is the most fundamental optimization strategy. The concept is that we start off with a set of criteria. The objective function's gradient with respect to the parameters is then calculated. In order to have the most impact on the objective function, we can adjust the parameters in this way. The parameters are then updated by shifting them in the opposite direction of the gradient. This will be the minor, local parameter change that will have the biggest impact on lowering the objective function. This process is continued until the objective function is no longer noticeably reduced as shown in figure 7.6.

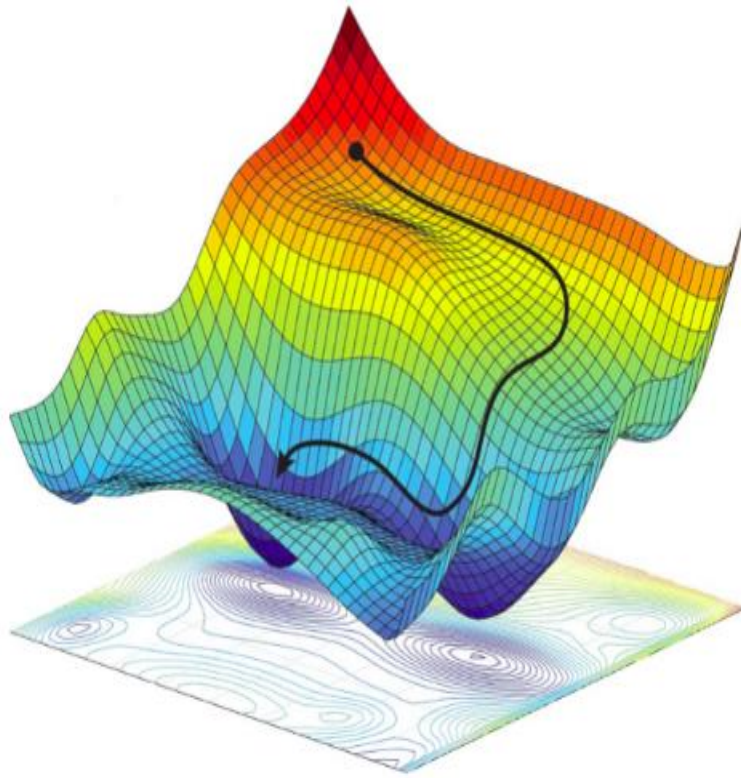


Fig. 7.6 The process of Gradient Descent (Salian, 2022)

7.4.2 The Gradient Descent Procedure

Let $f(x)$ be a function that yields a real number that is the cost, or the value of the objective function, and let x signify the parameters. This algorithm can be summed up as follows.

1. *Initialize x_c .*
2. *Set $x_n \leftarrow x_c - n \nabla f_x(x_c)$*
3. *If $||f(x_n) - f(x_c)|| < \epsilon$ stop.*
4. *Else $x_c \leftarrow x_n$ and return to step 2.*

We must select some initial values for the parameters in step 1. These are what we refer to as x_c , or current parameters. You may accomplish this by simply selecting some random values. This is how a neural network is initialized via `matconvnet`. Most of the time, we have to be a little careful to pick these random values in a suitable range, so we don't start somewhere absurd. Sometimes we can set the beginning values so that they are probably close to a good solution by using some prior knowledge. Because we don't have to go as far to find the best answer, the algorithm can run more quickly and accurately because we are less likely to converge to an undesirable local minimum. Later, more on that.

The parameters are updated in step 2 to move in the direction of the negative gradient. We refer to these by the new parameters' x_n . The cost function increases most quickly along the gradient, hence the direction where this is negative is the one where it declines most quickly. The algorithm's parameter instructs us on how big of a step to take. The learning rate is what we refer to as. This is frequently picked on the fly. There has been a lot of research on how to effectively determine good learning rates and even change the learning rate as the algorithm advances.

Then, in step 3, we determine whether this most recent action is actually bringing about a noticeable improvement in the cost function. We stop when we cease improving, or when we reach a certain point. Keep in mind that the gradient will be zero after we have determined the values that minimize the cost function, at which point we would stop advancing.

7.4.3 Convexity and Local Minima

The cost function can be reduced via gradient descent, as we saw before, and it can converge once it reaches a point where the gradient of the cost function is zero. When will this lead to the best solution to the optimization function is a crucial question. In essence, if the cost function is convex, there will only be one local minimum, and whenever the gradient is zero, we will be at the cost function's global minimum.

Consider the 3D space represented by $(x, y, f(x, y))$ as a way to illustrate this convexity (the same ideas will hold for higher dimensions) as shown in figure 7.7. The volume of space containing all of the points (x, y, z) such that $z \geq f(x, y)$ can be considered (x, y) . Its volume so resembles a 3D parabola in the aforementioned case. We can infer that the problem is convex if this volume is convex.

Why does convexity matter? We'll demonstrate that a convex issue will only have one local minima (by show we mean some very hand-wavy, intuitive explanation). Consider the following two points: (x_1, y_1) and (x_2, y_2) . A line segment between them is defined as all points of the form in equation (7.8).

$$(x_t, y_t) = t(x_1, y_1) + (1 - t)(x_2, y_2), 0 \leq t \leq 1 \quad (7.8)$$

There is a lot of talk about how to do this and we are here to help! With convexity in equation (7.9)

$$f(x_t, y_t) = tf(x_1, y_1) + (1 - t)f(x_2, y_2) \quad (7.9)$$

Because of this, if $f(x_1, y_1) > f(x_2, y_2)$, the cost must monotonically decrease from (x_1, y_1) to (x_2, y_2) .

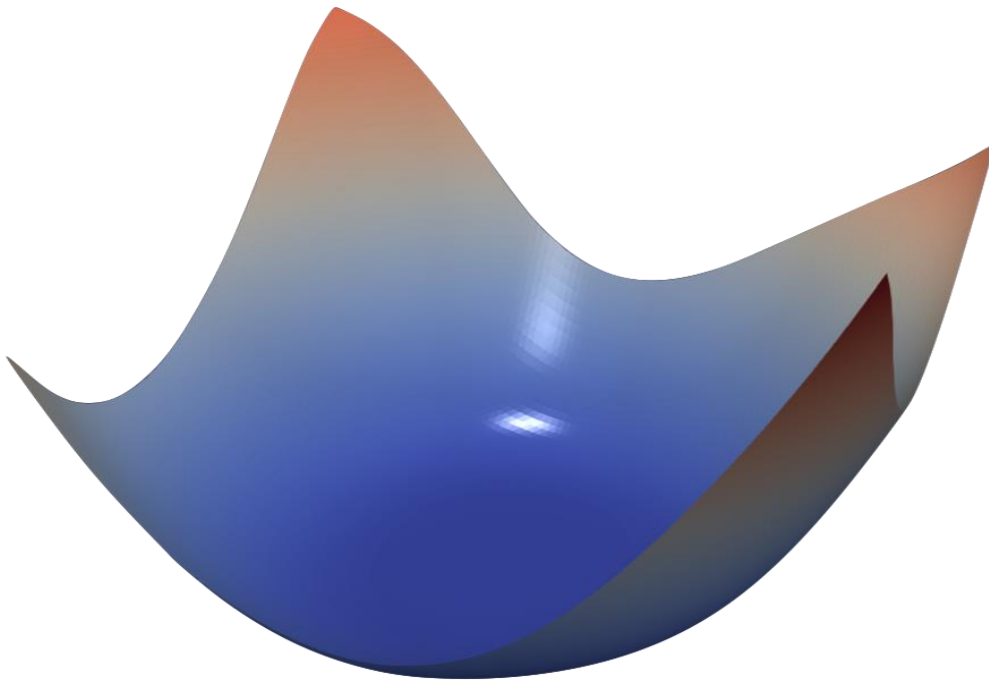


Fig. 7.7 3D Parabola (Visualizing the Loss Landscape of Neural Nets, 2023)

Say, however, that there are two local minima. The gradient at a point must be zero, and every other point in a close vicinity must have a higher cost for that point to be the local minimum of the cost function. Hence, suppose that both the local minima (x_1, y_1) and (x_2, y_2) exist. The price must not decrease anywhere in the vicinity of (x_1, y_1) . This runs counter to the previous sentence. If the second

point costs more or if both points cost the same, the same logic holds true. This indicates that we have discovered the problem's globally optimal solution if gradient descent converges, and the cost function is convex. (Visualizing the Loss Landscape of Neural Nets, 2023)

Many optimization issues, though, are not convex. Learning a perceptron can be defined as a convex optimization problem, but that's another story. In particular, learning the weights of a neural network is a non-convex optimization problem. A nonconvex problem may contain numerous local minima. And because they are all fixed points, depending on where we begin gradient descent, we might end up in any of those local minima. Furthermore, there is no assurance as to what the cost function's value will be at various local minima. One of them might have very low prices, while others might have quite high ones. As a result, we can arrive at a solution that is much less desirable than the ideal one.

7.4.4 Stochastic Gradient Descent

If we don't initialise gradient descent sufficiently close to the global minimum when there are several local minima, it won't be able to identify the global minimum. As the space is extremely high-dimensional and impossible to map out numerically and is difficult to evaluate, we are unsure of exactly how the energy landscape for deep networks appears. The loss landscape can be as intricate as figure 7.8 illustrates. Also, because it takes time to train a neural network, we often do not even let it run to convergence. As a result, we never reach even a local minimum; instead, we only expect to be close to one when we decide to stop. Nonetheless, there are a number of methods for avoiding undesirable local minima in optimization.

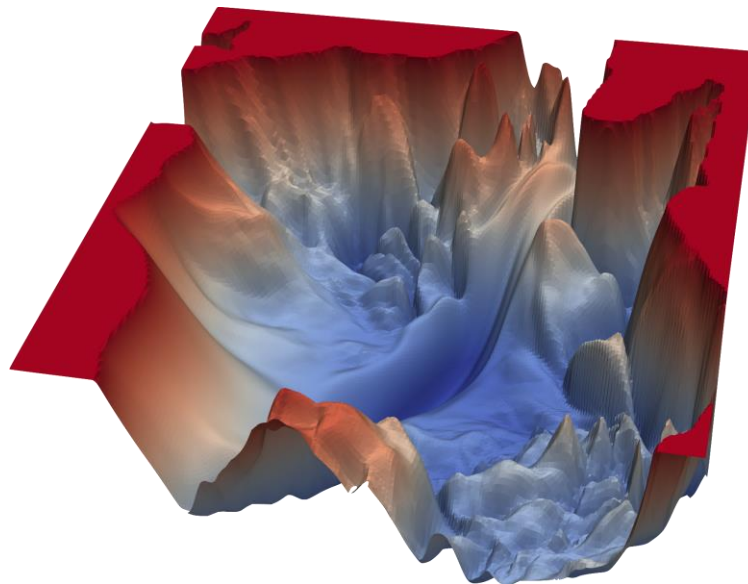


Fig. 7.8 A Complex loss landscape of a neural network (Visualizing the Loss Landscape of Neural Nets, 2023)

The first method involves using numerous random starting points for gradient descent and choosing the solution with the lowest energy. This way, we can select the global minimum if even one of these arbitrary beginning locations does. Given how long it takes to train a neural network just once, this might not be very useful. But, occasionally individuals will do this, or if they discover a number of solutions with comparable energies, they integrate the outcomes of all of them into an ensemble. The idea of assembling a variety of networks is related heuristically to several other strategies, such as dropout.

Using stochastic gradient descent is a second strategy. Here, it is intended to substitute a noisy gradient estimate—a random gradient whose anticipated value is the actual gradient—instead of using the precise gradient. If we do this, eventually we are, on average, moving in the direction of the gradient. Yet, since the gradient is noisy, we can travel in directions that aren't parallel to it. This can have the effect of removing us from a nearby local minimum and keeping us from becoming trapped in a small local minimum.

7.4.5 Delta Rule

Learning is governed by a method or a mathematical logic. The artificial neural network's functionality is enhanced by applying this rule over the entire network. A network's weights and bias levels are modified using learning rules as it simulates in a particular data environment.

One of the learning principles that is most frequently used is the delta rule. It is reliant on supervised learning. This rule indicates that when the error and input are multiplied, the synaptic weight of a node changes. Mathematically, the delta rule is stated in equation (7.10).

$$\Delta W = n(t - y)x \quad (7.10)$$

Compare the output vector to the input vector to determine the proper response. If the difference is zero, no learning occurs; if not, it reduces the difference by adjusting its weights using equation (7.11) for changing weight from u_i to u_j .

$$dw_{ij} = r * a_i * e_j \quad (7.11)$$

Where, a_i is the activation of u_i , r denotes the learning rate, and e_j denotes the difference between the expected output and the actual output of u_j . If the collection of input patterns is an independent set, use the delta rule to learn any links.

For networks with linear activation functions and no hidden units, it has been noted. In n -space, the error squared vs. weight graph exhibits a paraboloid shape. Because of the negative proportionality constant, this function has the lowest value and a concave upward graph. The vertex of this paraboloid represents the location where error is reduced. The weight vector that corresponds to this position is the ideal weight vector.

The delta learning rule can be used for both a single output unit and several output units. Suppose that while applying the delta rule, the error can be accurately determined. The delta rule's implementation aims to reduce the discrepancy between the output that was expected and what actually occurred.

References:

- Banoula, M. (2023) *What is Perceptron? A beginner's guide [updated]: Simplilearn, Simplilearn.com*. Simplilearn. Available at: [https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron#:~:text=A%20Perceptron%20is%20a%20neural,value%20%E2%80%9Df\(x\).](https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron#:~:text=A%20Perceptron%20is%20a%20neural,value%20%E2%80%9Df(x).) (Accessed: March 19, 2023).

Mitchell, T.M. (1997) *Machine learning*. New York: McGraw Hill.

Salian, C. “A Gentle Introduction to Multi-Objective Optimisation.” *Codemonk*, 23 February 2022, <https://codemonk.in/blog/a-gentle-introduction-to-multi-objective-optimization/>. Accessed 18 March 2023.

“Visualizing the Loss Landscape of Neural Nets.” *UMD Department of Computer Science* /, <https://www.cs.umd.edu/~tomg/projects/landscapes/>. Accessed 18 March 2023.

Chapter 8

Multi-layer Neural Network

8.1 Multilayer Neural Network

In the previous chapter 6 and 7, we studied artificial neural network and perceptron, we understand what neuron or nodes is. In this section, we will learn the different facts of the multilayer neural network, which stands for more than one layer of artificial neural, neurons or nodes over one another. Its designs are very different from ANN. Nowadays, the majority of networks use multi-layered neural networks instead of single-layer neural networks.

As we study perceptron in chapter 7, Only linear connections between input & output data can be recognized by perceptrons, a type of neural network with just one neuron. But in the Multilayer Perceptron, this neural network is able to consist of different layers of neurons & is prepared to study ever-more complicated pattern.

The following points highlight the features of the multilayer neural perceptrons:

- The model of the network's neurons includes a differentiable nonlinear activation function.
- Single or many of the network layers are hidden from the input and output nodes.
- The synaptic weights of the network indicate how highly connected the network is.

8.1.1 Advantages and Disadvantages of Multilayer Neural Network

The advantages of Multilayer Neural Networks are:

- It can be utilized to solve complex nonlinear problems.
- It effectively manages vast amounts of input data.
- Makes quick predictions after training the model.
- The following training makes fast predictions is simple.
- Even with smaller samples, the same accuracy ratio can be attained.

The disadvantage of Multilayer Neural Networks is:

- The entire numbers of parameters (the numbers of perceptrons in layers one multiplied by the entire number of p in layer two, multiplied by the total number of p in layer three, etc.) might

increase to a very high number. With the huge number of redundant dimensions, this is inefficient.

- The spatial information is disregarded. Flattened vectors are used as its inputs.

A multilayer perceptron architecture network's input layer, 2 hidden layers, and output layer are displayed in figure 8.1. Multilayer perceptrons were feed-forward neural networks having two hidden layers, input, and output. The signals for the input layer's analysis comes from the input layer and output layer take care of categorization and prediction duties. Between the input and output layers, infinite numbers of hidden layers make up Multilayer Perceptrons' precise computing engine. A multilayer perceptron transfers data in a similar manner from input layers to the output layers. The Multilayer Perceptrons' neurons are taught using the backpropagation learning algorithm. Because Multilayer Perceptron is designed to calculate approximate each and every continuous function, they can deal with problems that cannot be divided linearly.

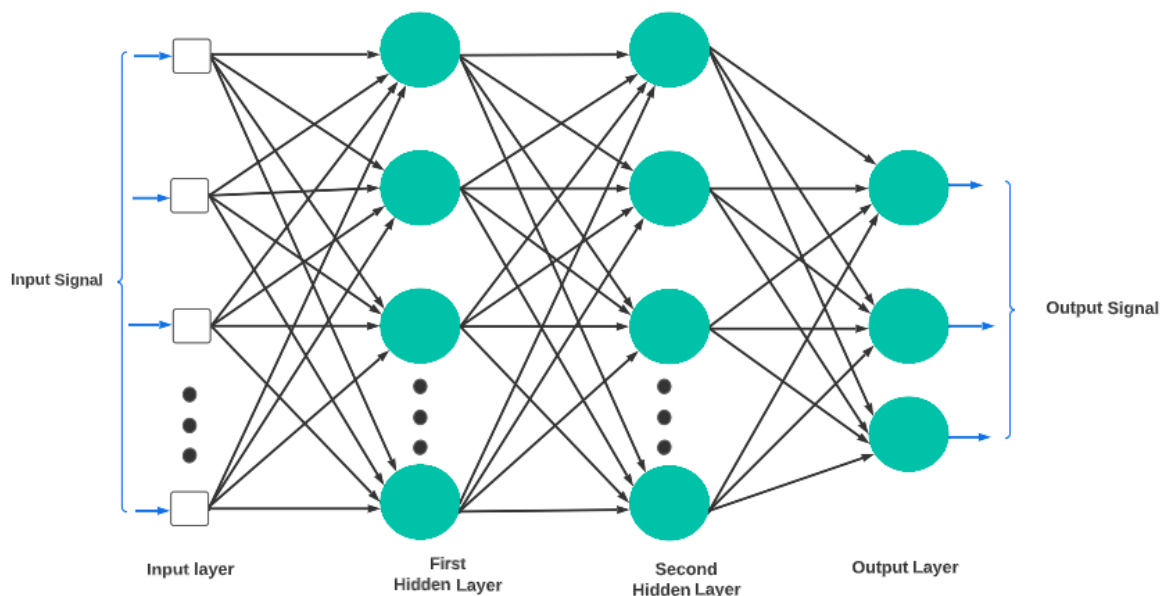


Fig. 8.1 Architectural Graph of the Multilayer Neural Network with two hidden layers

A unit of the layered perceptron is shown in figure 8.2. In this network, there are two different types of signals:

1. *Function Signals*

An input (stimulus) enters the network at its input end, moves ahead through the network neuron by neuron, and then emerges as such an output signal at the network's output end. For two reasons, we call this type of communication a "function signal". It is first assumed

that it serves a purpose at the network's output. Second, each neuron within network it passes through has its input and associated weights used to calculate the signal.

2. Error Signals

A backward-moving error signal starts its journey through all the networks at an output neuron (layer by layer). Because each cell within network must carry out some sort of error-dependent function in order to compute it, we refer it as an "error signal".

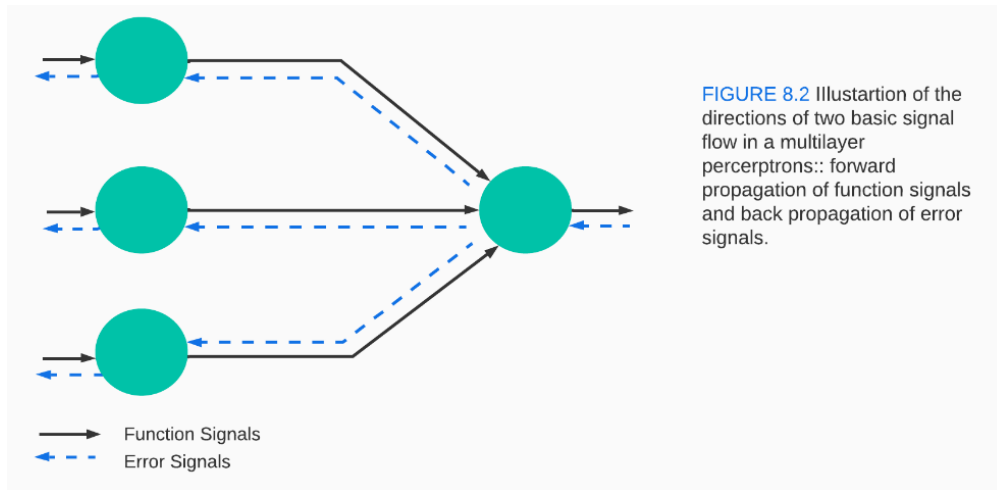


Fig. 8.2 Illustration of the direction of two basic signal

8.2 A Differentiable Threshold Unit

Multilayer Perceptrons can handle situations that cannot be partitioned linearly since they are built to approximate every continuous function. We require an apparatus whose output is a nonlinear & differentiable function of its inputs. The sigmoid unit, which is similar to a perceptron but is based on a smoothed, differentiable threshold function, is one option.

Although the sigmoid unit shares many similarities with a perceptron, it is based on a smoothed, discrete threshold function. After computing a linear mixture of its inputs and the sigmoid unit thresholds the outcome as shown in figure 8.3. However, the threshold output of the sigmoid unit is the constant function of its intake.

The sigmoid unit determines their output o given in equation (8.1)

$$o = \sigma(\vec{w} \cdot \vec{x}) \quad (8.1)$$

Where $\sigma(y)$ is defined as equation (8.2),

$$\sigma(y) = \frac{1}{1+e^{-y}} \quad (8.2)$$

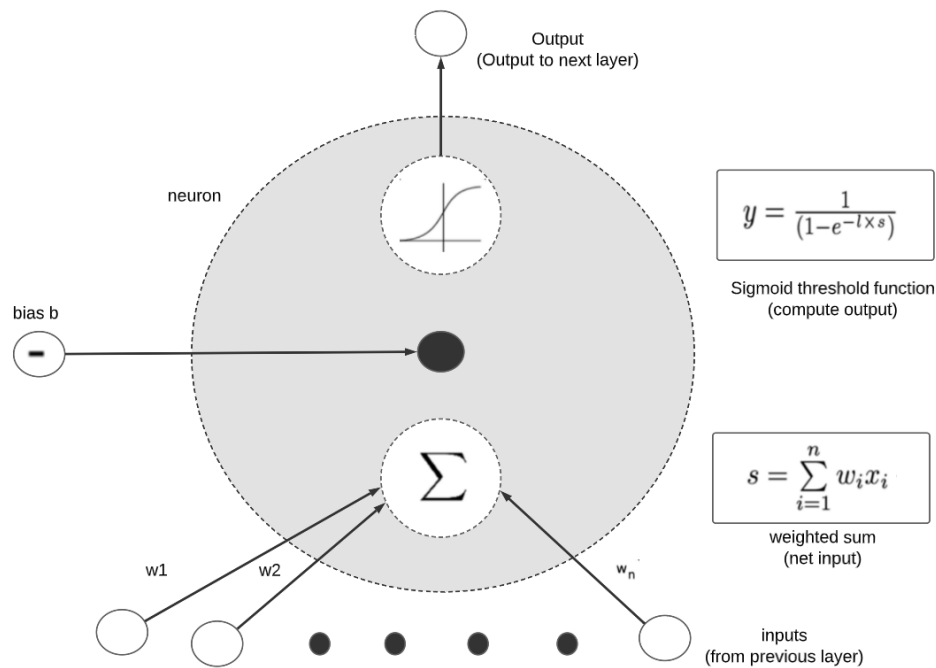


Fig. 8.3 Sigmoid Threshold Unit

8.3 Convergence and Local Minima

8.3.1 Convergence

The back-propagation technique has a stochastic aspect, which means it tends to zigzag about the true route to reduce errors to a minimum. The statistical technique known as a stochastic approximation, which Robbins and Monro first developed in 1951, is used in back-propagation learning. As a result, it tends to converge slowly. According to Jacobs (1988), there are two main reasons for this property.

- 1) As the error functions are comparatively flat across the weight dimension, the derivatives of the errors function for that weight are minimal. As a result of the minor adjustment made to the weight in this case, it may take many rounds of the method to significantly improve the network's error performance. In contrast, if somehow the error function is strongly curved along the weight dimension, the derivatives of the error function for that weight will be very big. The algorithm might overshoot the lowest value of the error surface in the second instance because the weight's modification was large.

- 2) The algorithm could move incorrectly as a result of the weight changes since the negatives gradient vectors (i.e., the negatives derivatives of the cost function in relation with vector of weights) might lead left after the error surface minima.

8.3.2 Local Minima

Another characteristic of the error surface that affects the effectiveness of the backpropagation method is the occurrence of local minima in along with global minima; normally, it is difficult to calculate the number of both global and local minima.

Being a hill-climbing method, back-propagation learning runs the risk of being stuck in a local minimum in which any minor change of synaptic weights raises the cost function. Although the network is locked at the local minimum, there is an additional set of synaptic weights inside the weight space for which the cost function is smaller.

8.4 Backpropagation Algorithm

The backpropagation algorithm, often referred to as backward propagation of mistakes, was developed to detect faults as data is returned from input nodes to output nodes. It's a crucial mathematical tool for data mining as well as artificial learning to improve prediction accuracy. Backpropagation is a technique for quickly calculating derivatives. (Zola, A. and Vaughan, J., 2022) By using learning algorithm backpropagation, artificial neural networks calculate a gradient descent w.r.t the weight for the various inputs. By compared intended outputs to actual system outputs, the system is tuned by adjusting connection weights in an effort to minimise the disparity between the two. (What is backpropagation ?, Eudureka 2023). As a result of the weights being revised backward, from input to output, the algorithm is known as a backward algorithm.

8.4.1 Why do we need Backpropagation algorithm?

The backpropagation algorithm is used to train the neural network related to particular datasets. For understanding why, we need this algorithm. We will understand it by some advantages of the Backpropagation Algorithm:

- The input has only one number, and the only parameter that can be tuned is how many inputs there are.
- It doesn't need any previous expertise and is very adaptive and effective.
- It is among the quickest, easiest, and most user-friendly ways to programme.

- Users do not need to master any specific expertise.
- It is regarded as a reliable standard procedure.
- The following are the disadvantages of the backpropagation algorithm:
- It prefers a matrix-based strategy to a mini-batch strategy.
- It is susceptible to noise and alterations.
- Errors may occur due to a lot of noisy data.
- Data from the input are very important for performance.
- Training requires a lot of time and resources.

8.4.2 Categories of Backpropagation

Backpropagation networks generally are classified into two different categories:

1. *Static Backpropagation:* A network designed to convert static inputs into static outputs is known as static backpropagation. Static backpropagation networks can be used to address issues involving static categorization, such as optical character recognition (OCR).
2. *Recurrent Backpropagation:* The recurrent backpropagation networks are utilised in recurrent backpropagation for fixed-point learning. Recurrent backpropagation's activation spreads it until reaches a fixed value.

The primary difference between the two is that static backpropagation offers instant mapping, recurring while backpropagation does not.

The application of the Backpropagation algorithm is as follows:

- The neural networks are utilized in the fields of voice recognition and trained for identify every letter in phrase and sentence.
- It is utilised in the fields of face and character recognition.

8.4.3 Working of Backpropagation Algorithm

How is the Backpropagation Algorithm implemented? The neural networks backpropagation algorithm (shown in figure 8.4) method uses the chain rule to determine the loss function gradient for a single weight. It computes one layer at a time effectively as opposed to a native direct calculation. Though it also computes the gradients, it does not define how to apply it. It expands the area of computation for the delta rule. (Haykin, S., 2009).

To understand, consider the diagram shown below of the backpropagation neural network:

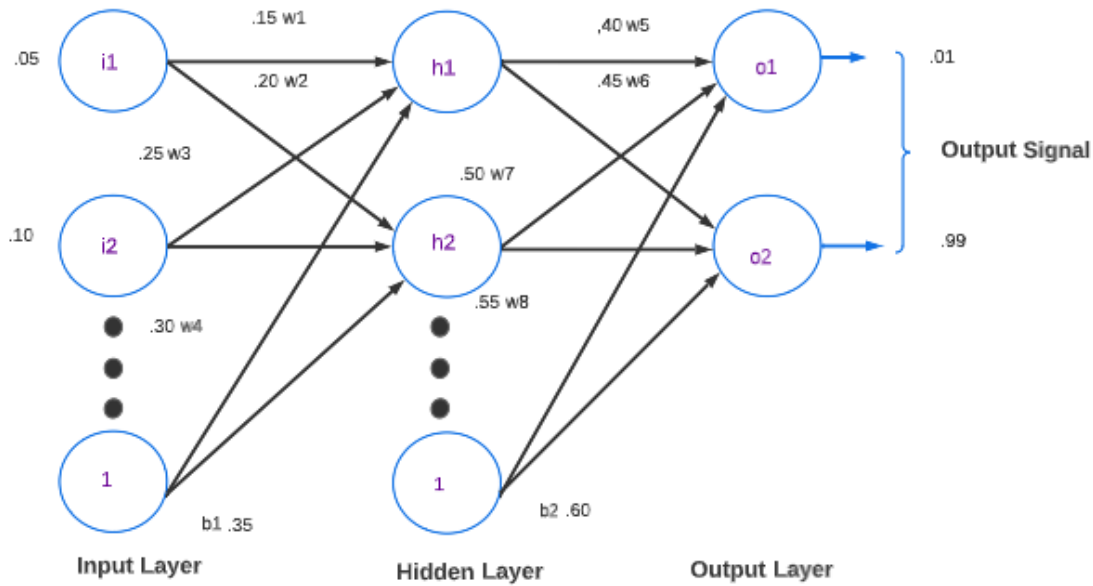


Fig. 8.4 Backpropagation Neural Network

The network mentioned above includes the following:

- Two input layer neurons.
- Two hidden layer neuron.
- Two output layer neurons.
- And two biases.

These are the steps used in Backpropagation Algorithm:

Step 1: Forward Propagation

Step 2: Backward Propagation

Step 3: Putting all the numbers in organized way to determine the updated weight value.

Step 1: Forward Propagation

In order to produce an output, input data is passed through a network in a forward direction using forward propagation. Hidden layers take in the data, process it in according with its activation function, and then pass it on to the layer below. The net input and net output are given in equation (8.3) and equation (8.4) respectively.

$$net\ h_n = w_n * i_n + w_n * i_n + b_n * n \quad (8.3)$$

$$out\ h_n = \frac{1}{1 + e^{-net\ h_n}} \quad (8.4)$$

Consider an example with two hidden layers $h1$ and $h2$, two inputs $i1$ and $i2$ with corresponding weights as $w1$ and $w2$ (figure 8.4). The output from each hidden layer is represented as $out\ h1$ and $out\ h2$.

Initially, the process started by propagating both input and output from the hidden layer to the next hidden layer i.e. output input as shown in figure 8.5.

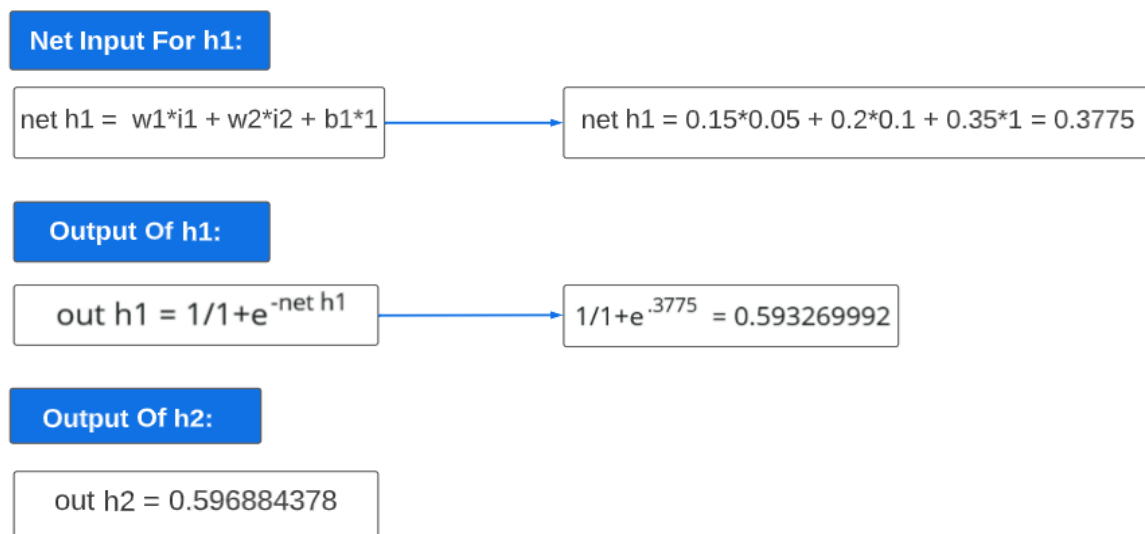


Fig. 8.5 Propagating forward for both input and output input

Next as shown in figure 8.6, the forward propagation is executed for the output layer of neurons. In the example, there are two neurons which takes input from the hidden layers' outputs.

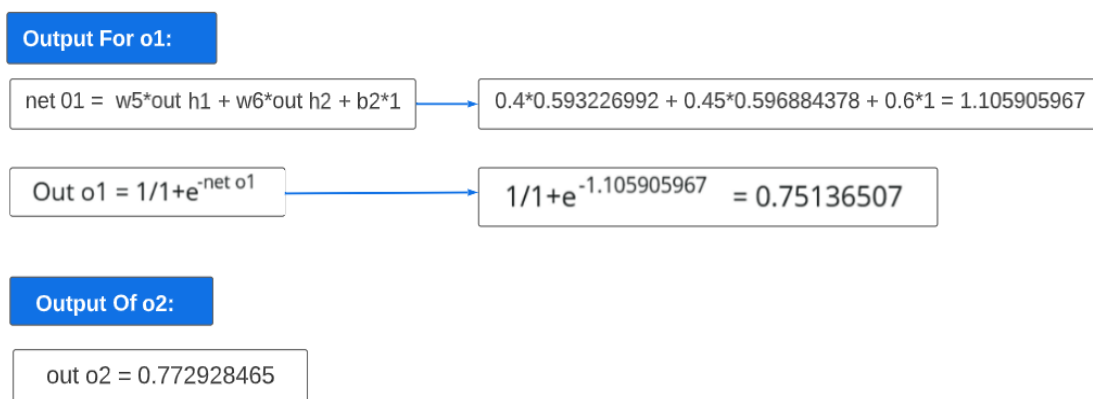


Fig. 8.6 Propagating forward for output input

After propagating through the output layer of neurons, it is required to propagate through the error values to update the model as required as given in figure 8.7.

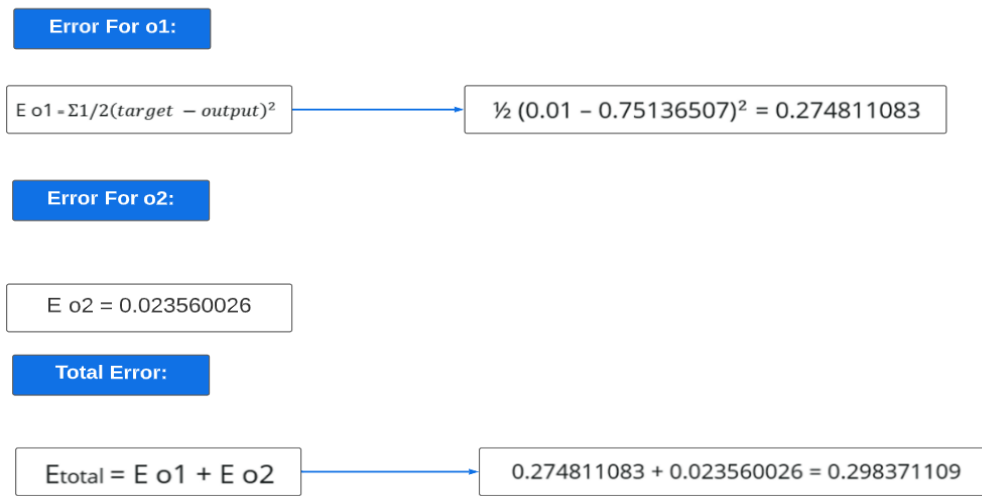


Fig. 8.7 Propagating forward for error value

Step 2: Backward Propagation

We will now propagate backward with error values as shown in figure 8.8. By altering the values of the weights and biases, we may try to reduce the error.

We will calculate the rate of change of error relative to the change in weight $W5$ using the equation (8.5).

$$\frac{\delta E_{total}}{\delta w_5} = \frac{\delta E_{total}}{\delta out\ o1} * \frac{\delta out\ o1}{\delta net\ o1} * \frac{\delta net\ o1}{\delta w_5} \quad (8.5)$$

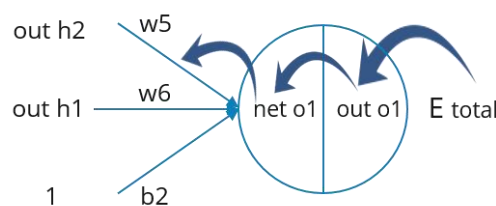


Fig. 8.8 Rate of change of error

First, since we are propagating backward, we must determine the change in total errors in comparison with outputs $o1$ and $o2$ using equation (8.6) and equation (8.7).

$$E_{total} = 1/2 (total\ o1 - out\ o1)^2 + 1/2 (total\ o2 - out\ o2)^2 \quad (8.6)$$

$$\frac{\delta E_{total}}{\delta out\ o1} = -(target\ o1 - out\ o1) = -(0.01 - 0.75136507) = 0.74136507 \quad (8.7)$$

We will now propagate even further back to calculate how *out o1* has changed with its overall net input is given in equation (8.8) and (8.9).

$$out\ o1 = \frac{1}{1 + e^{-neto1}} \quad (8.8)$$

$$\frac{\delta out\ o1}{\delta net\ o1} = out\ o1(1 - outo1) = 0.75136507(1 - 0.75136507) = 0.18681502 \quad (8.9)$$

Let's examine, using equation (8.11) that how *w5* has affected *net o1*(in equation (8.10)).

$$net\ o1 = w5 * out\ h1 + w6 * outh2 + b2 * 1 \quad (8.10)$$

$$\frac{\delta net\ o1}{\delta w5} = 1 * out\ h1\ w5^{(1-1)} + 0 + 0 = 0.593269992 \quad (8.11)$$

Step 3: Combining all the numbers to determine the updated weight value.

Putting all the results from equations (8.7), (8.9) and (8.11) in equation (8.5), we get

$$\frac{\delta E_{total}}{\delta w5} = \frac{\delta E_{total}}{\delta out\ o1} * \frac{\delta out\ o1}{\delta net\ o1} * \frac{\delta net\ o1}{\delta w5} = 0.82167041$$

To compensate the with error (propagated backwards), the new value of *w5* can be calculated using equation (8.12)

$$\begin{aligned} w5^+ &= w5 - n \frac{\delta E_{total}}{\delta w5} \Rightarrow w5^+ \\ &= 0.4 - 0.5 * 0.82167041 = 0.35891648 \end{aligned} \quad (8.12)$$

We can also determine the other weight adjustments by propagating the error and using equation (8.5) in the same manner.

Step 3. After that, we will move forward once more and compute the result. We'll compute the error once more.

Step 4. The step 2 and step 3 are repeated until the error is minimum.

8.5 Deep Learning

A subset of machine learning is deep learning, which has been the neural network having three or more than three layers. Artificial neural networks try to mimic how the human mind functions, but they are unable to match it in any way, allowing it all to "learn" from enormous volumes of data. Although even if a neural network only with one hidden layer can still produce reliable estimates, more layers can aid in optimization and improvement. Deep learning is considered as the subset of the artificial intelligence and machine learning as shown in figure 8.9.

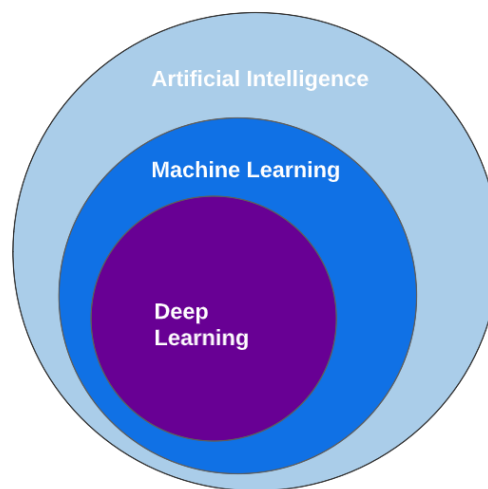


Fig. 8.9 Introduction to Deep Learning

Deep learning, also known as Deep Learning Neural Networks (DNNs) is a subset of machine learning that is built on ANNs with many layers. These neural networks were created to learn from huge amounts of data in an unsupervised or semi-supervised way. (Madhu K., 2023)

Several artificial intelligences (AI) applications and services are made possible by deep learning, which improves automation by carrying out both mental and physical tasks without the need for human intervention. Technologies like virtual assistants, voice-activated tv remote controls, and the detection of credit card fraud as well as upcoming technology like self-driving automobiles are all supported by science. A technique used in artificial intelligence (AI) called deep learning teaches computers to interpret data in a manner modelled after the human brain.

When we study deep learning there are many more questions arise in our mind like why we use deep learning why it is saying as deep learning, is machine learning not efficient, the difference between deep learning and machine learning and artificial intelligence, and many more.

8.5.1 Deep learning's workings

With the help of user input data, weights, as well as biases, artificial neural networks, often called as deep learning neural networks, attempt to replicate how the human brain functions. Collected, these components enable accurate item identification, classification, and description in the data.

Each layer of interconnected neurons that makes up the deep neural networks improves the categorization or prediction offered by the layer behind it. Calculations are transmitted via a network using a method called forward propagation. The output and input layers in a DNNs are the layers that are visible. The deep learning model then conducts the predictions or classification with in output layer after the data has already been analysed in the input layer.

Backpropagation is an alternative method that evaluates prediction errors using techniques like gradient descent until iteratively returning to the layers to modify the biases and weights of the function in order to train the model. The combination of back propagation algorithm and forward propagation allows a neural network to forecast the future and correct any errors. The algorithm's accuracy keeps improving over time.

The most fundamental kind of deep neural networks is described in the paragraph before. Deep learning techniques are quite complex compared to the various neural network models that can be used to solve specific problems or datasets. For example, CNNs (Convolutional Neural Networks), which have been frequently used in computer vision and image classification applications, have the capacity to recognise patterns and attributes in a picture, enabling actions like object recognition and detection. This is used for the first time in object recognition test, CNN surpassed a human in 2015. RNNs (Recurrent neural networks) are often used in applications for voice and natural language recognition because they utilise sequential or time series data.

8.5.2 Deep learning application

Although real-world deep learning applications are extensively utilised in daily life, customers frequently are unaware of the intricate data processing taking place in the background because of how well they are incorporated into goods and services. These are only a few examples.

1. Law enforcement

The effectiveness and efficiency of investigative inquiry can be improved by using computer vision, voice recognition, and other deep learning techniques in obtaining patterns as well as evidence in pictures, documents, audio and video recordings, or other media. This makes it possible for law enforcement to quickly and accurately process massive amounts of data.

2. Financial assistance

Financial institutions frequently use predictive analytics to promote algorithmic stock dealing, assess risks to businesses for loan approvals, spot fraud, and help clients manage their portfolios of investments and credit.

3. Consumer assistance

The technology that is used widely in businesses customer support procedures is deep learning. A simple type of AI is chatbots, which are used in many different applications, services, and customer support portals. Traditional chatbots, which are frequently found in call centres like menus, use natural language and even visual recognition. However, more advanced chatbot solutions try to determine whether there are multiple responses to ambiguous queries through machine learning. The chatbot then attempts to directly respond to these inquiries or direct the discussion to a human user depending on the responses it has received.

4. Healthcare

Since the digitization of health records and photos, the healthcare sector has greatly profited from deep learning techniques. Radiologists and imaging professionals can examine and evaluate more images in less time by using image recognition software.

5. Image recognition

Image recognition is the process of recognizing and classifying objects and people in photos to comprehend their context and content. Gambling, retail, tourism, and other industries are already using this region.

6. Earthquakes Prediction

Teaches a computer to carry out viscoelastic calculations, which are useful in earthquake prediction.

7. Image and Video Recognition

Deep learning models are employed to automatically categorize photos and videos, find objects and recognize faces. Applications include self-driving automobiles, surveillance systems, and picture and video search engines.

8. Gaming:

More realistic characters and environments are produced using deep learning models, which also enhances the gameplay.

9. Recommender Systems:

Deep learning models are employed in recommender systems, which allow users to receive individualized recommendations for products, movies, and news.

10. Social media:

Deep learning models are applied to filter spam, identify fake news, and flag hazardous information.

11. Autonomous systems:

Self-driving cars, drones, and other autonomous systems employ deep learning models to make judgments based on sensor input.

The advantages of deep learning are as follows:

- The best performance in class on problems.
- Makes feature engineering less necessary.
- Reduces unnecessary money.
- Easily detects flaws that are challenging to find.

The disadvantages of deep learning are as follows:

- A lot of information is needed.
- Costly to train in terms of computation is expensive.
- There is a lack of a strong theoretical foundation.

8.5.3 Difference between Machine Learning and Deep Learning.

Deep learning is a specific kind of machine learning. Selecting manually from photos the most crucial attributes is the initial stage of a machine learning process. The features are then used to create a prototypical that categorises the objects in the image. The relevant information is mechanically extracted from images using a deep-learning technique. The same fact is shown in figure 8.10. In

"end-to-end learning," some other component of deep learning, networks are giving raw data and a task to execute, such as classification, and it merely trains in what way to carry it out.

Alternative significant distinction is that while shallow learning methods converge, deep learning techniques expand with data. The performance of "shallow learning" machine learning methods is limited once you add additional examples as well as training data to the neural network. Deep learning networks frequently improve as the amount of your data increases, which is a significant advantage.

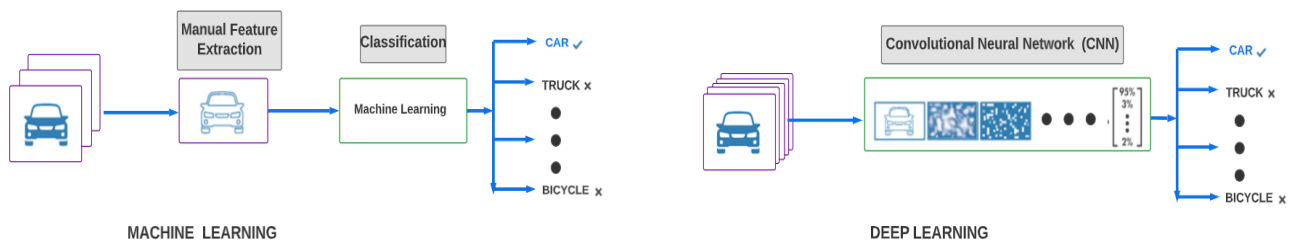


Fig. 8.10 comparison between deep learning and machine learning for categorizing cars.

To sort images in machine learning, characteristics and classifiers are manually chosen. Deep learning automates the steps of extracting features and modelling.

8.5.4 Future of Deep Learning.

Deep learning (DL) gained notoriety when a machine won the widely played game of AlphaGo against a human opponent. Many people have praised deep learning methods for teaching and learning because they "humanize" machines. Many of the advanced automation features already found in business AI platforms can be attributed to the quick advancement of Machine Learning (ML) and Deep Learning technologies.

This post compares AI, ML, and DL and examines the "ubiquitous" use of DL in Natural Language Processing (NLP) and computer vision applications, among other areas of AI. Automated systems, tools, and solutions that are AI- and DL-enabled are gradually encroaching on and All business sectors are being dominated by digital technology, from marketing to user experience, virtual reality to NLP.

References:

Deepak D. - Home (2023). Available at: https://deepakdvallur.weebly.com/uploads/8/9/7/5/89758787/module_3_ppt.pdf (Accessed: March 17, 2023).

Madhu K., says, J.C. and says, V.C. Introduction to deep learning, Intellipaat Blog. Available at: <https://intellipaat.com/blog/tutorial/machine-learning-tutorial/introduction-deep-learning/> (Accessed: March 17, 2023).

What is backpropagation? Training a neural network, Edureka. Available at: <https://www.edureka.co/blog/backpropagation/> (Accessed: March 17, 2023).

Haykin, S. (2009) Neural networks and learning machines. Upper Saddle River; Boston; Columbus etc.: Pearson Education.

Zola, A. and Vaughan, J. (2022) What is a backpropagation algorithm and how does it work? Enterprise AI. TechTarget. Available at: <https://www.techtarget.com/searchenterpriseai/definition/backpropagation-algorithm#:~:text=Backpropagation%2C%20or%20backward%20propagation%20of,data%20mining%20and%20machine%20learning>. (Accessed: March 17, 2023).

Chapter 9

Bayesian Learning

9.1 Introduction

Bayesian learning is a statistical framework for learning from data that is based on Bayes' theorem. In this framework, the goal is to update one's prior beliefs or probabilities about a hypothesis or model based on new data. In the Bayesian network or directed acyclic graphical model, a collection of random variables and their conditional connections are described by a Directed Acyclic Graph (DAG). A deep learning neural network called a DAG network has layers that are set up in the form of a directed acyclic graph. A DAG network's design might be more intricate, with layers receiving inputs from other layers and producing data to other layers. A Bayesian network, for instance, could display the probability of associations between illnesses and symptoms. The network can be used to determine how likely it is given a collection of signs that a particular set of diseases will be present. There are effective algorithms for inference and learning. (Ray, S. ,2019)

9.1.1 Features of Bayesian learning methods:

The estimated likelihood that a hypothesis is true can be progressively reduced or increased for each training example that is observed. Compared to algorithms that deny a theory directly if it is found to be irreconcilable with any specific case, this offers a more flexible method of learning. The ultimate probability of a hypothesis may be calculated using observed data and prior information. A prior probability for each event hypothesis in Bayesian learning is asserted, along with a probability distribution across the determined data for every conceivable hypothesis. Probabilistic prediction-making assumptions can be accommodated by Bayesian approaches. By integrating the predictions of several hypotheses and weighting them according to their probability, new examples may be categorized. Even in situations when Bayesian procedures are computationally infeasible, they can offer a benchmark for the best possible decision-making that can be used to compare alternative realistic approaches. (Ch, R., & MAP, M. ,1997)

Bayesian learning is a statistical approach to machine learning that involves using prior knowledge to make inferences and predictions about new data. Some of the key features of Bayesian learning methods includes the following:

Probability distributions:

Bayesian learning involves modelling data using probability distributions. These distributions are used to represent uncertainty and variability in the data, and to make predictions about new data.

Prior knowledge:

Bayesian learning incorporates prior knowledge into the model by using prior probability distributions. These priors represent what is known about the problem before any data is observed, and are updated as new data is collected.

Posterior distribution:

The probability distribution that is updated as a result of fusing previous information with observed data is known as the posterior distribution. For predicting fresh data, one uses the posterior.

Bayesian inference:

Bayesian learning involves using Bayesian inference to calculate the posterior distribution. Bayesian inference involves updating the preceding probability distribution based on the measured data using Bayes' theorem.

Model selection:

Bayesian learning allows for model selection by comparing the posterior probabilities of different models. This allows for the selection of the model that is most likely to generate the observed data.

Uncertainty quantification:

Bayesian learning provides a way to quantify uncertainty in model predictions. This is done using the posterior probability distribution, which provides a measure of the uncertainty in the predicted values.

Overall, Bayesian learning methods provide a powerful framework for machine learning that can be used to incorporate prior knowledge, quantify uncertainty, and make predictions based on probabilistic models.

9.2 Bayes Theorem

Using new data or information, Bayes' theorem, a foundational idea in statistics and probability theory, describes how to revise a hypothesis' probability. This technique is called after Thomas Bayes, an 18th-century British mathematician and philosopher. (Ch, R., & MAP, M. ,1997)

According to Bayes' Theorem given in equation (9.1), the likelihood of an explanation y given a hypothesis x is proportionate to the sum of the prior likelihood of x and the likelihood of y assumed x , split by the marginal likelihood of y :

$$P(x | y) = (P(y | x) * P(x)) / P(y) \quad (9.1)$$

Where,

$P(x | y)$ = posterior probability of the hypothesis x given the observed evidence y .

$P(y | x)$ = likelihood of the evidence y given the hypothesis x .

$P(x)$ = prior probability of the hypothesis x before observing the evidence y .

$P(y)$ = marginal likelihood of the evidence y , which is the probability of observing y across all possible hypotheses.

Bayes' theorem is often used in machine learning and data analysis to update the probability of a hypothesis or model based on new data or observations. It is a powerful tool for incorporating prior knowledge and uncertainty into statistical models, and for making probabilistic predictions based on observed data. In machine learning, we try to choose the superlative theory from a hypothesis space x with respect to the detected trained data y .

- The hypothesis with the greatest probability given the data y and any previous data regarding the odds of the several hypotheses in x is the best hypothesis in Bayesian learning.
- The Bayes theorem gives a technique to compute the probability of a hypothesis based on the probability of the predecessor, the probability of seeing different data provided the hypothesis, and the actual measured data.
- Given the training data as observed, we are concerned in the possibility $P(x|y)$ that x holds in ML tasks. y .
- A technique to estimate the posterior probability $P(x|y)$ from the prior probability $P(x)$, along with $P(y)$ and $P(y|x)$, is provided by the Bayes theorem.

- As per the Bayes theorem, $P(x|y)$ rises with $P(x)$ and $P(y|x)$..
- $P(x|y)$ tends to decrease as $P(y)$ raises, since y offers less proof supporting x , the more likely it is that y will be spotted independently of x .

Some examples and problems will make the concept clearer: -

Example 1: -

Users teach the system to recognise spam using Spam Assassin, which functions as a message filter. It analyses word trends in emails that users have flagged as *spam*. For instance, it might have discovered that 30% of the emails have the term *Release* flagged as *spam*. The term *Release* appears in 0.8% of *non spam* emails, and 40% of all emails that users receive are spam. If the phrase "issue" appears in a message, determine the likelihood that it is *spam*. (Naive Bayes classifiers, 2023).

Solution:

Given,

$$P(\text{Release} | \text{Spam}) = 0.3$$

$$P(\text{Release} | \text{Non Spam}) = 0.008$$

$$P(\text{Spam}) = 0.4$$

$$P(\text{Non Spam}) = 0.4$$

$$P(\text{Spam} | \text{Release}) = ?$$

Now, using Bayes' Theorem:

$$P(\text{Spam} | \text{Release}) = P(\text{Release} | \text{Spam}) * P(\text{Spam}) / P(\text{Release})$$

$$= \frac{0.3 * 0.4}{(0.4 * 0.3 + 0.3 * 0.008)}$$

$$= 0.98$$

Therefore, the requisite possibility is 0.98.

Example 2: -

Container 1 has four red and eight green balls, while Container 2 has five red and three green pens. One pen is randomly selected from one of the containers, and it turns out to be a green pen. Calculate the probability that the pen will be selected from Container 1.

Solution:

Given,

Let $E1$, $E2$ and A be the three events.

where, $E1$ = Event of selecting Container 1

$E2$ = Event of selecting Container 2

A = Event of drawing green pen

Now,

$$P(E1) = P(E2) = \frac{1}{2}$$

$$P(\text{drawing a green pen from Container 1}) = P(A|E1) = \frac{8}{12} = \frac{2}{3}$$

$$P(\text{drawing a green pen from Bag2}) = P(A|E2) = \frac{3}{8}$$

Using Bayes' Theorem, the possibility of selecting a green pen from container 1,

$$\begin{aligned} P(E1|A) &= \frac{P(A|E1) P(E1)}{P(A|E1) P(E1) + P(A|E2) P(E2)} \\ &= \frac{\left(\frac{2}{3} * \frac{1}{2}\right)}{\left(\frac{2}{3} * \frac{1}{2} + \frac{3}{8} * \frac{1}{2}\right)} \\ &= \frac{16}{25} \end{aligned}$$

Hereafter, the possibility that the pen is drawn from Container 1 = 0.64

9.2.1 Bayes Theorem Applications

An important concept in probability theory is the Bayes' theorem, that gives formulas for calculating the conditional probability of an occurrence provided information or evidence. It has many applications in various fields, including:

- i. *Medical diagnosis:* Bayes' theorem is widely used in medical diagnosis to calculate the probability of a patient having a disease given their symptoms and other medical history. By combining prior knowledge about the disease with new test results, doctors can make more accurate diagnoses.
- ii. *Spam filtering:* Bayes' theorem is used in spam filtering algorithms to calculate the probability that an incoming email is spam, given its content and other features. By using a large dataset of known spam and non-spam emails, the algorithm can learn to make accurate predictions.

- iii. *Fraud detection:* Bayes' theorem is also used in fraud detection systems to calculate the probability that a transaction is fraudulent, given the user's history and other factors. By combining prior knowledge about common fraud patterns with real-time transaction data, these systems can identify and prevent fraudulent activity.
- iv. *Image and speech recognition:* Baye's theorem is used in machine learning algorithms for image and speech recognition. By using prior knowledge about the characteristics of different images or speech patterns, the algorithm can make more accurate predictions about what it is seeing or hearing.
- v. *Risk analysis:* Bayes' theorem is used in risk analysis to calculate the likelihood of a particular event occurring, given some prior knowledge or evidence. This can be used in a variety of contexts, such as financial risk analysis or environmental risk assessment.
- vi. Overall, Bayes' theorem is a powerful tool that can be applied in various different areas to build more precise predictions and decisions based on available evidence.

9.3 Naïve Bayes Classifiers

A probabilistic machine learning method called the Naive Bayes classifier is built on the Bayes theorem and assumes that characteristics are independent. It is a quick and straightforward method that is effective with big databases and frequently applied to text categorization issues like spam filtering and sentiment analysis. Naive Bayes algorithm can be implemented in three types: Gaussian Naive Bayes, Multinomial Naive Bayes, and Bernoulli Naive Bayes.

Naive Bayes classifier is a class of probabilistic machine learning models depends on Bayes' theory, that assumes individuality amongst the features or variables. Naive Bayes classifier often used in text classification, spam filtering, and other various applications where the input data consists of a set of features or attributes. The “naïve” assumption in Naive Bayes classifiers is the attributes that are conditionally distinct from the class label. (Berrar, D., 2018) In other words, each feature contributes independently to the probability of the class, and the features are assumed to be unrelated to each other.

Calculating the posterior probability of every class given the observed characteristics and then selecting the class with the greatest probability is the fundamental principle of a Naive Bayes classifier. The classifier works by first estimating the prior probability of each class depends upon the training data, then calculating the probability of every feature given every class. These probabilities are then combined by Bayes' theorem to determine the posterior probability of every class.

Naive Bayes classifiers are easy to implement and computationally efficient, making them well-suited for large-scale classification tasks. However, their accuracy can be limited by the naive assumption of feature independence, which may not hold in some cases. Additionally, Naive Bayes classifiers can be sensitive to the presence of irrelevant features, which can reduce their performance. Despite these limitations, Naive Bayes classifiers are widely used in practice and have been shown to perform well in many applications (Berrar, D. ,2018).

Types of Naive Bayes Classifier:

Naive Bayes classifiers come in a variety of forms, and each one makes a distinct assumption regarding the distribution of the features and how they relate to the class labels. The most common available Naive Bayes classifiers (Naive Bayes classifiers, GFG, 2023) are:

i. Multinomial Naive Bayes (MNB)

This is mainly used to resolve text classification problems, like figuring out whether a document belongs in the category for games, politics, technology, etc. The classifiers use the document's word rate as one of its characteristics or predictions. This classifier is used for count data, where each feature represents the number of occurrences of an actual word or term in a document. It assumes that the features are generated by independent multinomial distributions and calculates the possibility of each class depending on the frequency of each feature.

A probabilistic algorithm called MNB is used for categorization jobs, especially text classification. It is an adaptation of the Naive Bayes algorithm, that is founded on the Bayes theory and the conception that the characteristics are conditionally distinct given the class. In MNB, the features are the frequencies of the words in the text. The algorithm calculates the possibility of a document fitting to a particular class (e.g., spam or not spam) based on the incidence of each text in the document and

the occurrence of each word in each class. The algorithm assumes that the distribution of word frequencies in each class follows a multinomial distribution.

To use MNB for classification, the algorithm is trained on a labelled dataset, where the class labels are known. During training, the algorithm calculates the probability of each word in each class, along with the prior probability of each class. During testing, the algorithm takes a new, unlabelled document as input, and estimates the possibility of the text fitting to each class based on the word frequencies in the document and probabilities calculated during training. The algorithm then allocates the text to the class with the uppermost probability. MNB is a quick and effective algorithm that handles large datasets with many features. It is frequently It is commonly employed in work involving text classification such as sentiment analysis, junk filtering, and topic classification. However, its performance may suffer if the independence assumption does not hold, or if the frequency of a particular word in a class is very low.

ii. Bernoulli Naive Bayes

Using binary factors as models, this approach is analogous to multinomial naive bayes. The parameters we use to identify the class variable can only have a yes or no response, such as whether a word exists in the text or not. This classifier is used for binary data, where each feature can take on only two values (e.g., 0 or 1). It considers that the features are engendered by independent Bernoulli distributions and calculates the probability of each class depending on the occurrence or absence of each feature. This classifier is used for continuous data, where each feature is a real-valued variable. Furthermore, the features are created by independent Gaussian distribution, and calculates the possibility of each class depends on mean and variance of features. (Ray, S., 2019)

Another version of the Naive Bayes algorithm is Bernoulli Naive Bayes, used for classification tasks, particularly for binary text classification. Features in this version are binary, i.e., each feature is either present or absent in the document. The Bernoulli Naive Bayes algorithm assumes that each feature is conditionally independent given the class and follows a Bernoulli distribution. It calculates the possibility of a text belonging to a specific class i.e., the presence or absence of each feature in the document and frequency of each feature in each class.

To use Bernoulli Naive Bayes for classifications, system is trained on a labelled dataset, where the class labels are known. During training, the algorithm calculates the probability of each feature being present or absent in each class, along with prior probability of each class. During testing, the algorithm

takes a new, unlabelled document as input and computes the possibility of the text belonging to each class based on the occurrence or absence of each feature in the text and the probabilities calculated during training. The algorithm then assigns the document to the class with the highest probability. Bernoulli Naive Bayes is a quick and efficient technique that can handle large datasets with many features. It is basically used for binary text classifications task like sentiment analysis, junk filtering, and text categorization. However, like other variants of Naive Bayes, its performance may suffer if the independence assumption does not hold.

iii. Gaussian Naive Bayes

Gaussian Naive Bayes (GNB) is a variant of the Naive Bayes algorithm used for classification tasks, particularly for continuous data. In this variant, the features are continuous and assumed to follow a Gaussian (normal) distribution. GNB calculates the probability of a document belonging to a certain class based on the mean and variance of each feature in each class. It assumes that each feature is conditionally independent given the class, and the probability density function of each feature in each class follows a Gaussian distribution.

To use GNB for classification, the algorithm is trained on a labelled dataset, where the class labels are known. The algorithm computes the prior possibility of each class, the mean and variance of each characteristic in each class, and both during training. During testing, the algorithm takes a new, unlabelled document as input and calculates the probability of the document belonging to each class based on the mean and variance of each feature in each class and the probabilities calculated during training. The algorithm then provides a text to the class with greatest probability.

GNB is a simple and efficient process that works well with small datasets and continuous features. It is mainly work for classification tasks like medical diagnosis, image recognition, and financial analysis. However, its performance may suffer if the independence assumption does not hold, or if the distribution of the features in each class is significantly different from a Gaussian Distribution. When the outcomes of the variables take up a continuous value rather than being discontinuous, we assume that these values are examples from a Gaussian distribution. Figure 9.1 shows the graphical representation of the GNB outcome.

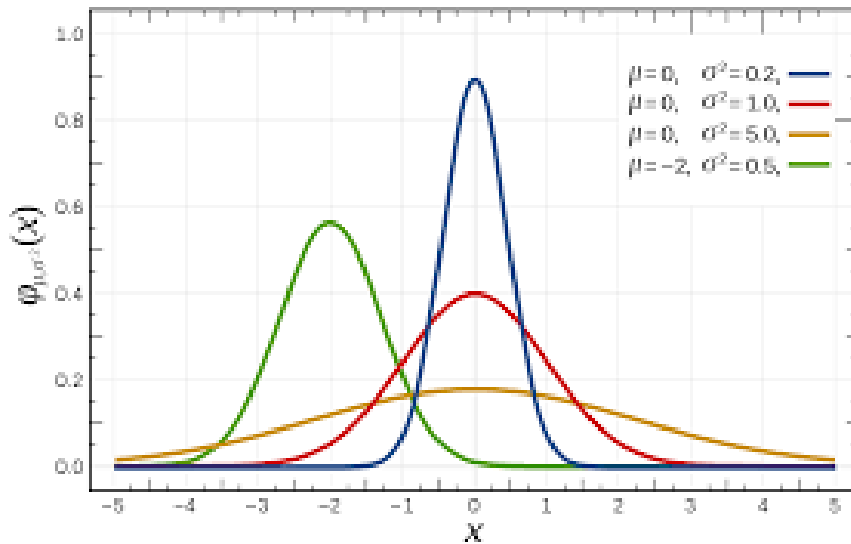


Fig. 9.1 Gaussian Naïve Bayes (Ch, R, 1997)

Due to variations in how the numbers are presented throughout the collection, the conditional probability formula also alters as given in equation (9.2).

$$P(x_i | y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \quad (9.2)$$

In addition to these basic types, there are also hybrid variants of naive Bayes classifiers that combine elements of multiple types, such as the Mixed Naive Bayes classifier, which can handle datasets with both continuous and discrete features. The choice of which type of naive Bayes classifier to use depends on the nature of the data and the assumptions that are most appropriate for the problem at hand. In practice, it is often a good idea to try multiple types of classifiers and compare their performance on a validation set.

Now, check out how Scikit-Learn implemented the Gaussian Naive Bayes algorithm.

```

1. # import gaussian naïve bayes functions from sklearn library
2. from sklearn.naive_bayes import GaussianNB as my_naive_model
3. # load the iris dataset
4. from sklearn.datasets import load_iris
5. my_iris_data = load_iris()
6. # store the feature matrix (X) and response vector (y)
7. X = my_iris_data.data
8. y = my_iris_data.target
9. # splitting X and y into training and testing sets
10. from sklearn.model_selection import train_test_split
11. X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1)
12. # training the model on training set
13. my_naive_model = GaussianNB()
14. my_naive_model.fit(X_train, y_train)
15. # making predictions on the testing set for target such as Iris-setosa, Iris-versicolor or
    Iris_virginica
16. y_prediction = my_naive_model.predict(X_test)
17. # comparing actual response values (y_test) with predicted response values (y_prediction)
18. from sklearn import metrics
19. print("Gaussian Naive Bayes model accuracy(in %):", metrics.accuracy_score(y_test,
    y_prediction)*100)

```

Output:

```
Gaussian Naive Bayes model accuracy (in %): 95.0
```

Here are some key items to consider as we draw to a close to this article:

- Despite what seem to be their overly simplistic conventions, naïve Bayes classifiers have done excellently in a variation of real-world situations, most notably documents categorization and junk filtering. The only prerequisite a slight quantity of training data to determine the needed parameters.
- In contrast to more sophisticated approaches, naïve Bayes classifiers and learners can work extremely quickly. Since the separation of the class conditional feature distributions, every distribution can be independently approximated as one-dimensional distribution. It helps to resolve the difficulties caused by the dimensions plague.

References:

- Ch, R., & MAP, M. (1997). Bayesian learning. *book: Machine Learning. McGraw-Hill Science/Engineering/Math*, 154-200.
- Berrar, D. (2018). Bayes' theorem and naive Bayes classifier. *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics*, 403, 412.
- Naive Bayes classifiers (2023) GeeksforGeeks. GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/naive-bayes-classifiers/> (Accessed: February 18, 2023).
- Ray, S. (2019, February). A quick review of machine learning algorithms. In *2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon)* (pp. 35-39). IEEE.

Chapter 10

Instance-Based Learning

Instance-based learning is a sort of machine learning calculation that learns directly from the training examples rather than from an explicit model. In this approach, the algorithm stores the entire training dataset and uses it to make predictions for new inputs. The basic idea behind instance-based learning is that new inputs that are similar to the training examples are likely to have similar outputs. Therefore, when a new input is presented to the algorithm, it finds the training examples that are closest to the new input and uses them to make a prediction. Such type of algorithms spends more time on prediction rather than training. Unlike classification algorithms, instance-based algorithms sit ideal until testing data appears; as soon as testing data appears algorithm starts process to compute the similarity among existing examples to predict the label, therefore, these algorithms are also termed as lazy learners. Though they are known as lazy learners but contradictory to their name such algorithms are generally faster in prediction as they don't require any explicit training for the model. They also capture the complex relationship between inputs and outputs that may be difficulty to capture using explicit modelling.

However, instance-based learning has some limitations also. One limitation is that it can be computationally expensive since it requires searching through the entire training dataset for each new input. Additionally, the performance of the algorithm can be sensitive to the choice of distance metric used to compare inputs. Finally, the algorithm may suffer from overfitting or underfitting if the training dataset contains noisy or irrelevant examples. (Ray, S. (2019). A quick review of machine learning algorithms)

10.1 Nearest Neighbor Learning

Nearest neighbour learning is a sort of machine learning calculation that belongs to the category of instance-based learning. In nearest neighbour learning, the model is not explicitly trained to learn a function that maps inputs to outputs. Instead, the model simply stores the training examples, and when a new input is given, it looks up the closest (nearest) training example and outputs the corresponding output. The idea behind nearest neighbour learning is that similar inputs have similar outputs.

Therefore, if enough training examples are stored than the closest example to the new instance is chosen for label prediction.

Nearest neighbor learning's key benefit is that it could be very effective when there is a large amount of data and the underlying relationship between inputs and outputs is complex and hard to define. However, it can also be computationally expensive since it requires searching through the entire dataset for each new input.

10.2 Remarks on k - Nearest Neighbor Algorithm (KNN)

A straightforward but efficient machine learning approach used for classification and regression problems is the k -nearest neighbours (KNN) algorithm. It is a non-parametric technique that means no specific probability distribution is assumed before progressing to the training and testing. Its predictions are based on the training set's k nearest data points. The KNN algorithm relegates a class name to a new data point in classification tasks in view of the class names of its k nearest neighbours. The anticipated class label for the new data point is given as the class with the highest frequency among the k nearest neighbours. The KNN algorithm predicts the value of a new data point in regression tasks by averaging or averaging out the values of its k closest neighbours. To determine the distance between data points, the KNN algorithm needs a distance measure. The Euclidean distance that calculates the straight-line separation between two points in n -dimensional space, is the most often used distance metric.

The KNN algorithm's lack of assumptions regarding the data's underlying distribution is one of its benefits. When working with huge datasets, it might be computationally expensive, and it could not work effectively if the feature space is high-dimensional, or the training set is unbalanced.

Overall, the k –Nearest Neighbor algorithm is useful and intuitive method for classification and regression tasks, particularly for small to medium-sized datasets. The following points must take for the KNN method.

- Among the most straight forward machine learning algorithms, in view of the supervised learning algorithm, the simplest one is k Nearest Neighbor.
- The k -NN algorithm puts the new case in the classification that is most equivalent to the accessible classes, expecting that the new case/information and the current cases are comparative.

- The k -NN algorithm saves the data that is all suitable and orders new information in light of similitude(similarity). This suggests that new information can be quickly and accurately classified into an appropriate class using the k -NN technique.
- The k -NN approach can be utilized for both grouping and relapse issues, yet it is most often used for order issues.
- k -NN is a non-parametric technique, and that implies it makes no suspicions about the basic information and probabilistic data distribution.
- It is generally alluded as an inactive beginner as it does not start learning from the examples in training phase until the test data appears i.e., it stores-scans-computes the training set for categorizing upcoming instance on its arrival.
- The k -NN algorithm fundamentally stores the dataset throughout the preparation step, classifying fresh data into a category that is highly similar to the new data.
- It is type of lazy learner that does not begin working with training data until testing instances appears.

10.2.1 Example of KNN classifier:

Assume, for example, it is to be determined if the creature (*input value*) depicted in figure 10.1, resembles to category *cat* or category *dog*. However, since the KNN calculation depends on a comparability metric, it could be used for distinguishing proof to label input creature as *cat* or *dog*. The KNN model(*K-Nearest Neighbor(KNN) algorithm for Machine Learning - Javatpoint (2023)*) will search for likenesses between the new informational index's elements and those in the photographs of felines and canines, and in view of those similitudes (attributes considered for the similarity), it will characterize the new informational collection as one or the other feline or canine related.

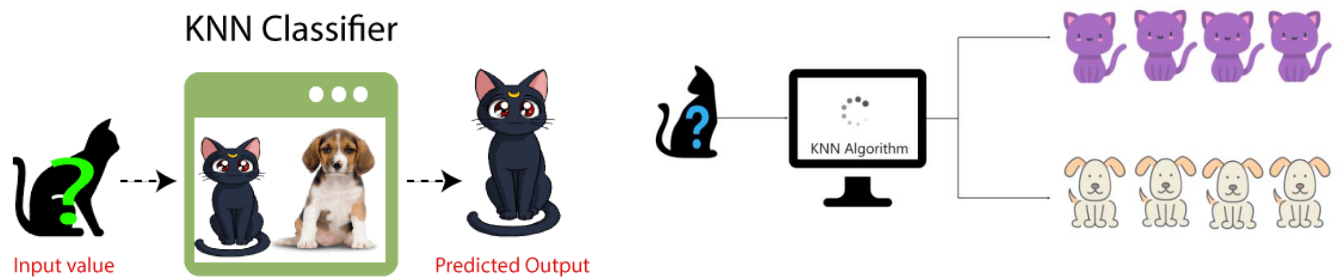


Fig. 10.1 k –NN Classifier Image Representation (*K-Nearest Neighbor(KNN) algorithm for Machine Learning - Javatpoint, 2023*)

10.2.2 Why KNN Algorithm is required?

On the off chance that there are two classifications, Class A and Class B, and we have another data of interest, x_1 , which classification does this information point have a place ready? We require a k –NN calculation to resolve this sort of issue. KNN simplifies it to decide the class or category of a given dataset.

Look at the graph in figure 10.2 to understand the learning of KNN model .

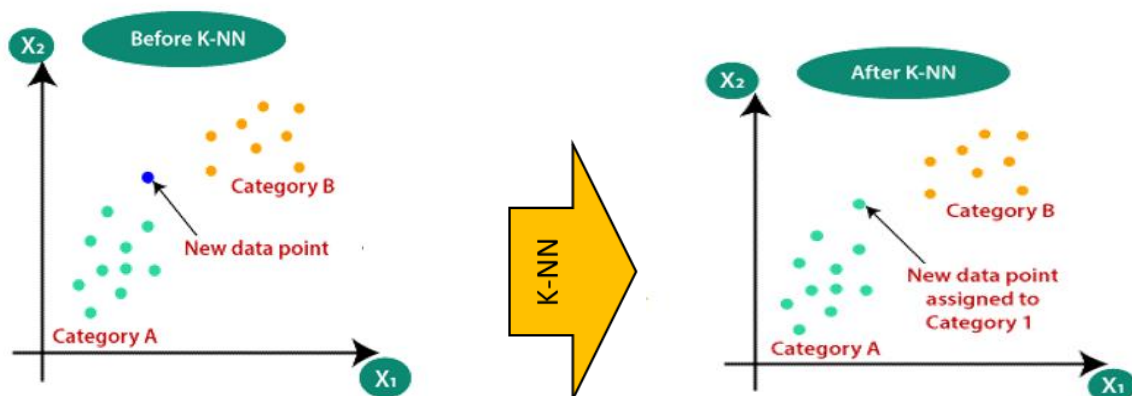


Fig. 10.2 - KNN Classifier Working (*K-Nearest Neighbor (KNN) algorithm for Machine Learning – Javatpoint, 2023*)

10.2.3 How does KNN function?

The accompanying calculation can be utilized to depict how the K-NN functions:

Step 1. k means the number of neighbours considered under study for learning the algorithm. Decide the value of k very carefully.

Step 2. Calculate the distance among the data points and the new test instance to be classified. The distance may be calculated using Manhattan, Euclidean distance measure, etc. The formula for distance measures is given in table 10.1.

Step 3. Extract the k nearest neighbour of new instance i.e., k data points that are closest to the new test instance.

Step 4. Find out the class labels of k nearest neighbours. The outcome-class label of new test instance is decided by the majority of class labels among k -nearest neighbours.

Step 5. Add this new test instance in the knowledge base of the KNN algorithm.

Step 6. Here, the learning-predicting cycle of KNN completes.

Table 10.1 Distance measures to compute distance between point x and point y in n -dimensional space

Distance Measure	Formula	Description
Manhattan Distance	$D(x, y) = \sum_{i=1}^n x_i - y_i $	Distance between point x and point y in the n -dimensional space.
Euclidean Distance	$D(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$	

KNN Algorithm with example given in figure 10.3.

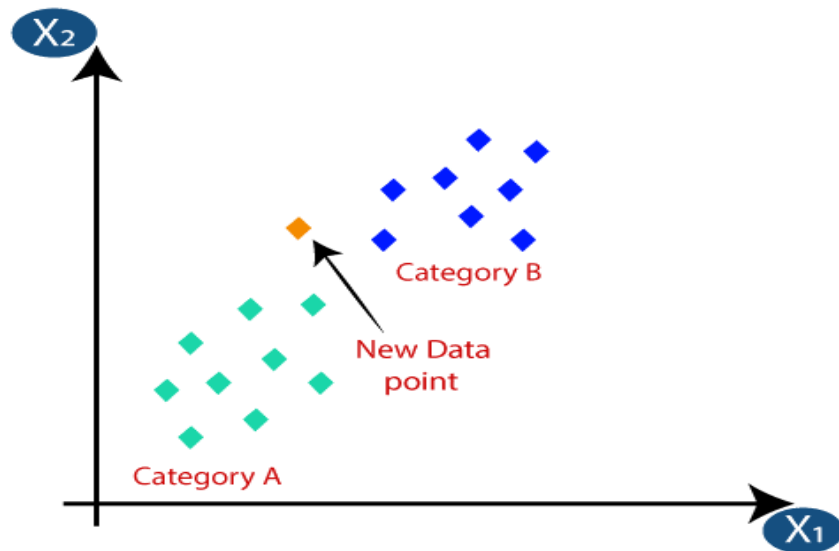


Fig. 10.3 KNN Classifier (*K-Nearest Neighbor(KNN) algorithm for Machine Learning – Javatpoint, 2023*)

- Let us assume the number of nearest neighbours to be, $k = 5$.
- Afterwards, this suggests that fresh material can be classified accurately and quickly using the K-NN method.
- Calculations can be made as shown in fig. 10.4.
- By calculating the Euclidean distance, the nearest neighbours could be traced. Among 5 nearest neighbours two belong to category B (represented in blue) and three belong to category A (represented in red) as shown in figure 10.5. The majority of class labels from 5 nearest neighbors votes for category A. Therefore, label of new instance is category A.

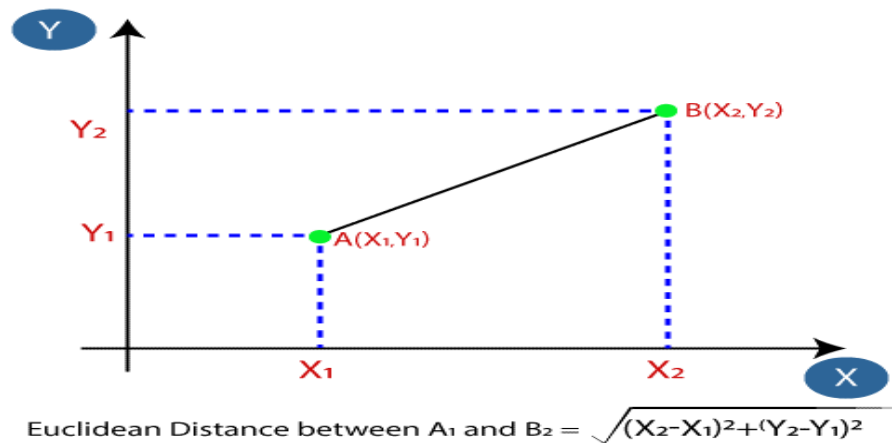


Fig. 10.4 - Euclidean Distance (*K-Nearest Neighbor(KNN) algorithm for Machine Learning - Javatpoint, 2023*)

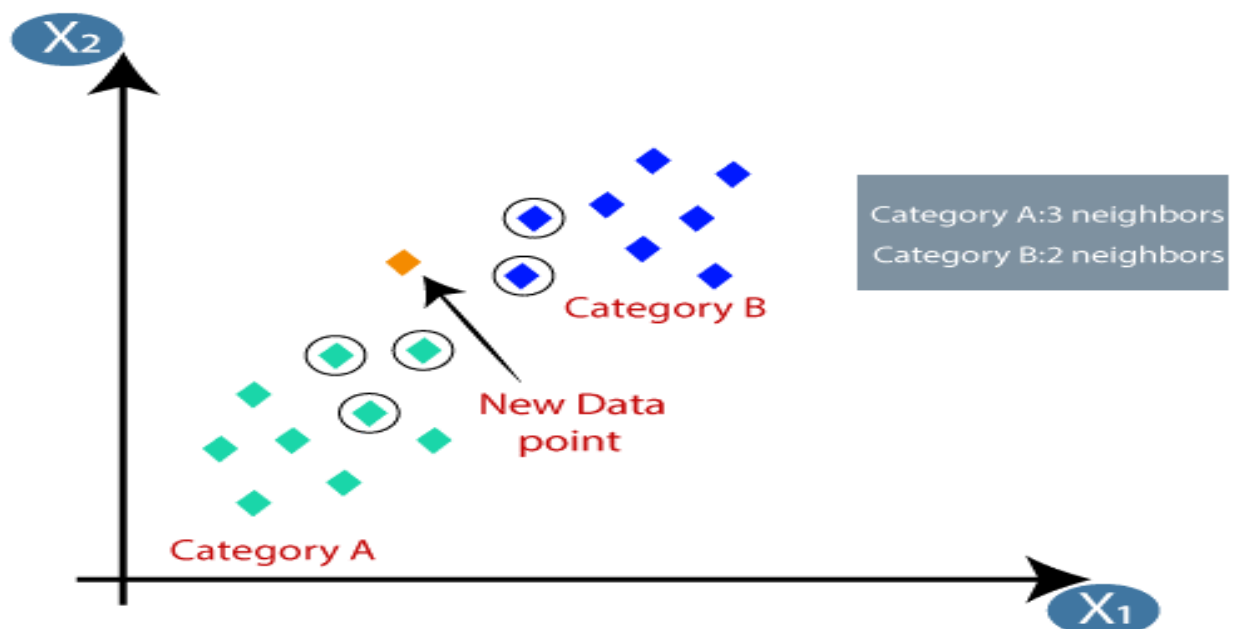


Fig. 10.5 KNN Applied (*K-Nearest Neighbor(KNN) algorithm for Machine Learning - Javatpoint ,2023*)

10.2.4 How is k in the K-NN Algorithm to be chosen?

While selecting k 's value for KNN algorithm, keep the following key points in mind:

- As there is no optimum method to select the ideal number for " k ," therefore, the algorithm must be tested with multiple values to discover which one performs the best. Mostly, k is most accurately portrayed by the number 5.
- An extremely low value of k , such as $k = 1$ or $k = 2$, may be unreliable and lead to model outlier effects.

- The smaller values of k may make model more sensitive to the exceptions and anomalies and may give weight to these particular points only.
- The too larger values of k may lead a generic model, that will cause problem of underfitting.
- k should be chosen as an odd discrete value, to define the win situation without any confusion.

10.2.5 The KNN Algorithms advantages include:

The k-nearest neighbours (KNN) algorithm (Ray, S. ,2019) has several advantages:

1) Simple to enforce:

KNN is a simplistic algorithm that is simple to comprehend and use. It doesn't call for any complex mathematical models or assumptions about underlying distribution of the information.

2) Multivariate:

KNN is a non-parametric algorithm, therefore doesn't make any assumptions about how the data are distributed at its core. This makes it appropriate for information sets with complicated or ambiguous distributions. Another assumption about the data set may be its linear or non-linear behaviour, KNN works independently without considering this fact.

3) No training phase:

KNN doesn't require a training phase, in contrast to other algorithms for machine learning. Instead, it uses every aspect of the training set for prediction, which can be useful in scenarios where information is constantly changing. It stores the training examples in its knowledge base and when testing data appears, it utilizes every training example to predict the class of new instance. This procedure makes algorithm run faster and accurately. This not only increases the speed but identify the hidden explicit relationships between input and output variables.

4) High accuracy:

K-NN is known for its high accuracy, especially when the number of neighbors is chosen carefully. Even when there are missing numbers or the data is noisy, it can still perform effectively.

5) Both classification and regression purposed

KNN is a flexible method that may be used for both tasks. KNN regression predicts the continuous dependent variable value by local average. KNN classification attempts to predict the class by the majority vote method among nearest neighbours.

6) Interpretable:

Since the technique simply allocates the new data point to the class with the highest frequency among its k closest neighbours, the predictions generated by KNN are simple to understand. KNN method is a popular option for many applications since it is a strong and adaptable algorithm with a number of advantages over other algorithms.

10.2.6 Disadvantages of KNN Algorithm:

The k -nearest neighbours (KNN) algorithm also has several disadvantages:

1. Computationally expensive:

KNN requires a lot of computation to find the k nearest neighbours, especially when dealing with large datasets. As the number of dimensions in the dataset increases, the computation required increases exponentially.

2. Sensitive to irrelevant features:

KNN is sensitive to irrelevant features in the dataset. When irrelevant features are present, the algorithm may give more weight to those features, leading to a decrease in accuracy.

3. Curse of dimensionality:

The instance is said to be well defined if its feature set covers all attributes. But as the number of attributes increases the distance among instances become meaningless. This is known as the Curse of Dimensionality. KNN is also affected by this problem as it is distance dependent; that may lead poor performances in higher dimensions.

4. Needs good choice of k :

The choice of k may significantly affect the performance of KNN. If k is too small, the algorithm may be sensitive to noise and outliers, while if k is too large, the algorithm may be biased towards the majority class.

5. Imbalanced data:

KNN can also perform poorly on imbalanced datasets, where the number of data points in each class is not equal. In such cases, the algorithm may be biased towards the majority class.

10.2.7 Features of KNN Algorithm

1. Supervised Machine Learning Algorithm

KNN is a supervised learning method/technique that predicts/outputs the result of the sample points using a labelled input/raw data set.

2. Simplest

One of the simplest and mathematically comprehensible machine learning models, can be used to resolve a wide variety of problems.

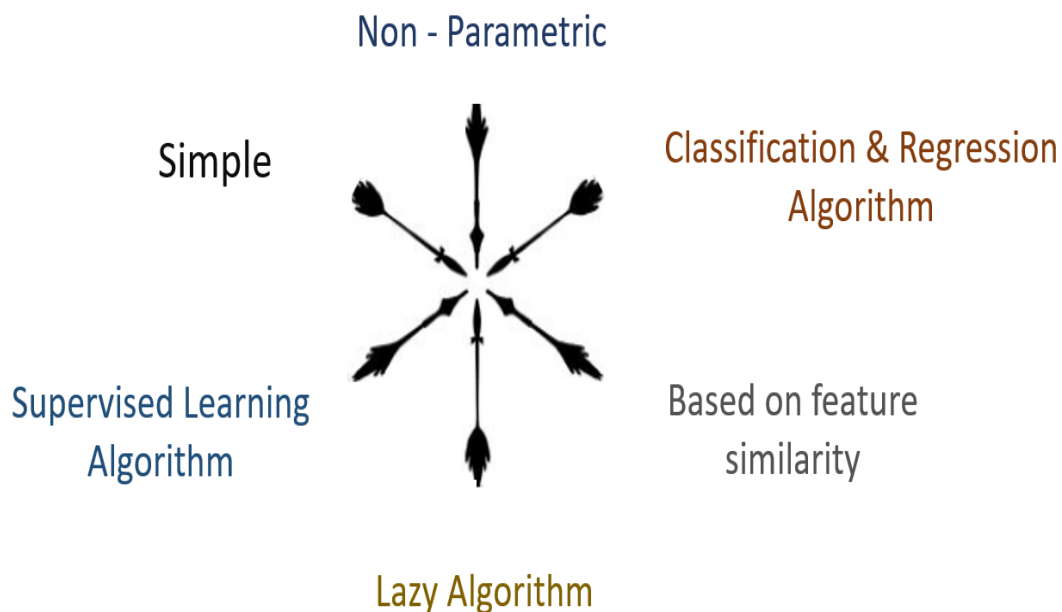


Fig.10.6 Features of KNN

3. Lazy Learning

It does not require the model to be trained initially. It utilizes the training examples when testing data appears. Therefore, it is considered as the lazy learner but faster in prediction results.

4. Non-Parametric

Unlike other algorithms, KNN is a non-parametric model, which means it makes no assumptions about the data set distribution. As a result of being able to handle actual data, the algo is more effective.

5. Classification and Regression

KNN can be used to address classification and regression problems. Classification relates to the discrete categorical bifurcations. On other hand regression is a way to compute the value of independent variable by using local averages.

6. Based on Feature Similarity

The similarity of features among the dataset is the basis for model. A data point is classified into the class to which it is most comparable using KNN, which examines how similar it is to its neighbor.

The main features are depicted in the figure 10.6, KNN do have more features also listed as below:

- It can identify the hidden explicit relationship of the input and output data.
- It utilizes the training examples in real time therefore, it is easier to consider any update in the training examples under study.
- Because KNN is a sluggish algorithm, it internalizes the data set used for training rather than drawing conclusions about the training data based on direct or indirect influences.

Overall, the KNN algorithm can be a powerful and effective machine learning algorithm, it finds its implementation in many domains such as text mining, agriculture, medical, finance, facial recognition, recommendation system. Every technique comes with limitations also, the same is the case with KNN. It also has several limitations that need to be considered when using it for a particular task.

References:

Ray, S. (2019, February). A quick review of machine learning algorithms. In 2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon) (pp. 35-39). IEEE.

Ch, R., & MAP, M. (1997). Bayesian learning. book: Machine Learning. McGraw-Hill Science/Engineering/Math, 154-200.

Iris Dataset | Kaggle, <https://www.kaggle.com/datasets/vikrishan/iris-dataset>

K-Nearest Neighbor(KNN) algorithm for Machine Learning - Javatpoint (2023) www.javatpoint.com. Available at: <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning> (Accessed: March 27, 2023).

Ch, R., & MAP, M. (1997). Bayesian learning. book: Machine Learning. McGraw-Hill Science/Engineering/Math, 154-200.

Chapter 11

Support Vector Machines

11.1 Introduction

For classification and regression analysis, Support Vector Machine (SVM), a supervised ML technique, is used. Finding the best hyperplane to divide the data into distinct classes is the aim of SVM. SVM determines the line that optimises the margin between the two classes in a two-class problem. SVM is very helpful when the data has a distinct margin of separation. The method works well for both linearly and non-linearly separable information, transforming the incoming data into a high-dimensional region where linear boundaries may be found.

SVM, one of the most popular supervised learning algorithms, is used to resolve classification and regression problems. However, it is widely implemented in ML Classification issues. The SVM algorithm focuses on finding the best arc or decision boundary that can quickly divide the n -dimensional regions into groups to quickly categorize information that will be received in the future. A hyperplane is the term of the optimal choice boundary. The maximum vector and spots which contributes to the hyperplane are chosen using SVM. The SVM method is built on support vectors, which are utilized to represent these extreme conditions. Have a look at figure 11.1 that distinguishes two distinct groups by using a decision boundary or hyperplane:

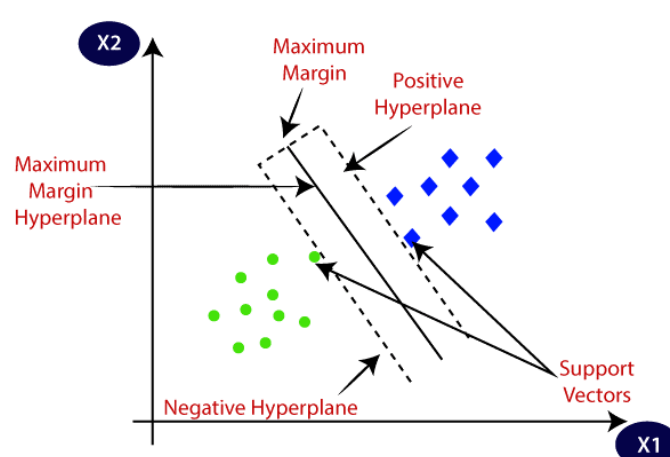


Fig. 11.1 Support Vector Machine (SVM) (Support Vector Machine (SVM) algorithm – javatpoint, 2023)

SVM may be understood using the KNN classifier example that we presented earlier. To develop a program that can accurately distinguish between a cat and a dog, let's say that we come across a strange cat which also mimics a dog. We can build such a model using the SVM method. Prior to actually testing it with this alien species, we would first teach our machine using various pictures of both cats and dogs to get it acclimated to the distinct traits of cats and dogs. Thus, the support vector will be able to recognise the extreme cases of cats and dogs when it creates a dividing line between each data set (cat and dog) i.e., shown in figure 11.2. The classification as a cat will be made using the support vectors.

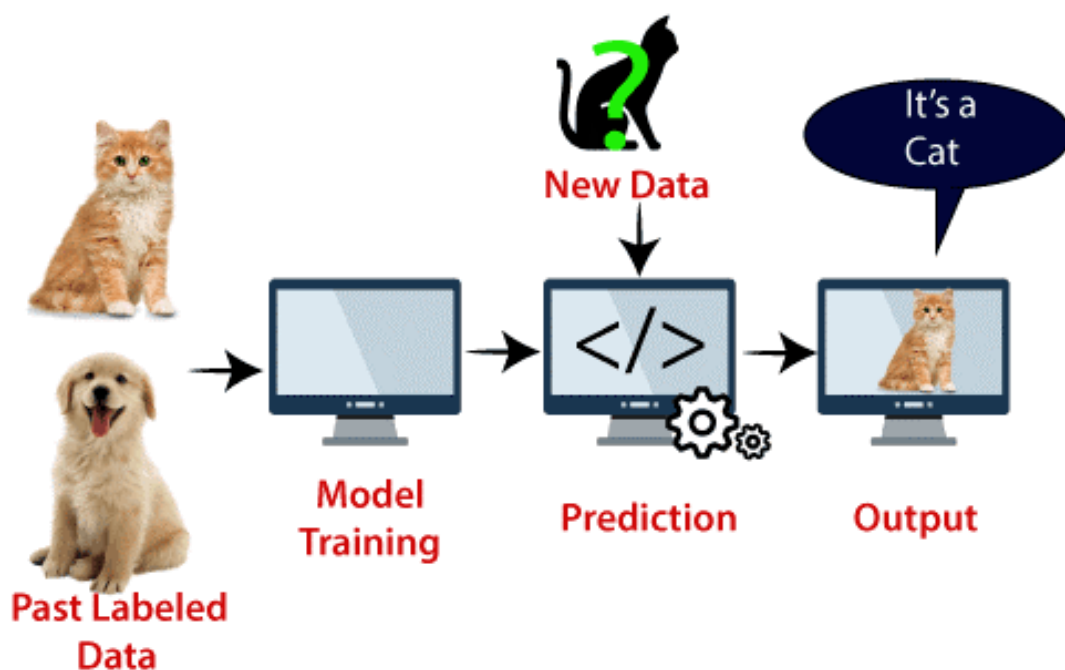


Fig.11.2 SVM Example

For example, text categorization, image classification, and face detection can all be done using the SVM algorithm.

11.1.1 Types of SVM

It is classified in two types:

- i. **Linear SVM:** When information that can be split b/w two categories using a linear line, or data that can be linearly separable, linear SVM is utilised. Under such circumstances, a Linear SVM classifier is employed.

- ii. **Non-linear SVM:** It is used for data that has been split into non-linear categories; non-linearity is the inability to classify a dataset by using a straight line. These kinds of datasets are employed by non-linear SVM classifiers.

11.1.2 Hyperplane and Support Vectors in the SVM algorithm:

It is necessary to identify the ideal decision boundary for classifying the data points. In n -dimensional space, many arcs or decision borders may be utilised to divide classes. This ideal boundary is known as the SVM hyperplane. A solid line will be the hyper-plane when there are only two characteristics because the dataset's features specify its dimensions (as in the illustrated image). (Ray, S, 2019)

In addition, if it has three major characteristics, the hyper-plane will only have two dimensions. Every time, constructing a hyper-plane with the highest margin/boundary, or the widest gap here b/w data points.

11.1.3 Support Vectors:

Support vectors are the information points or vectors that are nearer to the hyper-plane and have the greatest influence on its location. These vectors are referred to as support vectors since they aid the hyper-plane.

11.1.4 How does SVM works?

Linear SVM:

The operation of the SVM algorithm (Support Vector Machine (SVM) algorithm – javatpoint, 2023) can be better understood by an example. Consider a dataset shown in figure 11.3 that has two categories of instances represented in colours blue and green. Each category is represented by two features (x_1, x_2). The dataset is given a geometric representation in figure 11.3 and SVM will tend to find the hyperplane as a straight line to provide a maximum marginal separation between two categories.

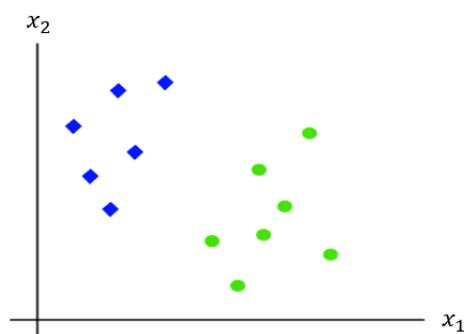


Fig. 11.3 Linear SVM (Support Vector Machine (SVM) algorithm – javatpoint, 2023)

We can simply distinguish between these two classes because it is a 2D space and can just draw a linear/non-curve line. However, these classes/groups may be splitted by more than 1 line as given in figure 11.4.

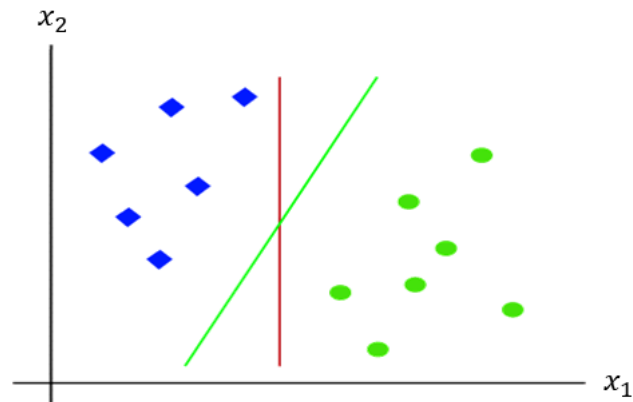


Fig.11.4 SVM Hyperplane (Support Vector Machine (SVM) algorithm – javatpoint,2023)

Hence, the SVM method assists in locating the best decision boundary or area, frequently referred to as a hyperplane. The nearest vector from each group is identified by the SVM algorithm. Support-vectors are the names for these points. The margin/line is the separation between the vectors and the hyper-plane. SVM aims to upright this margin/line. The hyperplane using the biggest margin is the optimum one (Support Vector Machine algorithm, 2022). In our case, slopped line has the maximum margin therefore, it will be used as separating hyperplane as shown in figure 11.5.

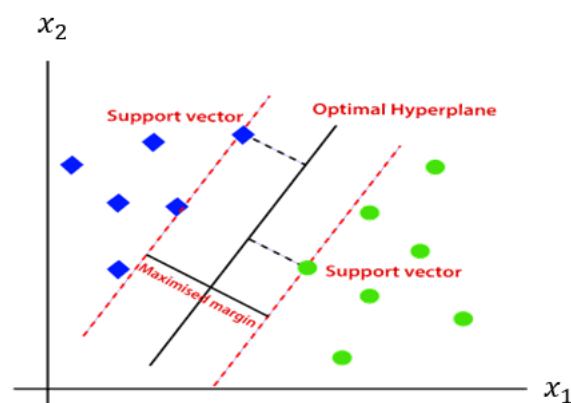


Fig. 11.5 SVM Hyperplane (Support Vector Machine (SVM) algorithm – javatpoint, 2023)

Non-Linear SVM:

Data that is structured linearly can be divided using a straight line, however non-linear data cannot be separated by linear SVM. Look at the data dispersion in figure 11.6. It cannot be separated by a linear hyper plane.

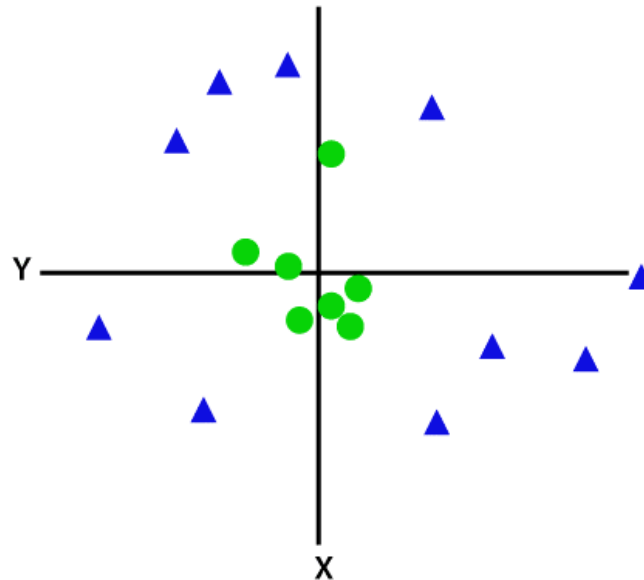


Fig.11.6 Non-linear SVM (Support Vector Machine (SVM) algorithm – javatpoint, 2023)

Hence, to segregate we need to include another dimension to these data values. We used the two-dimensional x and y for linear data, and the three-dimensional z for non-linear data. This is known as transformation to higher dimensional space.

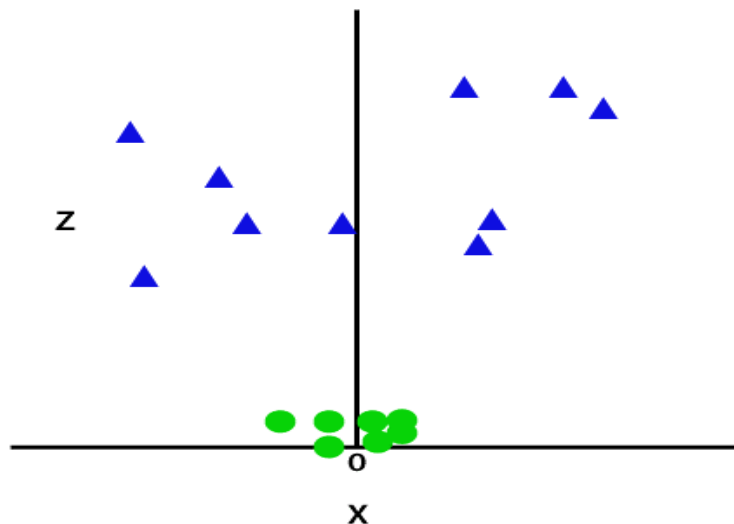


Fig. 11.7 Non-linear SVM (Support Vector Machine (SVM) algorithm – javatpoint, 2023)

The third dimension is calculated using equation (11.1).

$$z = x^2 + y^2 \quad (11.1)$$

After transforming to higher dimension, the plot looks like figure 11.7.

The datasets/information will now be split/divided as shown in figure 11.8

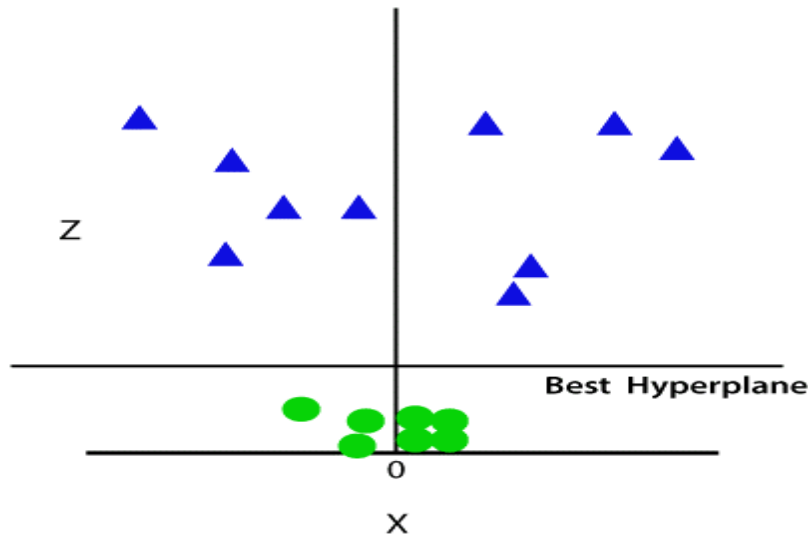


Fig. 11.8 Non-linear SVM Hyper-plane (Support Vector Machine (SVM) algorithm – javatpoint, 2023)

We are all in 3D, therefore it seems to be a plane along to the x- axis. If we translate it into 2D space/region with ($z = 1$), it will take the form as shown in figure 11.9.

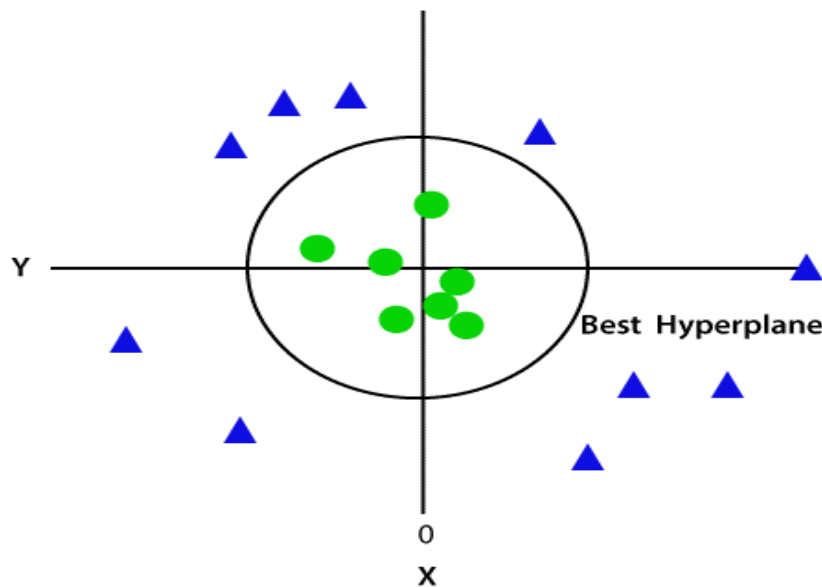


Fig. 11.9 Non-linear SVM Hyperplane (Support Vector Machine (SVM) algorithm – javatpoint, 2023)

Hence, when dealing with non-linear data/information, we find a circle of radius 1 as a hyperplane.

11.2 Optimal Separation

Optimal separation refers to finding the best possible division or arrangement of elements into distinct groups based on specific criteria. This can be used in various fields, such as engineering, economics, and machine learning, to maximize efficiency, productivity, or accuracy. The concept of optimal separation is dependent on the context and the objective being optimized. It refers to the process of finding the best or most efficient way to divide a set of items into groups or classes based on certain criteria. It aims to maximize the difference between the groups or classes, while minimizing the number of misclassified items. Optimal separation is commonly used in fields such as machine learning, data analysis, and image processing. (Ray, S. ,2019. A quick review of machine learning algorithms)

11.3 Kernels

Support vector machines (SVM), convolutional neural networks (CNN), and other machine learning methods require kernels, which are mathematical functions. Moving the raw data together into higher dimensional space, where simpler separation is feasible, is the purpose of a kernel's function. As a result, the model can recognise intricate non-linear correlations between the data and the desired outcomes. Kernels can be of various common sorts, such as linear, polynomial, sigmoid, and radial basis function (RBF). The data's structure and the issue being solved will determine which kernel is used.

11.4 Extensions to the Support Vector Machine

There are several extensions to the basic Support Vector Machine (SVM) algorithm, including:

Kernel SVM: This extension allows for non-linearly separable information to be transformed into a higher dimensional region(space) where it becomes linearly separable.

Multi-class SVM: This extension enables SVM to handle classification problems with more than two classes.

Regression SVM: This extension is used for regression problems, where the goal is to predict a continuous value instead of a class label.

Online SVM: This extension deals with problems where the training data arrives in a stream and needs to be processed incrementally.

Sparse SVM: This extension is designed to handle large sparse datasets more efficiently.

Cost-sensitive SVM: This extension allows for the specification of different misclassification costs for different classes.

Fuzzy SVM: This extension is designed to handle uncertainty and vagueness in data.

Each of these extensions is designed to address specific problems and improve the performance of the basic SVM algorithm. (Support Vector Machine algorithm ,2022 GeeksforGeeks)

Pros of SVM:

- Effective in situations with large dimensions.
- Its cognitive efficiency depends on the decision functions that use the subsets of training points i.e., called support vectors.
- For decision processes, various kernel functions can be formulated.

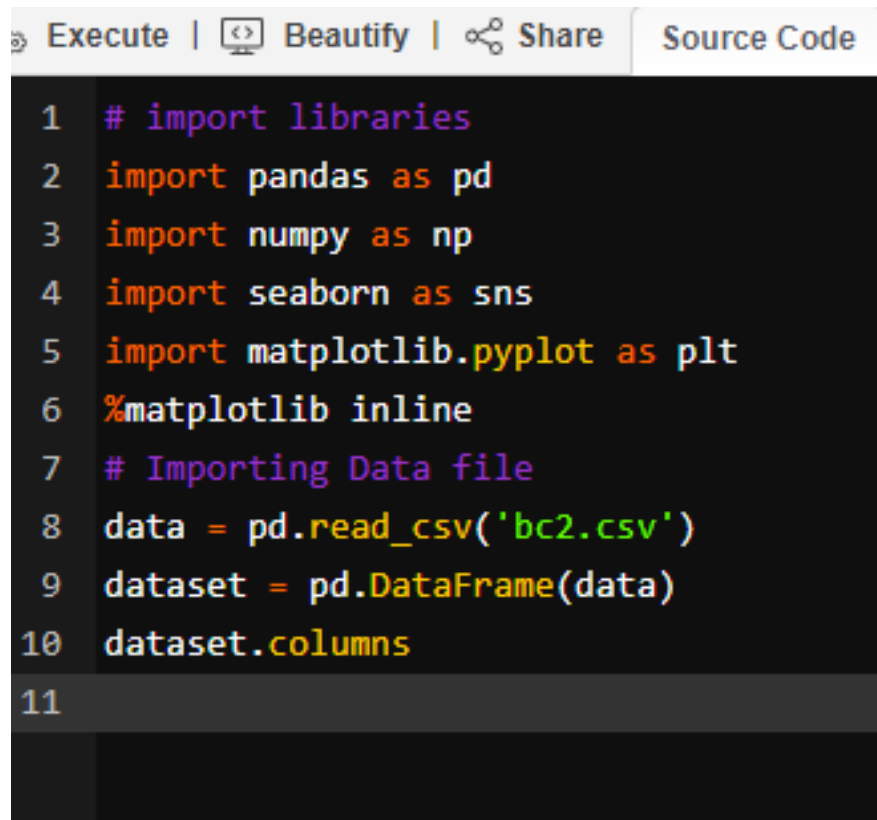
11.4.1 Python implementation of SVM:

Aim: Determine whether the tumour is benign or dangerous as the goal.

Give the physicians the ability to distinguish between tumours and benign cases provided the independent characteristics by using past data about patients identified with tumours. (Support Vector Machine (SVM) algorithm – javatpoint, 2023)

Dataset: [http://www.archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+\(Original\)](http://www.archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original))

For the implantation of SVM for the analyses of tumours, firstly we have to import the data of patients. Figure 11.10 shows the coding for importing the dataset in python. It gives the following output after the execution of this code.



```
1 # import libraries
2 import pandas as pd
3 import numpy as np
4 import seaborn as sns
5 import matplotlib.pyplot as plt
6 %matplotlib inline
7 # Importing Data file
8 data = pd.read_csv('bc2.csv')
9 dataset = pd.DataFrame(data)
10 dataset.columns
11
```

Fig. 11.10 Python implementation code

Output:

```
Index(['ID', 'ClumpThickness', 'Cell Size', 'Cell Shape', 'Marginal
Adhesion',
'Single Epithelial Cell Size', 'Bare Nuclei', 'Normal Nucleoli',
'Bland Chromatin',
'Mitoses', 'Class'], dtype='object')
```

Python

```
dataset.info()
```

Table 11.1 Data Set Info

	count	mean	std	min	25%	50%	75%	max
ID	699	1.07E+06	617095.7298	61634	870688.5	1171710	1238298	13454352
clump Thickness	699	4.42E+00	2.815741	1	2	4	6	10
Cell Size	699	4.42E+00	2.815741	1	1	1	5	10
Cell Shape	699	3.13E+00	3.051459	1	1	1	5	10
Marginal Adhesion	699	2.806867e+00	2.971913	1	1	1	4	10
Single Epithelial cell size	699	3.22E+00	2.855379	1	2	2	4	10
Normal Nucleoli	699	3.437768e+00	2.2143	1	2	3	5	10
Bland chromatin	699	2.866953e+00	2.438364	1	1	1	4	10
Mitoses	699	1.589413e+00	3.053634	1	1	1	1	10
class	699	2.69E+00	1.715078	2	2	2	4	4

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 699 entries, 0 to 698
Data columns (total 11 columns):
ID                                699 non-null int64
ClumpThickness                    699 non-null int64
Cell Size                         699 non-null int64
Cell Shape                       699 non-null int64
Marginal Adhesion                699 non-null int64
Single Epithelial Cell Size      699 non-null int64
Bare Nuclei                      699 non-null object
Normal Nucleoli                  699 non-null int64
Bland Chromatin                  699 non-null int64
Mitoses                         699 non-null int64
Class                           699 non-null int64
dtypes: int64(10), object(1)
memory usage: 60.1+ KB
```

Python

```
dataset.describe().transpose()
```

```
dataset = dataset.replace('?', np.nan)
dataset = dataset.apply(lambda x: x.fillna(x.median()),axis=0)

# converting the hp column from object 'Bare Nuclei'/ string type to
float
dataset['Bare Nuclei'] = dataset['Bare Nuclei'].astype('float64')
dataset.isnull().sum()
```

Output:

```
ID                                0
ClumpThickness                    0
Cell Size                          0
Cell Shape                        0
Marginal Adhesion                 0
Single Epithelial Cell Size       0
Bare Nuclei                       0
Normal Nucleoli                   0
Bland Chromatin                   0
Mitoses                           0
Class                             0
dtype: int64
```

```

from sklearn.model_selection import train_test_split

# To calculate the accuracy score of the model
from sklearn.metrics import accuracy_score, confusion_matrix

target = dataset["Class"]
features = dataset.drop(["ID", "Class"], axis=1)
X_train, X_test, y_train, y_test = train_test_split(features, target,
                                                    test_size = 0.2, random_state = 10)
from sklearn.svm import SVC

# Building a Support Vector Machine on train data
svc_model = SVC(C= .1, kernel='linear', gamma= 1)
svc_model.fit(X_train, y_train)

prediction = svc_model .predict(X_test)
# check the accuracy on the training set
print(svc_model.score(X_train, y_train))
print(svc_model.score(X_test, y_test))

```

Output:

```

0.9749552772808586
0.9642857142857143

```

Python

```

print("Confusion Matrix:\n",confusion_matrix(prediction,y_test))

```

Output:

```

Confusion Matrix:
[[95  2]
 [ 3 40]]

```

```
# Building a Support Vector Machine on train data
svc_model = SVC(kernel='rbf')
svc_model.fit(X_train, y_train)
```

Output:

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='rbf', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

```
print(svc_model.score(X_train, y_train))
print(svc_model.score(X_test, y_test))
```

Output:

```
0.998211091234347
0.9571428571428572
```

References:

Ray, S. (2019, February). A quick review of machine learning algorithms. In *2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon)* (pp. 35-39). IEEE.

Support Vector Machine algorithm (2022) GeeksforGeeks. GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/support-vector-machine-algorithm/> (Accessed: March 10, 2023).

Support Vector Machine (SVM) algorithm - javatpoint (2023) [www.javatpoint.com](https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm). Available at: <https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm> (Accessed: March 27, 2023)