As per the New Credit System Syllabus (2020 Course) of
Savitribai Phule Pune University w.e.f. academic year 2023-2024

Machine Learning

(Code : 417521) (Compulsory Subject)

Semester VII - Artificial Intelligence and Data Science

Pravin Goyal

UNIT I

UNIT II

UNIT III

Introduction : Need of Classification, Types of Classification (Binary and Multiclass), Binary-vs-Multiclass Classification, Balanced and Imbalanced Classification Problems.

Binary Classification : Linear Classification model, Performance Evaluation- Confusion Matrix, Accuracy, Precision, Recall, F measures.

Multiclass Classification : One-vs-One and One-vs-All classification techniques, Performance Evaluation- Confusion Matrix, Per Class Precision, Per Class Recall

Classification Algorithms : K Nearest Neighbor, Linear Support Vector Machines (SVM) – Introduction, Soft Margin SVM, Kernel functions– Radial Basis Kernel, Gaussian, Polynomial, Sigmoid.

UNIT IV

Introduction : What is clustering, Need of Clustering, Types of Clustering Hierarchical clustering algorithms /connectivity-based clustering): Agglomerative Hierarchical Clustering (AHC) algorithm, Divisive Hierarchical Clustering (DHC) algorithm.

Centroid-based clustering algorithms / Partitioning clustering algorithms : K-Means clustering algorithm, Advantages and disadvantages of K-Means clustering algorithm, Elbow method, The Silhouette method, K-Medoids, K-Prototype.

Density-based clustering algorithms : DBSCAN algorithm, how it works, Advantages and disadvantages of DBSCAN.

Distribution-based clustering algorithms : Gaussian mixture model.

Application of Clustering Technique : Market Segmentation, Statistical data analysis, Social network analysis, Image segmentation, Anomaly detection.

UNIT V

Ensemble Learning : Introduction to Ensemble Learning, Need of Ensemble Learning, Homogeneous and Heterogeneous ensemble methods, Advantages and Limitations of Ensemble methods, Applications of Ensemble Learning.

Basic Ensemble Learning Techniques : Voting Ensemble, Types of Voting: Max Voting, Averaging, Weighted Average.

Advanced Ensemble Learning Techniques:

Bagging : Bootstrapping, Aggregation.

Boosting : Adaptive Boosting (AdaBoost), Gradient Boosting, XGBoost .

Stacking : Variance Reduction, Blending, Random Forest Ensemble, Advantages of Random Forest.

UNIT VI

Reinforcement learning : What is Reinforcement Learning? Need for Reinforcement Learning, Supervised vs Unsupervised vs Reinforcement Learning, Types of Reinforcement, Elements of Reinforcement Learning, Real time applications of Reinforcement learning.

Markov's Decision Process : Markov property, Markov chain/process, Markov reward process (MRP), Markov decision process (MDP), Return, Policy, Value functions, Bellman equation

Q Learning : Introduction of Q-Learning, Important terms in Q learning, Q table, Q functions, Q learning algorithm.

❑❑❑

1 Introduction to Machine Learning

Unit 1

Syllabus

At the end of this unit, you should be able to understand and comprehend the following syllabus topics:

- Introduction
  - What is Machine Learning
  - Definitions and Real-life applications
  - Comparison of Machine learning with traditional programming
  - ML vs AI vs Data Science
- Learning Paradigms
  - Learning Tasks - Descriptive and Predictive Tasks
  - Supervised
  - Unsupervised
  - Semi-supervised
  - Reinforcement Learnings
- Models of Machine learning
  - Geometric model
  - Probabilistic Models
  - Logical Models
  - Grouping and grading models
  - Parametric and non-parametric models
- Feature Transformation
  - Dimensionality reduction techniques
    - PCA
    - LDA

1.1 Introduction to Machine Learning

- What do you mean by learning? What do you mean when you tell someone to "learn" something? The dictionary meaning of the word learn is "to gain knowledge or understanding of or skill in by study, instruction, or experience".
- You, as a reader or student of a particular course, learn something and then become equipped or capable of carrying out various tasks based on what you learnt. You might have heard about Bloom's Taxonomy. Bloom's Taxonomy is a classification of the different objectives and skills that learners could achieve out of a particular learning or a course. Fig. 1.1.1 outlines Bloom's Taxonomy.

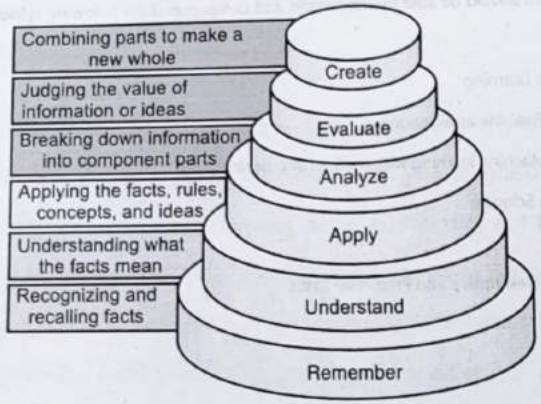


Fig. 1.1.1 : Bloom's Taxonomy

- So, as you understand, learning helps to gain skills and carry out various tasks. Like human beings, animals and other organisms also learn which helps them to carry out tasks required as per their lifecycle.
- You would also agree that, in general, the more you learn (through experience, courses, instructions, manuals, training, workshop or anything else) the more capable (and performant) you become at carrying out tasks based on learning objectives.

So, if

E = Experience

P = Performance for a given task

T = Task

- Then, you can mathematically say that performance at a given task is directly proportional to the experience. Isn't it?

$$P(T) \propto E$$

- As humans and other living organisms learn, similarly, it was thought, can machines made to learn and carry out tasks at a decent performance level?

- One of the most notable work towards Machine Learning (ML) was done by Alan Turing in 1950, yes 70 years ago! In his 1950 paper, "Computing Machinery and Intelligence," Alan Turing asked, "Can machines think?" (See http://www.csee.umbc.edu/courses/471/papers/turing.pdf for the full paper.).
- The paper describes the "Imitation Game", which involves three participants a human acting as a judge, another human, and a computer that is attempting to convince the judge that it is human. The judge would type into a terminal program to "talk" to the other two participants. Both the human and the computer would respond, and the judge would decide which response came from the computer.
- If the judge couldn't consistently tell the difference between the human and computer responses, then the computer won the game. The test continues today in the form of the Loebner Prize, an annual competition in artificial intelligence. The aim is simple enough: Convince the judges that they are chatting to a human instead of a computer chat bot program.
- In 1959, Arthur Samuel defined machine learning as, "[A] Field of study that gives computers the ability to learn without being explicitly programmed." Samuel is credited with creating one of the self-learning computer programs with his work at IBM. Samuel is widely known for his work in artificial intelligence.
- One the most common definition of Machine Learning came from Tom M. Mitchell, the Chair of Machine Learning at Carnegie Mellon University. It is as following.

✍ **Definition :** *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with the experience E.*

Hence,

✍ **Definition :** *Machine Learning is the study of computer algorithms and programs that automatically improve their performance, for a given set of tasks, with increase in their experience.*

- Machine learning is a branch of artificial intelligence (making machines intelligent). Machine learning uses various statistical (mathematical) models to learn from collected data and then uses those trained models to carry out various tasks such as predictions and classifications.

1.1.1 Learnability

- Learning is distinguished into a number of different forms. The simplest is learning by trial-and-error. For example, a simple program for solving mate-in-one chess problems might try out moves at random until one is found that achieves mate. The program remembers the successful move and next time the computer is given the same problem it is able to produce the answer immediately. The simple memorising of individual items, solutions to problems, words of vocabulary, etc. is known as rote learning.
- Rote learning is relatively easy to implement on a computer. More challenging is the problem of implementing what is called generalisation. Learning that involves generalisation leaves the learner able to perform better in situations not previously encountered. A program that learns past tenses of regular English verbs by rote will not be able to produce the past tense of e.g. "jump" until presented at least once with "jumped", whereas a program that is able to generalise from examples can learn the "add-ed" rule, and so form the past tense of "jump" in the absence of any previous encounter with this verb. Sophisticated modern techniques enable programs to generalise complex rules from data.

1.1.2 How does Machine Learning Work?

The Fig. 1.1.2 outlines how machine learning works at a high-level.

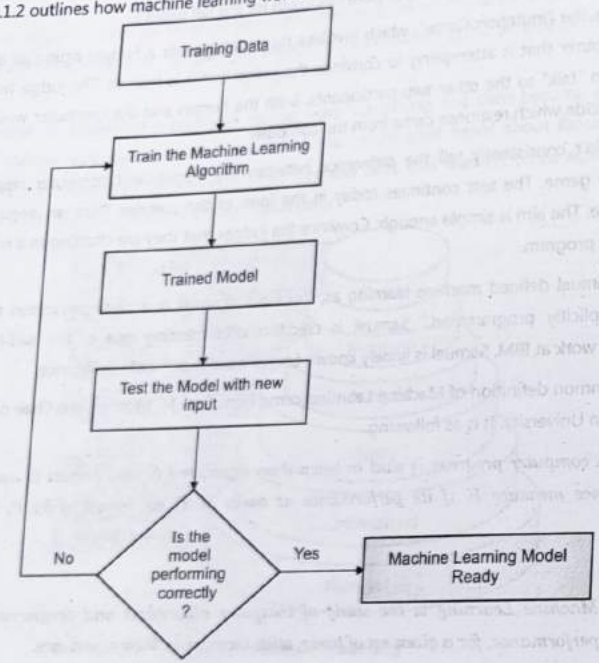
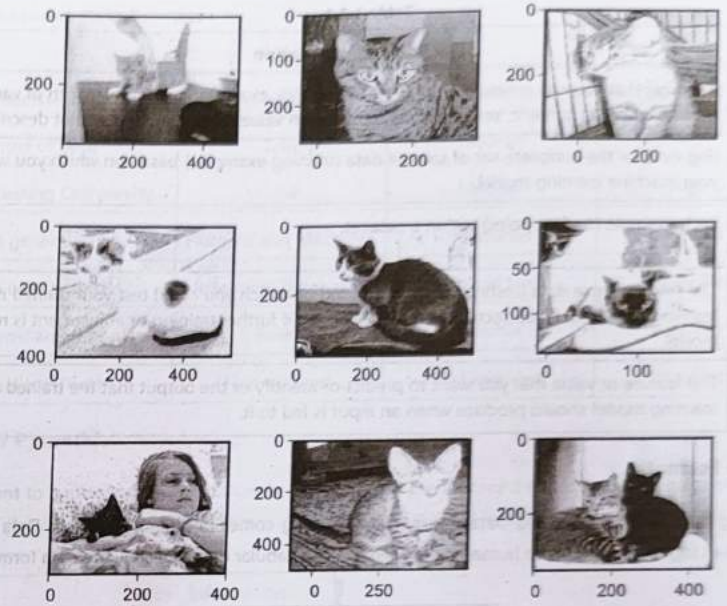


Fig. 1.1.2 : Machine Learning Model

- I am sure you would have made several predictions many a times in life already, isn't it? Whether it is going to rain, whether a team is going to win, whether the stock market would go up or down and likewise. Similarly, you can identify a cat pretty confidently even if it is of different shape and size and is in almost any position (sleeping, jumping, turning upside down, running, eating or whatever).
- How do you do these things? You can do this because you have some sort of mental model trained from your past experiences (and learning's). Based on those experiences, you can easily predict facts or identify things. If it occurs to you that what you predicted or identified was not right, you take that feedback and update your mental model so that you can do a better job next time. Probably, you are less likely to make the same mistake. Unfortunately, machines aren't as intelligent as you are, but we take the similar approach to make them intelligent!
- Machine learning is all about using a huge volume of training (sample) data (also called as big data analysis) and then using that data to build models that can carry out a set of tasks when a similar but new input is provided.
- For example, if you want a machine to automatically identify a cat from various animal pictures fed to it, you first train the machine to recognise different types of cats and also in various positions and environments. For example, assume that the following picture has the training data for identifying cats in a picture.



- Once the model has learnt "enough", it can then successfully carry out a task such as identifying a cat in a picture that was not in the training data. For example, it can identify the cat in the following picture.

- That's a very high-level view of how machine learning works. However, there are many more complications and effort required for building and training a complex machine learning model for real-life applications. But, a general high-level understanding of how machine learning works would go a long way to help you build the complex concepts later on.

1.1.3 Key Terms Associated with Machine Learning

Let's understand some of the key terms associated with machine learning that you would commonly encounter.

Table 1.1.1

Term	Description
Features or Attributes	Anything that you can measure and build data for. For example, the typical length of various animals. Feature could be numeric, set of characters, Boolean values, or anything else that describes the data.
Training Dataset	(Big data) or the complete set of sample data (training examples) based on which you would train your machine learning model.
Training Example	Each example (or data point) within a dataset.
Testing Dataset	The set of sample data (testing examples) based on which you could test your trained machine learning model for its correctness and determining if further training or adjustment is required to the model.
Target variable	The feature or value that you want to predict or identify or the output that the trained machine learning model should produce when an input is fed to it.

1.1.4 Data Formats

The training data (also called as Big Data) for machine learning comes in different formats. Data can be machine generated (such as log files) or could be human generated (such as tabular data). Overall, the data format is classified as the following.

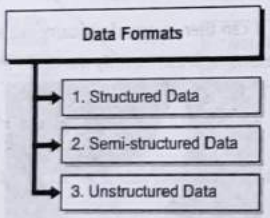


Fig. 1.1.3 : Data Formats

1. **Structured Data :** Structured data exhibits a particular order (also known as model or schema) for storing and working with the data. The data attributes are usually related and are often the basis of analysis. The structured data is usually generated by machines or compiled by humans. For example, spreadsheets, customer records, transaction records, sales reports, etc. are all structured data. The structured data is usually stored in relational databases or simple CSV or spreadsheet files.
2. **Semi-structured Data :** Semi-structured data has some definitive patterns for storage, but the data attributes may not be inter-related. The data could be hierarchical or graph-based in nature. The semi-structured data is usually stored in text files as XMLs, JSON or YAML format. The common sources for semi-structured data is usually machines such as sensors, website feeds, or other application programs.
3. **Unstructured Data :** Unstructured data does not exhibit a fixed pattern or a particular schema. This is the most common format of Big Data. Examples of unstructured data are video, audio, tweets, likes, shares, text documents, PDFs, and scanned images. Special tools and mechanisms are required to process unstructured data. Also, it is usually cleaned (sanitised) before it can be used for analysis.

Comparison between Data Formats

Table 1.1.2

Comparison Attributes	Structured Data	Semi-structured Data	Unstructured Data
Volume of Data	Low	Medium	High
Processing Complexity	Low	Medium	High
Data generated by	Humans and Machines	Machines	Humans
Data usually stored in	Relational Databases	Textual files	Binary files
Patterns and Schema	Fixed	Flexible	Random
Specialised Tools	Not required	Not required	Required

1.1.5 DIKW Pyramid

It is important to understand how data can be enriched and the journey it takes with each stage of enrichment. To understand this journey, typically DIKW Pyramid is referenced. DIKW is an acronym for the four stages of data enrichment that are

(a) Data (b) Information
(c) Knowledge and (d) Wisdom

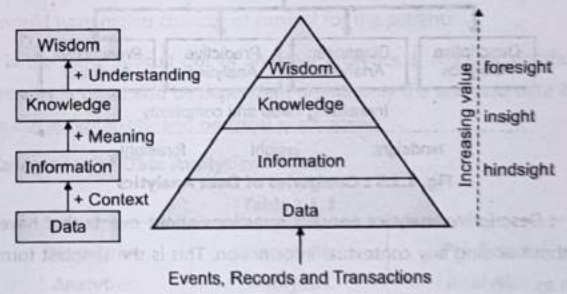


Fig. 1.1.4 : DIKW Pyramid

1. **Data :** This is the lowest bottom of the DIKW pyramid. This is the raw data collected from various events, records, and transactions around you. The data could be generated by machines or by humans. The data itself does not have very high value until it is enriched with more attributes that can be used for further analysis. For example, you could just have a data set that has list of millions of people with their demographic details and how they died. This data may not give you anything actionable.
2. **Information :** At the next level, the collected data is enriched with the context to give information. You start to build perception about the data to give you hindsight. The hindsight about the information reflects or acknowledges what is contained in the data. For example, you could begin to see that the list of people is actually the list of cancer patients and their life patterns.

3. **Knowledge :** When you add meaning to the information, you start to gain knowledge. This is where you precisely start analysing the information at hand and make it more useful and meaningful. You could gain deep insights about the information and be able to answer high-level questions. For example, you can analyse the information on cancer patients and build patterns around life expectancy after cancer detection with or without chemotherapy. You could further analyse the effect of various chemotherapy medicines to understand their dosage and effectiveness level. A company could then invest in building more effective chemotherapy medicines to improve life expectancy of cancer patients after diagnosis. So, understand here that the objectives of data analysis must be clear to derive meaningful knowledge from the information at hand.

4. **Wisdom :** The final level of wisdom is achieved when you add understanding to the derived knowledge. Note here that wisdom is not achieved using a technical algorithm or formula but is based on the human understanding of the data analysis that was carried out. For example, after analysis of data on cancer patients, you could understand what lifestyle to follow in terms of diet, sleep, and exercise to avoid or delay occurrence of cancer. This understanding could be shared with the world as foresight.

As you see, the DIKW pyramid helps to put a perspective on visualising the data analytics stages. Now let's take this discussion further and understand the categories of data analytics.

1.1.6 Categories of Data Analytics (Learning Tasks-Descriptive and Predictive Tasks)

Now that you have a fair understanding of what Big Data is, let's touch upon the spectrum of analytics that is possible on the given data sets. The different types of analytics require different tools and techniques.

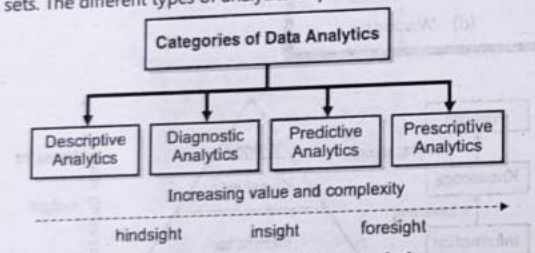


Fig. 1.1.5 : Categories of Data Analytics

1. **Descriptive Analytics :** Descriptive analytics answers questions about events that have already occurred. Usually raw data is queried without adding any contextual information. This is the simplest form of analytics and typically answers questions such as

   (a) How many units of a particular item sold in last 6 months?

   (b) How many patients died of a particular cancer type?

   (c) How many calls did you receive for a particular issue?

   This kind of analytics is usually done using database queries or simple spreadsheet filters, You could have periodic dashboards and reports that can be used to visualise results of the descriptive analytics.

2. **Diagnostic Analytics :** Diagnostic Analytics is done to find out cause of a phenomenon or derive reasoning behind events. This analytics goes a level deeper to provide information that can be used to fix a particular situation or event. Diagnostic analytics usually adds more context to the data to get information about a particular interest. For example, following are a few questions that can be answered using diagnostic analytics.

   (a) Why the sales in quarter 2 lower than quarter 1?

   (b) Why are people falling ill after eating a particular type of biscuits ?

   (c) Why the model X of the car preferable over the model Y of the car ?

   Diagnostic analytics require careful examination of data from multiple sources and is a little more involved and skilful exercise than descriptive analytics.

3. **Predictive Analytics :** Predictive analytics is carried out to forecast and predict future events. The information is further enriched by adding meaning to it to derive knowledge. The predictive data models are carefully created that can base off future predictions based on the past events. Predictive analytics could possibly answer questions such as

   (a) What would be the improved life expectancy if choosing medicine A over medicine B ?

   (b) What would be the sales figure for model X of the car in third quarter ?

   (c) Which team would likely win the world cup this year ?

   Predictive analytics assumes that certain set of conditions are met or would exist. If there are changes to those conditions, then predictive analytics may not be accurate.

4. **Prescriptive Analytics :** Prescriptive analytics takes the results from predictive analytics and further adds human judgment to prescribe or advise further actions. This reflects the wisdom level from the DIKW pyramid that you learnt earlier. The prescriptive analytics could answer questions such as

   (a) What should you do to delay cancer?

   (b) What is the best time to leave home to reach airport on time?

   (c) Which medicine would have higher chances of survival for the patient?

   Prescriptive analytics is the most difficult out of all other analytics. It requires significant skills and time to give effective actions and results. It could also be dependent on not only the analysed data but external conditions such as political pressure, social acceptability, and personal preferences.

Comparison between Categories of Data Analytics

Table 1.1.3

Comparison Attribute	Descriptive Analytics	Diagnostic Analytics	Predictive Analytics	Prescriptive Analytics
Complexity	Least	Medium	High	Highest
Time requirement to produce results	Low	Medium	High	Very High
Value of results	Short Term	Medium Term	Long Term	Very long term
Data enrichment level	Data	Information	Knowledge	Wisdom
Analytics Frequency	Most Common	Frequent	Not often	Rare

1.2 Types of Machine Learning (Learning Paradigms)

As a human learn in various ways (auditory, visual, kinesthesis), so does machines. You also understand that there could be different types of data formats and various analytics and learning requirements. At a high-level, machine learning algorithms could be categorized as following.

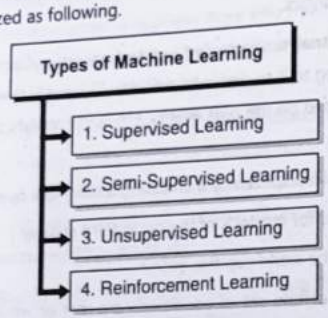


Fig. 1.2.1 : Types of Machine Learning

1.2.1 Supervised Learning

✍ ***Definition :*** *Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs.*

- In supervised learning, the machine learning algorithm is trained on labelled data (the training dataset has both input as well as output). This is very similar to you teaching a toddler by showing a picture of an apple and saying, "Look kid, this is an apple". Fig. 1.2.2 provides a high-level outline of how supervised machine learning algorithm work.

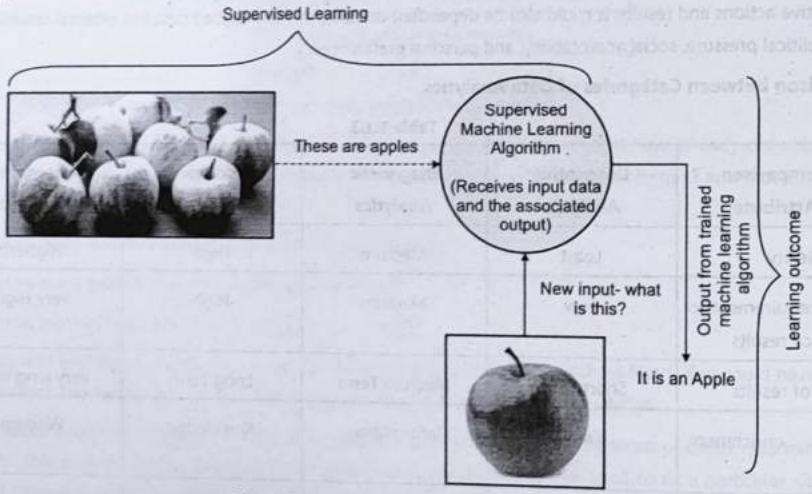


Fig. 1.2.2 : Supervised Machine Learning

- The training data set has various examples that contain both input (features) as well as output (target). Supervised learning is typically used for classification or predicting a particular value (regression). For example, classifying an animal picture as a cat or a dog or whether it is likely to rain. Some of the common supervised learning algorithms are k-Nearest Neighbours, Linear regression, Logistic regression, Naïve Bayes, Support Vector Machines, and Decision Trees.

1.2.2 Semi-Supervised Learning

- Semi-supervised learning is a class of machine learning techniques that make use of both labelled and unlabelled examples when learning a model.
- In one approach, labelled examples are used to learn class models and unlabelled examples are used to refine the boundaries between classes. This approach combines a small amount of labelled data with a large amount of unlabelled data during training.
- Semi-supervised learning falls between unsupervised learning (with no labelled training data) and supervised learning (with only labelled training data). It is a special instance of weak supervision.
- For a two-class problem, you can think of the set of examples belonging to one class as the positive examples and those belonging to the other class as the negative examples. In the Fig. 1.2.3, if you do not consider the unlabelled examples, the dashed line is the decision boundary that best partitions the positive examples from the negative examples. Using the unlabelled examples, you can refine the decision boundary to the solid line. Moreover, you can detect that the two positive examples at the top right corner, though labelled, are likely noise or outliers.

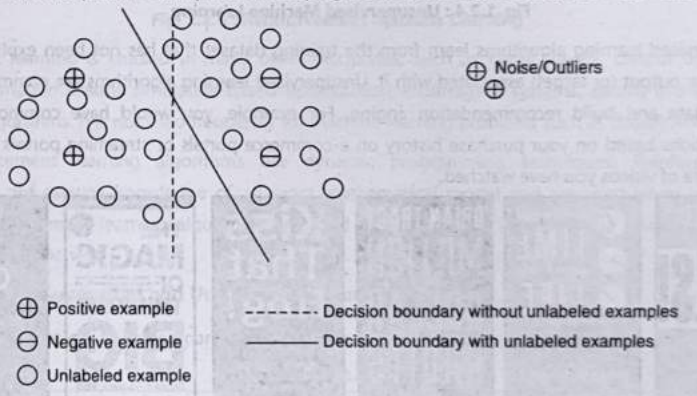


Fig. 1.2.3 : Semi-Supervised Learning

1.2.3 Unsupervised Learning

✍ ***Definition :*** *Unsupervised learning algorithms take a set of data that contains only inputs and automatically find structure in the data in order to group or arrange them in a cluster.*

- In unsupervised learning, once the model is given a training dataset, it automatically finds patterns and relationships in the dataset by creating clusters (or groups) within it. Unsupervised learning algorithms, however, cannot label the cluster or determine what those created clusters might mean.

- For example, if you feed a training dataset having thousands of pictures of cats, dogs, and monkeys, an unsupervised learning algorithm can potentially create three different clusters – one for each animal but it cannot tell you which cluster is what. It is up to the observer or human to make the sense out of the created clusters or data.
- When you feed a new input to such an algorithm, it can place it in one of the already created clusters or could create a new group if it is dissimilar to any of the already created clusters.
- Fig. 1.2.4 provides a high-level outline of how unsupervised machine learning algorithm work.

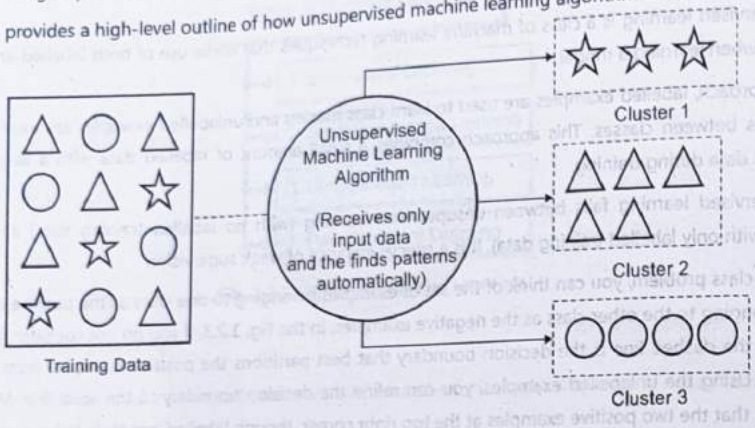


Fig. 1.2.4 : Unsupervised Machine Learning

- Thus, unsupervised learning algorithms learn from the training dataset that has not been explicitly labelled and neither has the output (or target) associated with it. Unsupervised learning algorithms are commonly used to find patterns of data and build recommendation engine. For example, you would have commonly seen various recommendations based on your purchase history on e-commerce portals or streaming portals such as YouTube based on genre of videos you have watched.

Fig. 1.2.5

- Some of the common unsupervised learning algorithms are k-Means clustering, Expectation maximization, Hidden Markov Model, DBSCAN, and Parzen window.

1.2.4 Reinforcement Learning

- Treat reinforcement learning as learning from mistakes. The reinforcement learning algorithms work very similar to how you learn by yourself without any guidance – basically through hit and trial. When you get something right, you get a reward, you feel happy, and you move ahead. When you get something wrong, you get a penalty, you take a step back, and then you try to avoid the incorrect path while exploring another correct path.

✍ **Definition :** *Reinforcement machine algorithms improves upon themselves and learn from new situations using a trial-and-error method.*

- The favourable outputs are encouraged, or 'reinforced', and non-favourable outputs are discouraged or 'punished'. Fig. 1.2.6 outlines how reinforcement learning works at a high-level.

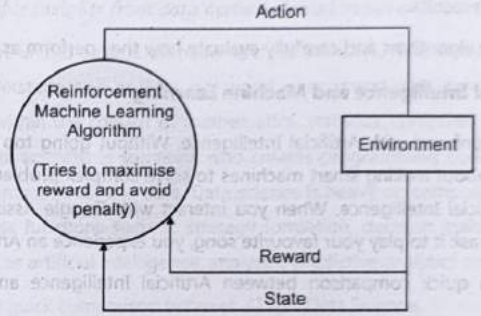


Fig. 1.2.6 : Reinforcement Machine Learning

- Reinforcement learning is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics, and genetic algorithms. It is not used frequently for machine learning problems such as classification or prediction.
- Many reinforcement learning algorithms use dynamic programming techniques. Reinforcement learning algorithms do not assume knowledge of an exact mathematical model and are used when exact models are infeasible. Reinforcement learning algorithms are used in autonomous vehicles or in learning to play a game against a human opponent.

Comparison between Supervised and Unsupervised Learning

The Table 1.2.1 provides a quick comparison between supervised and unsupervised learning.

Table 1.2.1

Comparison Attribute	Supervised Learning	Unsupervised Learning
Training Dataset Contains	Both input and output	Only input
Used for	Classification and Prediction	Finding patterns and understanding data
Training Data	Is Labelled	Not labelled
Number of targets	Known beforehand	Not known
Feedback from user	Provided	Not provided
Complexity	High	Low

1.2.5 How to Choose the Right Machine Learning Algorithm?

- So, how do you choose which machine learning type to go with? It is simple – if you need to predict a target value, then you should choose supervised learning. For example, if you want to find out chances of raining tomorrow, a team winning a tournament, a picture being an apple or a cat, or problems such as these, then you should choose supervised learning method.
- However, if you are not solving a prediction or classification problem and your goal is to group the data and find "interesting" patterns, then you are better of using unsupervised learning.
- Once you have decided which type of machine learning algorithm you need, you then need to choose a suitable algorithm from that machine learning type. Note here that there is no single answer to what the best algorithm is or what will give you the best results.
- You will need to try different algorithms and carefully evaluate how they perform as per your requirements.

Comparison between Artificial Intelligence and Machine Learning

- Machine learning is often confused with Artificial Intelligence. Without going too much into details, understand that artificial intelligence is about making smart machines to solve complex problems similar to humans. Machine learning is a subset of Artificial Intelligence. When you interact with Google Assistant or Apple Siri or Alexa to check weather conditions or ask it to play your favourite song, you experience an Artificially built Intelligent system.
- The Table 1.2.2 provides a quick comparison between Artificial Intelligence and Machine Learning for your reference.

Table 1.2.2

Comparison Attribute	Machine Learning	Artificial Intelligence
Focus	Learn from data	Solve complex problems
Complexity	Low	High
Scope	Narrow	Broad
Human interaction	Minimal	High

Comparison between Data Mining and Machine Learning

- Data mining is a process used by companies to turn raw data into useful information. By using software to look for patterns in large batches of data, businesses can learn more about their customers to develop more effective marketing strategies, increase sales and decrease costs. However, there are significant differences between data mining and machine learning.
- The Table 1.2.3 provides a quick comparison between data mining and machine learning.

Table 1.2.3

Comparison Attribute	Machine Learning	Data Mining
Building a trained model	Required	Not required
Human effort required	Only for building model	For extracting information from data
Use of specific algorithms	Frequent	Rare

Comparison Attribute	Machine Learning	Data Mining
Accuracy	High	Low
Tasks carried out by	Machines	Humans
Self-learning	Yes	No

1.2.6 Comparison of AI with Data Science

✍ ***Definition :*** *Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from noisy, structured and unstructured data and apply knowledge and actionable insights from data across a broad range of application domains.*

- Data science as a concept unifies several domains such as statistics, data analysis, informatics, and their related methods in order to understand and analyse real-world phenomena with data. It uses techniques and theories drawn from many fields within the context of mathematics, statistics, computer science, information science, and domain knowledge. A data scientist is someone who creates programming code and combines it with statistical knowledge to create insights on business data. Data science is heavy on computer science and mathematics. Data science is used in business functions such as strategy formation, decision making and operational processes. It touches on practices such as artificial intelligence, analytics, predictive analytics and algorithm design.
- The Table 1.2.4 provides a quick comparison between AI and Data Science.

Table 1.2.4

Comparison Attribute	Artificial Intelligence	Data Science
Focus on	Making machines intelligence like humans	Extracting useful and actionable insights from data
Complexity	Very high	Medium
Useful applications	Limited, still evolving	Several
Response	In terms of voice or action	In terms of prediction or visuals

1.2.7 Comparison between Machine Learning and Traditional Programming

- As you know, computer programming is the process of performing a particular computation (or more generally, accomplishing a specific computing result), usually by designing and building an executable computer program. For example, if you want to display something specific on the screen, you use a programming language's syntax to display it on the screen. Machine learning, on the other hand, looks at the data and builds a model that can be used for making predictions.
- The Table 1.2.5 provides a quick comparison between machine learning and traditional programming.

Table 1.2.5

Comparison Attribute	Machine Learning	Traditional Programming
Task focus	Build prediction model	Automation
Works through	Machine learning techniques	Programming languages
Dependence on data	Very High	Low
Complexity	High	Low
Output	Machine learning model	Task results

1.3 Issues in Machine Learning

Some of the common issues or challenges involved in machine learning are as following.

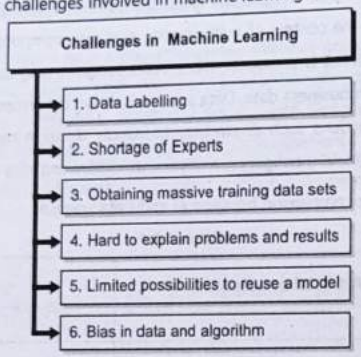


Fig. 1.3.1 : Challenges in Machine Learning

1. **Data Labelling** : Today, there is a massive amount of data that is unlabelled and raw. As you know, supervised machine learning works on labelled data. Without adequate data labels in the training dataset, it is not feasible to build robust machine learning models. Companies are putting thousands of man hours to label data so that it can be appropriately used for machine learning. This is an active area of research where the labels can be attached to the data as it is used. But today data labelling is far most one of the biggest challenges in machine learning.
2. **Shortage of Experts** : Machine learning is an emerging field and there are not many experts around the world. You require experts who can
   (a) Understand the wide variety of data
   (b) Model the data correctly so as to meet the desired objectives
   (c) Build and manage software and hardware tools and techniques required for machine learning
   (d) Design appropriate visual interfaces and
   (e) Also communicate the findings effectively

   Building a team of experts that have all the required capabilities is challenging and time consuming.

3. **Obtaining massive training data sets** : As you understand, the more intensive is the training dataset the better and robust is the trained machine learning model. However, it is difficult to obtain massive training datasets for various areas and applications. You may lack historical data. Also, the quality of data for the training dataset matters. If the dataset obtained does not represent a fair sample size, then the resultant machine learning models could be erroneous.

   For example, if you are trying to build a machine learning model to predict a particular type of cancer from a given set of symptoms, lifestyle, and blood related parameters, then you would require quality data for thousands of patients that have had that particular type of cancer and the details of their symptoms, lifestyle, and blood related parameters. You would not be able to build a robust and trained machine learning model without having a decent sized and high-quality training data set. If there are areas that are new, you may not have the training dataset available for it.
4. **Hard to explain problems and results :** Larger and more complex machine learning models make it hard to explain, in human terms, why a certain decision was reached (and even harder when it was reached in real time).

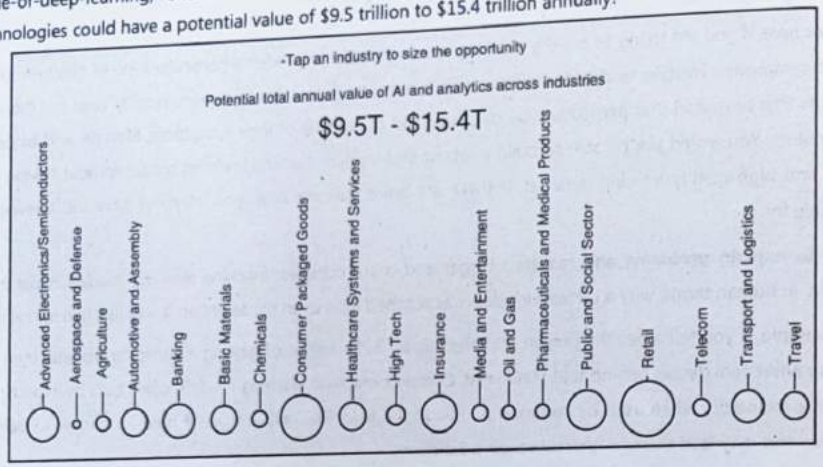
   For example, if you tell a healthy person that she has an 80% chance of getting a particular disease, then she may require additional details behind that statement. Complex machine learning models, often built by experts, may not be self-explanatory when used by common people in the field. This might make it hard for people to believe such systems and may lead them to question their predictions.
5. **Limited possibilities to reuse a model :** Unlike humans, who can apply their learnings from one circumstance to another circumstance, machine learning models are built specifically per use case. It is difficult to reuse an existing machine learning model for other use cases or solving problems in even related domains. Hence, companies have to invest time and resources to build new model for solving new use cases.
6. **Bias in data and algorithm :** What you read is what you learn, isn't it? Haven't you heard your parents and teacher say that learn from right sources and do not trust everything you get to read. Similarly, the training datasets use to build machine learning models could have hidden biases either knowingly or unknowingly. You will agree for sure if I tell you that our society does have its own space for prejudices and biases and not every decision made is fair to every group. This bias can creep-in into the training dataset and make the machine learning models biased towards a particular group, decision, or societal element.

   The process and frequency of data collection itself could be uneven across groups and observed behaviours which could further distort how algorithms analyse that data, learn, and make predictions. Negative consequences can include misinformed recruiting decisions, misrepresented scientific or medical prognoses, distorted financial models and criminal-justice decisions, and several other predictions and actions. In many cases, these biases go unrecognised or disregarded as they are believed to be beyond general human understanding and treated as "advanced data sciences" , "proprietary data and algorithms," or "objective analysis".

1.4 Application of Machine Learning

- There is hardly any area or domain today that does not use machine learning (or artificial intelligence based on machine learning) in some or the other ways. So, it is hard to just talk about one or more applications of machine learning.

- But, let me first give you some perspective that would help you to understand what machine learning has made possible for human life. The following snapshot from McKinsey's report, available at https://www.mckinsey.com/featured-insights/artificial-intelligence/notes-from-the-ai-frontier-applications-and-value-of-deep-learning, shows that Artificial Intelligence (based on machine learning) and surrounding technologies could have a potential value of $9.5 trillion to $15.4 trillion annually!



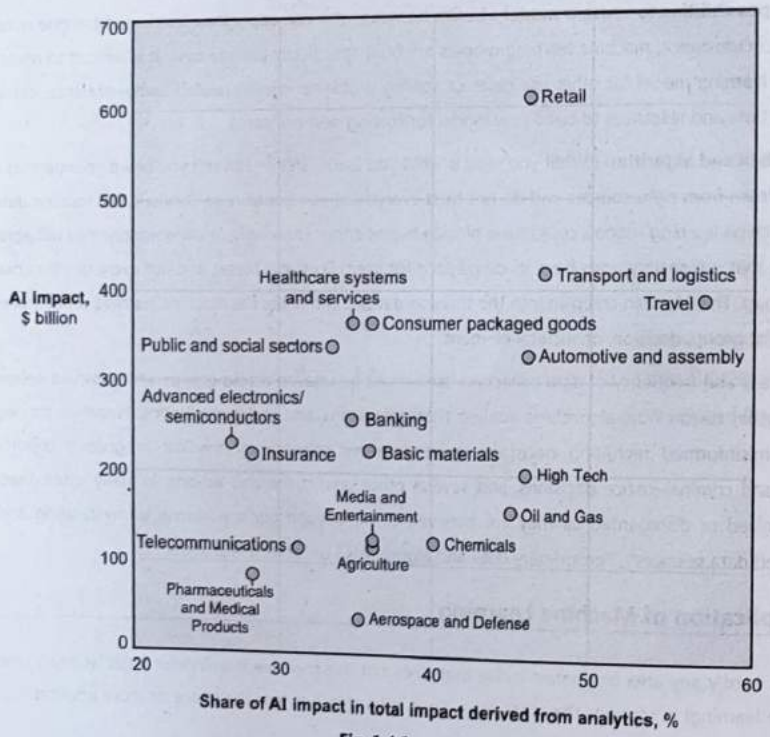
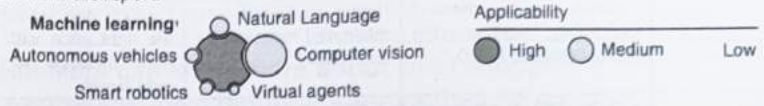


Fig. 1.4.1

- In each of these areas, Artificial intelligence can create value across the value chain in four ways as highlighted in the following snapshots from the report.



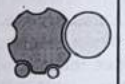
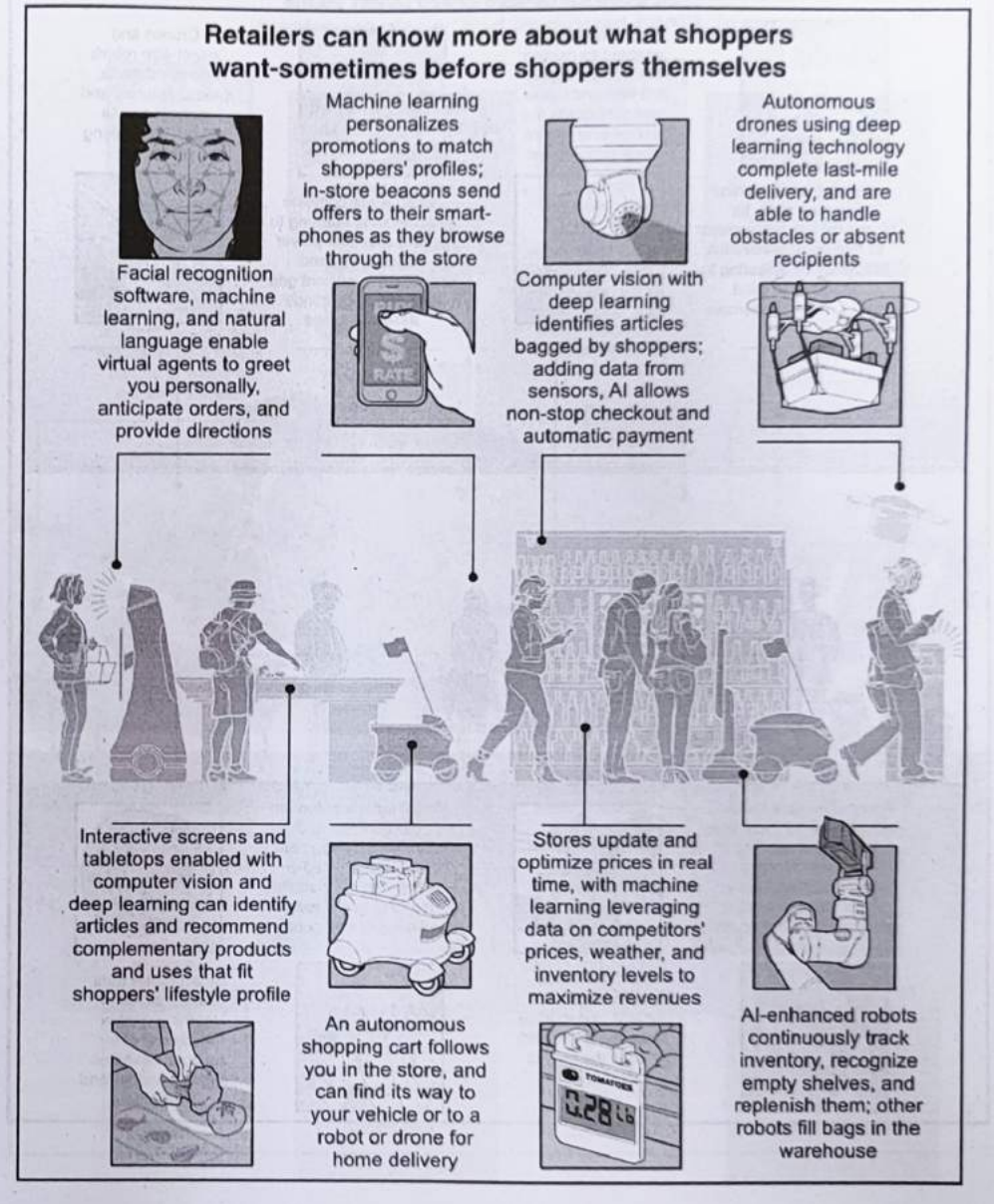
	Applicable technologies	Project	Produce	Promote	Provide
		Enlightened R and D, real-time forecasting, and smart sourcing	Operations with higher productivity, lower cast, and better efficiency	Products and services at the right price, with the right message, and to the right targets	Enriched, tailored, and convenient user experience
Retail		Anticipate demand trends, while optimizing and automating supplier negotiation and contracting	Automate warehouse and store operations; optimize merchandising, product assortment, and microspace	Optimize pricing, personalize promotions, and tailor website displays in real time	Personalize tips and suggestions, offer immediate assistance with virtual agents, automate in-store checkout, and complete last-mile delivery by drones
Electric utilities		Enhance demand and supply prediction, assess reliability of integrated generation assets, and automate demand-side response	Optimize preventive maintenance, improve electricity production yield, reduce energy waste, and prevent electricity theft	Optimize pricing with time-of-day and dynamic tariffing; match producers and consumers in real time	Automate supplier selection, provider consumption insight, automate customer service with virtual agents, and tailor usage to consumer's preferences
Manufacturing		Improve product design yield and efficiency, automate supplier assessment, and anticipate parts requirements	Improve processes by the task, automate assembly lines, reduce errors, limit product rework, and reduce material delivery time	Predict sales of maintenance services, optimize pricing, and refine sales-leads prioritization	Optimize flight planning and rout and fleet allocation; enhance maintenance engineer and pilot training
Health care		Predict disease, identify high-risk patient groups, and launch prevention therapies	Automate and optimize hospital operations; automate diagnostic tests and make them faster and more accurate	Predict cost more accurate, focus on patients' risk reduction	Adapt therapies and drug formulations to patients, use virtual agents to help patients navigate their hospital journey
Education		Anticipate job market demand, identify new drivers of performance to assess students, and help graduates highlight their strengths	Automate teachers' routine tasks, identify early disengagement signs, and optimize group formation for learning objectives		Personalize learning, shift from stop-and-test model to continuous learning cadenced by virtual coaches and tutors, and build student self-awareness

Table 1.4.1

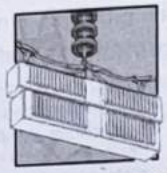
	Project	Produce	Promote	Provide
	Accurate demand forecasting, smart sourcing and enlightened R & D	Higher productivity and minimized maintenance and repairs	Products and services at the right price, with the right message to the right targets	Enriched, tailored and convenient user experience
Retail	• 1-2% EBIT improvement using machine learning to anticipate fruit and vegetable sales • 20% stock reduction using deep learning to predict e-commerce purchases. • 2 million fewer product returns per year	• 30% reduction of stocking time using autonomous vehicles in warehouses.	• 50 % improvement of assortment efficiency • 4-6 % sales increase using geospatial modelling to improve micro market attractiveness • 30% online sales increase by using dynamics pricing and personalization.	-
Electric utilities	• Objective to cut 10% in national electricity usage by using deep learning to predict power demand and supply	• 20% energy production increase using machine learning and smart sensors to optimize assets yield. • 10-20 % EBIT improvement by using machine leaning and smart sensors to optimize assets yield	-	• $10-$30 saving on monthly bills using machine learning to automatically switch electricity supply deals
Manufacturing	• 10% yield improvement for integrated using AI to improve R & D process • 39% IT staff reduction by using AI to fully automate procurement processes	• 30% increase of material delivery time using machine learning to determine timing of goods transfer • 3-5% production yield improvement	• 13% EBIT improvement by using machine learning to predict sources of servicing revenues and optimize sales efforts	• 12% fuel saving for manufactures customers airlines by using machine learning to optimize flight routes
Healthcare	• $300 billion possible savings in the United States using machine learning tools for population health forecasting	• 30-50% productivity improvement for nurses supported by AI tools • Up to 2% GDP savings for operational efficiencies in developed countries	• 5-9 % health expenditure reduction by using machine learning to tailor treatments and keep patients engaged	• $2 trillion - $10 trillion savings globally by tailoring drugs and treatments. • 0.2-1.3 additional years of average life expectancy.
Education	-	• Virtual teaching assistance can answer 40% of students routine questions	• 1% increases in enrolment by using virtual assistant to follow up to applicants	• 85 % match with human grading, using machine leanjng and predictive modelling



AI can make the smart grid smarter and reduce the need for utilities to add power plants

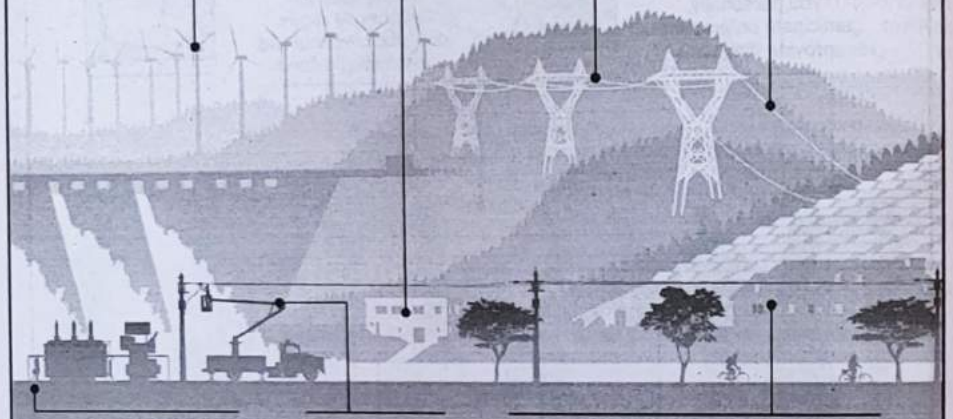
Sensors and machine learning allow for by-the-minute adjustments to maximize generation efficiency by adjusting to changes in wind conditions, for example

Machine learning-enabled forcasting anticipates supply and demand peaks, and maximizes the use of intermittent renewable power

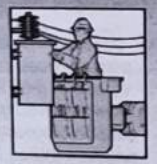


Smart wires combine with machine learning to enable real-time power dispatching, and optimize it to current grid load and to buildings' asset portfolios

Drones and insect-size robots identify defects, predict failures, and inspect assets without interrupting production



Few technicians remain, but they spend more time on problem solving; in place of logging inspection status by hand, documents are automatically logged and routed



Field workforce receives real-time updates to decrease response times and reduce the impact of outages

Virtual agents automate call centers, and automatically segment consumers based on service history; machine learning offers early warning of bad debts

Smart-meter data and machine learning enable utilities to offer services based on usages, weather and other factors

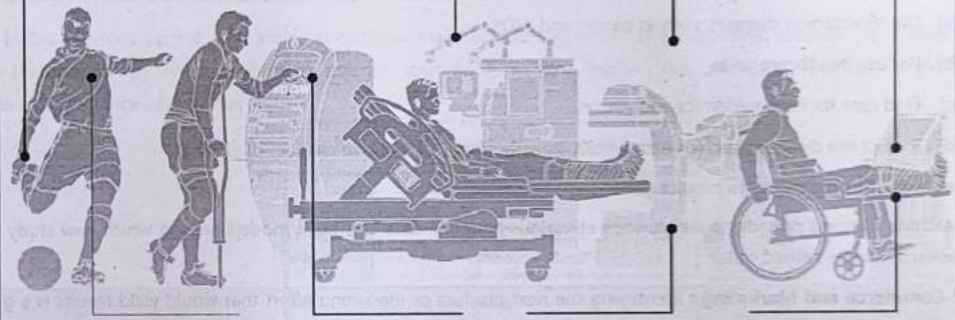
AI in health care; quicker diagnoses, better treatment plans, and improved health insurance

Machine learning program analyzes patients' health remotely via mobile device, compares it to medical records, and recommends a fitness routine or warns of possible disease

Autonomous diagnostic devices using machine learning and other AI techno-logies can conduct simple medical tests without human assistance, relieving doctors and nurses of routine activities

AI-powered diagnostic tools identify disease faster and with greater accuracy, using historical medical data and patient records

AI algorithms optimize hospital operation, staffing schedules, and inventory by using medical and environmental factors to forecast patient behavior and disease probabilities



AI tools analyze patients' medical histories and environmental factors to identify people at risk of an illness and steer them to preventive care programs

Virtual agents in the form of interactive kiosks register patients and refer them to appropriate doctors, improving their experience and reducing waiting time

Personalized treatment plans designed by machine learning tools improve therapy efficiency by tailoring treatment to specific patients' needs and medical

AI insights from population health analyses give payers an opportunity to reduce hospitalization and treatment costs by encouraging care providers to manage patients' wellness

- Alright, so you got some perspective now, isn't it? That is what machine learning can potentially do!
- Today, machine learning serves various purposes. However, some of the most common application of machine learning are as following.

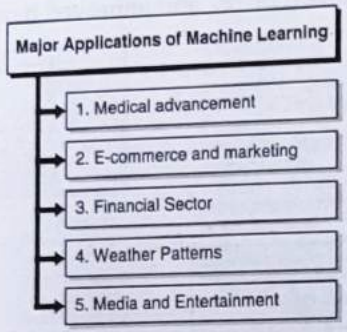


Fig. 1.4.2 : Applications of Machine Learning

1. **Medical advancement :** The healthcare industry is one of the biggest proponents of machine learning. Large data sets are used for research and operations in various areas such as
   (a) Life threatening diseases such as cancer and AIDS
   (b) Reduce healthcare costs
   (c) Find cure for new epidemics and viruses
   (d) Predict the diseases, such as Alzheimer's, in early stages to improve chances of survival
   (e) Genetic sequencing to predict genes-based diseases

   Machine learning provides a way to look at various patient's data and build models around which new study and research can be carried out.
2. **E-Commerce and Marketing :** Identifying the next product or marketing effort that would yield results is a great way for companies to build adequate strategies. Consumer patterns, their likes, their purchase potential, time of the year when the users buy, market trends, global economy etc. are all accounted to come out with the new products and services. The consumer's data is analysed to identify patterns and make predictions about the future demands. For example, if you are shopping early stage pregnancy products, it is likely that you would purchase infant's products soon. Companies can use this information to push new schemes and offers to you and also equip their warehouses to ship baby products faster to you by placing the inventory closer to you.
3. **Financial Sector :** Machine learning is also heavily used in the financial sector. Some of the common applications of machine learning in financial sector are
   (a) Detecting frauds. For example, if you made a physical transaction in Delhi and then another physical transaction in Mumbai within one hour, it is highly unlikely that you could have made both the transactions.
   (b) Credit score analysis based on your past reputation to pay dues
   (c) Quoting customised insurance premiums based on your lifestyle
   (d) Selling new products and services to you. For example, if you have surplus balance in your account throughout the year, then you could be sold investment products.

Apart from these, the financial sector also uses machine learning for detecting money laundering, shell companies, fraudulent transactions, and reporting. Based on the transactions carried out by individuals, the companies can build financial health profile of its consumers and identify future spend patterns and requirements.

4. **Weather Patterns :** Machine learning is crucial for detecting changes in the weather patterns. You would have heard about
   (a) The rising ocean temperatures (b) Global warming
   (c) Melting glaciers in Antarctica (d) Reducing Oxygen level

   There is a huge amount of data that can be used to predict the weather changes and report how it is affecting our environment. It can be used to predict weather forecast, natural disasters and any other changes that could affect our well-being.
5. **Media and Entertainment :** The media and entertainment industry use machine learning to understand viewing and liking patterns for the media content. Based on the time of the day, season, device you are on, your personal interests and taste, the content can automatically be recommended for you. I am sure you would have seen YouTube recommendations as you watch YouTube videos. Similarly, companies like Spotify can automatically create curated and customised playlists for you based on your listening profile.

1.5 Steps in Developing a Machine Learning Application

In the previous section, you learnt about how machine learning works at a high-level. Extending the discussion from there, at a high-level, developing a machine learning application involves the following steps. Each of the steps could be a sequence of activities in itself.

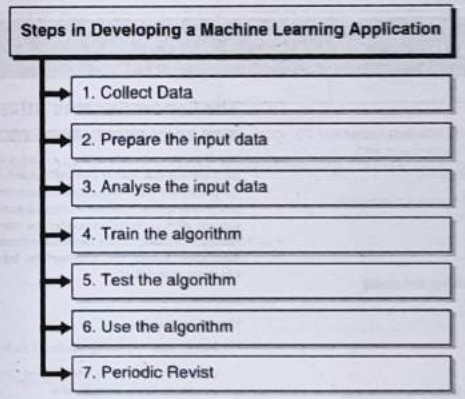


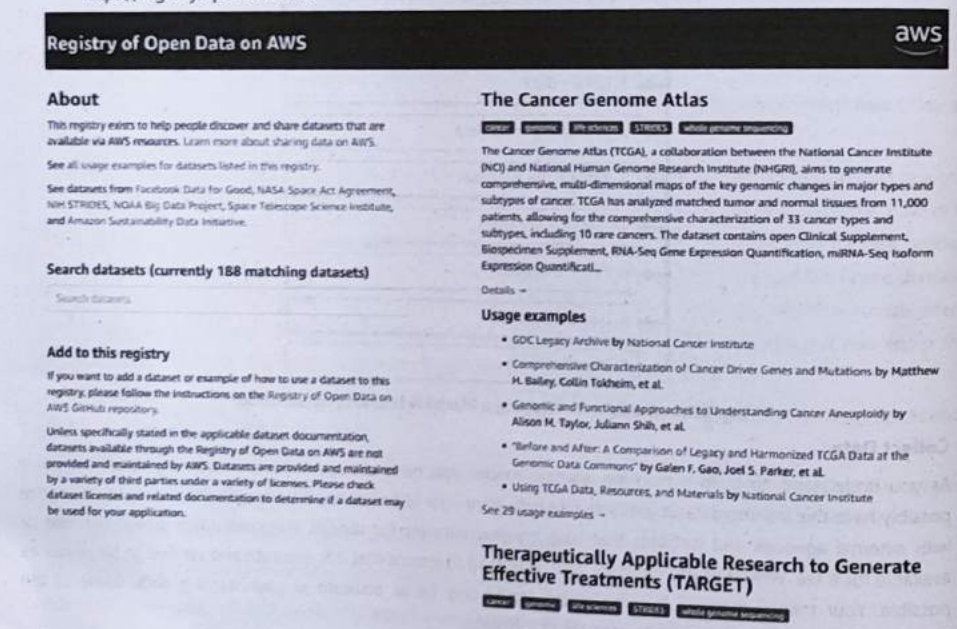
Fig. 1.5.1 : Steps in Developing a Machine Learning Application

1. Collect Data

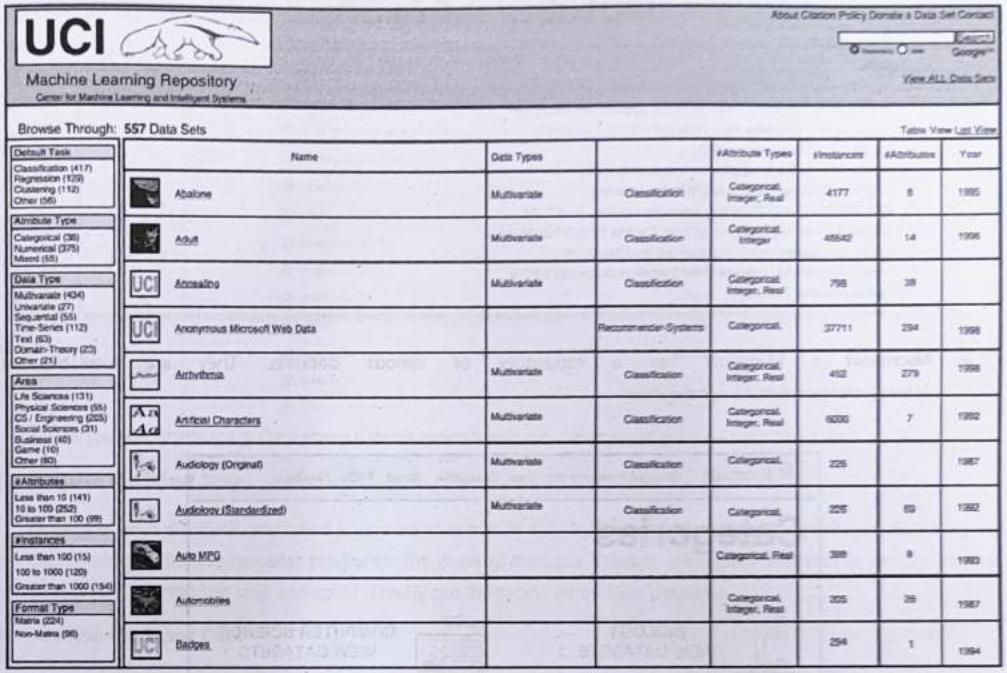
As you understand, to build a machine learning model, you need a huge volume of training data. You could possibly have this training dataset available internally from your historical business operations or you could engage with external agencies and websites that have training datasets for various purposes either freely available or available for a fee. Wherever you get data from, you need to ensure that it is accurate and verified to be as real as possible. Your trained machine learning model would only be as accurate as your training data. Some of the popular, publicly available, dataset sources are as following.

a. **Kaggle Dataset :** As of Sep 2020, Kaggle has 54,876 datasets on various topics. They are available on https://www.kaggle.com/datasets.

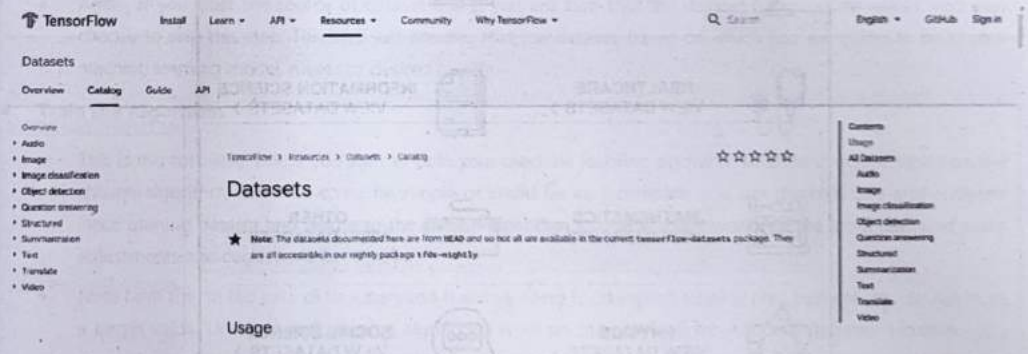
b. **Amazon Web Services (AWS) :** As of Sep 2020, AWS has 188 datasets on various topics. They are available on https://registry.opendata.aws.



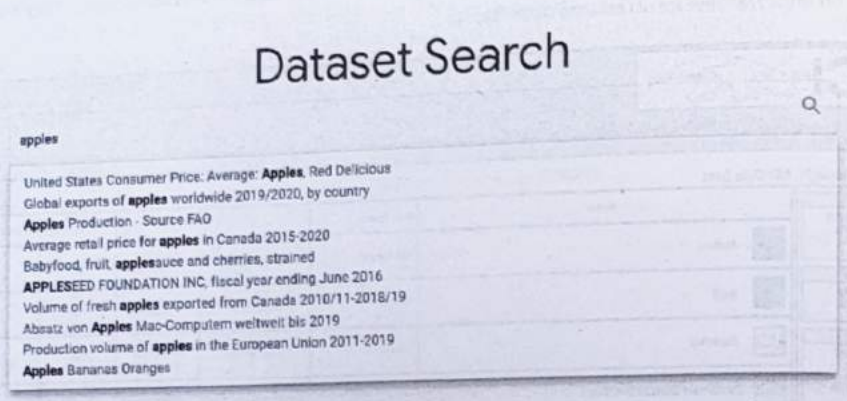
c. **UCI Machine Learning Repository :** As of Sep 2020, UCI has 557 datasets on various topics. They are available on https://archive.ics.uci.edu/ml/datasets.php.



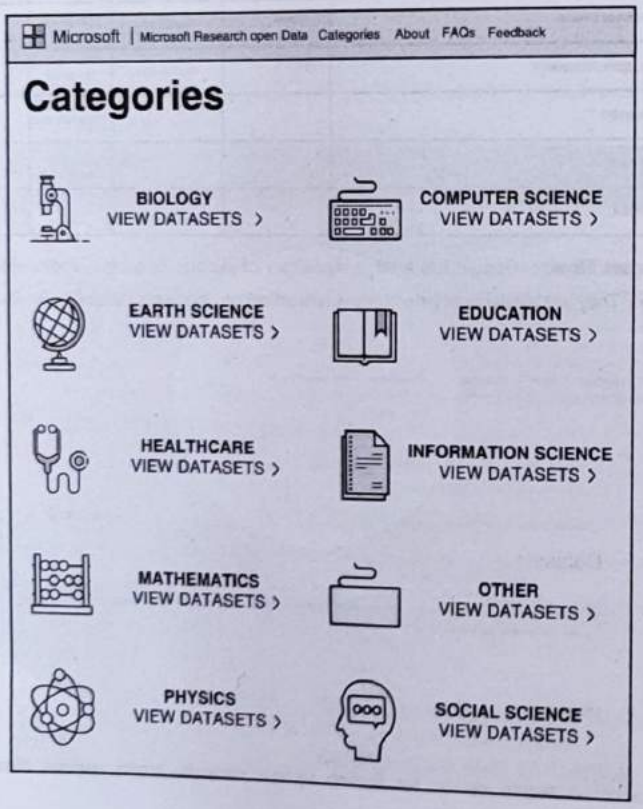
d. **Google's Tensor Flow :** Google has a huge repository of various datasets – audio, video, images, text, and various others. They are available at https://www.tensorflow.org/datasets/catalog/overview.



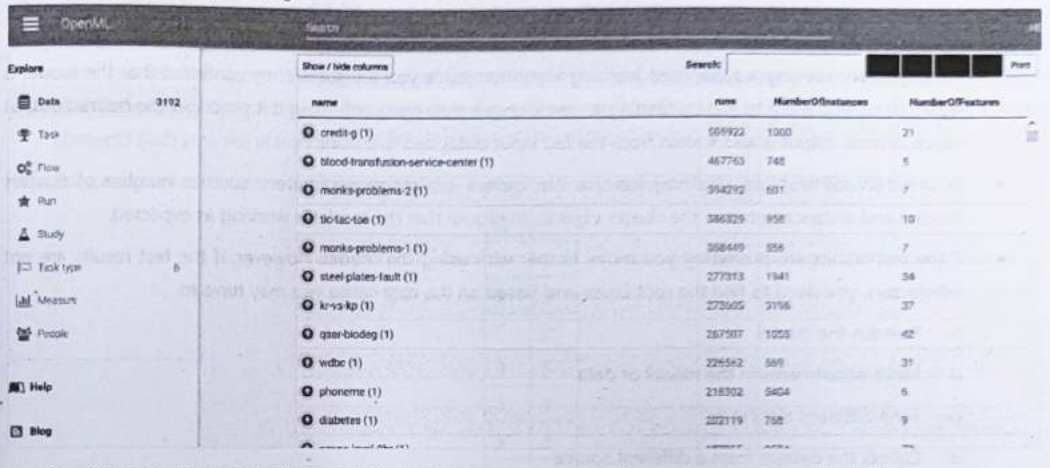
Google also provides a search option for finding various datasets across various dataset repositories at https://datasetsearch.research.google.com.



e. **Microsoft** : Microsoft has a repository of various datasets. They are available at https://msropendata.com/categories.



f. **OpenML** : As of Sep 2020, OpenML has 3192 datasets on various topics. They are available on https://www.openml.org.



So, as you see, there are quite a many dataset repositories around from where you can collect data.

2. Prepare the Input Data

Once you have the data, you need to ensure that it is in the right format such that it can be processed by the chosen algorithm and computer programs. The publicly available datasets are usually available in various formats so that you can skip this step and could directly use the procured training dataset.

3. Analyse the Input Data

- This is a crucial step where you need to ensure that the input dataset could be parsed properly for your chosen computer program. You also need to ensure that the examples are complete (they are not missing values) and are also not skewed (too high or too low compared to rest of the examples).
- Again, if you trust the source of dataset and if you are sure that the dataset has accurate values, you may choose to skip this step. This step just ensures that the dataset, based on which you are going to build your machine learning model, meet the desired quality.

4. Train the Algorithm

- This is the core step where you start to train your machine learning algorithm to build a model. Based on the chosen algorithm, this step could be simple or could be very complex. You use the collected and analysed input training dataset and feed it to the chosen algorithm to check how it works on the input data and make adjustments and corrections as required.
- Note here that in the case of unsupervised learning, there is no explicit training step because you do not have a target value. Unsupervised learning algorithms work on the provided input to find patterns. However, you may have to choose which features would you choose to feed the unsupervised learning algorithm so that the discovered patterns are meaningful for the purpose.

5. Test the Algorithm

- As a matter of practice, when you get a dataset, you partition it into 80-20, where 80% of the examples in th dataset are used to train the model and 20% of the examples in the dataset are used to test the trained mode When you are training a supervised learning algorithm, once you are sufficiently confident that the model well-trained, you put it to test by feeding it new known inputs and confirming if it produces the desired outpu (since desired output is also known from the fed input data).
- In unsupervised learning, you may have to use various evaluation parameters, such as number of cluste created and distance between the cluster objects, to ensure that the model is working as expected.
- If the test results are promising you move further with using the model. However, if the test results are no satisfactory, you need to find the root cause and based on the root cause you may have to
  - Re-train the model
  - Make adjustments in the model or data
  - Try a different algorithm
  - Collect the dataset from a different source
- Testing the algorithm before use is a crucial step to ensuring that your model does not product false result Do not skip it.

6. Use the Algorithm

You spent a lot of time collecting and cleaning the data and then building and testing the model. Once you ar through these steps, you are good to use the model and sit back and enjoy the success from your hard work. Yo may develop an application based out of the model. For example, based on someone's health parameters, you machine learning model could deduce health related problems that the person may face in near future. Based o someone's credit history, your application may infer the chances of a new loan getting approved. The application and usage are plenty as you learned earlier!

7. Periodic Revisit

As you often revise to ensure that your learning is still effective, you should periodically review the results that th model is producing and evaluate if there are opportunities for improving it in light of new data. You may carry ou minor adjustments to the model or may re-train it with latest data to fine tune it. This step is very similar to yo getting a master health check done for yourself annually to ensure that your body's vital parameters are doing well If any parameter indicates a potential problem, then you either make lifestyle changes or seek medical advice.

1.6 Models of Machine Learning

As you understand, models form the central concept in machine learning as they are what is being learned from the data, in order to solve a given task. There is a wide range of machine learning models to choose from based on th task at hand. You had previously learnt about various types of machine learning models based on the way they operate supervised, semi-supervised, unsupervised, and reinforcement. Now, let's learn about some of the core principles o domains on which such models of machine learning are based off. Note here that these groupings are not meant to be mutually exclusive, and sometimes a particular kind of model can be based off multiple principles or domains.

1.6.1 Geometric Models

- Machine learning models based on the principles of geometry fall under geometric models. The instance space is the set of all possible or describable instances, whether they are present in your data set or not. Usually this set has some geometric structure. For instance, if all features are numerical, then you can use each feature as a coordinate in a Cartesian coordinate system. A geometric model is constructed directly in instance space, using geometric concepts such as lines, planes and distances.
- For example, for the given data, you can plot a regression line to predict the value of Y if X = 10.

X	Y
0	1
1	3
2	2
3	5
4	7
5	8
6	8
7	9
8	10
9	12

Regression Line Plot

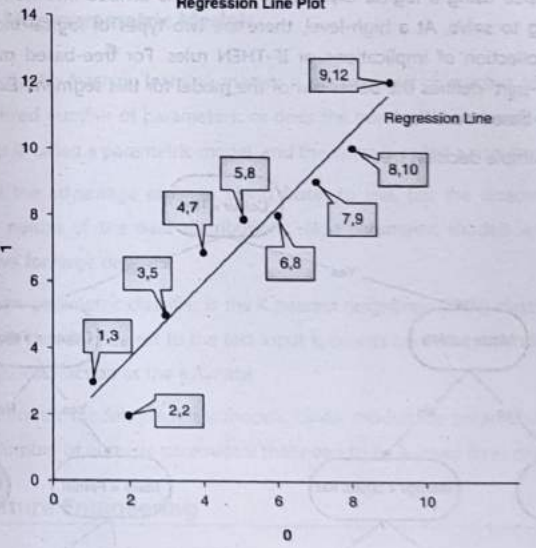


Fig. 1.6.1

- Some examples of geometric models are linear regression, k-means clustering, support vector machines, etc. You will learn about some of these later in the course.

1.6.2 Probabilistic Models

- Machine learning models based on the concepts of probability fall under probabilistic models. Typically, in probabilistic models, you have, say, X denote the variables that you know about, e.g., your instance's feature values, and Y denote the target variables you are interested in, e.g., the instance's class.
- The key question in machine learning is how to model the relationship between X and Y. The statistician's approach is to assume that there is some underlying random process that generates the values for these variables according to a well-defined but unknown probability distribution. You want to use the data to find out more about this distribution. Examples of probabilistic models are Naïve Bayes classifier and Markov model.
- For example, you could use the Markov model to predict the probability of a sunny day followed by a cloud day based on probability.

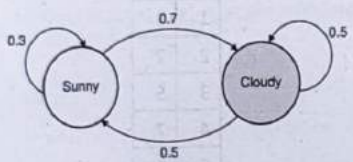


Fig. 1.6.2

1.6.3 Logical Models

- Logical models, drawing inspiration from computer science and engineering, use a logical expression to divide the instance space into segments. A logical expression returns a Boolean value, i.e., a True or a False outcome.
- Once the data is grouped using a logical expression, the data is divided into homogeneous groupings for the problem you are trying to solve. At a high-level, there are two types of logical models – trees and rules. Rule models consist of a collection of implications or IF-THEN rules. For tree-based models, the 'if-part' defines a segment and the 'then-part' defines the behaviour of the model for this segment. Examples of logical models are decision trees and rule-based classifier.
- The Fig. 1.6.3 shows a simple decision tree.

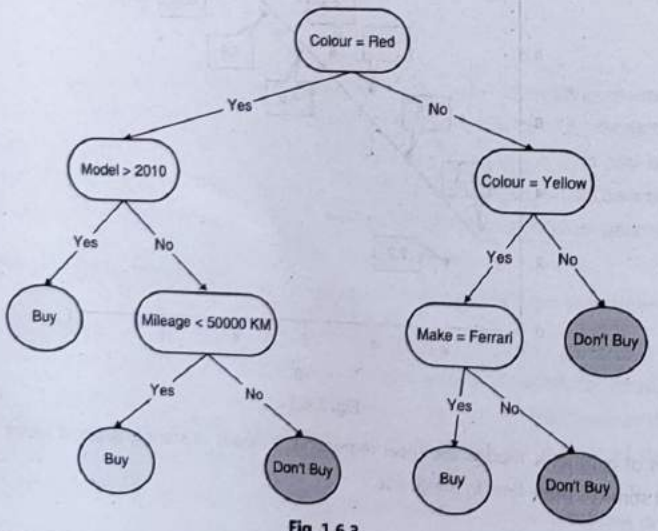


Fig. 1.6.3

1.6.4 Grouping and Grading Models

- Grouping models break up the instance space into groups or segments, the number of which is determined at training time. You could say that grouping models have a fixed and finite 'resolution' and cannot distinguish between individual instances beyond this resolution. What grouping models doat this finest resolution is often something very simple, such as assigning the majority class to all instances that fall into the segment. The main emphasis of training a grouping model is then on determining the right segments so that you can get away with this very simple labelling at the local segment level. A good example of grouping models are the tree-based models. They work by repeatedly splitting the instance space into smaller subsets. Because trees are usually of limited depth and don't contain all the available features, the subsets at the leaves of the tree, partition the instance space with some finite resolution. Instances filtered into the same leaf of the tree are treated the same, regardless of any features not in the tree that might be able to distinguish them.
- Grading models, on the other hand, do not employ such a notion of segment. Rather than applying very simple, local models, they form one global model over the instance space. Consequently, grading models are (usually) able to distinguish between arbitrary instances, no matter how similar they are. Their resolution is, in theory, infinite, particularly when working in a Cartesian instance space. Support vector machines and other geometric classifiers are examples of grading models. Because they work in a Cartesian instance space, they are able to represent and exploit the minutest differences between instances. As a consequence, it is always possible to come up with a new test instance that receives a score that has not been given to any previous test instance.

1.6.5 Parametric and Non-parametric Models

- One of the ways to classify the machine learning models is on the basis of number of parameters that they take. Does the model have a fixed number of parameters, or does the number of parameters grow with the amount of training data? The former is called a parametric model, and the latter is called a non-parametric model.
- Parametric models have the advantage of often being faster to use, but the disadvantage of making stronger assumptions about the nature of the data distributions. Non-parametric models are more flexible, but often computationally expensive for large datasets.
- A simple example of a non-parametric classifier is the K nearest neighbour (KNN) classifier. It simply "looks at" the K points in the training set that are nearest to the test input x, counts how many members of each class are in this set, and returns that empirical fraction as the estimate.
- A simple example of parametric models are linear models. Linear models are parametric, meaning that they have a fixed form with a small number of numeric parameters that need to be learned from data.

1.7 Feature and Feature Engineering

Before diving into feature engineering, let's take a moment and do a recap of what you have learnt so far. Let's take a look at the overall machine learning pipeline.

1.7.1 Data

- The raw data, or just data is a collection of observations of real-world phenomena. For instance, stock market data might involve observations of daily stock prices, announcements of earnings by individual companies, and even opinion articles from pundits. Sports data could have information on matches, environment in which those matches were played, player performances, and several other observations. Similarly, personal biometric data can include measurements of your minute-by-minute heart rate, blood sugar level, blood pressure, oxygen level, etc. You can come up with endless examples of data across different domains.
- Each piece of data provides a small window into a limited aspect of reality. The collection of all of these observations gives you a picture of the whole. But the picture is messy because it is composed of a thousand little pieces, and there's always measurement noise and missing pieces.

1.7.2 Tasks

Why do you collect data? I am sure you would say that there are several questions that data can help you answer (or predict). Some of the popular questions are

- How likely is that a customer buying product A will also buy product B?
- Which team is likely to win?
- How will be the weather next month?
- What food you should eat to get healthier?
- What is the risk of getting diabetes based on your biometric data?

The path from data to answers is full of false starts and dead ends.

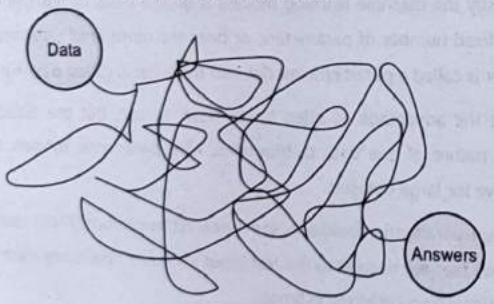


Fig. 1.7.1

- What starts out as a promising approach may not work in reality. What was originally just a hunch may end up leading to the best solution. Workflows with data are frequently multistage and iterative processes. For instance, stock prices are observed at the exchange, aggregated by an intermediary like Thomson Reuters, stored in a database, bought by a company, converted into a Hive store on a Hadoop cluster, pulled out of the store by a script, subsampled, massaged, and cleaned by another script, dumped to a file, and converted to a format that you can try out in your favourite modelling library in R, Python, or Scala. The predictions are then dumped back out to a CSV file and parsed by an evaluator, and the model is iterated multiple times, rewritten in C++ or Java by your production team, and run on all of the data before the final predictions are pumped out to another database.

- However, if you disregard the mess of tools and systems for a moment, you might see that the process involves two mathematical entities that are at the centre of machine learning models and features.

1.7.3 Models

- Trying to understand the world through data is like trying to piece together reality using a noisy, incomplete jigsaw puzzle with a bunch of extra pieces. This is where mathematical modelling in particular statistical modelling comes in. The language of statistics contains concepts for many frequent characteristics of data, such as wrong, redundant, or missing. Wrong data is the result of a mistake in measurement.
- Redundant data contains multiple aspects that convey exactly the same information. For instance, the day of week may be present as a categorical variable with values of "Monday," "Tuesday," ... "Sunday," and again included as an integer value between 0 and 6. If this day-of-week information is not present for some data points, then you have got missing data on your hands.
- A mathematical model of data describes the relationships between different aspects of the data. For instance, a model that predicts stock prices might be a formula that maps a company's earning history, past stock prices, and industry to the predicted stock price. A model that recommends music might measure the similarity between users (based on their listening habits) and recommend the same artists to users who have listened to a lot of the same songs.
- Mathematical formulas relate numeric quantities to each other. But raw data is often not numeric. For example, the action "Rohit bought Motorola G9 on Friday", is not numeric. Similarly, product reviews may not be numeric. This is where features come in.

1.7.4 Features (Concept of Feature)

- Earlier, you read that feature is anything that you can measure and build data for. For example, the typical length of various animals. Feature could be numeric, set of characters, Boolean values, or anything else that describes the data.
- But, for most machine learning mathematic models, features are required to be numeric so that they can be used in various computation.

So, let's redefine features as,

✍ ***Definition :*** *A feature is a numeric representation of raw data.*

- As you know, the features in a data set are also called its dimensions. So a data set having n features is called a n – d dimensional data set. For example, consider the following data set.

Gender	Marks
Girl	65
Girl	46
Boy	56
Boy	43
Boy	53
Boy	49

Gender	Marks
Girl	42
Boy	84
Boy	44
Girl	42
Girl	40

- How many features or dimensions does it have? Two, right? Yes, this is a two-dimensional data set, or you could also say that this data set has 2 features. I know you are saying out loud that "hey look, the gender field is not numeric". I understand, that is where feature engineering would come in where you would understand how you could convert this raw data set into something more meaningful and computationally more appropriate. For example, you could assign a value of "0" for boys and a value of "1" for girls. Now that is numeric, isn't it?

1.7.5 Feature Engineering

- There are many ways to turn raw data into numeric measurements (I just showed you one earlier), which is why features can end up looking like a lot of things. Naturally, features must derive from the type of data that is available. Features are also tied to the model. Some models are more appropriate for some types of features, and vice versa. The right features are relevant to the task at hand and should be easy for the model to ingest.

✍ ***Definition :*** *Feature engineering is the process of formulating the most appropriate features given the data, the model, and the task.*

- The Fig. 1.7.2 depicts where feature engineering sits in the machine learning pipeline.

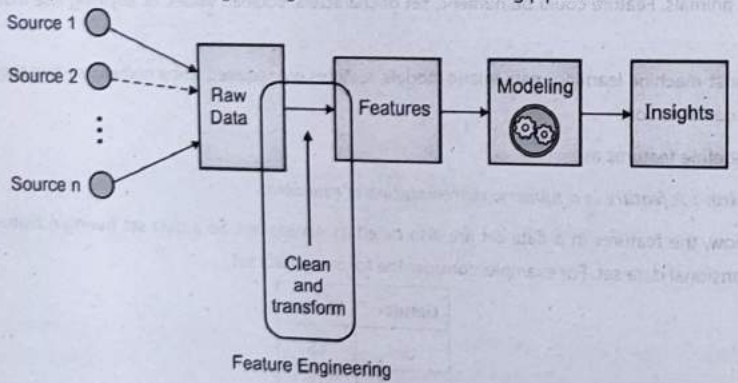


Fig. 1.7.2

- Features and models sit between raw data and the desired insights. In a machine learning workflow, you pick not only the model, but also the features. This is a double-jointed lever, and the choice of one affects the other. Good features make the subsequent modelling step easy and the resulting model more capable of completing the desired task. Bad features may require a much more complicated model to achieve the same level of performance.

- The number of features is also important. If there are not enough informative features, then the model will be unable to perform the ultimate task. If there are too many features, or if most of them are irrelevant, then the model will be more expensive and trickier to train. Something might go wrong in the training process that impacts the model's performance.
- Feature engineering typically includes feature creation, feature transformation, feature extraction, and feature selection.

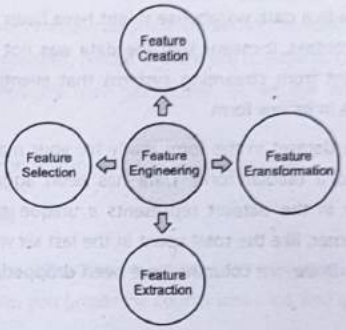


Fig. 1.7.3

- Feature creation identifies the features in the dataset that are relevant to the problem at hand.
- Feature transformation manages replacing missing features or features that are not valid.
- Feature extraction is the process of creating new features from existing features, typically with the goal of reducing the dimensionality of the features.
- Feature selection is the filtering of irrelevant or redundant features from your dataset. This is usually done by observing variance or correlation thresholds to determine which features to remove.

At a high-level, the feature engineering process looks like shown in Fig. 1.7.4.

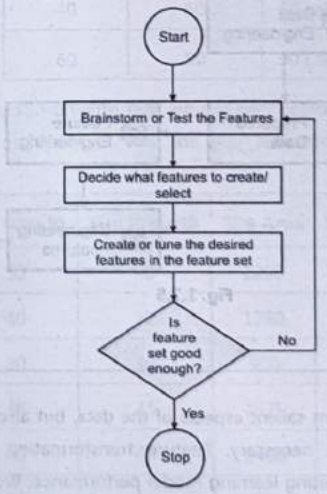


Fig. 1.7.4

1.7.6 Data Engineering -vs- Feature Engineering

Often raw data engineering (data pre-processing) is confused with feature engineering. Data engineering is the process of converting raw data into prepared data. Feature engineering then tunes the prepared data to create the features expected by the machine learning model. These terms have specific meanings as outlined here.

- **Raw data (or just data) :** This refers to the data in its source form, without any prior preparation for machine learning. Note that in this context, the data might be in its raw form (in a data lake) or in a transformed form (in a data warehouse). Transformed data in a data warehouse might have been converted from its original raw form to be used for analytics, but in this context, it means that the data was not prepared specifically for your machine learning task. In addition, data sent from streaming systems that eventually call machine learning models for predictions is considered to be data in its raw form.
- **Prepared data :** This refers to the dataset in the form ready for your machine learning task. Data sources have been parsed, joined, and put into a tabular form. Data has been aggregated and summarized to the right granularity-for example, each row in the dataset represents a unique customer, and each column represents summary information for the customer, like the total spent in the last six weeks. In the case of supervised learning tasks, the target feature is present. Irrelevant columns have been dropped, and invalid records have been filtered out.
- **Engineered features :** This refers to the dataset with the tuned features expected by the model that is, performing certain machine learning specific operations on the columns in the prepared dataset, and creating new features for your model during training and prediction. Some of the common examples are scaling numerical columns to a value between 0 and 1, clipping values, and one-hot-encoding categorical features.
- In practice, data from the same source is often at different stages of readiness. For example, a field from a table in your data warehouse could be used directly as an engineered feature. At the same time, another field in the same table might need to go through transformations before becoming an engineered feature. Similarly, data engineering and feature engineering operations might be combined in the same data pre-processing step.
- The Fig. 1.7.5, highlights the placement of data engineering and feature engineering tasks.

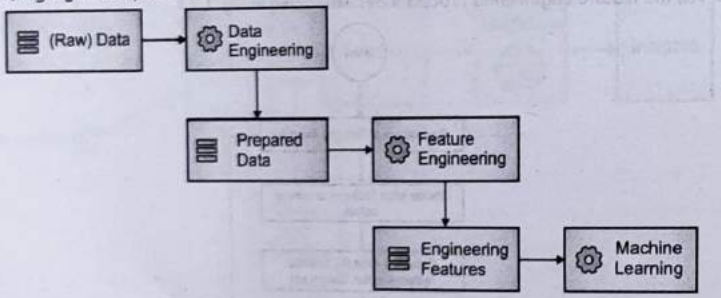


Fig. 1.7.5

1.8 Feature Transformation

Good features should not only represent salient aspects of the data, but also conform to the assumptions of the model. Hence, transformations are often necessary. Feature transformation is used as an effective tool for dimensionality reduction and hence for boosting learning model performance. Broadly, there are two distinct goals of feature transformation :

- Achieving best reconstruction of the original features in the data set
- Achieving highest efficiency in the learning task

Feature transformations could be applied to numeric features or non-numeric features such as text and images. Feature transformation generally involves feature construction and feature extraction. Let's learn about them.

1.8.1 Feature Construction (Statistical Feature Engineering)

Feature construction process discovers missing information about the relationships between features and expands the feature space by creating additional features. Hence, if there are n features or dimensions in a data set, after feature construction m more features or dimensions may get added. So at the end, the data set will become n + m dimensional.

Let's learn about some of the common feature construction techniques.

1.8.1(A) Quantization or Binning

- Quantization or binning is a feature construction technique where you could combine features to create segments or bins of information. In other words, you group the counts into bins, and get rid of the actual count values.
- For example, consider the following data set for real-estate sites.

Site Length	Site Breadth	Site Price
30	40	40 Lakhs
40	32	40 Lakhs
30	30	30 Lakhs
35	45	45 Lakhs
40	60	60 Lakhs
60	80	90 Lakhs

- Instead of having site length and site breadth that are not very useful for establishing a modelling pattern, you could create a new feature such as "site area". The "site area" feature could then provide a good estimate of site price.

Site Length	Site Breadth	Site Area	Site Price
30	40	1200	40 Lakhs
40	32	1280	40 Lakhs
30	30	900	30 Lakhs
35	45	1575	45 Lakhs
40	60	2400	60 Lakhs
60	80	4800	90 Lakhs

- There could be several other examples of binning. For example, it is common to see custom-designed age ranges that better correspond to stages of life, such as:
  - 0-12 years old
  - 12-17 years old
  - 18-24 years old
  - 25-34 years old
  - 35-44 years old
  - 45-54 years old
  - 55-64 years old
  - 65-74 years old
  - 75 years or older
- You could get rid of the actual age in a large data set and create bins based on age groups. You could then model things such as product or service preferences, reviews, demographics, eating habits, etc. more elegantly.
- It is up to your requirements to create bins as you need. For example, you may need to create bins for income. You could then have something like the following (remember how income tax or electricity bills have various slabs?).
  - Below 5 Lakhs
  - 5-10 Lakhs
  - 10-20 Lakhs
  - 20-50 Lakhs
  - Over 50 Lakhs
- You can create bins on fixed value range or take value range based on other mathematical derivations such as quantile, percentile, median, etc.
- Note here that binning can be carried out on both categorical and numerical data. For example, you would have seen a BMI chart that classifies you in various fitness categories based on your body weight and height. Hence, you are binning the categorical data (fitness category). Instead of worrying about each possible combination of BMI $\left(\frac{\text{weight in kg}}{(\text{height in meter})^2}\right)$, you just combine the data and bin it as per the respective categories.

BMI	Nutritional status
Below 18.5	Underweight
18.5 – 24.9	Normal weight
25.0 – 29.9	Pre-obesity
30.0 – 34.9	Obesity class I
35.0 – 39.9	Obesity class II
Above 40	Obesity class III

1.8.1(B) Log Transform

- The log transform is a powerful tool for dealing with large positive numbers with a heavy-tailed distribution. A heavy-tailed distribution places more entries towards the tail end of the plot rather than centre. It compresses the long tail in the high end of the distribution into a shorter tail and expands the low end into a longer head. Let's understand how.
- The log function is the inverse of the exponential function. It is defined such that $\log_a(a^x) = x$, where a is a positive constant, and x can be any positive number. Since $a^0 = 1$, you get $\log_a(1) = 0$. This means that the log function maps the small range of numbers between (0, 1) to the entire range of negative numbers $(-\infty, 0)$. The function $\log_{10}(x)$ maps the range of [1, 10] to [0, 1], [10, 100] to [1, 2], and so on. In other words, the log function compresses the range of large numbers and expands the range of small numbers. The larger x is, the slower log (x) increments.
- For example, in the Fig. 1.8.1, note how the horizontal x values from 100 to 1,000 get compressed into just 2.0 to 3.0 in the vertical y range, while the tiny horizontal portion of x values less than 100 are mapped to the rest of the vertical range.

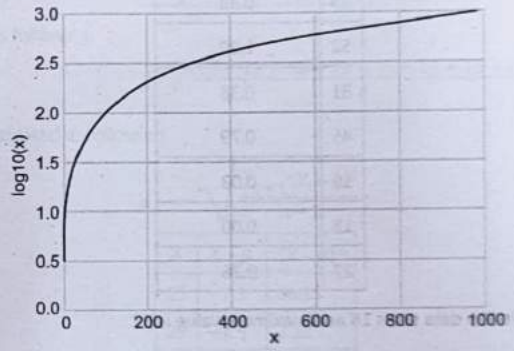


Fig. 1.8.1

- Log transform is commonly used when you are dealing with large numbers. For example, some businesses have a lot of reviews (say over 2000) and some have only few (say in 20s).
- In such a scenario, it becomes difficult to compare and correlate one business with the other because a large count in one element of the data set would outweigh the similarity in all other elements, which could throw off the entire similarity measurement for various machine learning algorithms. Log transformation helps to normalise skewed data.

1.8.1(C) Feature Scaling or Normalisation

- Some features, such as latitude or longitude, are bounded in value. Other numeric features, such as counts, may increase without bound. Models that are smooth functions of the input, such as linear regression, logistic regression, or anything that involves a matrix, are affected by the scale of the input.
- If your model is sensitive to the scale of input features, feature scaling could help. As the name suggests, feature scaling changes the scale of the feature. It is also called as feature normalisation. Feature scaling is usually done individually to each feature. Let's see some examples.

1.8.1(D) Min-Max Scaling

- Min-Max scaling scales all values in a fixed range between 0 and 1. This transformation does not change the distribution of the feature. Before applying min-max scaling, you should separately handle the outliers to reduce their effect.
- Min-Max scaling is carried out as following.
- Let x be an individual feature value, and min (x) and max (x), respectively, be the minimum and maximum values of this feature over the entire dataset. Then, min-max scaling could be applied using the following formula.

$$\bar{x} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

- For example, min-max scaling for the following data set is as following.

X	Min-Max values
23	0.15
29	0.32
52	1.00
31	0.38
45	0.79
19	0.03
18	0.00
27	0.26

- The minimum value of X in the data set is 18 and maximum value is 52.

  So, for the first value of X, the min-max scaled value is calculated as following.

$$\bar{x} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

$$\bar{x} = \frac{23 - 18}{52 - 18} = \frac{5}{34} = 0.15$$

- Likewise, all the elements are calculated. Notice that irrespective of how large or small the value is, the resultant min-max value is between 0 and 1. That is the power of scaling!

1.8.1(E) Standardisation (Variance Scaling)

- Standardisation (or Z-score normalisation or variance scaling) scales the values taking standard deviation of the features into account. Thus, it reduces the effect of the outliers in the features.
- In this scaling method, the mean of the feature (over all data points) is subtracted and divided by the standard deviation of the feature. The resulting scaled feature has a mean of 0 and a standard deviation of 1. If the original feature is a normal distribution, then the scaled feature is also a normal distribution.

- It is calculated as following.

$$Z = \frac{x - \mu}{\sigma}$$ where μ = mean and σ = standard deviation

For example, normalised values for the following data set is as following.

X	Normalised Values
23	−0.62
29	−0.12
52	1.77
31	0.04
45	1.19
19	−0.95
18	−1.03
27	−0.29

- The mean is calculated as following.

$$\mu = \frac{23 + 29 + 52 + 31 + 45 + 19 + 18 + 27}{8} = 30.5$$

- Standard deviation is calculated as following.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (X_i - \mu)^2}{n - 1}}$$

X	X − μ	(X − μ)²
23	−7.5	56.25
29	−1.5	2.25
52	21.5	462.25
31	0.5	0.25
45	14.5	210.25
19	−11.5	132.25
18	−12.5	156.25
27	−3.5	12.25
	Total	**1032**

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (X_i - \mu)^2}{n - 1}} = \sqrt{\frac{1032}{7}} = 12.14$$

Hence, for the first value of X, the normalised value is calculated as

$$z = \frac{x - \mu}{\sigma} = \frac{23 - 30.5}{12.14} = -0.62$$

Likewise, all the elements are calculated.

l^2 Normalisation(Length-based)

- This technique normalises (divides) the original feature value by what's known as the l^2 norm (pronounced l Norm). It is also known as the Euclidean norm. It is calculated as following.

$$\bar{x} = \frac{x}{\|x\|_2}$$

Where, $\|x\|_2 = \sqrt{x_1^2 + x_2^2 + x_3^2 \ldots x_n^2}$

- The l^2 norm measures the length of the vector in coordinate space. The definition can be derived from the well known Pythagorean theorem that gives us the length of the hypotenuse of a right triangle given the lengths of th sides.
- For example, l^2 normalised values for the following data set is as following.

X	l^2 Normalised Values
23	0.25
29	0.32
52	0.56
31	0.34
45	0.49
19	0.21
18	0.20
27	0.29

- It was calculated as following.

X	X^2
23	529
29	841
52	2704
31	961
45	2025
19	361
18	324
27	729
Total	**8474**

$$\|x\|_2 = \sqrt{x_1^2 + x_2^2 + x_3^2 \ldots x_n^2} = \sqrt{8474} = 92.05$$

So, for the first value of X

$$\bar{x} = \frac{x}{\|x\|_2} = \frac{23}{92.05} = 0.25$$

Likewise, all the elements are calculated.

1.8.1(F) Encoding Categorical Variables

What is a categorical variable?

✍ ***Definition :*** *A categorical variable is used to represent categories or labels.*

- For example, a categorical variable could represent major cities in the world, the four seasons in a year, or the industry (oil, travel, technology) of a company. The number of category values is always finite in a real-world dataset. The values may be represented numerically. However, unlike other numeric variables, the values of a categorical variable cannot be ordered with respect to one another. For example, Oil is neither greater than nor less than travel as an industry type, isn't it? Such type of categorical variables are called nonordinal (meaning that they cannot be specifically ordered).
- Large categorical variables are particularly common in transactional records. For instance, many web services track users using an ID, which is a categorical variable with hundreds to hundreds of millions of values, depending on the number of unique users of the service. The IP address of an internet transaction is another example of a large categorical variable.
- They are categorical variables because, even though user IDs and IP addresses are numeric, their magnitude is usually not relevant to the task at hand. For instance, the IP address might be relevant when doing fraud detection on individual transactions-some IP addresses or subnets may generate more fraudulent transactions than others. But a subnet of 164.203.x.x is not inherently more fraudulent than 164.202.x.x; the numeric value of the subnet does not matter.
- So, as you know, the categories of a categorical variable are usually not numeric. For example, eye colour can be "black," "blue," "brown," etc. Thus, an encoding method is needed to turn these non-numeric categories into numbers. It is tempting to simply assign an integer, say from 1 to k, to each of k possible categories but the resulting values would be orderable against each other (means you could start treating them as actual numbers mistakenly), which should not be permissible for categories. So, let's learn about some of the ways in which categorical variables could be encoded to construct features out of them.

1.8.1(G) One-Hot Encoding

- One-hot encoding is one of the most common encoding methods in machine learning. It creates new (binary) columns, indicating the presence of each possible value from the original data. Each bit represents a possible category. If the variable cannot belong to multiple categories at once, then only one bit in the group can be "on". Each of the bits is a feature.
- For example, consider a categorical feature called "eye colour" that can take one of the three values – black, brown, and blue, in a data set.

Age	Eye colour
24	Black
25	Brown
26	Blue

- What you could do is that you split the feature into k columns where k is the number of possible values that categorical feature could possibly take. Assign "1" where the particular value is present and "0" where the particular value is not present.
- So, the sample data set could be one-hot encoded as the following.

Age	Eye colour
24	Black
25	Brown
26	Blue

Sample data set

⇒

Age	Black	Brown	Blue
24	1	0	0
25	0	1	0
26	0	0	1

One-hot encoding

- The first row has the new column "Black" marked as "1" to represent the sample record where "Eye colour" is Black. Columns "Brown" and "Blue" are marked as "0" as those values do not apply to the first record. Similarly, rest of the records are encoded to match the sample data set. This way, you have converted a categorical variable to numeric.
- One-hot encoding is very simple to understand, but it uses one more bit than is strictly necessary. If you see that k – 1 of the bits are 0, then the last bit must be 1 because the variable must take on one of the possible k values. Mathematically, you can write this constraint as "the sum of all bits must be equal to 1".

$e_1 + e_2 + ... + e_k = 1$ [where e is encoded variable values]

1.8.1(H) Dummy Coding

- The problem with one-hot encoding is that it allows for k degrees of freedom, while the variable itself needs only k – 1. Dummy coding removes the extra degree of freedom by using only k – 1 features (feature values) in the representation. One feature is disregarded and is represented by the vector of all zeros. This is known as the reference category.
- You don't lose any information as the removed feature value can be derived knowing that If you see that k – 1 of the bits are 0, then the last bit must be 1 because the variable must take on one of the possible k values. The sum of all bits must be equal to 1.
- So, for example, one-hot encoding could be dummy coded as following.

Age	Black	Brown	Blue
24	1	0	0
25	0	1	0
26	0	0	1

One-hot encoding

⇒

Age	Black	Brown
24	1	0
25	0	1
26	0	0

Dummy coding

- The column "Blue" is deleted as it contained 0 for first two rows and the last row has both "Black" and "Brown" as 0, meaning that "Blue" must be 1.

1.8.1(I) Feature Hashing

- Large categorical features, such as user ID, advertisement ID, website URL, IP addresses, etc., pose computation challenges in terms of memory efficiency and storage.

- For such features, a technique called feature hashing could be used that makes working with large categorical variables less computation intensive and yet produces accurate models that are fast to train.

✍ ***Definition :*** *Hashing, in general, is the process of taking any length of input information and finding a unique fixed length representation of that input information.*

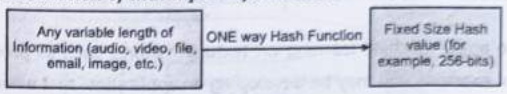


Fig. 1.8.2

- Hashing is the process of finding a unique message digest (or hash value) that corresponds to the input information. The length of input could be just one character or a huge video file. Hashing always produces a fixed size representation of the information.
- Hashing, in general, is used in several domains such as information security, cryptocurrency, high-performance programming, and for creating quick lookup tables.
- In machine learning, hash functions can be constructed for any object that can be represented numerically (which is true for any data that can be stored on a computer): numbers, strings, complex structures, etc.

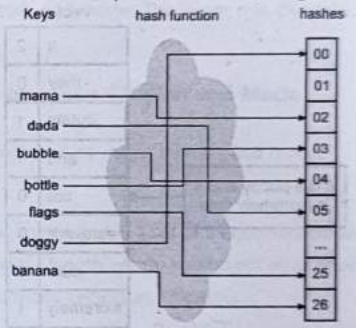


Fig. 1.8.3

- When there are many features, storing the feature vector could take up a lot of space. Feature hashing compresses the original feature vector into an m-dimensional vector by applying a hash function to the feature ID (also called as keys). For example, if the original features were words in a document, then the hashed version would have a fixed vocabulary size of m, no matter how many unique words there are in the input.

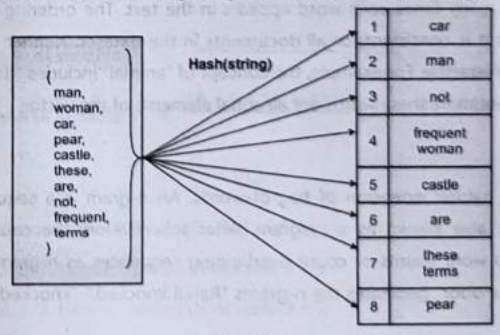


Fig. 1.8.4

- Note here that since the input range is potentially larger than the output range, multiple features may get mapped to the same output (hash value). This is called a collision. However, you rely on the model learning that some shared representation of the categories in the same index (same hash value) works well for the given problem.

1.8.1(J) Handling Textual Features

- Often times, you need to apply machine learning on textural features such as product reviews, comments, stock line, news reports, etc. For example, you may be developing an application, that when provided an information, can tell whether it is fake or true. How do you relate the information with what is known to be true or fake?

Let's learn some techniques to handle textual features.

1. Bag-of-Words (Count-based)

- In bag-of-words (BoW), a text document is converted into a vector of counts. The vector contains an entry for every possible word in the vocabulary. If the word say, "aardvark" appears three times in the document, then the feature vector has a count of 3 in the position corresponding to that word. If a word in the vocabulary doesn't appear in the document, then it gets a count of 0.

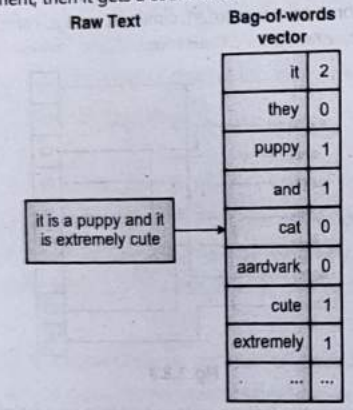


Fig. 1.8.5

- Bag-of-words converts a text document into a flat vector. It is "flat" because it does not contain any of the original textual structures. The original text is a sequence of words. But a bag-of-words has no sequence. It just remembers how many times each word appears in the text. The ordering of words in the vector is not important, as long as it is consistent for all documents in the dataset. Neither does bag-of-words represent any concept of word hierarchy. For example, the concept of "animal" includes "dog," "cat," "raven," etc. But in a bag-of-words representation, these words are all equal elements of the vector.

2. Bag-of-n-Grams

- Bag-of-n-Grams is a natural extension of bag-of-words. An n-gram is a sequence of n tokens. A word is essentially a 1-gram, also known as a unigram. After tokenisation, the counting mechanism can group individual tokens into word counts or count overlapping sequences as n-grams. For example, the sentence "Rahul knocked on the door" generates the n-grams "Rahul knocked," "knocked on," "on the," and "the door" for n = 2.

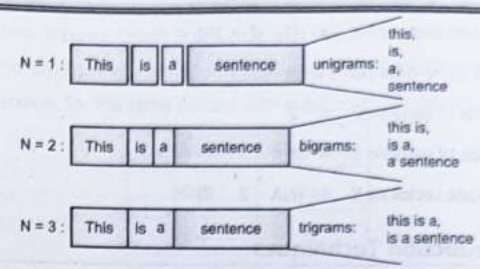


Fig. 1.8.6

- n-grams retain more of the original sequence structure of the text, and therefore the bag-of-n-grams representation can be more informative. However, this comes at a cost. Theoretically, with k unique words, there could be k^2 unique 2-grams (also called bigrams). In practice, there are not nearly so many, because not every word can follow every other word. Nevertheless, there are usually a lot more distinct n-grams (n > 1) than words.
- This means that bag-of-n-grams is a much bigger and sparser feature space. It also means that n-grams are more expensive to compute, store, and model. The larger n is, the richer the information, and the greater the cost.

1.8.2 Statistical Vectors based on Mean, Median and Mode

- You can also create statistical vectors based on mean, median, and mode.
- Let's take an example.

Suppose that you have the following matrix that is a set of 5 observations and measuring 3 variables. Assume that the three variables, from left to right are length, width, and height of a certain object. You can then describe these variables as their mean, median, and mode vectors as following.

$$X = \begin{bmatrix} 5 & 2 & 10 \\ 4 & 1 & 9 \\ 6 & 2 & 11 \\ 7 & 3 & 12 \\ 3 & 1 & 9 \end{bmatrix}$$

$$\text{Mean of variable 1} = \frac{5+4+6+7+3}{5} = \frac{25}{5} = 5$$

$$\text{Mean of variable 2} = \frac{2+1+2+3+1}{5} = \frac{9}{5} = 1.8$$

$$\text{Mean of variable 3} = \frac{10+9+11+12+9}{5}$$

$$= \frac{51}{5} = 10.2$$

$$\text{Mean vector of X} = (5 \quad 1.8 \quad 10.2)$$

Sort variable 1 to get 3, 4, 5, 6, 7. Hence, median of variable 1 = 5

Sort variable 2 to get 1, 1, 2, 2, 3. Hence, median of variable 2 = 2

Sort variable 3 to get s9,9, 10, 11, 12. Hence, median of variable 3 = 10

Median vector of X = [5 2 10]

The is no mode of variable 1 as no data point is repeated

First mode of variable 2 = 2

Mode of variable 3 = 9

Mode vector of X = {NA 2 9}

1.9 Dimensionality Reduction Techniques

- Assume that you are watching cricket on your 4K TV. A 4K TV has 3,840 horizontal pixels and 2,160 vertical pixe for a total of about 8.3 million pixels! Most of the times, you are focusing on the ball. Say that the ball occupi 10,000 pixels to form a 3D image. Your brain is filtering out rest of the pixels and helping you to focus on 10,0 pixels out of the total 8.3 million pixels presented to it. That is close to just 0.12% of the entire set of pixels that in front of you. This is precisely what happens in dimensional reduction. You reduce the number of dimensions your dataset to just what matters the most.
- Often times, you find that your dataset could have 100s of features (or dimensions). Practically, you know that n all dimensions are equally important for analysis or classification of data. Also, it becomes computationa intensive and visually difficult to understand which dimensions have the most influence on the dataset if you ha 100s of dimensions.

✍ **Definition :** *Dimensionality reduction techniques help you to reduce the number of dimensions to only ke important dimensions of data and discard all other dimensions.*

- In most learning algorithms, the complexity depends on the number of input dimensions, d, as well as on the si of the data sample, N. As you increase the number of dimensions, you would also require collecting increas number of samples to support those many dimensions (in order to ensure that every combination of features well represented in the dataset). As the number of dimensions increase, working with it becomes increasing harder. This problem is often cited as "the curse of dimensionality".
- To make it less computationally intensive, you reduce the dimensionality of the problem. Decreasing d al decreases the overall complexity of the problem and make it more plausible to understand most importa dimensions of data.

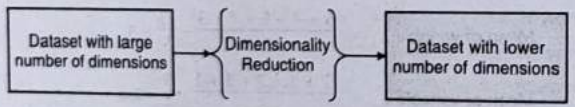


Fig. 1.9.1

Let's take a simple example.

Income	Credit Score	Age	Location	Give Loan
50,000	High	34	Mumbai	Yes
75,000	Low	33	Bangalore	No
80,000	High	37	Mumbai	Yes
90,000	High	29	Kolkata	Yes

- This dataset has 4 dimensions (Income, Credit Score, Age, and Location) based on which loan approval seems to be granted. But, if you look closely, then you would find that the other dimensions do not influence the decision as much as Credit Score dimension. So, the same dataset with reduced dimensions could be as following.

Credit Score	Give Loan
High	Yes
Low	No
High	Yes
High	Yes

- This dimensional reduction not only makes the algorithms computationally less intensive but also makes it simple to understand and visualise the dataset as well as the results.
- Note here that the goal of dimensionality reduction is NOT to reduce (or compromise on) the quality of data when discarding unnecessary dimensions but to only keep the dimensions that matter the most without any significant loss of quality. This is very similar to how you compress a file without losing information or how you switch to a lower resolution for a video if your internet speed is not optimal. Lowering the resolution does not change the video much, it is just that it does not look that sharp!

Need for Dimensionality Reduction (Advantages)

So, as you understand by now, dimensionality reduction is required for the following.

1. To reduce complexity when working with data
2. To make the learning models computationally less intensive
3. Reduce overall training time for the algorithm (lesser the number of dimensions, shorter is the time required for the algorithm to learn about those dimensions)
4. Reduce noise and minimise errors (simpler models are more robust on small datasets)
5. Reduce storage requirements (larger datasets require more storage)
6. Be able to visualise and describe the dataset and the results more prominently.

1.9.1 Types of Dimensionality Reduction Techniques

There are two major types of dimensionality reduction techniques.

1. **Feature Selection :** In feature selection, you are interested in finding k of the d dimensions that give you the most information and you discard the other (d – k) dimensions. The aim is to find the best subset of the set of features. The best subset contains the least number of dimensions that most contribute to accuracy. You discard the remaining unimportant dimensions. Using a suitable error function, this can be used in both regression and classification problems. There are 2^d possible subsets of d variables, but you cannot test for all of them unless d is small. You use heuristics (experienced guess) to get a reasonable (but not optimal) solution in reasonable time.

   There are two approaches for carrying out feature selection.

   (a) Forward Selection : In forward selection, you start with no variables and add them one by one, at each step adding the one that decreases the error the most, until any further addition does not decrease the error.

(b) **Backward Selection :** In backward selection, you start with all variables and remove them one by one, at ea step removing the one that decreases the error the most (or increases it only slightly), until any furth removal increases the error significantly.

2. **Feature Extraction :** In feature extraction, you are interested in finding a new set of k dimensions that a combinations of the original d dimensions. These methods may be supervised or unsupervised depending o whether they use the output information or not. The best known and most widely used feature extraction metho are Principal Components Analysis (PCA) and Linear Discriminant Analysis (LDA). They both are linear projecti methods. PCA is unsupervised method of dimensionality reduction whereas LDA is a supervised method.

1.10 Principal Component Analysis (PCA)

PCA is an unsupervised dimensional reduction method as it does not use output (target) information. PCA aims finding the most important dimensions (called principal components) for the given dataset.

Definition : *A principal component is a direction in which the data has the largest variation.*

At a high-level, the algorithm takes the following steps.

1. The algorithm first centres the data by subtracting off the mean.
2. It then chooses the direction with the largest variation and places an axis in that direction.
3. It then looks at the remaining variations and finds another axis that is orthogonal (at 90°) to the first and covers much of the remaining variation as possible.
4. It then iterates this (step 3) until it has run out of possible axes and covered all the variations.
5. The end result is that all the variation is along the axes of the coordinate set.
6. Some of the axes that are found last have very little variation, and so they can be removed without affecting th variability in the overall data.

Let's take an example to understand it better. Consider a simple dataset as shown in Fig. 1.10.1.

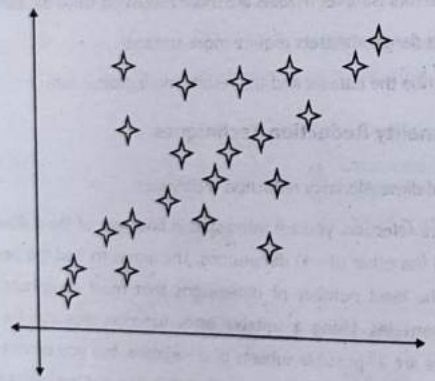


Fig. 1.10.1

If I tell you to draw a line such that it covers the maximum number of datapoints, how would you likely draw it ?

Perhaps, you would draw it like the Fig. 1.10.2.

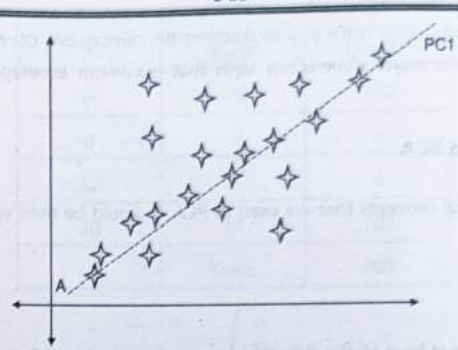


Fig. 1.10.2

The line A covers the maximum variation in the datapoints. This is called principal component 1 and is denoted as PC1. Now, if I tell you to draw a second line, at to the first line, to cover the next maximum variation in datapoints, how would you draw such a line ?

Probably, you would draw it as shown in Fig. 1.10.3.

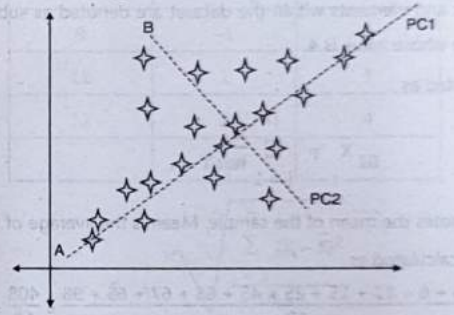


Fig. 1.10.3

The line B further covers the remaining variations in the datapoints. This is called principal component 2 and is denoted as PC2.

You then realign PC1 and PC2 axes to make them further easier to comprehend.

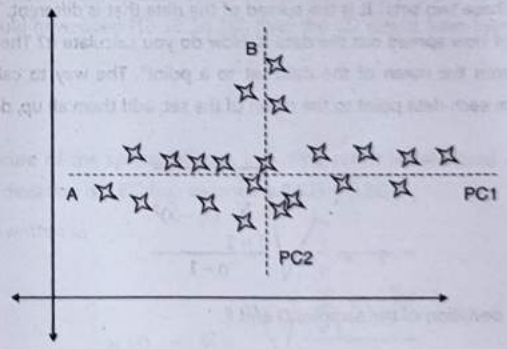


Fig. 1.10.4

This is what PCA does in a nutshell. It helps you to discover the dimensions covering different variations in data points and you then pick only the major dimensions such that maximum coverage is obtained and discard the remaining dimensions.

1.10.1 Mathematics behind PCA

Let's revise some mathematical concepts that are used in PCA. It would be then easier to understand how PCA is carried out.

1.10.2 Standard Deviation

- As you understand, you may not have all the data that you want every time. Most of the times, you draw samples from a large population in the hope that it represents the entire population to a great extent. By using a sample of the population, you can work out what is most likely to be the measurement if you used the entire population.

  Assume a simple dataset.

$$X = [1\ 2\ 4\ 6\ 12\ 15\ 25\ 45\ 68\ 67\ 65\ 98]$$

- X refers to the entire dataset and elements within the dataset are denoted as subscript. For example, X_3 refers to the 3rd element in the sample whose value is 4.

  Mean of the sample is calculated as

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

$\bar{X}$, pronounced as (X-bar), denotes the mean of the sample. Mean is the average of the sample values.

So, for the given dataset, mean is calculated as

$$\bar{X} = \frac{1+2+4+6+12+15+25+45+68+67+65+98}{12} = \frac{408}{12} = 34$$

- Unfortunately, the mean doesn't tell us a lot about the data except for a sort of middle point. For example, these two data sets have exactly the same mean (10) but are obviously quite different.

$$D = [0\ 8\ 12\ 20] \text{ and } E = [8\ 9\ 11\ 12]$$

- So what is different about these two sets? It is the spread of the data that is different. The Standard Deviation (SD) of a data set is a measure of how spread out the data is. How do you calculate it? The English definition of the SD is "The average distance from the mean of the data set to a point". The way to calculate it is to compute the squares of the distance from each data point to the mean of the set, add them all up, divide by n – 1, and then take the positive square root.

$$S = \sqrt{\frac{\sum_{i=1}^{n} (X_i - \bar{X})^2}{n-1}}$$

- Let's calculate the standard deviation of the samples D and E.

X	$X - \bar{X}$	$(X - \bar{X})^2$
0	–10	100
8	–2	4
12	2	4
20	10	100
	Total	**208**

$$S = \sqrt{\frac{\sum_{i=1}^{n} (X_i - \bar{X})^2}{n-1}} = \sqrt{\frac{208}{3}} = 8.32$$

X	$X - \bar{X}$	$(X - \bar{X})^2$
8	–2	4
9	–1	1
11	1	1
12	2	4
	Total	**10**

$$S = \sqrt{\frac{\sum_{i=1}^{n} (X_i - \bar{X})^2}{n-1}}$$

$$= \sqrt{\frac{10}{3}} = 1.82$$

- So, what does SD calculation tell you about the samples? The first set has a much larger standard deviation due to the fact that the data is much more spread out from the mean whereas the data is closer to the mean in the second set. If the set would have been [10 10 10 10], then the SD would have been 0 as none of the datapoints deviate from the mean.

1.10.3 Variance

- Variance is another measure of the spread of data in a data set. It is calculated as s^2 where s is the standard deviation. So, if standard deviation is 1.82 then variance is $1.82^2 = 3.31$.
- Mathematically, it can be written as

$$\text{var}(X) = S^2 = \sqrt{\frac{\sum_{i=1}^{n} (X_i - \bar{X})^2}{n-1}}$$

1.10.4 Covariance

- Standard deviation and variance work on 1-dimensional data. However, many data sets have more than one dimension, and the aim of the statistical analysis of these data sets is usually to see if there is any relationship between the dimensions. For example, you might have a data set having both, the height of all the students in a class, and the marks they received for a test. You could then perform statistical analysis to see if the height of a student has any effect on her marks.
- Standard deviation and variance only operate on 1-dimension. So, you can only calculate the standard deviation for each dimension of the data set independently of the other dimensions in the dataset. However, it is useful to have a similar measure to find out how much each of the dimensions vary from the mean with respect to each other. Covariance is such a measure. Covariance is always measured between 2 dimensions. If you calculate the covariance between one dimension and itself, you get the variance for that dimension.
- So, if you have a 3-dimensional data set (x, y, z), then you could measure the covariance between x and y dimensions, x and z dimensions, and y and z dimensions. Measuring the covariance between x and x, y and y, or z and z would give you the variance of the x, y, and z dimensions respectively.
- The mathematical formula for covariance is very similar to the mathematical formula for variance. The formula for variance could also be written like

$$\text{var } X = \frac{\sum_{i=1}^{n} (X_i - \bar{X})^2}{n-1} = \frac{\sum_{i=1}^{n} (X_i - \bar{X})(X_i - \bar{X})}{n-1}$$

- Covariance is mathematically computed as

$$\text{cov}(X, Y) = \frac{\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})}{n-1}$$

- Note here that cov (X, Y) = cov (X, Y) as the calculation would remain unchanged.
- Let's take a sample data set having 2-dimesions, number of hours of study and marks obtained. Does number of hours affect marks obtained?

Hours (H)	Marks (M)
9	39
15	56
25	93
14	61
10	50
18	75
0	32
16	85

Hours (H)	Marks (M)
5	42
19	70
16	66
20	80

- You could calculate the covariance between hours of study and marks obtained to establish if there is any relationship between them.

Hours (H)	Marks (M)	$H_i - \bar{H}$	$M_i - \bar{M}$	$(H_i - \bar{H})(M_i - \bar{M})$
9	39	– 4.92	– 23.42	115.23
15	56	1.08	– 6.42	– 6.93
25	93	11.08	30.58	338.83
14	61	0.08	– 1.42	– 0.11
10	50	– 3.92	– 12.42	48.69
18	75	4.08	12.58	51.33
0	32	– 13.92	– 30.42	423.45
16	85	2.08	22.58	46.97
5	42	– 8.92	– 20.42	182.15
19	70	5.08	7.58	38.51
16	66	2.08	3.58	7.45
20	80	6.08	17.58	106.89
				1352.42

$$\text{cov}(H, M) = \frac{1352.42}{11} = 122.94$$

So what does it tell you?

The exact value is not as important as it's sign (positive or negative).

- If the value is positive, as it is here, then it indicates that both dimensions increase together, meaning that, in general, as the number of hours of study increased, so did the marks obtained.
- If the value is negative, then as one of the dimensions increases, the other decreases. If you had ended up with a negative covariance here, then that would have meant that as the number of hours of study increased, the marks obtained decreased.
- In the last case, if the covariance is zero, it indicates that the two dimensions are independent of each other.

1.10.5 Covariance Matrix

- If you have a data set with more than 2 dimensions, then there is more than one covariance measurement that can be calculated. For example, if you have a 3-dimensional data set having x, y, z as the dimensions, then you can calculate cov (x, y), cov (x, z), and cov (y, z). You can also calculate variance within the respective dimensions cov (x, x), cov (y, y), and cov (z, z).
- A useful way to get all the possible covariance values between all the different dimensions is to calculate them all and put them in a matrix.

$$C = \begin{pmatrix} \text{cov}(x, x) & \text{cov}(x, y) & \text{cov}(x, z) \\ \text{cov}(y, x) & \text{cov}(y, y) & \text{cov}(y, z) \\ \text{cov}(z, x) & \text{cov}(z, y) & \text{cov}(z, z) \end{pmatrix}$$

Note that cov (x, y) = cov (y, x), cov (x, z) = cov (z, x),

and cov (y, z) = cov (z, y).

- Let's take an example.
- Calculate the covariance matrix for the following 3-dimensional data.

x	y	z
1	2	1
−1	1	3
4	3	−1

x	y	z	$x_i - \bar{x}$	$y_i - \bar{y}$	$z_i - \bar{z}$	$(x_i - \bar{x})^2$	$(x_i - \bar{x})(y_i - \bar{y})$	$(x_i - \bar{x})(z_i - \bar{z})$
1	2	1	−0.33	0	0	0.1089	0	0
−1	1	3	−2.33	−1	2	5.4289	2.33	−4.66
4	3	−1	2.67	1	−2	7.1289	2.67	−5.34
						Total = 12.66	**Total = 5**	**Total = −10**

$$\text{cov}(x, x) = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1} = \frac{12.66}{2} = 6.33$$

$$\text{cov}(x, y) = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{n-1} = \frac{5}{2} = 2.5$$

$$\text{cov}(x, z) = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(z_i - \bar{z})}{n-1} = \frac{-10}{2} = -5$$

y	z	$(y_i - \bar{y})$	$(y_i - \bar{y})^2$	$(z_i - \bar{z})$	$(y_i - \bar{y})(z_i - \bar{z})$
2	1	0	0	0	0
1	3	−1	1	2	−2
3	−1	1	1	−2	−2
			Total = 2		**Total = −4**

$$\text{cov}(y, y) = \frac{\sum_{i=1}^{n} (y_i - \bar{y})^2}{n-1} = \frac{2}{2} = 1$$

$$\text{cov}(y, z) = \frac{\sum_{i=1}^{n} (y_i - \bar{y})(z_i - \bar{z})}{n-1} = \frac{-4}{2} = -2$$

z	$(z_i - \bar{z})$	$(z_i - \bar{z})^2$
1	0	0
3	2	4
−1	−2	4
		Total = 8

$$\text{cov}(z, z) = \frac{\sum_{i=1}^{n} (z_i - \bar{z})^2}{n-1} = \frac{8}{2} = 4$$

Let's arrange all the values in the covariance matrix format.

$$C = \begin{pmatrix} \text{cov}(x, x) & \text{cov}(x, y) & \text{cov}(x, z) \\ \text{cov}(y, x) & \text{cov}(y, y) & \text{cov}(y, z) \\ \text{cov}(z, x) & \text{cov}(z, y) & \text{cov}(z, z) \end{pmatrix}$$

$$C = \begin{pmatrix} 6.33 & 2.5 & -5 \\ 2.5 & 1 & -2 \\ -5 & -2 & 4 \end{pmatrix}$$

1.10.6 Eigenvectors

- Let's take two examples of matrix multiplication.

$$M = \begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 11 \\ 5 \end{pmatrix}$$

$$N = \begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

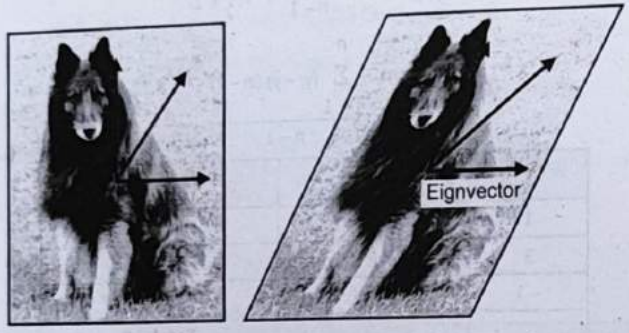
- In the first example, the resulting vector is not an integer multiple of the original vector, whereas in the second example, the resulting vector is exactly 4 times the vector you began with. This is called eigenvector.

- So, if you multiply the matrix on the left of an eigenvector, the answer is another vector that is transformed from its original position. It is the nature of the transformation that the eigenvectors arise from. Imagine a transformation matrix, that when multiplied, reflected vectors on the line y = x. Then, you can see that if there was a vector that lay on the line y = x , it's reflection of itself. This vector (and all multiples of it because it does not matter how long the vector was), would be an eigenvector of that transformation matrix.

So, if you scale the eigenvector $2 \times \begin{pmatrix}3\\2\end{pmatrix} = \begin{pmatrix}6\\4\end{pmatrix}$, it will still transform the matrix in the similar manner.

$$\begin{pmatrix}2 & 3\\2 & 1\end{pmatrix} \times \begin{pmatrix}6\\4\end{pmatrix} = \begin{pmatrix}24\\16\end{pmatrix} = 4 \times \begin{pmatrix}6\\4\end{pmatrix}$$

- So, in a nutshell, an eigenvector does not change direction in a transformation. Think of it as in the following picture.



Eigenvectors have the following properties.

1. Eigenvectors can only be found for square matrices. For example, 2 × 2, 3 × 3, 4 × 4, etc.
2. Not every square matrix has eigenvectors.
3. Given that an n × n matrix does have eigenvectors, there are n eigenvectors for that matrix. So, a 2 × 2 matrix would have 2 eigenvectors, and a 3 × 3 matrix would have 3 eigenvectors.
4. Even if you scale the vector by some amount before you multiply it, you still get the same multiple of it as a result.
5. All the eigenvectors of a matrix are perpendicular, i.e. at right angles to each other (also called orthogonal to each other), no matter how many dimensions you have. It means that you can express the data in terms of these perpendicular eigenvectors, instead of expressing them in terms of the x and y axes.
6. As you know, the length (or scale) of an eigenvector does not matter. So, in order to keep eigenvectors standard, whenever you find an eigenvector, you usually scale it to make it have a length of 1. It makes the computations easier to do as all eigenvectors have the same length.

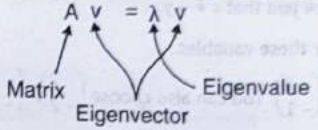
For example, if $\begin{pmatrix}3\\2\end{pmatrix}$ is an eigenvector, then the length of the vector is $\sqrt{3^2 + 2^2} = \sqrt{13}$.

So, you divide the original eigenvector with this value to make its length 1.

Hence, $\begin{pmatrix}3\\2\end{pmatrix}$ can be re-written as $\begin{pmatrix}\frac{3}{\sqrt{13}}\\\frac{2}{\sqrt{13}}\end{pmatrix}$ to make its length as 1.

1.10.7 Eigenvalue

- Eigenvalues are closely related to eigenvectors, in fact, you saw an eigenvalue previously in the eigenvectors section. It was 4. Eigenvalue is the number by which the original vector was scaled after multiplication by the square matrix. No matter what multiple of the eigenvector you took before you multiplied it by the square matrix, you would always get 4 times the scaled vector as your result in that example.
- Mathematically, the relationship between eigenvector and eigenvalue is given as following.



- So, how do you calculate eigenvalue and eigenvector?
- You start with the formula given previously.

$$Av = \lambda v$$

- You can put an identity matrix on the right hand side as identity matrix does not change its value.

$$Av = \lambda Iv$$

Bring all to the left-hand side and take off v assuming that it is non-zero.

$$A - \lambda I = 0$$

You then solve for λ and once you have λ, you can find v.

Let's see an example. Let's take the matrix from eigenvector section and calculate eigenvalue and eigenvector for it.

$$\begin{pmatrix}2 & 3\\2 & 1\end{pmatrix} - \lambda \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix} = 0$$

$$\begin{pmatrix}2-\lambda & 3\\2 & 1-\lambda\end{pmatrix} = 0$$

- The determinant of the matrix $\begin{pmatrix}a & b\\c & d\end{pmatrix}$ could be calculated as ad – bc.

Hence,

$$(2-\lambda)(1-\lambda) - 2 \times 3 = 0$$

$$2 - 2\lambda - \lambda + \lambda^2 - 6 = 0$$

$$\lambda^2 - 3\lambda - 4 = 0$$

Solving this quadratic equation, you get λ = – 1 and λ = 4.

Now, find out eigenvectors for eigenvalues –1 and 4 respectively. Recall that a n × n square matrix has n eigenvectors. So, in this example, you have a 2 × 2 matrix and hence you would have 2 eigenvectors, one for each eigenvalue.

- Let's find the 1st eigenvector corresponding to λ = – 1. Let's call it $v = \begin{pmatrix}x\\y\end{pmatrix}$.

As you know,

$$Av = \lambda v$$

$$\begin{pmatrix}2 & 3\\ 2 & 1\end{pmatrix} \times \begin{pmatrix}x\\ y\end{pmatrix} = -1 \times \begin{pmatrix}x\\ y\end{pmatrix}$$

$$\begin{pmatrix}2x+3y\\ 2x+y\end{pmatrix} + \begin{pmatrix}x\\ y\end{pmatrix} = 0$$

$$\begin{pmatrix}3x+3y\\ 2x+2y\end{pmatrix} = 0$$

So, you now have 2 equations.

$3x + 3y = 0$ and $2x + 2y = 0$ which tell you that $x = -y$.

You can take any non-zero values for these variables.

So, eigenvector for eigenvalue -1 is $\begin{pmatrix}1\\ -1\end{pmatrix}$. You can also choose $\begin{pmatrix}2\\ -2\end{pmatrix}, \begin{pmatrix}3\\ -3\end{pmatrix}, \begin{pmatrix}1000\\ -1000\end{pmatrix}$, literally any value.

Let's test out this eigenvector.

$$\begin{pmatrix}2 & 3\\ 2 & 1\end{pmatrix} \times \begin{pmatrix}1\\ -1\end{pmatrix} = \begin{pmatrix}-1\\ 1\end{pmatrix} = -1 \times \begin{pmatrix}1\\ -1\end{pmatrix}$$

So, it does satisfy the equation $Av = \lambda v$.

- To make the length of this eigenvector as 1, divide the eigenvector with its current length $\sqrt{(-1)^2 + (1)^2} = \sqrt{2}$.

So, the 1st eigenvector with unit length is $\begin{pmatrix}\frac{1}{\sqrt{2}}\\ -\frac{1}{\sqrt{2}}\end{pmatrix}$

- Let's find the 2nd eigenvector corresponding to $\lambda = 4$. Let's call it $v = \begin{pmatrix}x\\ y\end{pmatrix}$.

As you know,

$$Av = \lambda v$$

$$\begin{pmatrix}2 & 3\\ 2 & 1\end{pmatrix} \times \begin{pmatrix}x\\ y\end{pmatrix} = 4 \times \begin{pmatrix}x\\ y\end{pmatrix}$$

$$\begin{pmatrix}2x+3y\\ 2x+y\end{pmatrix} - \begin{pmatrix}4x\\ 4y\end{pmatrix} = 0$$

$$\begin{pmatrix}-2x+3y\\ 2x-3y\end{pmatrix} + \begin{pmatrix}x\\ y\end{pmatrix} = 0$$

So, you now have two equations.

$$-2x + 3y = 0 \text{ and } 2x - 3y = 0.$$

Solving the two equations, you get $x = \frac{3y}{2}$ or $\frac{x}{y} = \frac{3}{2}$.

- Hence, the 2nd eigenvector is $\begin{pmatrix}3\\ 2\end{pmatrix}$. Yes, this is what you know from the previous example as well. You could use any multiple of $\begin{pmatrix}3\\ 2\end{pmatrix}$. So $\begin{pmatrix}3\\ 2\end{pmatrix}, \begin{pmatrix}6\\ 4\end{pmatrix}, \begin{pmatrix}150\\ 100\end{pmatrix}$ are all equivalent and scaled eigenvectors.

- To make the length of this eigenvector as 1, divide the eigenvector with its current length $\sqrt{(3)^2 + (2)^2} = \sqrt{13}$.

So, the 2nd eigenvector with unit length is $\begin{pmatrix}\frac{3}{\sqrt{13}}\\ \frac{2}{\sqrt{13}}\end{pmatrix}$

- So, this is how you calculate eigenvectors and eigenvalues for a given square matrix. Note here that eigenvectors give the direction of transformation whereas eigenvalues give magnitude.

1.10.8 Carrying out PCA

Now that you have a revised the mathematics behind PCA, let's understand how PCA is carried out. PCA is carried out in the following steps

1. Adjust the dataset by subtracting the means of the respective dimensions. This produces a data set whose mean is zero.
2. Calculate the covariance matrix of the data set
3. Calculate the eigenvectors and eigenvalues of the covariance matrix
4. Form a feature vector
5. Derive the new dataset (also called as transformation)

Note here that PCA involves intensive computations. In the real-life scenarios, it is carried out using specific computer programs and mathematical programming libraries.

But, for your understanding, let's solve a simplistic example for carrying out PCA.

Practice Question

Ex. 1.10.1 : Carry out PCA for the following dataset.

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

Soln. :

Step 1 : Adjust the dataset by subtracting the means of the respective dimensions

This produces a data set whose mean is zero.

$$\bar{x} = \frac{2.5 + 0.5 + 2.2 + 1.9 + 3.1 + 2.3 + 2 + 1 + 1.5 + 1.1}{10} = \frac{18.1}{10} = 1.81$$

$$\bar{y} = \frac{2.4 + 0.7 + 2.9 + 2.2 + 3 + 2.7 + 1.6 + 1.1 + 1.6 + 0.9}{10} = \frac{19.1}{10} = 1.91$$

x	y	$x_i - \bar{x}$	$y_i - \bar{y}$
2.5	2.4	0.69	0.49
0.5	0.7	–1.31	–1.21
2.2	2.9	0.39	0.99
1.9	2.2	0.09	0.29
3.1	3	1.29	1.09
2.3	2.7	0.49	0.79
2	1.6	0.19	–0.31
1	1.1	–0.81	–0.81
1.5	1.6	–0.31	–0.31
1.1	0.9	–0.71	–1.01

So, the mean adjusted dataset is as following.

$x_i - \bar{x}$	$y_i - \bar{y}$
0.69	0.49
– 1.31	– 1.21
0.39	0.99
0.09	0.29
1.29	1.09
0.49	0.79
0.19	– 0.31
– 0.81	– 0.81
– 0.31	– 0.31
– 0.71	– 1.01

Step 2 : Calculate the covariance matrix of the adjusted data

This is 2-dimensional data. Its covariance matrix is written as following.

$$C = \begin{pmatrix} \text{cov}(x, x) & \text{cov}(x, y) \\ \text{cov}(y, x) & \text{cov}(y, y) \end{pmatrix}$$

Let's calculate the respective covariance's.

Covariance is mathematically computed as

$$\text{cov}(X, Y) = \frac{\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})}{n-1}$$

$x_i - \bar{x}$	$y_i - \bar{y}$	$(x_i - \bar{x})(y_i - \bar{y})$	$(x_i - \bar{x})^2$	$(y_i - \bar{y})^2$
0.69	0.49	0.3381	0.4761	0.2401
–1.31	–1.21	1.5851	1.7161	1.4641
0.39	0.99	0.3861	0.1521	0.9801
0.09	0.29	0.0261	0.0081	0.0841
1.29	1.09	1.4061	1.6641	1.1881
0.49	0.79	0.3871	0.2401	0.6241
0.19	–0.31	–0.0589	0.0361	0.0961
–0.81	–0.81	0.6561	0.6561	0.6561
–0.31	–0.31	0.0961	0.0961	0.0961
–0.71	–1.01	0.7171	0.5041	1.0201
		Total = 5.539	**Total = 5.549**	**Total = 6.449**

$$\text{cov}(x, y) = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{n-1} = \frac{5.539}{9} = 0.6154$$

$$\text{cov}(x, x) = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1} = \frac{5.549}{9} = 0.6165$$

$$\text{cov}(y, y) = \frac{\sum_{i=1}^{n} (y_i - \bar{y})^2}{n-1} = \frac{6.449}{9} = 0.7165$$

Arranging these covariance's in the covariance matrix format, you get

$$C = \begin{pmatrix} \text{cov}(x, x) & \text{cov}(x, y) \\ \text{cov}(y, x) & \text{cov}(y, y) \end{pmatrix}$$

$$C = \begin{pmatrix} 0.6165 & 0.6154 \\ 0.6154 & 0.7165 \end{pmatrix}$$

Step 3 : Calculate the eigenvectors and eigenvalues of the covariance matrix

First calculate the eigenvalues using the formula $A - \lambda I = 0$

$$\begin{pmatrix} 0.6165 & 0.6154 \\ 0.6154 & 0.7165 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = 0$$

$$\begin{pmatrix} 0.6165 - \lambda & 0.6154 \\ 0.6154 & 0.7165 - \lambda \end{pmatrix} = 0$$

Taking the determinants, you get

$$(0.6165 - \lambda)(0.7165 - \lambda) - 0.6154 \times 0.6154 = 0$$

$$\lambda^2 - 1.333\lambda + 0.063 = 0$$

Solving the quadratic equation, you get

$$\lambda = 0.049 \text{ and } \lambda = 1.28$$

These two are eigenvalues. Now, calculate eigenvectors using the calculated eigenvalues.

Let's find the 1st eigenvector corresponding to $\lambda = 0.049$. Let's call it $v = \begin{pmatrix} x \\ y \end{pmatrix}$.

As you know,

$$Av = \lambda v$$

$$\begin{pmatrix} 0.6165 & 0.6154 \\ 0.6154 & 0.7165 \end{pmatrix} \times \begin{pmatrix} x \\ y \end{pmatrix} = 0.049 \times \begin{pmatrix} x \\ y \end{pmatrix}$$

$$\begin{pmatrix} 0.6165x + 0.6154y \\ 0.6154x + 0.7165y \end{pmatrix} - \begin{pmatrix} 0.049x \\ 0.049y \end{pmatrix} = 0$$

$$\begin{pmatrix} 0.5675x + 0.6154y \\ 0.6154x + 0.6675y \end{pmatrix} = 0$$

So, you now have two equations.

$$0.5675x + 0.6154y = 0$$

$$\text{and } 0.6154x + 0.6675y = 0$$

Solving the two equations, you get $x = -1.08y$

$$\text{Or } \frac{x}{y} = \frac{-1.08}{1}$$

So, the 1st eigenvector is $\begin{pmatrix} -1.08 \\ 1 \end{pmatrix}$. Let's make this eigenvector of unit length by diving the vector by its curren length $\sqrt{(-1.08)^2 + 1^2} = 1.47$

So, the 1st eigenvector with unit length is $\begin{pmatrix} -0.734 \\ 0.68 \end{pmatrix}$

Let's find the 2nd eigenvector corresponding to $\lambda = 1.28$. Let's call it $v = \begin{pmatrix} x \\ y \end{pmatrix}$.

As you know,

$$Av = \lambda v$$

$$\begin{pmatrix} 0.6165 + 0.6154 \\ 0.6154 + 0.7165 \end{pmatrix} \times \begin{pmatrix} x \\ y \end{pmatrix} = 1.28 \times \begin{pmatrix} x \\ y \end{pmatrix}$$

$$\begin{pmatrix} 0.6165x + 0.6154y \\ 0.6154x + 0.7165y \end{pmatrix} - \begin{pmatrix} 1.28x \\ 1.28y \end{pmatrix} = 0$$

$$\begin{pmatrix} -0.6635x + 0.6154y \\ 0.6154x - 0.5635y \end{pmatrix} = 0$$

So, you now have two equations.

$$-0.6635x + 0.6154y = 0$$

$$\text{and } 0.6154x - 0.5635y = 0$$

Solving the two equations, you get $x = 0.927y$

$$\text{or } \frac{x}{y} = \frac{0.927}{1}$$

So, the 2nd eigenvector is $\begin{pmatrix} 0.927 \\ 1 \end{pmatrix}$. Let's make this eigenvector of unit length by diving the vector by its current length $\sqrt{(0.927)^2 + 1^2} = 1.36$.

So, the 2nd eigenvector with unit length is $\begin{pmatrix} 0.681 \\ 0.735 \end{pmatrix}$.

Step 4 : Form a feature vector

So, now you have two eigenvalues and eigenvectors.

Eigenvalue 1 = 0.049 and its corresponding eigenvector 1 = $\begin{pmatrix} -0.734 \\ 0.68 \end{pmatrix}$.

Eigenvalue 2 = 1.28 and its corresponding eigenvector 2 = $\begin{pmatrix} 0.681 \\ 0.735 \end{pmatrix}$

The eigenvector with the highest eigenvalue is the principal component of the data set. In this example, the eigenvector with the largest eigenvalue is the 2nd one. It is the most significant relationship between the data dimensions.

In general, once eigenvectors are found from the covariance matrix, the next step is to order them by eigenvalue, highest to lowest. This gives you the components in order of significance. Now, if you like, you can decide to ignore the components of lesser significance. You do lose some information, but if the eigenvalues are small, you don't lose much. That's how you reduce the number of dimensions in the data set.

So, a feature vector is a matrix of vectors ordered from highest eigenvalue to the lowest.

$$\text{Feature Vector} = (\text{eigenvector}_{\text{highest eigenvalue}}, \ldots, \text{eigenvector}_{\text{lowest eigenvalue}})$$

In this example,

Feature Vector = $\begin{pmatrix} 0.681 \\ 0.735 \end{pmatrix}$ as it corresponds to the highest eigenvalue which is eigenvalue 2 (1.28). If you had 4-dimensional data and you wanted to choose two top-most dimensions, then the feature vector would have two vectors.

Step 5 : Derive the new dataset

This is the final step in PCA. Once you have chosen the principal components (eigenvectors) that you wish to keep, you simply multiply it with the mean adjusted data set.

$$\text{Final Data} = \text{Feature Vector} \times \text{Row Data Adjusted}$$

Final Data is the final data set, with data items in columns, and dimensions along rows.

It will give you the original data solely in terms of the vectors you chose to keep. Your original data set had two axes, x and y, so your data set was in terms of them. You have changed the data from being in terms of the axes x and y, and now they are in terms of two eigenvectors that you computed. In the case of when the new data set has reduced dimensionality, i.e. you have left some of the eigenvectors out, the new data is only in terms of the vectors that you decided to keep.

So, in the nutshell, you transformed your dataset from being represented in terms of dimensions to being represented by the chosen eigenvectors. You have transformed the data set so that it is expressed in terms of the patterns between the dimensions. These patterns closely describe the relationships between the data.

Let's carry out this step.

The chosen Feature Vector = $\begin{pmatrix} 0.681 \\ 0.735 \end{pmatrix}$

$(x_i - \bar{x})$	$(y_i - \bar{y})$	Final data $(x_i - \bar{x}) \times 0.681 + (y_i - \bar{y}) \times 0.735$
0.69	0.49	0.83004
–1.31	–1.21	– 1.78146
0.39	0.99	0.99324
0.09	0.29	0.27444
1.29	1.09	1.67964
0.49	0.79	0.91434
0.19	–0.31	– 0.09846
–0.81	–0.81	– 1.14696
–0.31	–0.31	– 0.43896
–0.71	–1.01	– 1.22586

So, in this example, you have reduced 2-dimensional data to 1-dimensional data.

1.11 Singular Value Decomposition (SVD)

- You can use Singular Value Decomposition (SVD) to represent original data set with a much smaller data set. It is one of the techniques to carry out PCA. Using SVD, you can extract knowledge from data. It is also sometimes referred to as Singular Vector Decomposition.
- The history of the SVD is over a century old. But it has found more use with the adoption of computers in the last several decades. One of first uses was in the field of information retrieval. The method that uses SVD is called latent Semantic Indexing (LSI) or latent semantic analysis. In LSI, a matrix is constructed of documents and words. When the SVD is done on this matrix, it creates a set of singular values.
- The singular values represent concepts or topics contained in the documents. This was developed to allow more efficient searching of documents. A simple search that looks only for the existence of words may have problems if the words are misspelled. Another problem with a simple search is that synonyms may be used and looking for the existence of a word would not tell you if a synonym was used to construct the document.
- If a concept is derived from thousands of similar documents, both of the synonyms will map to the same concept. For example, it is common to divide restaurants by the type of food they serve – Chinese, Italian, North Indian, South Indian, American, etc. When you are searching for a restaurant, you can filter your search based on the dishes you would like to eat.

- For example, if you want to eat Pasta, you will likely search for authentic Italian restaurants in your area. Each restaurant can serve hundreds of food varieties but at a high-level, you can categorise them by a single word such as Chinese and Italian.
- SVD is also commonly used to build recommendation systems. In fact, it became extremely popular after Netflix organising a competition having a prize of $1 million in 2006 to build a recommendation system!!
- The following excerpt is about the Netflix's competition.

> - The concept of a recommender system was first brought to widespread attention by the announcement of the Netflix Prize. In 2006, Netflix announced a competition to produce a better movie rating prediction system than their current one, at the time called Cine match.
> - The idea is that Netflix would like to predict how well its users might like individual movies, so that it could recommend movies to them. The more accurately they would be able to predict users' ratings of movies, the better for their business. The grand prize was $1 million.
> - Each movie that is rated by a user is given a 1 to 5 star rating, 5 being the best. One way to measure the accuracy of a movie rating prediction is to compute the Root-Mean-Square-Error, or RMSE for predictions. That means you add up the square of the errors of all its predictions and take the square root. This is a common metric for measuring errors. Cine match had an RMSE of 0.9514 on the test data, which means that, on average, it predicted within 0.9514 stars of a user's actual rating. The contest was to see (and award $1 million) to a team that could improve that accuracy by at least 10%, meaning an RMSE of 0.8572 or better.
> - The actual contest data looked like this. There are 480,189 users. There are 17,770 movies. If everyone rated every movie, there would be 480,189 x 17,770 ratings, but of course, not everyone watched or rated every movie. They give you 100,480,507 ratings (1-5 each), which is about 1% of all the possible ratings.
> - The idea is that they use about 1.4 million ratings (the test data) that have NOT been provided to you to determine the winners – you have to provide your estimates for those 1.4 million entries, and the RMSE of your estimates provides your final score.
> - There were some other details, like they had a separate 1.4 million ratings (also not provided, the qualifying data) that people could use to submit interim estimates and get scores and be posted on the leader board. This was to prevent people from gaming the system by constantly getting a score on the qualifying data to determine what some of the true answers were.

Interesting, isn't it?

1.11.1 How SVD Works ?

- From you understanding on PCA, you know that oftentimes a few columns in the dataset could be more relevant and containing the most valuable information than others. The other information could be noise or just irrelevant for the overall purpose.
- In linear algebra, there are many techniques for decomposing matrices. The decomposition is done to put the original matrix in a new form that is easier to work with. The new form is a product of two or more matrices. This decomposition can be thought of like factoring in algebra. How can you factor 12 into the product of two numbers? (1,12), (2,6), and (3,4) are all valid answers. The idea behind factoring is that it then becomes easier to work with smaller factored numbers than the equivalent large number.

- The various matrix factorisation techniques have different properties that are more suited for one application or another. One of the most common factorisations is the SVD. The SVD takes an original data set matrix say Data and decomposes it into three matrices denoted by U, ε, and V^T.U and V are orthogonal matrices (i.e. the inverse of the matrix is its transpose, so $U^TU = UU^T = I$, where I is the identity matrix).
  - If the original data set Data is of size m x n, then
  - Left singular vectors U will be m × m
  - Singular value matrix ε will be m × n.
  - Right singular vectors V^T will be n × n
- This can be mathematically written as following.

$$Data_{m \times n} = U_{m \times m} \varepsilon_{m \times n} V^T_{n \times n}$$

- The decomposition creates ε, which has only diagonal elements. All other elements of this matrix are 0. Another convention is that the diagonal elements of ε are sorted from largest to smallest.
- These diagonal elements are called singular value sand they correspond to the singular values of your original data set, Data. If you recall from the previous section on PCA, you found the eigen values of a matrix. These eigen values tell you what features are most important in the given data set.
- The same thing is true about the singular values in ε. The singular values and eigen values are related. Singular values are the square root of the eigen values of Data × $Data^T$. Also, after a certain number of singular values (say r) of a data set, the other values will drop to 0. This means that the data set has only r important features, and the rest of the features are noise or repeats.
- Visually, the SVD could be represented as shown in Fig. 1.11.1.

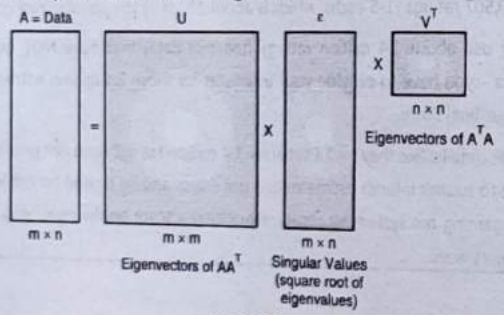


Fig. 1.11.1

Let's see some solved examples for SVD.

Practice Questions

Ex. 1.11.1 : Carry out SVD on $\begin{bmatrix} 1 & 1 \\ 7 & 7 \end{bmatrix}$

Soln. :

$$\text{Let } A = \begin{bmatrix} 1 & 1 \\ 7 & 7 \end{bmatrix}$$

$$\text{Transpose A to get } A^T = \begin{bmatrix} 1 & 7 \\ 1 & 7 \end{bmatrix}$$

Calculate $A \times A^T$

$$A \times A^T = \begin{bmatrix} 1 & 1 \\ 7 & 7 \end{bmatrix} \times \begin{bmatrix} 1 & 7 \\ 1 & 7 \end{bmatrix} = \begin{bmatrix} 2 & 14 \\ 14 & 98 \end{bmatrix}$$

Calculate $A^T \times A$

$$A^T \times A = \begin{bmatrix} 1 & 7 \\ 1 & 7 \end{bmatrix} \times \begin{bmatrix} 1 & 1 \\ 7 & 7 \end{bmatrix} = \begin{bmatrix} 50 & 50 \\ 50 & 50 \end{bmatrix}$$

To find the value of left singular vector U, let's find the eigen values and eigenvectors of $A \times A^T = \begin{bmatrix} 2 & 14 \\ 14 & 98 \end{bmatrix}$

$$\begin{bmatrix} 2-\lambda & 14 \\ 14 & 98-\lambda \end{bmatrix} = 0$$

$$(2-\lambda)(98-\lambda) - 14 \times 14 = 0$$

$$\lambda^2 - 100\lambda + 196 - 196 = 0$$

$$\lambda^2 - 100\lambda = 0$$

Hence $\lambda = 0$ or $\lambda = 100$

Let's find the 1st eigenvector corresponding to $\lambda = 0$. Let's call it $v = \begin{bmatrix} x \\ y \end{bmatrix}$

$$\begin{bmatrix} 2 & 14 \\ 14 & 98 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = 0 \times \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} 2x + 14y \\ 14x + 98y \end{bmatrix} = 0$$

$$\text{Hence, } x = -7y$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -7 \\ 1 \end{bmatrix}$$

To make the length of this eigenvector as 1, divide the eigenvector with its current length

$$\sqrt{(-7)^2 + (1)^2} = \sqrt{50} = 7.07$$

Hence, 1st eigenvector of unit length is

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -7/7.07 \\ 1/7.07 \end{bmatrix} = \begin{bmatrix} -0.99 \\ 0.14 \end{bmatrix}$$

Let's find the 2nd eigenvector corresponding to $\lambda = 100$. Let's call it $v = \begin{bmatrix} x \\ y \end{bmatrix}$

$$\begin{bmatrix} 2 & 14 \\ 14 & 98 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = 100 \times \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} 2x + 14y \\ 14x + 98y \end{bmatrix} - \begin{bmatrix} 100x \\ 100y \end{bmatrix} = 0$$

$$\begin{bmatrix} -98x + 14y \\ 14x - 2y \end{bmatrix} = 0$$

$$\text{Hence, } x = \frac{y}{7}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 7 \end{bmatrix}$$

To make the length of this eigenvector as 1, divide the eigenvector with its current length

$$\sqrt{(1)^2 + (7)^2} = \sqrt{50} = 7.07.$$

Hence, 2nd eigenvector of unit length is

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1/7.07 \\ 7/7.07 \end{bmatrix} = \begin{bmatrix} 0.14 \\ 0.99 \end{bmatrix}$$

You sort the eigenvectors according to eigen values.

Hence,

$$U = \begin{bmatrix} 0.14 & -0.99 \\ 0.99 & 0.14 \end{bmatrix}$$

To find the value of right singular vector V^T, let's find the eigenvalues and eigenvectors of $A^T \times A = \begin{bmatrix} 50 & 50 \\ 50 & 50 \end{bmatrix}$

$$\begin{bmatrix} 50-\lambda & 50 \\ 50 & 50-\lambda \end{bmatrix} = 0$$

$$(50-\lambda)(50-\lambda) - 50 \times 50 = 0$$

$$\lambda^2 - 100\lambda + 2500 - 2500 = 0$$

$$\lambda^2 - 100\lambda = 0$$

Hence $\lambda = 0$ or $\lambda = 100$

Let's find the 1st eigenvector corresponding to $\lambda = 0$. Let's call it $v = \begin{bmatrix} x \\ y \end{bmatrix}$.

$$\begin{bmatrix} 50 & 50 \\ 50 & 50 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = 0 \times \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} 50x + 50y \\ 50x + 50y \end{bmatrix} = 0$$

$$\text{Hence, } x = -y$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

To make the length of this eigenvector as 1, divide the eigenvector with its current length

$$\sqrt{(-1)^2 + (1)^2} = \sqrt{2} = 1.41.$$

Hence, 1st eigenvector of unit length is

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -1/1.41 \\ 1/1.41 \end{bmatrix} = \begin{bmatrix} -0.70 \\ 0.70 \end{bmatrix}$$

Let's find the 2nd eigenvector corresponding to $\lambda = 100$. Let's call it $v = \begin{bmatrix} x \\ y \end{bmatrix}$.

$$\begin{bmatrix} 50 & 50 \\ 50 & 50 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = 100 \times \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} 50x + 50y \\ 50x + 50y \end{bmatrix} - \begin{bmatrix} 100x \\ 100y \end{bmatrix} = 0$$

$$\begin{bmatrix} -50x + 50y \\ 50x - 50y \end{bmatrix} = 0$$

$$\text{Hence, } x = y$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

To make the length of this eigenvector as 1, divide the eigenvector with its current length

$$\sqrt{(1)^2 + (1)^2} = \sqrt{2} = 1.41.$$

Hence, 2nd eigenvector of unit length is

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1/1.41 \\ 1/1.41 \end{bmatrix} = \begin{bmatrix} 0.70 \\ 0.70 \end{bmatrix}$$

You sort the eigenvectors according to eigenvalues.

Hence, $$V^T = \begin{bmatrix} 0.70 & -0.70 \\ 0.70 & 0.70 \end{bmatrix}$$

Singular values are square root of eigenvalues and are sorted in descending order. It is a matrix of m × n where all elements are 0 except the diagonal elements.

$$\varepsilon = \begin{bmatrix} \sqrt{100} & 0 \\ 0 & \sqrt{0} \end{bmatrix} = \begin{bmatrix} 10 & 0 \\ 0 & 0 \end{bmatrix}$$

Hence, SVD of $\begin{bmatrix} 1 & 1 \\ 7 & 7 \end{bmatrix}$ can be written as following.

$$\begin{bmatrix} 1 & 1 \\ 7 & 7 \end{bmatrix} = U \varepsilon V^T = \begin{bmatrix} 0.14 & -0.99 \\ 0.99 & 0.14 \end{bmatrix} \times \begin{bmatrix} 10 & 0 \\ 0 & 0 \end{bmatrix} \times \begin{bmatrix} 0.70 & -0.70 \\ 0.70 & 0.70 \end{bmatrix}$$

Just for fun and confirmation, if you go ahead and multiply the matrices you got after decomposition, you will get the original matrix (very close values).

So, $\begin{bmatrix} 0.14 & -0.99 \\ 0.99 & 0.14 \end{bmatrix} \times \begin{bmatrix} 10 & 0 \\ 0 & 0 \end{bmatrix} \times \begin{bmatrix} 0.70 & -0.70 \\ 0.70 & 0.70 \end{bmatrix}$ will give you $\begin{bmatrix} 0.98 & -0.98 \\ 6.93 & -6.93 \end{bmatrix}$ which is very close to the matrix that you started with $\begin{bmatrix} 1 & 1 \\ 7 & 7 \end{bmatrix}$

Note here that signs of matrices U and V do not matter. You may get different signs. As long as the absolute values are correct, it is a correct SVD.

Ex. 1.11.2 : Find the singular value decomposition of $A = \begin{bmatrix} 2 & 2 \\ -1 & 1 \end{bmatrix}$

Soln. :

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

Transpose A to get $A^T = \begin{bmatrix} 2 & -1 \\ 2 & 1 \end{bmatrix}$

Calculate $A \times A^T$ $$A \times A^T = \begin{bmatrix} 2 & 2 \\ -1 & 1 \end{bmatrix} \times \begin{bmatrix} 2 & -1 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 8 & 0 \\ 0 & 2 \end{bmatrix}$$

Calculate $A^T \times A$ $$A^T \times A = \begin{bmatrix} 2 & -1 \\ 2 & 1 \end{bmatrix} \times \begin{bmatrix} 2 & 2 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix}$$

To find the value of left singular vector U, let's find the eigenvalues and eigenvectors of $A \times A^T = \begin{bmatrix} 8 & 0 \\ 0 & 2 \end{bmatrix}$

$$\begin{bmatrix} 8-\lambda & 0 \\ 0 & 2-\lambda \end{bmatrix} = 0$$

$$(8-\lambda)(2-\lambda) - 0 \times 0 = 0$$

Hence $\lambda = 8$ or $\lambda = 2$

Let's find the 1st eigenvector corresponding to $\lambda = 8$. Let's call it $v = \begin{bmatrix} x \\ y \end{bmatrix}$.

$$\begin{bmatrix} 8 & 0 \\ 0 & 2 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = 8 \times \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} 8x \\ 2y \end{bmatrix} - \begin{bmatrix} 8x \\ 8y \end{bmatrix} = 0$$

Hence, you get two equations.

$$8x - 8x = 0 \text{ and } 2y - 8y = 0$$

From these equations, you can pick the value of $x = 1$ and y could only be 0.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

To make the length of this eigenvector as 1, divide the eigenvector with its current length $\sqrt{(1)^2 + (0)^2} = 1$.

Hence, 1st eigenvector of unit length is

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1/1 \\ 0/1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Let's find the 2nd eigenvector corresponding to $\lambda = 2$. Let's call it $v = \begin{bmatrix} x \\ y \end{bmatrix}$.

$$\begin{bmatrix} 8 & 0 \\ 0 & 2 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = 2 \times \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} 8x \\ 2y \end{bmatrix} - \begin{bmatrix} 2x \\ 2y \end{bmatrix} = 0$$

Hence, you get two equations.

$$8x - 2x = 0 \text{ and } 2y - 2y = 0$$

From these equations, you can pick the value of $y = 1$ and x could only be 0.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

To make the length of this eigenvector as 1, divide the eigenvector with its current length $\sqrt{(0)^2 + (1)^2} = 1$.

Hence, 2nd eigenvector of unit length is

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0/1 \\ 1/1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

You sort the eigenvectors according to eigenvalues.

Hence,

$$U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

To find the value of right singular vector V^T, let's find the eigenvalues and eigenvectors of $A^T \times A = \begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix}$

$$\begin{bmatrix} 5-\lambda & 3 \\ 3 & 5-\lambda \end{bmatrix} = 0$$

$$(5-\lambda)(5-\lambda) - 3 \times 3 = 0$$

$$\lambda^2 - 10\lambda + 25 - 9 = 0$$

$$\lambda^2 - 10\lambda + 16 = 0$$

Hence $\lambda = 8$ or $\lambda = 2$

Let's find the 1st eigenvector corresponding to $\lambda = 8$. Let's call it $v = \begin{bmatrix} x \\ y \end{bmatrix}$.

$$\begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = 8 \times \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} 5x + 3y \\ 3x + 5y \end{bmatrix} - \begin{bmatrix} 8x \\ 8y \end{bmatrix} = 0$$

$$\begin{bmatrix} -3x + 3y \\ 3x - 3y \end{bmatrix} = 0$$

Hence, $x = y$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

To make the length of this eigenvector as 1, divide the eigenvector with its current length

$$\sqrt{(1)^2 + (1)^2} = \sqrt{2} = 1.41.$$

Hence, 1st eigenvector of unit length is

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1/1.41 \\ 1/1.41 \end{bmatrix} = \begin{bmatrix} 0.70 \\ 0.70 \end{bmatrix}$$

Let's find the 2nd eigenvector corresponding to $\lambda = 2$. Let's call it $v = \begin{bmatrix} x \\ y \end{bmatrix}$.

$$\begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = 2 \times \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} 5x + 3y \\ 3x + 5y \end{bmatrix} - \begin{bmatrix} 2x \\ 2y \end{bmatrix} = 0$$

$$\begin{bmatrix} 3x + 3y \\ 3x + 3y \end{bmatrix} = 0$$

Hence, $x = -y$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

To make the length of this eigenvector as 1, divide the eigenvector with its current length

$$\sqrt{(-1)^2 + (1)^2} = \sqrt{2} = 1.41.$$

Hence, 2nd eigenvector of unit length is

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -1/1.41 \\ 1/1.41 \end{bmatrix} = \begin{bmatrix} -0.70 \\ 0.70 \end{bmatrix}$$

You sort the eigenvectors according to eigenvalues.

Hence,

$$V^T = \begin{bmatrix} 0.70 & -0.70 \\ 0.70 & 0.70 \end{bmatrix}$$

Singular values are square root of eigenvalues and are sorted in descending order. It is a matrix of $m \times n$ where all elements are 0 except the diagonal elements.

$$\varepsilon = \begin{bmatrix} \sqrt{8} & 0 \\ 0 & \sqrt{2} \end{bmatrix} = \begin{bmatrix} 2.82 & 0 \\ 0 & 1.41 \end{bmatrix}$$

Hence, SVD of $\begin{bmatrix} 2 & 2 \\ -1 & 1 \end{bmatrix}$ can be written as following.

$$\begin{bmatrix} 2 & 2 \\ -1 & 1 \end{bmatrix} = U\varepsilon V^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} 2.82 & 0 \\ 0 & 1.41 \end{bmatrix} \times \begin{bmatrix} 0.70 & -0.70 \\ 0.70 & 0.70 \end{bmatrix}$$

Just for fun and confirmation, if you go ahead and multiply the matrices you got after decomposition, you will ge the original matrix (very close values).

So, $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} 2.82 & 0 \\ 0 & 1.41 \end{bmatrix} \times \begin{bmatrix} 0.70 & -070 \\ 0.70 & 0.70 \end{bmatrix}$ will give you $\begin{bmatrix} 1.974 & -1.974 \\ 0.987 & 0.987 \end{bmatrix}$ which is very close to the matrix that you started with $\begin{bmatrix} 2 & 2 \\ -1 & 1 \end{bmatrix}$.

Note here that signs of matrices U and V do not matter. You may get different signs. As long as the absolute value are correct, it is a correct SVD.

1.11.2 Linear Discriminant Analysis (LDA)

✍ ***Definition :*** *Linear Discriminant Analysis (LDA) is a supervised learning method for dimensionality reduction for classification problems.*

- For example, if you have two categories of data on a 2D plot, then LDA helps you to plot the data in 1-dimensiona axis so as to clearly separate the data between the two categories.
- The general LDA approach is very similar to a Principal Component Analysis (PCA) that you learnt earlier. But i addition to finding the component axes that maximise the variance of data as in PCA, you also want to derive th axes that maximise the separation between multiple classes in LDA.

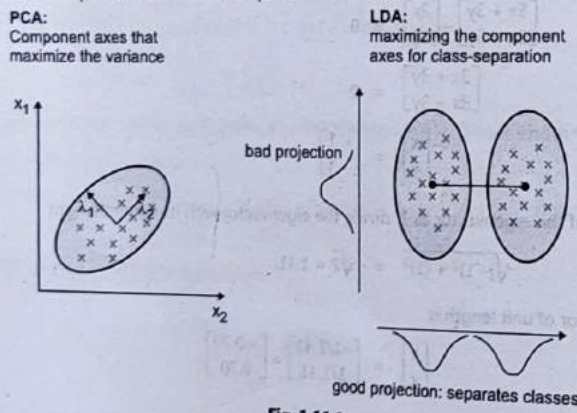


Fig. 1.11.1

- Given samples from two classes C_1 and C_2, you want to find the direction, as defined by a vector w, such that when the data are projected onto w, then the data points from the two classes are as well separated as possible.
- The Fig. 1.11.2 illustrates two-dimensional two-class data projected on w.

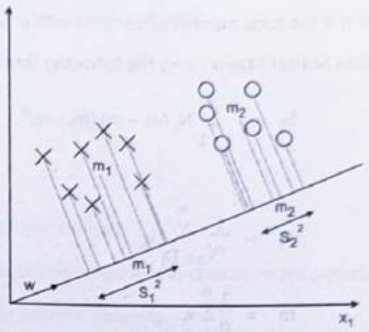


Fig. 1.11.2

After projection, for the two classes to be well separated, you would like their means to be as far apart as possible and the data points of classes be scattered in as small a region as possible. You perform LDA so that you can clearly distinguish the categories of data and within the categories there is as less variance as possible.

Hence, the two criteria used by LDA to create a new axis are as following.

1. Maximise the distance between means of the two classes.
2. Minimise the variation within each class.

LDA is thus the projected axis w that maximises the following function.

$$J(w) = \frac{(m_1 - m_2)^2}{S_1 + S_2}$$

where, m_1 and m_2 are the respective means of the samples and S_1 and S_2 are the respective scatters of the samples from C_1 and C_2 after projection. You want to maximise $(m_1 - m_2)^2$ and minimise $S_1 + S_2$.

At a high-level, following steps are carried out for LDA.

1. Compute the within class and between class scatter matrices.
2. Compute the eigenvectors and corresponding eigen values for the scatter matrices.
3. Sort the eigen values and select the top values based on number of dimensions you want.
4. Create a new matrix containing eigenvectors that map to the eigen values.
5. Obtain the new features (i.e. LDA components) by taking the dot product of the data and the matrix from step 4.

You can calculate the Within Class Scatter Matrix using the following formula.

$$S_w = \sum_{i=1}^{c} S_i$$

Where,

c is the total number of distinct classes

$$S_i = \sum_{x \in D_i}^{n} (x - m_i)(x - m_i)^T$$

$$m_i = \frac{1}{n_i} \sum_{x \in D_i}^{n} x_k$$

where x is a sample (i.e. row) and n is the total number of samples with a given class.

You can calculate the Between Class Scatter Matrix using the following formula.

$$S_B = \sum_{i=1}^{c} N_i (m_i - m)(m_i - m)^T$$

where,

$$m_i = \frac{1}{n_i} \sum_{x \in D_i}^{n} x_k$$

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

Review Questions

Here are a few review questions to help you gauge your understanding of this chapter. Try to attempt these questions and ensure that you can recall the points mentioned in the chapter.

Introduction to Machine Learning

Q. 1	Define Machine Learning. Explain the core idea behind machine learning.	(4 Marks)
Q. 2	How are experience, performance, and tasks related with respect to machine learning.	(4 Marks)
Q. 3	With a block diagram, explain how machine learning works at a high-level.	(6 Marks)
Q. 4	Explain the key terms associated with machine learning.	(4 Marks)
Q. 5	Explain the various Big Data formats.	(6 Marks)
Q. 6	Explain the various training data formats that could be used for machine learning.	(6 Marks)
Q. 7	With examples, explain the types of Big Data formats.	(6 Marks)
Q. 8	Compare various Data Formats that could be used in machine learning.	(6 Marks)
Q. 9	Compare structured, semi-structured and unstructured data.	(6 Marks)
Q. 10	Write a short note on DIKW pyramid.	(6 Marks)
Q. 11	Draw DIKW pyramid and explain.	(6 Marks)
Q. 12	Explain the journey of data enrichment as it moves from hindsight to foresight.	(6 Marks)
Q. 13	Describe the various categories of data analytics.	(6 Marks)
Q. 14	What types of analytics can be performed on a dataset?	(6 Marks)
Q. 15	Write a short note on Descriptive Analytics.	(4 Marks)
Q. 16	Write a short note on Diagnostic Analytics.	(4 Marks)
Q. 17	Write a short note on Predictive Analytics.	(4 Marks)
Q. 18	Write a short note on Prescriptive Analytics.	(4 Marks)
Q. 19	Compare the various categories of data analytics.	(6 Marks)

Q. 20	Explain the various types of machine learning algorithms.	(6 Marks)
Q. 21	Explain supervised learning.	(4 Marks)
Q. 22	Explain semi-supervised learning.	(4 Marks)
Q. 23	Explain unsupervised learning.	(4 Marks)
Q. 24	Explain reinforcement learning.	(4 Marks)
Q. 25	Compare supervised and unsupervised learning.	(4 Marks)
Q. 26	Joe is confused on which machine learning algorithm to choose for his project. What do you tell him?	(4 Marks)
Q. 27	Compare Artificial Intelligence and Machine Learning.	(4 Marks)
Q. 28	Compare Data Mining and Machine Learning.	(4 Marks)
Q. 29	Compare Artificial Intelligence and Data Science.	(4 Marks)
Q. 30	Compare Machine Learning and Traditional Programming.	(4 Marks)
Q. 31	Explain common issues in machine learning.	(6 Marks)
Q. 32	Explain common challenges in machine learning.	(6 Marks)
Q. 33	Describe a few applications of machine learning.	(6 Marks)
Q. 34	John is suggesting that he built a machine learning model using a training dataset of 10 examples. Would you trust such a model?	(4 Marks)
Q. 35	Machine learning models could be biased. Comment.	(4 Marks)
Q. 36	Describe the various steps involved in developing a machine learning application.	(6 Marks)
Q. 37	You could collect training dataset internally or externally. Comment.	(4 Marks)

Models of Machine learning

Q. 38	Write a short note on geometric models.	(4 Marks)
Q. 39	Write a short note on probabilistic models.	(4 Marks)
Q. 40	Write a short note on logical models.	(4 Marks)
Q. 41	Write a short note on grouping and grading models.	(4 Marks)
Q. 42	Write a short note on parametric and non-parametric models.	(4 Marks)

Feature and Feature Engineering

Q. 43	What is feature engineering?	(4 Marks)
Q. 44	With a flow-chart, explain the high-level process of feature engineering.	(6 Marks)
Q. 45	What does feature engineering typically include?	(4 Marks)
Q. 46	Write a short note on feature engineering.	(4 Marks)
Q. 47	Write a short note on feature transformation.	(4 Marks)
Q. 48	With a suitable example, explain binning.	(6 Marks)

Note

2 Regression

Unit 2

Syllabus

At the end of this unit, you should be able to understand and comprehend the following syllabus topics:

- Introduction
  - Regression
  - Need of Regression
  - Difference between Regression and Correlation
  - Types of Regression
    - Univariate vs. Multivariate
    - Linear vs. Nonlinear
    - Simple Linear vs. Multiple Linear
  - Bias-Variance trade-off
    - Overfitting
    - Underfitting
- Regression Techniques
  - Polynomial Regression
  - Stepwise Regression
  - Decision Tree Regression
  - Random Forest Regression
  - Support Vector Regression
  - Ridge Regression
  - Lasso Regression
  - ElasticNet Regression
  - Bayesian Linear Regression
- Evaluation Metrics
  - Mean Squared Error (MSE)
  - Mean Absolute Error (MAE)
  - Root Mean Squared Error (RMSE)
  - R-squared
  - Adjusted R-squared

2.1 Regression Analysis

The statistical definition of regression analysis is

✍ ***Definition :*** *A functional relationship between two or more correlated variables that is often empirically determined from data and is used specially to predict values of one variable when given values of the others.*

- Do not panic! You do this almost every day. For example, when you visit a petrol bunk, you have a good estimate of how much you would likely drive in the next few days and how much petrol you would need correspondingly, isn't it? How do you predict your petrol usage? Based on your past data points, right? That is precisely what regression analysis is.
- You try to establish a correlation between one variable (independent variable) with another variable (dependent variable) and use that relationship to predict other values.

  So, for example,

  - Petrol required (X) = 5 litres
  - Kilometres driven (Y) = 50
  - Relationship : Y = 10X
  - Prediction : for 10 litres of petrol, you could go around 10× 10 = 100 Kms.
- In a nutshell, regression analysis aims to establish a relationship (or influence) of a set of variables on another variable (outcome).

✍ ***Definition :*** *The variables that have influence on the outcome are called input, independent, explanatory or predictor variable.*

- In the previous example, the number of litres of petrol (X) is the variable that influences or controls the number of kilometres that you may drive. Here, the variable X is an independent variable.

✍ ***Definition :*** *The variable whose outcome (or value) depends on other variables is called response, outcome, or dependent variable.*

- In the previous example, number of kilometres (Y) is a dependent variable.
- Now, you may argue with me that it is not always true that you get exact same mileage all the time. It also depends on traffic jams, speed at which you drive, quality of petrol, condition of the automobile engine, time of the day, weather conditions, road conditions, etc. I get that.
- Wish life was so simple all the time! That is the precise idea of doing regression analysis identify the complex relationship between variables and try to establish it as closely as possible. It is obvious that there is nothing as 100% accurate relationship and prediction.
- Any relationship and predictions based on that would have errors. It is just important that those errors are minimised as much as possible is not that true for life as well? Philosophy later on my friend, back to data analytics.
- Now that you basically understand what a regression analysis, let's learn about a few regression analysis techniques.

2.1.1 Why Do We Need Regression Analysis ?

- As you now understand, regression analysis is a statistical technique used to model and analyse the relationship between a dependent variable and one or more independent variables. It is widely used in various fields, including economics, finance, social sciences, engineering, and business.
- Primary reasons why you need regression analysis are as following.

1. **Identifying Relationships**

   Regression analysis helps you to understand the nature and strength of the relationship between variables. It allows you to determine how changes in one variable affect another. By estimating the coefficients, you can quantify the impact of independent variables on the dependent variable.

2. **Prediction and forecasting**

   Regression models can be used for prediction and forecasting purposes. Once a relationship between variables is established, you can use the model to estimate the values of the dependent variable based on the values of the independent variables. This is particularly useful when trying to predict outcomes or trends based on historical data.

3. **Causal inference**

   Regression analysis can help infer causal relationships between variables. While correlation does not imply causation, regression analysis provides a framework to control for other factors and establish a more robust causal relationship. This is achieved by including relevant independent variables and using appropriate statistical techniques.

4. **Variable selection**

   Regression analysis allows you to determine which independent variables are most relevant in explaining the variation in the dependent variable. Through statistical tests and measures such as p - values and adjusted R - squared, you can identify significant predictors and eliminate irrelevant variables, thus simplifying the model.

5. **Model assessment**

   Regression analysis provides tools to assess the goodness of fit and validity of the model. Measures like R-squared, adjusted R-squared, and hypothesis testing help evaluate how well the model fits the data and whether the estimated coefficients are statistically significant.

6. **Decision-making and policy analysis**

   Regression analysis provides insights that aid decision-making processes. For example, it can be used to analyse the impact of price changes on demand, evaluate the effectiveness of marketing campaigns, or assess the influence of policy interventions. By understanding the relationships between variables, organisations can make informed decisions and formulate evidence-based policies.

2.1.2 Difference between Regression and Correlation

The Table 2.1.1 provides a quick comparison between regression and correlation.

Table 2.1.1

Comparison Attributes	Regression Analysis	Correlation
Purpose	To model and analyse the relationship between variables and predict the dependent variable.	To measure the strength and direction of the linear relationship between variables.
Focus	Dependent variable and its relationship with independent variables	Association between variables, without determining cause and effect.
Goal	Understand the impact of independent variables on the dependent variable and make predictions.	Measure the degree of association between variables, typically using correlation coefficients.
Direction	Can determine the direction and nature (positive or negative) of the relationship.	Only determines the direction (positive or negative) of the linear relationship.
Causality	Can infer causality and determine the cause and effect relationship between variables.	Does not establish causality. It simply shows the strength of the linear relationship.
Analysis Technique	Estimates coefficients for independent variables and assesses their statistical significance. Measures the goodness of fit and evaluates the model's validity.	Calculates correlation coefficients (Pearson's correlation coefficient) to quantify the linear relationship.

2.1.3 Types of Regression

Regression analysis could be of various types. Let's learn about them briefly.

Univariate vs. Multivariate

- Univariate regression analysis (also called as Simple Linear regression analysis) is a statistical technique that examines the relationship between a single dependent variable and one independent variable. It aims to understand how changes in the independent variable affect the dependent variable and to predict the values of the dependent variable based on the values of the independent variable. For example, let's say you want to examine the relationship between the number of hours studied and the exam scores of a group of students. The number of hours studied is the independent variable, and the exam scores are the dependent variable.
- Multivariate regression analysis is a statistical technique that examines the relationship between multiple independent variables and multiple dependent variables simultaneously. It allows you to understand how multiple predictors collectively affect multiple response variables.

Note : Many resources use multivariate and multiple regression interchangeably. That is incorrect. Multiple regression has just one output variable whereas multivariate regression could have several output variables. For both multiple regression and multivariate regressions, there are multiple input variables. So, just be mindful of the terms. Multivariate is not the same thing as multiple.

- So, you understand that in a multivariate regression there are more than one independent (or input) variables and multiple possible outputs (response). The multivariate regression model can be represented as the following.

  Let,
- Y be the $n \times p$ response matrix.
- X be an $n \times (q + 1)$ matrix such that all entries of the first column are 1s and q predictors (input variables).
- β be an $(q + 1) \times p$ matrix of fixed parameters called regression coefficients.
- e be an $n \times p$ matrix for error.

  Then, the simple form of model representation is given as following.

$$Y = \beta X + \epsilon$$

In the matrix (expanded) form, the model representation is given as shown in Fig. 2.1.1.

$$\begin{pmatrix} y_{11} & y_{12} & \cdots & y_{1p} \\ y_{21} & y_{22} & \cdots & y_{2p} \\ y_{31} & y_{32} & \cdots & y_{3p} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n1} & y_{n2} & \cdots & y_{np} \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1q} \\ 1 & x_{21} & x_{22} & \cdots & x_{2q} \\ 1 & x_{31} & x_{32} & \cdots & x_{3q} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nq} \end{pmatrix} \begin{pmatrix} \beta_{01} & \beta_{02} & \cdots & \beta_{0p} \\ \beta_{11} & \beta_{12} & \cdots & \beta_{1p} \\ \beta_{21} & \beta_{22} & \cdots & \beta_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{q1} & \beta_{q2} & \cdots & \beta_{qp} \end{pmatrix} + \begin{pmatrix} \epsilon_{11} & \epsilon_{12} & \cdots & \epsilon_{1p} \\ \epsilon_{21} & \epsilon_{22} & \cdots & \epsilon_{2p} \\ \epsilon_{31} & \epsilon_{32} & \cdots & \epsilon_{3p} \\ \vdots & \vdots & \ddots & \vdots \\ \epsilon_{n1} & \epsilon_{n2} & \cdots & \epsilon_{np} \end{pmatrix}$$

Fig. 2.1.1

- So, how do you know where to apply multiple regression and where multivariate regression? Let's take an example to understand.
- For example, suppose that a university wishes to refine its admission criteria so that they admit 'better' students. Also, suppose that a student's Grade Point Average (GPA) is what the university wishes to use as a performance metric for students. They have several criteria in mind such as high school GPA (HSGPA), SAT scores (SAT), Gender etc and would like to know which one of these criteria matter as far as GPA is concerned.
- In this case, you would use multiple regression as there is one dependent variable (GPA) and you have multiple independent variables (HSGPA, SAT, Gender etc). You want to find out which one of the independent variables are good predictors for your dependent variable. You would use multiple regression to make this assessment.
- Now consider another example. Instead of the above situation, suppose the admissions office wants to track student performance across time and wishes to determine which one of their criteria drives student performance across time. In other words, they have GPA scores for the four years that a student stays in school (say, GPA1, GPA2, GPA3, GPA4) and they want to know which one of the independent variables predict GPA scores better on a year-by-year basis. The admissions office hopes to find that the same independent variables predict performance across all four years so that their choice of admissions criteria ensures that student performance is consistently high across all four years.
- In this case, you would use multivariate regression as you have multiple dependent variables (i.e., GPA1, GPA2, GPA3, GPA4) and multiple independent variables.
- The Table 2.1.2 provides a quick comparison between univariate and multivariate regression analysis.

Table 2.1.2

Comparison Attribute	Univariate Regression Analysis	Multivariate Regression Analysis
Number of dependent variables	One	Multiple
Number of independent variables	One	Multiple
Purpose	Examine the relationship between a single independent variable and the dependent variable, make predictions, and assess the impact of the independent variable.	Examine the relationship between multiple independent variables and multiple dependent variables.
Example	Examining the relationship between study hours and exam scores.	Predicting housing prices based on square footage, number of bedrooms, proximity to amenities, and other factors.
Regression Equation	$Y = \beta_0 + \beta_1 X$ (Where Y is the dependent variable, X is the independent variable, β_0 and β_1 are coefficients.)	$Y = \beta X + \epsilon$
Model Complexity	Simpler model with fewer independent variables.	More complex model that considers the joint effects of multiple independent variables on multiple dependent variables.

Linear vs. Nonlinear

- Linear regression is a statistical technique that is used to model the relationship between two variables by fitting a straight line to the data points. It assumes that there is a linear relationship between the input variable (independent variable) and the output variable (dependent variable). The goal is to find the best-fitting line that minimises the distance between the line and the data points.
- For example, let's say you have a dataset of house prices and their corresponding sizes (in square feet). In linear regression, you would plot the house sizes on the x-axis and the prices on the y-axis. Then, you would draw a straight line that best represents the relationship between house size and price. This line can be used to predict the price of a house given its size.
- Nonlinear regression is used when the relationship between the variables cannot be adequately represented by a straight line. In this case, the relationship is described by a nonlinear equation or a curved line. Nonlinear regression aims to find the best-fitting curve that minimises the difference between the curve and the data points.
- An example of nonlinear regression is when you have a dataset of temperature readings and the corresponding time. If you plot the temperature on the y-axis and time on the x-axis, you might observe that the temperature initially increases rapidly and then levels off over time. In this case, a straight line wouldn't accurately represent the relationship. Instead, a curve (such as an exponential, logarithmic, or sigmoid curve) could better capture the pattern and predict the temperature at different time points.

- The Table 2.1.3 provides a quick comparison between simple linear and nonlinear regression analysis.

Table 2.1.3

Comparison Attributes	Linear Regression	Nonlinear Regression
Relationship	Assumes a linear relationship	Assumes a nonlinear relationship
Linearity	Uses a straight line as a model	Uses curves to model the relationship
Equation	$y = mx + b$	$y = f(x, \beta)$
Complexity	Simpler and easier to interpret	Can be more complex and harder to interpret
Assumptions	Assumes constant variance and independence of residuals	Assumptions may vary depending on the specific nonlinear model
Model Selection	Limited to linear relationships	Can handle a wide range of relationships
Interpretation	Coefficients represent the slope and intercept of the line	Interpretation can vary depending on the specific nonlinear equation
Flexibility	Limited flexibility in modelling complex relationships	More flexibility in capturing complex relationships
Performance	Good for simple, linear relationships	Better for capturing nonlinear patterns
Examples	Predicting house prices based on size	Modelling the growth rate of a population over time

Simple Linear vs. Multiple Linear

Simple Linear regression analysis (also called as Univariate regression analysis) is a statistical technique that examines the relationship between a single dependent variable and one independent variable. It aims to understand how changes in the independent variable affect the dependent variable and to predict the values of the dependent variable based on the values of the independent variable. For example, let's say you want to examine the relationship between the number of hours studied and the exam scores of a group of students. The number of hours studied is the independent variable, and the exam scores are the dependent variable.

Multiple Linear regression analysis is a statistical technique that examines the relationship between a single dependent variable and multiple independent variables. It allows for the simultaneous consideration of the effects of multiple predictors on the dependent variable. For example, suppose that you want to predict housing prices based on various factors such as square footage (X_1), number of bedrooms (X_2), and proximity to the city center (X_3). Here, housing price is the dependent variable (Y), and square footage, number of bedrooms, and proximity to the city center are the independent variables.

The Table 2.1.4 provides a quick comparison between simple linear and multiple linear regression analysis.

Table 2.1.4

Comparison Attribute	Simple Linear Regression Analysis	Multiple Linear Regression Analysis
Number of dependent variables	One	One
Number of independent variables	One	Multiple
Purpose	Examine the relationship between a single independent variable and the dependent variable, make predictions, and assess the impact of the independent variable.	Examine the relationship between a dependent variable and multiple independent variables to understand how multiple factors affect the dependent variable.
Example	Examining the relationship between study hours and exam scores.	Predicting housing prices based on square footage, number of bedrooms, proximity to amenities, and other factors.
Regression Equation	$Y = \beta_0 + \beta_1 X$ (Where Y is the dependent variable, X is the independent variable, β_0 and β_1 are coefficients.)	$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta \|\|X\|\|$ (Where Y is the dependent variable, X_1, X_2, ..., $X\|\|$ are the independent variables, β_0, β_1, ..., $\beta\|\|$ are the coefficients.)
Model Complexity	Simpler model with fewer independent variables.	More complex model that considers the joint effects of multiple independent variables.

2.1.4 Linear Regression

In simple words,

Definition : The process of finding a straight line that best approximates a set of points on a graph is called linear regression.

The word linear signifies that the type of relationship that you try to establish between the variables tends to be a straight line. Linear regression is one of the most simple and popular techniques of regression analysis.

There are two types of linear regression.

1. **Simple Linear Regression (SLR) :** This has only one independent (or input) variable. For example, number of litres of petrol and kilometres driven.
2. **Multiple Linear Regression (MLR) :** This has more than one independent (or input) variables. For example, number of litres of petrol, age of the vehicle, speed and kilometres driven.

The general formula (or model) for linear regression analysis is

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + ... \beta_n X_n + \varepsilon$$

Where,

- Y is the outcome (dependent) variable
- X_i are the values of independent variables
- β_0 is the value of Y when each X_i is equal to 0. It is also called as y-intercept
- β_i is the change in Y based on the unit change in X_i. It is also called as regression coefficient or slope of the regression line.
- ε is the random error or noise that represents the difference between the predicted value (based on the regression model) and actual value.

- For Simple Linear Regression (just one input or independent variable), the formula is exactly what you learnt in your school (remember y = mx + c)? It is

$$Y = \beta_0 + \beta_1 X_1 + \varepsilon$$

Most of the times, ε is omitted from the calculation to give the simple formula as

$$Y = \beta_0 + \beta_1 X_1$$

The goal is to find the regression line that best approximates (or fits) the relation between the input variable and output variable. You are required to calculate the values of β_0 and β_1(the regression coefficients). To do so, you could use the least square method in which you calculate the square of distance between each observed point and the probable regression line. The objective is to find a linear model where the sum of squares of the distances is minimal.

To calculate β_0 and β_1, you can use the following formulae.

$$\beta_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

where $\bar{x}$ is the mean of x and $\bar{y}$ is the mean of y.

$$\beta_o = \bar{y} - \beta_1 \bar{x}$$

Practice Questions

Ex. 2.1.1 : For the following data set, find the linear regression line. Predict the value of Y if X = 10.

X	Y
0	1
1	3
2	2
3	5
4	7
5	8
6	8
7	9
8	10
9	12

Soln. :

Calculate the mean of X and Y and then calculate the other values as required.

X	Y	$X_i - \bar{X}$	$Y_i - \bar{Y}$	$(X_i - \bar{X}) \times (Y_i - \bar{Y})$	$(X_i - \bar{X})^2$
0	1	– 4.5	– 5.5	24.75	20.25
1	3	– 3.5	– 3.5	12.25	12.25
2	2	– 2.5	– 4.5	11.25	6.25
3	5	– 1.5	– 1.5	2.25	2.25
4	7	– 0.5	0.5	– 0.25	0.25
5	8	0.5	1.5	0.75	0.25
6	8	1.5	1.5	2.25	2.25
7	9	2.5	2.5	6.25	6.25
8	10	3.5	3.5	12.25	12.25
9	12	4.5	5.5	24.75	20.25
				Total = 96.5	**Total = 82.5**
$\bar{X}$**(Mean of X) = 4.5**	$\bar{Y}$**(Mean of Y) = 6.5**				

Hence,

$$\beta_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

$$\beta_1 = \frac{96.5}{82.5} = 1.1696$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

$$\beta_0 = 6.5 - 1.1696 \times 4.5 = 1.2368$$

Hence, the regression line is

$$Y = 1.2368 + 1.1696X$$

A sample plot of the regression line is as shown in Fig. P. 2.1.1.

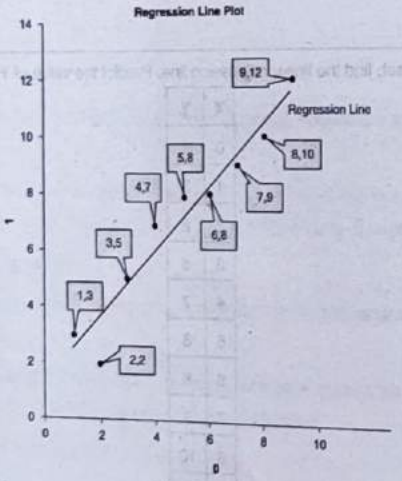


Fig. P. 2.1.1

To predict the value of Y when X = 10, put the value of X in the regression line. You get

$Y = 1.2368 + 1.1696 \times 10 = 12.932$ or 13 when rounded.

Ex. 2.1.2 : A clinical trial gave the following data for BMI and Cholesterol level for 10 patients. Predict the likely value of cholesterol level for someone who has a BMI of 27.

BMI	Cholesterol
17	140
21	189
24	210
28	240
14	130
16	100
19	135
22	166
15	130
18	170

Soln. :

Calculate the mean of BMI and Cholesterol and then calculate the other values as required.

BMI	Cholesterol	$X_i - \bar{X}$	$Y_i - \bar{Y}$	$(X_i - \bar{X}) \times (Y_i - \bar{Y})$	$(X_i - \bar{X})^2$
17	140	– 2.4	– 21	50.4	5.76
21	189	1.6	28	44.8	2.56
24	210	4.6	49	225.4	21.16
28	240	8.6	79	679.4	73.96
14	130	– 5.4	– 31	167.4	29.16
16	100	– 3.4	– 61	207.4	11.56
19	135	– 0.4	– 26	10.4	0.16
22	166	2.6	5	13	6.76
15	130	– 4.4	– 31	136.4	19.36
18	170	– 1.4	9	– 12.6	1.96
$\bar{X}$**(Mean of X) = 19.4**	$\bar{Y}$**(Mean of Y) = 161**			**Total = 1522**	**Total = 172.4**

Hence,

$$\beta_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

$$\beta_1 = \frac{1522}{172.4} = 8.8283$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

$$\beta_0 = 161 - 8.8283 \times 19.4 = -10.2690$$

Hence, the regression line is

$$Y = -10.2690 + 8.8283X$$

Hence, someone having a BMI of 27 would likely have cholesterol level as

$$\text{Cholesterol} = -10.2690 + 8.8283 \times 27$$

$$\text{Cholesterol} = 228$$

Ex. 2.1.3 : The following table records the number of balls that Shreyansh took for scoring runs. In how many balls is he likely to score a century?

Runs Scored	Number of Balls
8	10
35	20
47	31
54	23
11	5
85	47
84	35
93	67
89	73
2	1

Soln. :

Calculate the mean of Runs scored and Number of balls and then calculate the values as required.

Runs	Balls	$X_i - \bar{X}$	$Y_i - \bar{Y}$	$(X_i - \bar{X}) \times (Y_i - \bar{Y})$	$(X_i - \bar{X})^2$
8	10	-42.8	-21.2	907.36	1831.84
35	20	-15.8	-11.2	176.96	249.64
47	31	-3.8	-0.2	0.76	14.44
54	23	3.2	-8.2	-26.24	10.24
11	5	-39.8	-26.2	1042.76	1584.04
85	47	34.2	15.8	540.36	1169.64
84	35	33.2	3.8	126.16	1102.24
93	67	42.2	35.8	1510.76	1780.84
89	73	38.2	41.8	1596.76	1459.24
2	1	-48.8	-30.2	1473.76	2381.44
$\bar{X}$(Mean of X) = 50.8	$\bar{Y}$(Mean of Y) = 31.2			Total = 7349.4	Total = 11583.6

Hence,

$$\beta_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

$$\beta_1 = \frac{7349.4}{11583.6} = 0.6344$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

$$\beta_0 = 31.2 - 0.6344 \times 50.8 = -1.0275$$

Hence, the regression line is

$$Y = -1.0275 + 0.6344X$$

Hence, to score a century the number of balls required is

$$\text{Balls} = -1.0275 + 0.6344 \times 100$$

$$\text{Balls} = 62$$

Ex. 2.1.4 : Given the following data for the sales of car of an automobile company for six consecutive years. Predict the sales for next two consecutive years.

Year	Sales
2013	110
2014	100
2015	250
2016	275
2017	230
2018	300

Soln. :

Calculate the mean of Year and Sales and then calculate the values as required.

Year	Sales	$X_i - X_{mean}$	$Y_i - Y_{mean}$	$(X_i - X_{mean}) \times (Y_i - Y_{mean})$	$(X_i - X_{mean})^2$
2013	110	-2.5	-100.8333333	252.0833333	6.25
2014	100	-1.5	-110.8333333	166.25	2.25
2015	250	-0.5	39.16666667	-19.58333333	0.25
2016	275	0.5	64.16666667	32.08333333	0.25
2017	230	1.5	19.16666667	28.75	2.25
2018	300	2.5	89.16666667	222.9166667	6.25
$\bar{X}$(Mean of X) = 2015.5	$\bar{Y}$(Mean of Y) = 210.83			682.5	17.5

$$\beta_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

Hence,

$$\beta_1 = \frac{682.5}{17.5} = 39$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

$$\beta_0 = 210.83 - 39 \times 2015.5$$

$$= -78,393.67$$

Hence, the regression line is

$$Y = \beta_0 + \beta_1 X_1$$

$$Y = -78,393.67 + 39X$$

For the year 2019

$$\text{Sales } Y = -78,393.67 + 39 \times 2019$$

$$= 347 \text{ units}$$

For the year 2020

$$\text{Sales } Y = -78,393.67 + 39 \times 2020$$

$$= 386 \text{ units}$$

2.1.4(A) Use Cases (or Applications of) for Linear Regression

As you learnt through the various solved examples, linear regression could be very effective in establishing the relationship between independent and dependent variables and predict the outcomes. Hence, linear regression is extensively used in various business, medical, government and day to day scenarios. Some of the common use cases (or applications) of linear regression are as following.

1. **Healthcare**

   As you understand by now, healthcare industry is evolving and there are several researches that are going on. Linear regression could be used to establish the relationship between treatment and its effects or to understand complex operations of the human body to derive certain relationships. For example, you could study the effect of a particular drug chemical substance to reduce the level of infection in blood. 1 mg of substance could reduce the infection by 20% and 3 mg could reduce by 50% and so on.

2. **Demand forecasting**

   Businesses are always looking to maximise sales and reduce inventory. Sales might depend on several factors and it could be really helpful to determine the relationship of sales with those factors. Businesses can then try to modify those factors (through various promotional schemes) and appropriately forecast sales.

3. **Other predictions**

   There are several other areas where predictions can be made using the established linear relationship between the variables. It could be sports outcomes, crop output, machinery performance, fitness, and other similar areas.

2.1.5 Ridge Regression (L2 Regularisation)

- Multicollinearity is a statistical concept where several independent variables in a model are correlated. Two variables are considered to be perfectly collinear if their correlation coefficient is +/- 1.0. Multicollinearity among independent variables results in less reliable statistical inferences.
- It is better to use independent variables that are not correlated or repetitive when building multiple regression models that use two or more variables. The existence of multicollinearity in a data set can lead to less reliable results due to larger standard errors.
- As you understand, the linear regression fits a linear model with coefficients $w = (w_1, w_2, \ldots, w_p)$ to minimise the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation. Mathematically it solves a problem of the following form.

$$\min_w \|Xw - y\|_2^2$$

- The coefficient estimates for Ordinary Least Squares rely on the independence of the features. When features are correlated and the columns of the design matrix have an approximately linear dependence, the design matrix becomes close to singular and as a result, the least-squares estimate becomes highly sensitive to random errors in the observed target, producing a large variance. This situation of multicollinearity can arise, for example, when data are collected without an experimental design.
- Ridge regression is used in the case of multicollinearity. It places restrictions on the magnitudes of the estimated coefficients to avoid distortion. A penalty term proportional to the sum of the squares of the coefficients is added to reduce the standard errors. It is also called as L2 regularisation.
- The cost function of Ridge regression is given as following.

$$\min_w \|Xw - y\|_2^2 + \alpha \|w\|_2^2$$

- The complexity parameter $\alpha \geq 0$ controls the amount of shrinkage: the larger the value of α, the greater the amount of shrinkage and thus the coefficients become more robust to collinearity. The Fig. 2.1.1 illustrates Ridge coefficients as a function of the regularisation.

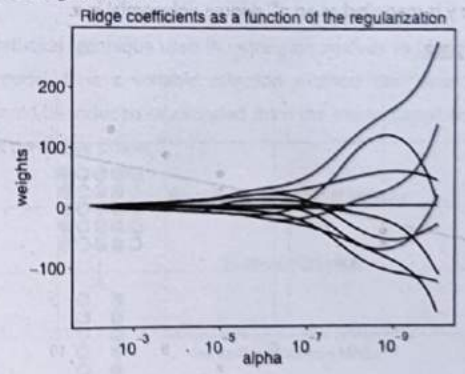


Fig. 2.1.1

- L2 regularisation results in smaller overall weight values and stabilises the weights when there is high correlation between the input features. L2 regularisation tries to estimate the mean of the data to avoid overfitting. L2 regularisation is more commonly used than L1 regularisation (Lasso Regression).

2.1.6 Lasso Regression (L1 Regularisation)

- Similar to ridge regression, lasso regression adds a penalty term proportional to the sum of the absolute values of the coefficients. It is also called as L1 regularisation.
- L1 regularisation has the effect of reducing the number of features used in the model by pushing to zero the weights of features that would otherwise have small weights. As a result, L1 regularisation results in sparse models and reduces the amount of noise in the model. L1 regularisation tries to estimate the median of the data to avoid overfitting.
- Mathematically, it consists of a linear model with an added regularisation term. The objective function to minimise is as following.

$$\min_{w} \left(\frac{1}{2n_{samles}}\right) \|Xw - y\|_2^2 + \alpha \|w\|_1$$

- The lasso estimate thus solves the minimisation of the least-squares penalty with $\alpha \|w\|_1$ added, where α is a constant and $\|w\|_1$ is the l_1 – norm of the coefficient vector.

2.1.7 Support Vector Regression (SVR)

Note : This topic is covered under Support Vector Machines in Unit 3. Please refer Section 3.8.

2.1.8 Polynomial Regression

- The most common type of regression analysis is simple linear regression, which is used when a predictor variable and a response variable have a linear relationship.
- However, sometimes the relationship between a predictor variable and a response variable is nonlinear. In that case, you could use polynomial regression.
- Polynomial regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modelled as an n^{th} degree polynomial in x.
- For example, look at the Fig. 2.1.2.

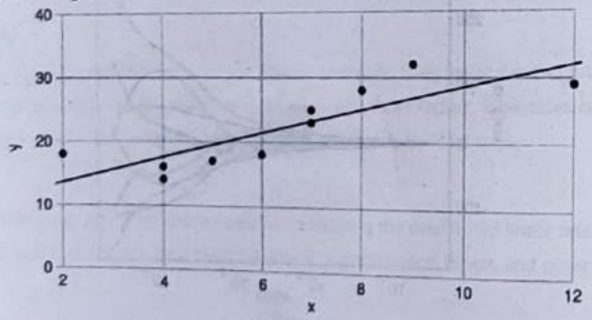


Fig. 2.1.2

- The simple linear regression line does not quite fit the data points.
- However, the polynomial regression line fits the data points much better.

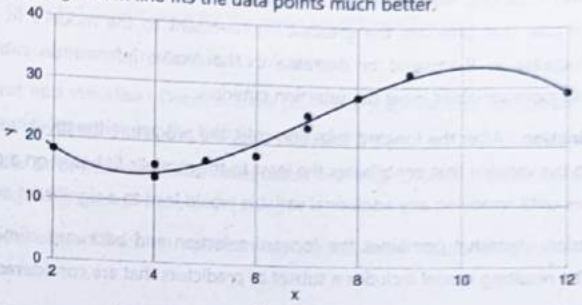


Fig. 2.1.3

- Similarly, here is another example, where polynomial regression is better suited than simple linear regression.

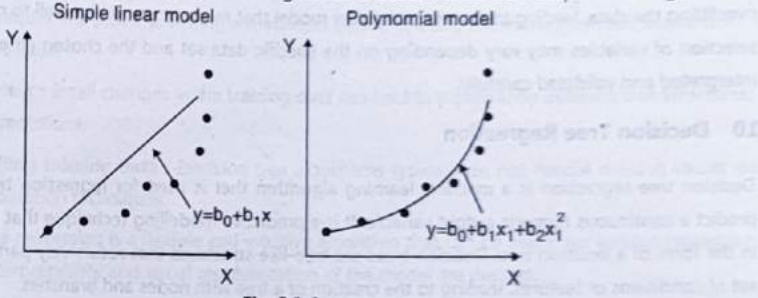


Fig. 2.1.4

- The polynomial regression formula is very similar to simple linear regression formula and is given as following.

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2$$

- Note here that the polynomial relationship may not be limited to just quadratic. It could be anything that fits the data well. For example, it could be cubic as in $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3$. The resulting regression line is a curve instead of a straight line.

2.1.9 Stepwise Regression

- Stepwise regression is a statistical technique used in regression analysis to identify the most relevant variables to include in a regression model. It is a variable selection method that aims to determine which predictors (independent variables) should be included or excluded from the model based on their statistical significance and contribution to the model's predictive power.

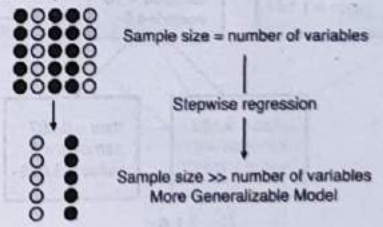


Fig. 2.1.5

- The stepwise regression procedure typically involves two main steps.
  1. **Forward selection :** Starting with an empty model, the procedure iteratively adds one predictor at a time, selecting the variable that provides the greatest improvement to the model's fit based on a pre-defined criterion (e.g., increase in R-squared or decrease in the Akaike information criterion, AIC). This process continues until no more variables meet the selection criterion.
  2. **Backward elimination :** After the forward selection step, the procedure iteratively removes one predictor at a time, eliminating the variable that contributes the least to the model's fit based on a pre-defined criterion. This process continues until removing any additional variable would lead to a significant decrease in model fit.
- The stepwise regression algorithm combines the forward selection and backward elimination steps until a final model is achieved. The resulting model includes a subset of predictors that are considered the most relevant based on the selected criterion.
- It's important to note that stepwise regression has limitations and should be used with caution. It can be prone to overfitting the data, leading to an overly complex model that may not generalise well to new data. Additionally, the selection of variables may vary depending on the specific data set and the chosen criterion, so results should be interpreted and validated carefully.

2.1.10 Decision Tree Regression

- Decision tree regression is a machine learning algorithm that is used for regression tasks, where the goal is to predict a continuous numeric output variable. It is a predictive modelling technique that builds a regression model in the form of a decision tree. Decision trees are tree-like structures that recursively partition the data based on a set of conditions or features, leading to the creation of a tree with nodes and branches.
- In decision tree regression, the tree is constructed by repeatedly splitting the data based on the values of different input features. Each internal node of the tree represents a condition or feature test, while the leaf nodes represent the predicted output values. The splitting process is performed in a way that minimises the variance or mean squared error (MSE) of the predicted values within each resulting subset. The Fig. 2.1.6 illustrates a sample decision tree for regression.

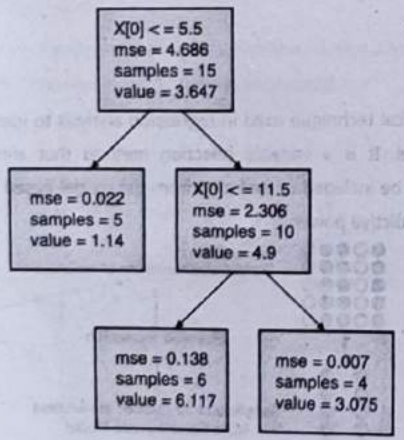


Fig. 2.1.6

- To make predictions using a decision tree regression model, you start at the root node and follow the path through the branches based on the feature values of the input data. Eventually, you reach a leaf node that corresponds to the predicted output value for the given set of input features.
- The advantages of decision tree regression are as following.
  1. **Easy to interpret and visualise :** Decision trees provide a graphical representation that is easy to understand, making them useful for explaining the decision-making process.
  2. **Ability to handle both numerical and categorical features :** Decision trees can handle a mixture of feature types without requiring extensive pre-processing.
  3. **Nonlinear relationships :** Decision trees can capture nonlinear relationships between the input features and the output variable.
- However, decision tree regression also has some limitations as following.
  1. **Prone to overfitting :** Decision trees can easily overfit the training data, resulting in poor generalisation to new data. Techniques like pruning, setting a maximum depth, or using ensemble methods like random forests can mitigate this issue.
  2. **Lack of robustness :** Small changes in the training data can lead to significantly different tree structures, which can affect the predictions.
  3. **Difficulty handling missing data :** Decision tree algorithms typically do not handle missing values well and may require imputation techniques.
- Overall, decision tree regression is a flexible and intuitive algorithm that can be useful for various regression tasks, particularly when interpretability and visual representation of the model are desired.

2.1.11 Random Forest Regression

- Random forest regression is a machine learning algorithm that combines the power of decision trees and ensemble learning for regression tasks. It is an extension of the decision tree algorithm that builds multiple decision trees and aggregates their predictions to make more accurate and robust predictions for regression problems.
- The Fig. 2.1.7 illustrates the concept of random forest regression.

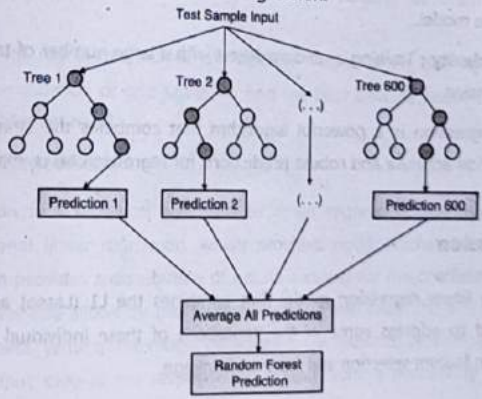


Fig. 2.1.7

- Following are the high-level steps of how random forest regression works.
  1. **Data Preparation :** As with any machine learning task, you start by preparing your dataset, splitting it into a training set and a test set.
  2. **Ensemble of Decision Trees :** Random forest regression builds an ensemble of decision trees. Each decision tree is constructed using a subset of the training data and a random selection of input features.
  3. **Random Feature Subsets :** When constructing each decision tree, a random subset of input features is chosen. This randomness helps to reduce correlation among the trees and increase the diversity in their predictions.
  4. **Bootstrap Aggregation (Bagging) :** The training data for each decision tree is sampled with replacement from the original training set. This sampling technique, known as bootstrapping, creates multiple training sets with potentially different instances.
  5. **Training :** Each decision tree in the random forest is trained independently using its respective bootstrapped training set and randomly selected features. The trees are grown until a stopping criterion is met (e.g., reaching a maximum depth or minimum number of samples per leaf).
  6. **Prediction :** To make predictions with a random forest regression model, the predictions of all the individual decision trees are aggregated. For regression, this typically involves averaging the predicted values from each tree to obtain the final prediction.
- The key advantages of random forest regression are as following.
  1. **Improved accuracy :** Random forests can provide more accurate predictions compared to individual decision trees by reducing overfitting and capturing more complex relationships in the data.
  2. **Robustness :** Random forests are less sensitive to noisy or outlier data points and can handle missing values in the input features.
  3. **Feature Importance :** Random forests provide a measure of feature importance, indicating which features have the most influence on the predictions.
- However random forest regression comes with some trade-offs as following.
  1. **Interpretability :** Random forests are more difficult to interpret compared to single decision trees due to the ensemble nature of the model.
  2. **Computational Complexity :** Training a random forest with a large number of trees or complex trees can be computationally expensive.
- Overall, random forest regression is a powerful algorithm that combines the strengths of decision trees and ensemble learning to provide accurate and robust predictions for regression tasks, making it well-suited for a wide range of applications.

2.1.12 ElasticNet Regression

- ElasticNet regression is a linear regression model that combines the L1 (Lasso) and L2 (Ridge) regularisation techniques. It is designed to address some of the limitations of these individual regularisation methods and provide a balance between feature selection and feature shrinkage.

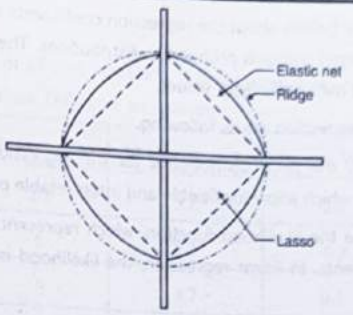


Fig. 2.1.8

- In ElasticNet regression, the model aims to minimise the following objective function.

  Loss function + $\lambda_1 \times$ L1 regularization + $\lambda_2 \times$ L2 regularization term

  where:
- The loss function measures the discrepancy between the predicted values and the actual values.
- λ_1 and λ_2 are hyperparameters that control the amount of L1 and L2 regularisation applied, respectively.
- The L1 regularisation term (also known as the Lasso regularisation) encourages sparsity by adding the absolute values of the regression coefficients to the objective function. This leads to some coefficients being exactly zero, effectively performing feature selection and reducing the number of variables used in the model.
- The L2 regularisation term (Ridge regularisation) penalises the sum of squared regression coefficients. It shrinks the coefficients towards zero without enforcing exact sparsity. This helps to reduce the impact of multicollinearity and stabilise the model by shrinking less important variables.
- The advantage of ElasticNet regression lies in its ability to handle situations where there are many correlated features and some level of feature selection is desired. It can automatically select relevant features while shrinking the coefficients of less important variables. By adjusting the values of λ_1 and λ_2, you can control the degree of regularisation and strike a balance between feature selection and coefficient shrinkage.
- ElasticNet regression is commonly used in scenarios where the number of predictors is large compared to the number of observations, and there is a possibility of collinearity among the predictors. It provides a flexible and adaptive approach for linear regression with regularization, helping to improve model performance and interpretability.
- When fitting an ElasticNet regression model, the optimal values of λ_1 and λ_2 can be determined through techniques such as cross-validation or grid search to find the best balance between the L1 and L2 regularization terms for the specific problem at hand.

2.1.13 Bayesian Linear Regression

- Bayesian linear regression is a statistical approach to linear regression that incorporates Bayesian inference principles. Unlike traditional linear regression, which provides point estimates for the regression coefficients, Bayesian linear regression provides a distribution of possible values for the coefficients. This distribution captures uncertainty in the estimates and allows for more comprehensive inference. The aim of Bayesian Linear Regression is not to find the single "best" value of the model parameters, but rather to determine the posterior distribution for the model parameters. Not only is the response generated from a probability distribution, but the model parameters are assumed to come from a distribution as well.

- In Bayesian linear regression, prior beliefs about the regression coefficients are specified before observing the data. These prior beliefs can be expressed through probability distributions. The prior distributions represent the prior knowledge or assumptions about the coefficients' values.
- The key steps in Bayesian linear regression are as following.
  1. **Prior Specification :** Specify the prior distributions for the regression coefficients. Commonly used priors include normal distributions, which allow for flexible and interpretable prior beliefs.
  2. **Likelihood Function :** Define the likelihood function, which represents the probability of the observed data given the regression coefficients. In linear regression, the likelihood is typically assumed to follow a normal distribution.
  3. **Posterior Distribution :** Combine the prior distributions with the likelihood function to obtain the posterior distribution of the regression coefficients. This is done using Bayes' theorem, which updates the prior beliefs with the observed data.
  4. **Posterior Inference :** Analyse the posterior distribution to make inferences about the regression coefficients. This can involve summarising the distribution (e.g., computing the mean or median) or obtaining credible intervals that represent ranges of likely coefficient values.
  5. **Prediction :** Use the posterior distribution of the coefficients to make predictions on new, unseen data. This involves drawing samples from the posterior distribution and computing predictions based on these samples.
- One advantage of Bayesian linear regression is that it provides a principled way to incorporate prior knowledge and quantify uncertainty in the estimates. The posterior distribution of the coefficients reflects both the observed data and the prior beliefs, allowing for a more nuanced understanding of the regression relationship.

2.2 Cost Functions (Evaluation Metrics)

- In mathematics, a function is a general way to establish mathematical relationship between entities or variables. Based on the type of relationship between the entities, the function is given a meaningful name. For example, to buy a packet of biscuit, you pay its full price. The more you buy, the higher is the amount to be paid. But you could negotiate or go to a wholesaler who starts to give you some discounts on bulk buying. This can be represented as a simple cost function such as the following.

$$\text{Total Amount} = 0.9 \times \text{Total Price}$$

  So, the above function offers a 10% discount on the total price.
- A cost function establishes the relationship between events or a set of values to represent loss or penalty incurred in gaining something (say an event or value). It is also called as loss or error function. An optimisation problem seeks to minimise a cost function so that the maximum gains could be accomplished. For example, while training a machine learning model, you seek to minimise the error occurred in training. Typically, the error is represented (or named) as a cost function. The idea behind representing a cost function is that it can be minimised using mathematical operations.
- Similarly, you could have other objective functions that you need to maximise. These are typically called a reward function, a profit function, a utility function, a fitness function, etc. So, keep it in mind, a function is just a relationship and based on the relationship and objective you have in mind, you would either minimise it or maximise it.
- Let's learn about some of the common cost functions used in machine learning.

2.2.1 Mean Error (ME)

- Mean Error (ME) is the simplest of all cost functions. It is simply a mean (average) of the difference between the actual value and the predicted values. Let's take an example.

Y (Actual)	Y' (Predicted)	Error = Y - Y'
10	9.2	0.8
8	8.3	-0.3
5	4.7	0.3
7	7.9	-0.9
4	3.1	0.9
	Total	0.8
	Mean Error	0.8 / 5 = **0.16**

- But there is a problem with this cost function. The errors can be both negative or positive and during summation, they tend to cancel each other out. In fact, it is theoretically possible that the errors are such that positive and negatives cancel each other to give zero error which could incorrectly mean a perfect model! So, Mean Error is not a recommended cost function.

2.2.2 Mean Squared Error (MSE)

- Mean Squared Error (MSE), an improvement upon Mean Error (ME) cost function, squares the difference between the actual and predicted value to avoid negative values cancelling out positive values in error calculation. Let's take an example.

Y (Actual)	Y' (Predicted)	Error = Y - Y'	Error Squared
10	9.2	0.8	0.64
8	8.3	– 0.3	0.09
5	4.7	0.3	0.09
7	7.9	-0.9	0.81
4	3.1	0.9	0.81
		Total	2.44
		Mean Squared Error	$\frac{2.44}{5}$ = **0.488**

- If you compare the MSE value (0.488) with the ME value (0.16) that you found for the same data, you find that MSE is a better way to calculate error when compared with ME. MSE is generally higher than ME for the same data indicating the error in the function.

2.2.3 Mean Absolute Error (MAE)

- Mean Absolute Error (MAE) also attempts to solve the cancelling out problem with Mean Error (ME). MAE takes the absolute value (without sign) of the difference between the actual value and the predicted value and then calculates the mean of the sum. Let's take an example.

Y (Actual)	Y' (Predicted)	Absolute Error = |Y - Y'|
10	9.2	0.8
8	8.3	0.3
5	4.7	0.3
7	7.9	0.9
4	3.1	0.9
	Total	3.2
	Mean Absolute Error	$\frac{3.2}{5} = 0.64$

- If you compare the MAE value (0.64) with the ME value (0.16) that you found for the same data, you find that MAE is a better way to calculate error when compared with ME. MAE is generally higher than ME for the same data indicating the error in the function.

2.2.4 Root Mean Squared Error (RMSE)

- Root Mean Squared Error (RMSE) also attempts to solve the cancelling out problem with Mean Error (ME). RMSE takes the root of Mean Squared Error (MSE). Let's take an example.

Y (Actual)	Y' (Predicted)	Error = Y - Y'	Error Squared
10	9.2	0.8	0.64
8	8.3	−0.3	0.09
5	4.7	0.3	0.09
7	7.9	−0.9	0.81
4	3.1	0.9	0.81
		Total	2.44
		Root Mean Squared Error	$\sqrt{\frac{2.44}{5}} = 0.698$

- If you compare the RMSE value (0.698) with the ME value (0.16) that you found for the same data, you find that RMSE is a better way to calculate error when compared with ME. RMSE is generally higher than ME for the same data indicating the error in the function.

2.2.5 R-Squared

- R-Squared (R^2 or the coefficient of determination) is a statistical measure in a regression model that determines the proportion of variance in the dependent variable that can be explained by the independent variable. In other words, R^2 shows how well the data fit the regression model (the goodness of fit). For example, R^2 of 60% reveals that 60% of the data fit the regression model. Generally, a higher R^2 indicates a better fit for the model.
- The Fig. 2.2.1 illustrates the model's performance for various values of R2.

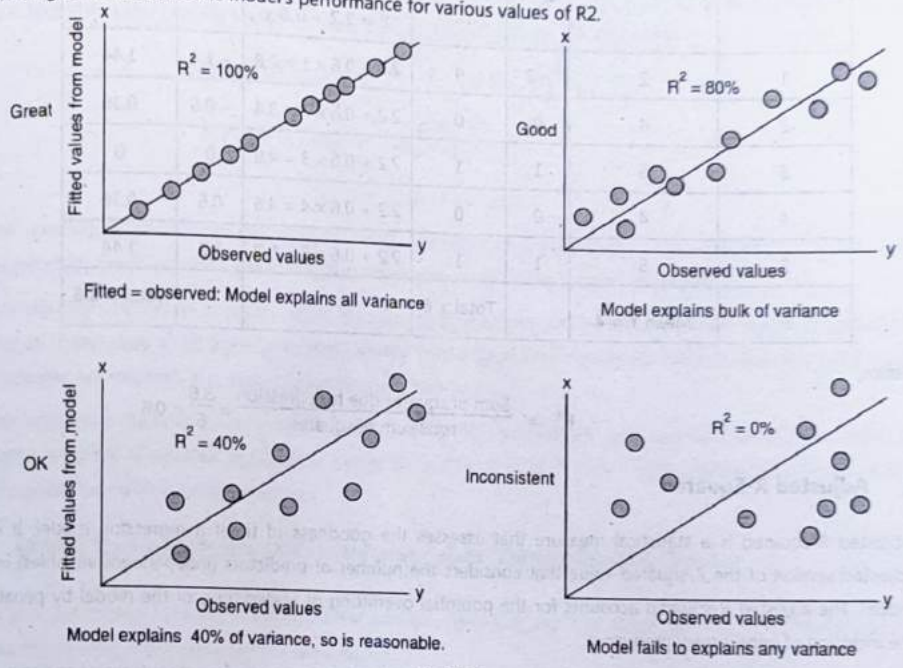


Fig. 2.2.1

- Note here that it is not always the case that a high R^2 is good for the regression model. The quality of the statistical measure depends on many factors, such as the nature of the variables employed in the model, the units of measure of the variables, and the applied data transformation. Thus, sometimes, a high R^2 can indicate the problems with the regression model.
- A low R^2 figure is generally a bad sign for predictive models. However, in some cases, a good model may show a small value. There is no universal rule on how to incorporate the statistical measure in assessing a model. The context of the experiment or forecast is extremely important and, in different scenarios, the insights from the metric can vary. R2 can take any values between 0 to 1 (as it is denoted in percentage).
- The formula for calculating R-squared is as following.

$$R^2 = \frac{\text{Sum of squares due to regression}}{\text{total sum of squares}}$$

- The sum of squares due to regression measures how well the regression model represents the data that was used for modelling. The total sum of squares measures the variation in the observed data (data used in regression modelling).

- Let's take an example.
- Assume that the regression line is given as the following function.

$y = 2.2 + 0.6\,x$. For a given set of data points x, R2 could be calculated for the regression line $y = 2.2 + 0.6\,x$ as following.

Data Point x	Actual Value Y	$Y - \bar{Y}$	$(Y - \bar{Y})^2$	Estimated Value $\hat{Y} = 2.2 + 0.6\,x$	$\hat{Y} - \bar{Y}$	$(\hat{Y} - \bar{Y})^2$
1	2	−2	4	2.2 + 0.6 × 1 = 2.8	−1.2	1.44
2	4	0	0	2.2 + 0.6 × 2 = 3.4	−0.6	0.36
3	5	1	1	2.2 + 0.6 × 3 = 4.0	0	0
4	4	0	0	2.2 + 0.6 × 4 = 4.6	0.6	0.36
5	5	1	1	2.2 + 0.6 × 5 = 5.2	1.2	1.44
	Mean $\bar{Y} = 4$		Total = 6			Total = 3.6

Hence,

$$R^2 = \frac{\text{Sum of squares due to regression}}{\text{total sum of squares}} = \frac{3.6}{6} = 0.6$$

2.2.6 Adjusted R-Squared

- Adjusted R-squared is a statistical measure that assesses the goodness of fit of a regression model. It is an adjusted version of the R-squared value that considers the number of predictors (independent variables) in the model. The adjusted R-squared accounts for the potential overfitting or underfitting of the model by penalizing the inclusion of unnecessary predictors.
- To understand adjusted R-squared, let's consider an example. Suppose you are a car manufacturer and you want to predict the fuel efficiency (measured in kilometres per litre, or kmpl) of your cars based on various features such as engine size, weight, and horsepower. You collect a dataset of 100 cars and their corresponding fuel efficiency and other attributes.
- You decide to build a regression model to predict the fuel efficiency using three predictors: engine size, weight, and horsepower. After fitting the model, you obtain an R-squared value of 0.85.
- R-squared measures the proportion of the variance in the dependent variable (fuel efficiency) that is explained by the independent variables (engine size, weight, and horsepower). In this case, an R-squared of 0.85 indicates that 85% of the variance in fuel efficiency is accounted for by the three predictors.

However, R-squared alone does not consider the complexity of the model or the number of predictors used. If you add more predictors to the model, the R-squared value will naturally increase, even if those additional predictors have little or no meaningful contribution to the model. This is where adjusted R-squared comes in. It adjusts the R-squared value to penalise the inclusion of unnecessary predictors, thereby providing a more reliable measure of model fit.

- The adjusted R-squared value is calculated using the following formula.

$$\text{Adjusted } R_{\text{squared}} = 1 - \left[(1 - R_{\text{squared}}) \times \frac{(n-1)}{(n-p-1)}\right]$$

- In the formula, n represents the number of observations in the dataset, and p represents the number of predictors in the model.
- Returning to our example, suppose the model you built has three predictors (p = 3) and 100 observations (n = 100). Using the formula, you can calculate the adjusted R-squared as following.

$$\text{Adjusted } R_{\text{squared}} = 1 - \left[(1 - 0.85) \times \frac{(100-1)}{(100-3-1)}\right]$$

$$\text{Adjusted } R_{\text{squared}} = 1 - \frac{0.15 \times 99}{96}$$

$$= 1 - 0.155 = 0.845$$

- The resulting adjusted R-squared value of 0.845 indicates that around 84.5% of the variance in fuel efficiency is explained by the three predictors in the model, after adjusting for the number of predictors.
- The adjusted R-squared increases when the new term improves the model more than would be expected by chance. It decreases when a predictor improves the model by less than expected. Typically, the adjusted R-squared is positive, not negative. It is always lower than the R-squared.
- The adjusted R-squared value is useful for comparing different models with varying numbers of predictors. A higher adjusted R-squared indicates a better fit, considering both the model's explanatory power and the complexity introduced by the predictors.

2.3 Model Representation and Interpretability (Generalisation Issues)

- As you understand, a model is based off the data you feed to it and how closely you tune it so that it can make accurate predictions or carry out the required job.
- The training data is limited and there could be noise, outliers, or variation in the data that may not accurately fit a model. Earlier, you have already learnt about several data quality issues that arise while training a model.
- You would read in subsequent chapters that a machine learning model is often a statistical function called as target function.
- The aim of training the model is to ensure that the target function is a close match to the data that is fed to it so that it can accurately carry out machine learning tasks for new data.

✍ ***Definition :*** *In statistics, a fit refers to how well you approximate a target function.*

- Fitness of a target function, approximated by a learning algorithm, determines how correctly the learning algorithm is able to classify a set of data points that it has never seen. Fitness is a measure of how well the concepts (or features) learned by a machine learning model, during training, apply to specific examples not seen by the model when it was learning.
- There are two common problems that occur when adjusting and tuning fitness of a target function – Overfitting and Underfitting.

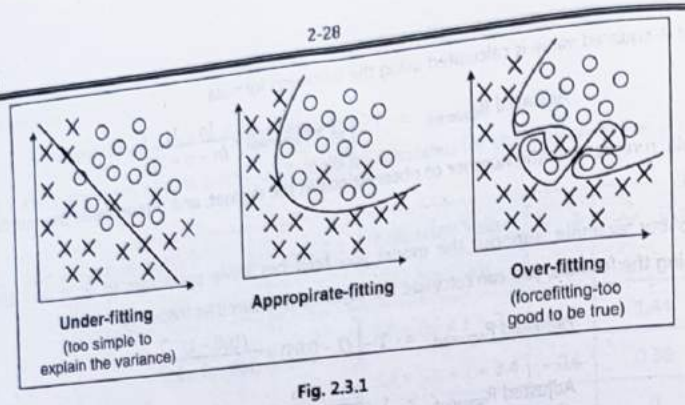


Fig. 2.3.1

Let's learn about them.

2.3.1 Overfitting

✍ ***Definition :*** *Overfitting occurs when the model matches the training data too closely, causing it to perform poorly on new data.*

- Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. The problem is that these concepts do not apply to new data and negatively impact the model's ability to generalise and carry out machine learning tasks.
- Overfitting is more likely with nonparametric and nonlinear models that have more flexibility when learning a target function. As such, many nonparametric machine learning algorithms also include parameters or techniques to limit and constrain how much detail the model learns.
- For example, decision trees are a nonparametric machine learning algorithm that is very flexible and is subject to overfitting training data.
- This problem can be addressed by pruning a tree after it has learned in order to remove some of the detail it has picked up.

2.3.2 Underfitting

✍ ***Definition :*** *Underfitting occurs when the model can neither model the training data nor work with the new data, causing it to perform poorly.*

- If the target function is kept too simple, it may not be able to learn the essential features and characteristics of the training data. Underfitting could potentially happen due to unavailability of sufficient training data or incorrect selection of features.
- An underfit machine learning model is not a suitable model and it has a poor performance on the training data. Underfitting is often not discussed as it is easy to detect given a good performance metric. You continue to tune the parameters until you get a satisfactory model. Underfitting can also be avoided by using more training data and also by carefully selecting the features on which the model is trained.

2.3.3 Bias vs Variance

- The general statistical dictionary meaning of bias is "deviation of the expected value of a statistical estimate from the quantity it estimates". In simpler terms, bias is the difference between the predicted value and the expected value.

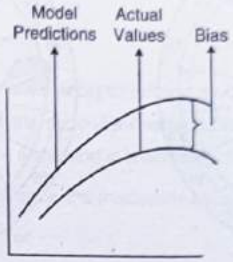


Fig. 2.3.2

- A dataset could have biased datapoints that got introduced into sampling or testing by selecting or encouraging one outcome or answer over others or due to systematic error.
- When you train a machine learning model, such biases creep into the model and thus causing it to make poor predictions.
- You have learnt about variance earlier. Variance is the variability of model prediction for a given data point or a value which tells you about the spread of datapoints in your dataset.
- A machine learning model with high variance pays a lot of attention to training data and does not generalise enough on the data which it has not seen before. As a result, such a model performs very well on training data but has high error rates on test data.

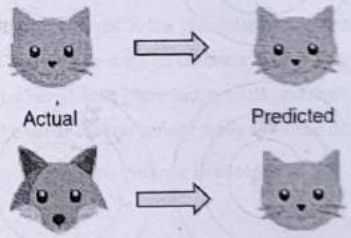
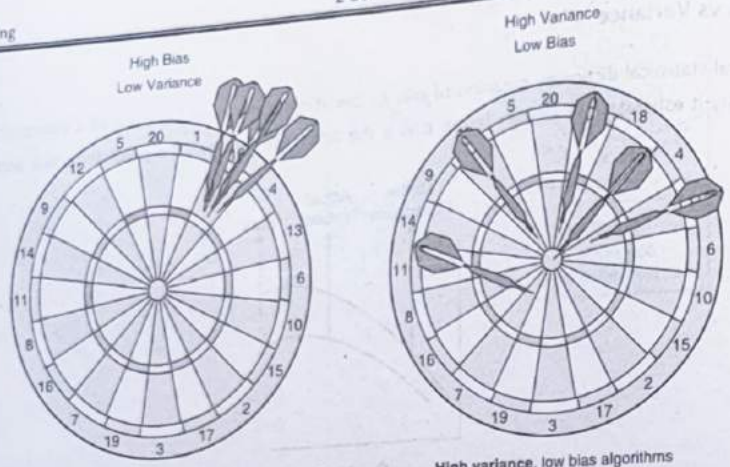


Fig. 2.3.3

- So as you see in the example, the model has learned extremely well for training data, which has taught it to identify cats. But when given new data, such as the picture of a fox, the model predicts it as a cat, as that is what it has learned. This happens when the Variance is high. The model will capture all the features of the data given to it, including the noise, will tune itself to the data, and predict it very well but when given new data, it cannot predict on it as it is too specific to training data.
- To make it further simple enough for you to understand the difference between bias and variance, consider a dart board. If all your throws land around the same position on the dart board, then there is high bias in your throws. Instead, if your dart throws are all over the place, then that is high variance. The Fig. 2.3.4 illustrates the concept.



High bias, low variance algorithms train models that are consistent, but inaccurate on average

High variance, low bias algorithms train models that are accurate on average, but inconsistent

Fig. 2.3.4

- So, if you were to plot all combinations of bias and variance, the dart board would look something like the Fig. 2.3.5.

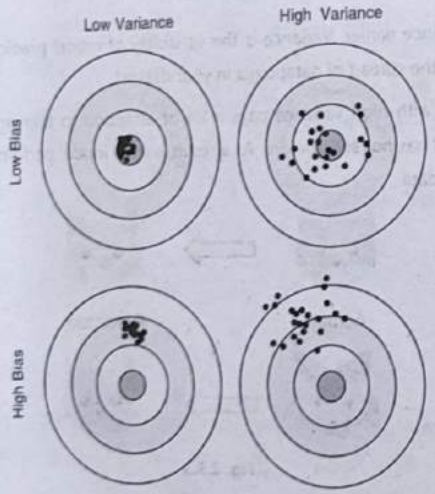


Fig. 2.3.5

- Bias and variance for some of the common machine learning algorithms are as following.

Algorithm Name	Bias	Variance
Linear Regression	High	Low
Decision Tree	Low	High
Bagging	Low	High
Random Forest	Low	High

Note that,

- High bias produces underfitting models
- High variance produces overfitting models

2.3.4 Bias-Variance Trade-Off

- There is a trade-off (need to balance bias and variance) to get the models just right for the job. If you denote the variable you are trying to predict as Y and the dependent variables as X, then you may assume that there is a relationship such that $Y = f(X) + \epsilon$ where the error term ϵ is normally distributed.
- The error term is an aggregation of reducible error and irreducible error.

Total Error = Reducible Error + Irreducible Error

The Fig. 2.3.6 illustrates what it looks like to balance bias and variance.

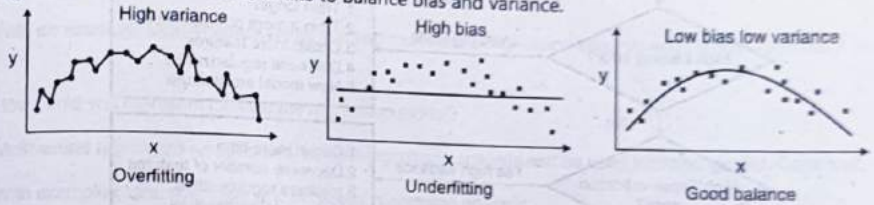


Fig. 2.3.6

Bias and variance errors can, however, be minimised. Mathematically, the error function could be described as following.

$$\text{Err}(x) = \text{Total Error} = \text{Bias}^2 + \text{variance} + \text{Irreducible Error}$$

- That third term, irreducible error, is the noise term in the true relationship that cannot fundamentally be reduced by any model. Given the true model and infinite data to calibrate it, you should be able to reduce both the bias and variance terms to 0. However, in a world with imperfect models and finite data, there is a trade-off between minimising the bias and minimising the variance. You cannot really minimise both perfectly at the same time.
- To build a good model, you need to find a good balance between bias and variance such that it minimises the total error.

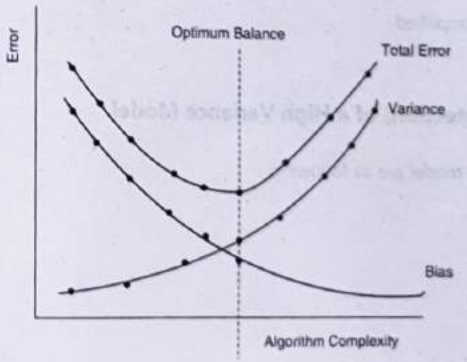


Fig. 2.3.7

- Since bias is high for a simpler model and decreases with an increase in model complexity, the line representing bias exponentially decreases as the model complexity increases.
- Similarly, Variance is high for a more complex model and is low for simpler models. Hence, the line representing variance increases exponentially as the model complexity increases.
- You can see that on either side, the generalisation error is quite high. Both high bias and high variance lead to a higher error rate.
- The most optimal complexity of the model is right in the middle, where the bias and variance intersect. These values of bias and variance would produce the least error and are preferred. An optimal balance of bias and variance would not overfit or underfit the model.

You can use the flow chart shown in Fig. 2.3.8 for balancing bias and variance.

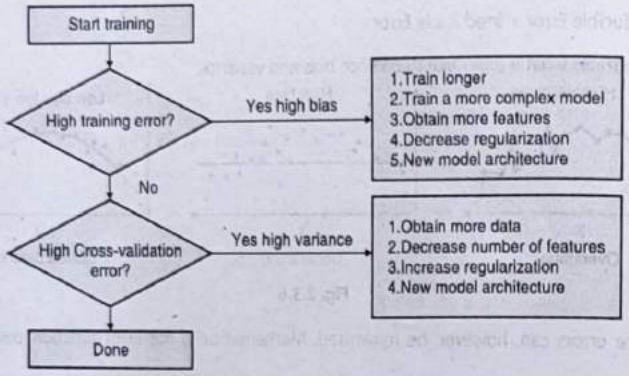


Fig. 2.3.8

2.3.5 Characteristics (Detection) of a High Bias Model

Characteristics of a high bias model are as following.

1. Failure to capture proper data trends
2. Low training accuracy
3. Potential towards underfitting
4. More generalised or overly simplified
5. High error rate

2.3.6 Characteristics (Detection) of a High Variance Model

Characteristics of a high variance model are as following.

1. Noise in the data set
2. Low testing accuracy
3. Potential towards overfitting
4. Complex models
5. Trying to put all data points as close as possible

Review Questions

Here are a few review questions to help you gauge your understanding of this chapter. Try to attempt these questions and ensure that you can recall the points mentioned in the chapter.

Regression Analysis

Q. 1 Write a short note on regression analysis. **(4 Marks)**

Q. 2 Why Do We Need Regression Analysis? **(4 Marks)**

Q. 3 Compare regression and correlation. **(4 Marks)**

Q. 4 Write a short note on univariate regression analysis. **(6 Marks)**

Q. 5 Write a short note on multivariate regression analysis. **(4 Marks)**

Q. 6 Compare univariate and multivariate regression analysis. **(4 Marks)**

Q. 7 With an example, explain where would you use multivariate regression analysis and multiple regression analysis? **(6 Marks)**

Q. 8 How could you represent multivariate regression model? **(4 Marks)**

Q. 9 Multivariate regression analysis and multiple regression analysis can be used interchangeably. Comment. **(4 Marks)**

Q. 10 With examples, describe linear and nonlinear regression analysis. **(4 Marks)**

Q. 11 Compare linear and nonlinear regression analysis. **(6 Marks)**

Q. 12 With examples, describe simple linear regression and multiple linear regression. **(6 Marks)**

Q. 13 Compare simple linear regression and multiple linear regression. **(6 Marks)**

Q. 14 Define dependent and independent variables with a suitable example. **(4 Marks)**

Q. 15 What is linear regression? **(4 Marks)**

Q. 16 For the following data set, find the linear regression line. Predict the value of Y if X = 10. **(8 Marks)**

X	Y
0	1
1	3
2	2
3	5
4	7
5	8
6	8
7	9
8	10
9	12

3 Classification

Unit 3

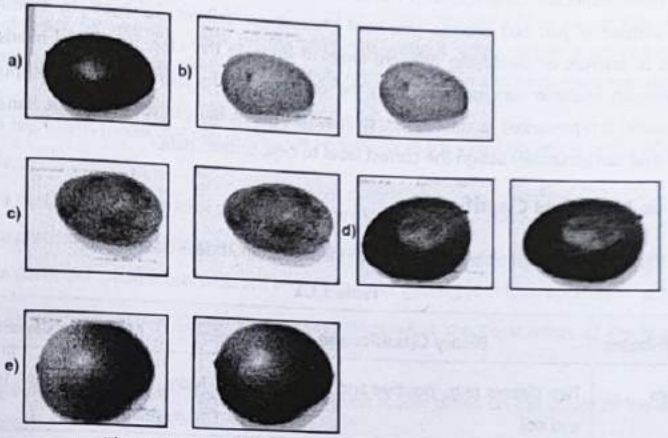
Syllabus

At the end of this unit, you should be able to understand and comprehend the following syllabus topics:

- Introduction
  - Need of Classification
  - Types of Classification (Binary and Multiclass)
  - Binary-vs-Multiclass Classification
- Binary Classification
  - Linear Classification model
- Multiclass Classification
  - Balanced and Imbalanced Classification Problems
  - One-vs-One
  - One-vs-All
- Performance Evaluation
  - Confusion Matrix
  - Accuracy
  - Precision
  - Recall
  - F measures
  - Per Class Precision
  - Per Class Recall
- Classification Algorithms
  - K Nearest Neighbour
  - Linear Support Vector Machines (SVM)
    - Introduction
    - Soft Margin SVM
    - Kernel functions
- Radial Basis Kernel
- Gaussian
  - Polynomial
  - Sigmoid

3.1 Classification Model (Need of Classification)

- Quite a few times in data analytics you do not need to predict something but rather classify objects or entities based on previous similar datasets. For example, in food industry it is common to sort out export quality mangoes from domestic quality mangoes. Today, this sorting method is increasingly becoming automated where mangoes are laid on a conveyor belt and a computer screens them to classify them as export quality (based on shape, size, weight, looks, etc.) or not.



a. The final look of the system **b. Mature small size mangoes**
c. Mature big size mangoes **d. Raw small size mangoes**
e. Raw big size mangoes

Fig. 3.1.1

- Such classification systems can tremendously help in automation. Some of the common applications of classification are

1. Classifying spam emails
2. Identifying fake user account
3. Classifying stage of a disease
4. Classifying entity breeds or sorting objects

3.1.1 Types of Classification (Binary and Multiclass)

- At a high-level, there are two types of classifications – Binary and Multiclass.

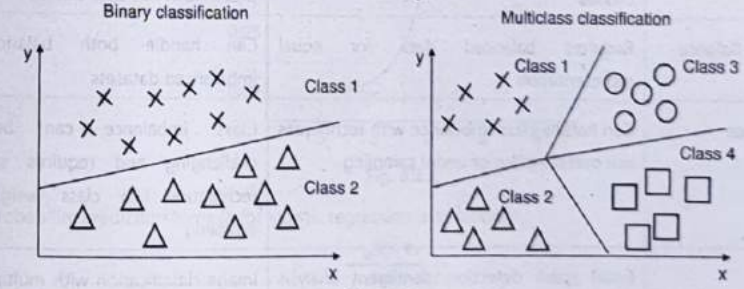


Fig. 3.1.2

- Binary classification is a type of classification problem where you have to make a decision between two options. It is like answering a yes or no question or choosing between two possibilities. For example, determining whether an email is spam or not spam, or predicting whether a customer will churn or not churn. The goal is to build a model that can learn from past data and make accurate predictions about new, unseen data. The output of a binary classification model is usually represented as either 0 or 1, where 0 represents one option (e.g., not spam, no churn) and 1 represents the other option (e.g., spam, churn).
- On the other hand, multiclass classification is a problem where you have more than two options or categories to choose from. Instead of just two choices, you have multiple possibilities. For example, classifying images into different types of animals or identifying different types of fruits. In this case, the model needs to learn how to distinguish between multiple classes and assign the correct label to each input. The output of a multiclass classification model is represented as class labels, such as red, green, blue, or apple, orange, banana. The goal is to build a model that can accurately assign the correct label to new, unseen data.

3.1.2 Binary-vs-Multiclass Classification

- The Table 3.1.1 provides a quick comparison between binary and multiclass classification.

Table 3.1.1

Comparison Attributes	Binary Classification	Multiclass Classification
Number of Classes	Two classes (e.g., positive and negative, yes and no)	More than two classes (e.g., red, green, blue, yellow)
Output Representation	Typically represented as 0 or 1	Represented as multiple class labels
Classification Models	Logistic Regression, Support Vector Machines, etc.	Logistic Regression, Decision Trees, Random Forests, etc.
Evaluation Metrics	Accuracy, Precision, Recall, F1-Score, ROC-AUC, etc.	Per Class - Accuracy, Precision, Recall, F1-Score, ROC-AUC, etc.
Decision Boundaries	A single decision boundary to separate two classes	Multiple decision boundaries to distinguish between classes
Complexity	Relatively simpler as there are only two classes	Can be more complex due to the presence of multiple classes
Training Data Balance	Requires balanced data for equal representation	Can handle both balanced and imbalanced datasets
Class Imbalance	Can handle class imbalance with techniques like oversampling or under sampling	Class imbalance can be more challenging and requires specialised techniques like class weighting or sampling
Examples	Email spam detection, sentiment analysis (positive/negative)	Image classification with multiple object categories

3.2 Binary Classification

Let's learn about a few linear binary classification techniques.

3.2.1 Logistic Regression

- So far you learnt about the linear regression where the independent and dependent variable were continuous in nature (they could have any value). They had a relationship which could have been plotted linearly.
- However, sometimes the outcome (or the dependent variable) is actually a binary event having yes / no, pass / fail, 0 / 1 kind of result. For example, given the humidity level either it would rain, or it will not rain. If you apply for a loan, it either gets approved or rejected. If you apply for admission in a college, the application is either approved or rejected.
- You need a regression model that can predict the probability of the event occurring or not occurring.
- For example,
  - Given a particular humidity level on a day, what are the chances of raining?
  - Given a particular score in the entrance examination, what are your chances of getting through the admission?
  - Given a particular credit score, what are your chances of getting your loan application approved?
  - Given the number of runs scored in a cricket match, what are the chances of the team chasing the score winning?
- Logistic regression provides such a probability prediction model based on the values of the independent variables (such as scores).

✍ ***Definition :*** *Logistic regression is a regression technique used to model and estimate the probability of an event occurring based on the values of the independent variables.*

Unlike the linear regression line, the logistic regression line is a curve. This is called Sigmoid curve or S-curve in short. All the outcome data points are either concentrated on 0 or 1. The curve depicts various possible probabilities of an event, occurring between 0 (totally uncertain) and 1 (totally certain), on the y-axis.

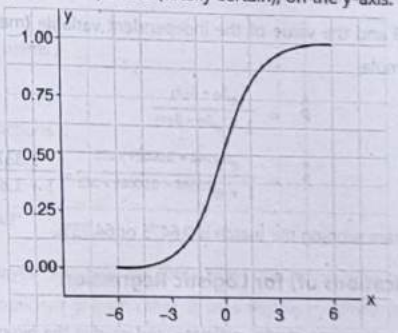


Fig. 3.2.1

The probability prediction formula for logistic regression is as following.

$$\hat{P} = \frac{e^{\beta_0 + \beta_1 x_1}}{1 + e^{\beta_0 + \beta_1 x_1}}$$

Where, $\hat{P}$ (pronounced p – hat) is the predicted probability

- e is the exponential function
- β_0 and β_1 are regression coefficients
- and X_1 is the value of the independent variable.

Note here that calculating the values of logistic regression coefficients β_0 and β_1 requires statistical software and is impractical to calculate by hand. Also, do not get confused seeing $\beta_0 + \beta_1 X_1$ in the logistic regression formula. It has nothing to do with linear coefficient calculation.

Practice Questions

Ex. 3.2.1 : You are applying for a home loan and your credit score is 720. Assuming logistic regression coefficient to be – 9.346 and 0.0146 respectively, calculate the probability of your home loan application getting approved.

Soln. :

The logistic regression coefficients are given to be

$\beta_0 = -9.346$ and $\beta_1 = 0.0146$ and the value of the independent variable (credit score) $X_1 = 720$. Putting these values in the logistic regression formula.

$$\hat{P} = \frac{e^{\beta_0 + \beta_1 x_1}}{1 + e^{\beta_0 + \beta_1 x_1}}$$

$$\hat{P} = \frac{e^{-9.346 + 0.0146 \times 720}}{1 + e^{-9.346 + 0.0146 \times 720}} = \frac{3.209}{1 + 3.209} = 0.7624$$

Hence, the probability of your loan application getting approved is 0.7624 or 76.24%.

Ex. 3.2.2 : A team scored 285 runs in a cricket match. Assuming regression coefficients to be 0.3548 and 0.00089 respectively, calculate its probability of winning the match.

Soln. :

The logistic regression coefficients are given to be

$\beta_0 = 0.3548$ and $\beta_1 = 0.00089$ and the value of the independent variable (match score) $X_1 = 285$. Putting these values in the logistic regression formula.

$$\hat{P} = \frac{e^{\beta_0 + \beta_1 x_1}}{1 + e^{\beta_0 + \beta_1 x_1}}$$

$$\hat{P} = \frac{e^{0.3548 + 0.00089 \times 285}}{1 + e^{0.3548 + 0.00089 \times 285}} = \frac{1.8375}{1 + 1.8375} = 0.6475$$

Hence, the probability of the team winning the match is 0.6475 or 64.75%.

3.2.1(A) Use Cases (or Applications of) for Logistic Regression

As you understand, logistic regression can model, estimate, and predict the binary outcome based on independent variables. It has applications in several fields and in the day to day scenarios. Some of the common applications are as following.

1. **Finance :** Based on various input parameters such as credit score, potential income, age, etc. the probability of a loan application getting approved or rejected could be estimated. This probability prediction can be used to make the right judgement after a review.

2. **Sports :** Based on various variables such as score, weather condition, pitch condition, player's past track record, etc. you could predict the chances of the team winning or losing a match.
3. **Maintenance :** You can build up proactive maintenance schedules based on various factors such as machine age, hours of operations, working conditions, etc. The machinery breakdown in plants can cause downtimes that affects business. Having a predictive maintenance schedules (say when the probability of breakdown goes higher than 50% based on the input parameters) could ensure that the breakdown is minimal.
4. **Classification or categorisation :** Logistic regression could also be used to classify the entities and objects based on input variables. For example, based on the size of mangoes, their colour and weight, you could classify them into export quality or not export quality.

3.3 Naïve Bayes (Classification by Bayesian Belief Networks)

Naïve Bayes provides another approach for classifying the data based on input variables.

Definition : *Naïve Bayes is a family of probabilistic classifiers based on Bayes' theorem (or Bayes' law) that uses the relationship between the probabilities of events for classification.*

- Given an observation of an input, a probabilistic classifier predicts a probability distribution over a set of classes. It does not just provide the most likely class that the datapoint should belong to but rather the probability distribution of the observation falling under multiple classes.
- Naïve Bayes takes a naive approach towards classification. The dictionary meaning of the word naive is "unaffected simplicity" or simply put "foolish". The naive assumption is that the occurrence of a certain feature (or data attribute) is independent of other features (or data attributes). It uses Bayes' algorithm and takes a naive approach towards classification and hence the name Naïve Bayes. Just remember that for now.
- Naïve Bayes has similar usage and application as other classifiers have such as
  1. Email spam filtering
  2. Fraud detection
  3. Diagnosing healthcare problems
  4. Text classification
  5. General classification predictions

Let's learn about it in detail.

3.3.1 Naïve Bayes Classifier

- Naïve Bayes classifier uses Bayes' theorem for probability calculation with a naive assumption that each data attribute is conditionally independent of each other. It uses slightly modified version of Bayes' theorem where the prior probability of the evidence (P(B)) is ignored from the probability calculation for various attributes because it is repetitive. Let's understand these simplifications.
- Let's take a simple scenario where you need to classify whether a particular fruit is banana or not. You have two input features (or data attributes) to make this classification – colour and length of the fruit.

- According to Bayes' theorem

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

- Which means, you would compare the probability of a fruit to be banana or not banana based on the data attributes colour and length as following.

$$P(\text{Banana} \mid \text{Colour}) = \frac{P(\text{Colour} \mid \text{Banana}) \times P(\text{Banana})}{P(\text{Colour})}$$

$$P(\text{Not Banana} \mid \text{Colour}) = \frac{P(\text{Colour} \mid \text{Not Banana}) \times P(\text{Not Banana})}{P(\text{Colour})}$$

Is P (Banana | Colour) > (Not Banana | Colour) ?

$$P(\text{Banana} \mid \text{Length}) = \frac{P(\text{Length} \mid \text{Banana}) \times P(\text{Banana})}{P(\text{Length})}$$

$$P(\text{Not Banana} \mid \text{Length}) = \frac{P(\text{Length} \mid \text{Not Banana}) \times P(\text{Not Banana})}{P(\text{Length})}$$

Is P (Banana | Length) > P (Not Banana | Length) ?

- Now, for data attributes colour and length, as you see, the denominator is same for both the scenarios – fruit being banana and fruit not being banana. Hence, the denominator can be ignored as it would not affect the probability comparison of the respective data attribute meaning to say that

1. When you are comparing the probabilities P (Banana | Colour) and P (Not Banana | Colour), the common term P (Colour) can be safely ignored without affecting the overall comparison as they cancel out.
2. Similarly, when you are comparing the probabilities P (Banana | Length) and P (Not Banana | Length) the common term P (Length) can be safely ignored without affecting the overall comparison as they cancel out.

- Now, let's understand what it means to make a naive assumption in Naïve Bayes Classifier. Continuing the same fruit classification example, the naive assumption is that *colour* and *length* are independent data attributes that give respective classification probabilities of a fruit being banana or not. These probabilities can be multiplied to get the overall classification probability.

Hence, $P(\text{Banana}) = P(\text{Banana} \mid \text{Colour}) \times P(\text{Banana} \mid \text{Length})$

Let's see a few examples of Naïve Bayes Classifier.

Practice Questions

Ex. 3.3.1 : You have designed a spam email detector based on Naïve Bayes classifier that marks emails as spam if the email body has both the words "gift" and "won". It is observed that for 100 emails

(i) 20 out of 25 spam emails have the word "gift" in the email body

(ii) 5 out of 75 non-spam emails have the word "gift" in the email body

(iii) 15 out of 25 spam emails have the word "won" in the email body

(iv) 10 out of 75 non-spam emails have the word "won" in the email body

What would be the accuracy of your spam detector ?

Soln. :

$$P(\text{Spam} \mid \text{Gift}) = \frac{P(\text{Gift} \mid \text{Spam}) \times P(\text{Spam})}{P(\text{Gift})} = \frac{\frac{20}{25} \times \frac{25}{100}}{\frac{25}{100}} = \frac{4}{5}$$

$$P(\text{Spam} \mid \text{Won}) = \frac{P(\text{Won} \mid \text{Spam}) \times P(\text{Spam})}{P(\text{Won})} = \frac{\frac{15}{25} \times \frac{25}{100}}{\frac{25}{100}} = \frac{3}{5}$$

$$P(\text{Not Spam} \mid \text{Gift}) = \frac{P(\text{Gift} \mid \text{Not Spam}) \times P(\text{Not Spam})}{P(\text{Gift})} = \frac{\frac{5}{75} \times \frac{75}{100}}{\frac{25}{100}} = \frac{1}{5}$$

$$P(\text{Not Spam} \mid \text{Won}) = \frac{P(\text{Won} \mid \text{Not Spam}) \times P(\text{Not Spam})}{P(\text{Won})} = \frac{\frac{10}{75} \times \frac{75}{100}}{\frac{25}{100}} = \frac{2}{5}$$

According to Naïve Bayes classifier,

$$P(\text{Spam} \mid \text{Gift, Won}) = P(\text{Spam} \mid \text{Gift}) \times P(\text{Spam} \mid \text{Won})$$

$$P(\text{Spam} \mid \text{Gift, Won}) = = \frac{4}{5} \times \frac{3}{5} = \frac{12}{25}$$

Hence, out of 25 spam emails, $\frac{12}{25} \times 25 = 12$ emails have gift and won

$$P(\text{Not Spam} \mid \text{Gift, Won}) = P(\text{Not Spam} \mid \text{Gift}) \times P(\text{Not Spam} \mid \text{Won})$$

$$P(\text{Not Spam} \mid \text{Gift, Won}) = \frac{1}{5} \times \frac{2}{5} = \frac{2}{25}$$

Hence, out of 75 not spam emails, $\frac{2}{25} \times 75 = 6$ emails have gift and won

Hence, accuracy of the spam detector is

$$\text{Accuracy} = \frac{\text{Spam}}{\text{Spam + Not Spam}}$$

$$= \frac{12}{12+6} = \frac{12}{18} = \frac{2}{3} = 66.67\ \%$$

Ex. 3.3.2 : Given the following dataset, find out if it is likely that the Blue Jeans from Brand X would be on Sale using Naïve Bayes Classifier.

Colour	Cloth Type	Brand	On Sale?
Blue	Jeans	Y	Yes
Blue	Shirt	X	No
Blue	Jeans	Y	Yes
Black	Jeans	Y	Yes
Black	Jeans	X	No
Black	Shirt	X	Yes
Black	Shirt	Y	Yes
Blue	Jeans	Y	No
Black	Jeans	X	Yes
Blue	Shirt	X	Yes

Soln. : There are totally 10 datapoints each having three attributes – Colour, Cloth Type and Brand.

$$P(\text{Sale}) = \frac{7}{10}$$

$$P(\text{Not on Sale}) = \frac{3}{10}$$

For *Colour* data attribute

$$P(\text{Blue} \mid \text{Sale}) = \frac{3}{7}$$

$$P(\text{Blue} \mid \text{Not Sale}) = \frac{2}{3}$$

$$P(\text{Black} \mid \text{Sale}) = \frac{4}{7}$$

$$P(\text{Black} \mid \text{Not Sale}) = \frac{1}{3}$$

For *Cloth Type* data attribute

$$P(\text{Jeans} \mid \text{Sale}) = \frac{4}{7}$$

$$P(\text{Jeans} \mid \text{Not Sale}) = \frac{2}{3}$$

$$P(\text{Shirt} \mid \text{Sale}) = \frac{3}{7}$$

$$P(\text{Shirt} \mid \text{Not Sale}) = \frac{1}{3}$$

For *Brand* data attribute

$$P(\text{X} \mid \text{Sale}) = \frac{3}{7}$$

$$P(\text{X} \mid \text{Not Sale}) = \frac{2}{3}$$

$$P(\text{Y} \mid \text{Sale}) = \frac{4}{7}$$

$$P(\text{Y} \mid \text{Not Sale}) = \frac{1}{3}$$

- Now, you have to find out the probability of Blue Jeans from Brand X to be on Sale or not on Sale using Naïve Bayes classifier.

$$P(\text{Blue Jeans from Brand X} \mid \text{Sale}) = P(\text{Blue} \mid \text{Sale}) \times P(\text{Jeans} \mid \text{Sale}) \times P(\text{Brand X} \mid \text{Sale})$$

$$P(\text{Blue Jeans from Brand X} \mid \text{Sale}) = \frac{3}{7} \times \frac{4}{7} \times \frac{3}{7} = \frac{36}{343} = 0.10 = 10\,\%$$

$$P(\text{Blue Jeans from Brand X} \mid \text{Not Sale}) = P(\text{Blue} \mid \text{Not Sale}) \times P(\text{Jeans} \mid \text{Not Sale}) \times P(\text{Brand X} \mid \text{Not Sale})$$

$$P(\text{Blue Jeans from Brand X} \mid \text{Not Sale}) = \frac{2}{3} \times \frac{2}{3} \times \frac{2}{3} = \frac{8}{27} = 0.296 = 29.6\%$$

P(Blue Jeans from Brand X | Not Sale) > P(Blue Jeans from Brand X | Sale). Hence, it is not likely that Blue Jeans from Brand X would be on sale based on Naïve Bayes classifier.

Ex. 3.3.3 : Given the following datapoints, what would be the fruit classification based on Naïve Bayes classifier?

Fruit Name	Yellow	Sweet	Long	Total Samples
Orange	350	450	0	650
Banana	400	300	350	400
Guava	50	100	50	150

Soln. :

Let's first calculate the probabilities of various data attributes with respect to each fruit classification.

Fruit Name	Yellow	Sweet	Long	Total Samples
Orange	350	450	0	650
Banana	400	300	350	400
Guava	50	100	50	150
Total Samples	800	850	400	1200

For Orange

$$P(\text{Yellow} \mid \text{Orange}) = \frac{P(\text{Orange} \mid \text{Yellow}) \times P(\text{Yellow})}{P(\text{Orange})} = \frac{\frac{350}{800} \times \frac{8000}{1200}}{\frac{650}{1200}} = 0.53$$

$$P(\text{Sweet} \mid \text{Orange}) = \frac{P(\text{Orange} \mid \text{Sweet}) \times P(\text{Sweet})}{P(\text{Orange})} = \frac{\frac{450}{850} \times \frac{850}{1200}}{\frac{650}{1200}} = 0.69$$

$$P(\text{Long} \mid \text{Orange}) = \frac{P(\text{Orange} \mid \text{Long}) \times P(\text{Long})}{P(\text{Orange})} = \frac{\frac{0}{400} \times \frac{400}{1200}}{\frac{650}{1200}} = 0$$

According to Naïve Bayes Classifier

$$P(\text{Orange} \mid \text{Yellow, Sweet, Long}) = P(\text{Yellow} \mid \text{Orange}) \times P(\text{Sweet} \mid \text{Orange}) \times P(\text{Long} \mid \text{Orange})$$

$$= 0.53 \times 0.63 \times 0 = 0$$

Hence, if the fruit is Yellow, Sweet and Long, its probability to be Orange is 0.

For Banana

$$P(\text{Yellow} \mid \text{Banana}) = \frac{P(\text{Banana} \mid \text{Yellow}) \times P(\text{Yellow})}{P(\text{Banana})}$$

$$= \frac{\frac{400}{800} \times \frac{800}{1200}}{\frac{400}{1200}} = 1$$

$$P(\text{Sweet} \mid \text{Banana}) = \frac{P(\text{Banana} \mid \text{Sweet}) \times P(\text{Sweet})}{P(\text{Banana})} = \frac{\frac{300}{850} \times \frac{850}{1200}}{\frac{400}{1200}} = 0.75$$

$$P(\text{Long} \mid \text{Banana}) = \frac{P(\text{Banana} \mid \text{Long}) \times P(\text{Long})}{P(\text{Banana})} = \frac{\frac{350}{400} \times \frac{400}{1200}}{\frac{400}{1200}} = 0.875$$

According to Naïve Bayes Classifier

$$P(\text{Banana} \mid \text{Yellow, Sweet, Long}) = P(\text{Yellow} \mid \text{Banana}) \times (\text{Sweet} \mid \text{Banana}) \times P(\text{Long} \mid \text{Banana})$$

$$= 1 \times 0.75 \times 0.875 = 0.66$$

Hence, if the fruit is Yellow, Sweet and Long, its probability to be Banana is 0.66.

For Guava

$$P(\text{Yellow} \mid \text{Guava}) = \frac{P(\text{Guava} \mid \text{Yellow}) \times P(\text{Yellow})}{P(\text{Guava})} = \frac{\frac{50}{800} \times \frac{800}{1200}}{\frac{150}{200}} = 0.33$$

$$P(\text{Sweet} \mid \text{Guava}) = \frac{P(\text{Guava} \mid \text{Sweet}) \times P(\text{Sweet})}{P(\text{Guava})} = \frac{\frac{100}{850} \times \frac{850}{1200}}{\frac{150}{1200}} = 0.66$$

$$P(\text{Long} \mid \text{Guava}) = \frac{P(\text{Guava} \mid \text{Long}) \times P(\text{Long})}{P(\text{Guava})} = \frac{\frac{50}{400} \times \frac{400}{1200}}{\frac{150}{1200}} = 0.33$$

According to Naïve Bayes Classifier

$$P(\text{Guava} \mid \text{Yellow, Sweet, Long}) = P(\text{Yellow} \mid \text{Guava}) \times P(\text{Sweet} \mid \text{Guava}) \times P(\text{Long} \mid \text{Guava})$$

$$= 0.33 \times 0.66 \times 0.33 = 0.072$$

Hence, if the fruit is Yellow, Sweet and Long, its probability to be Guava is 0.072.

Out of the three, Banana has the highest probability to be classified given that a fruit is Yellow, Sweet and Long based on Naïve Bayes Classifier.

Ex. 3.3.4 : Using Naïve Bayes classifier, find out if the following weather condition is favourable for playing.
Outlook = Rainy, Temperature = Cool, Humidity = High and Wind = Strong

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes

Day	Outlook	Temperature	Humidity	Wind	Play
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Soln. :

Let's calculate the probability of playing and not playing for each data attribute.

$$P(\text{Yes}) = \frac{9}{14}$$

$$P(\text{No}) = \frac{5}{14}$$

For Outlook

$$P(\text{Yes} \mid \text{Sunny}) = \frac{P(\text{Sunny} \mid \text{Yes}) \times P(\text{Yes})}{P(\text{Sunny})} = \frac{\frac{2}{9} \times \frac{9}{14}}{\frac{5}{14}} = 0.4$$

$$P(\text{No} \mid \text{Sunny}) = \frac{P(\text{Sunny} \mid \text{No}) \times P(\text{No})}{P(\text{Sunny})} = \frac{\frac{3}{5} \times \frac{5}{14}}{\frac{5}{14}} = 0.6$$

$$P(\text{Yes} \mid \text{Overcast}) = \frac{P(\text{Overcast} \mid \text{Yes}) \times P(\text{Yes})}{P(\text{Overcast})} = \frac{\frac{4}{9} \times \frac{9}{14}}{\frac{4}{14}} = 1$$

$$P(\text{No} \mid \text{Overcast}) = \frac{P(\text{Overcast} \mid \text{No}) \times P(\text{No})}{P(\text{Overcast})} = \frac{\frac{0}{5} \times \frac{5}{14}}{\frac{4}{14}} = 0$$

$$P(\text{Yes} \mid \text{Rain}) = \frac{P(\text{Rain} \mid \text{Yes}) \times P(\text{Yes})}{P(\text{Rain})} = \frac{\frac{3}{9} \times \frac{9}{14}}{\frac{5}{14}} = 0.6$$

$$P(\text{No} \mid \text{Rain}) = \frac{P(\text{Rain} \mid \text{No}) \times P(\text{No})}{P(\text{Rain})} = \frac{\frac{2}{5} \times \frac{5}{14}}{\frac{5}{14}} = 0.4$$

For Temperature

$$P(\text{Yes} \mid \text{Hot}) = \frac{P(\text{Hot} \mid \text{Yes}) \times P(\text{Yes})}{P(\text{Hot})} = \frac{\frac{2}{9} \times \frac{9}{14}}{\frac{4}{14}} = 0.5$$

$$P(\text{No} \mid \text{Hot}) = \frac{P(\text{Hot} \mid \text{No}) \times P(\text{No})}{P(\text{Hot})} = \frac{\frac{2}{5} \times \frac{5}{14}}{\frac{4}{14}} = 0.5$$

$$P(\text{Yes} \mid \text{Mild}) = \frac{P(\text{Mild} \mid \text{Yes}) \times P(\text{Yes})}{P(\text{Mild})} = \frac{\frac{4}{9} \times \frac{9}{14}}{\frac{6}{14}} = 0.67$$

$$P(\text{No} \mid \text{Mild}) = \frac{P(\text{Mild} \mid \text{No}) \times P(\text{No})}{P(\text{Mild})} = \frac{\frac{2}{5} \times \frac{5}{14}}{\frac{6}{14}} = 0.33$$

$$P(\text{Yes} \mid \text{Cool}) = \frac{P(\text{Cool} \mid \text{Yes}) \times P(\text{Yes})}{P(\text{Cool})} = \frac{\frac{3}{9} \times \frac{9}{14}}{\frac{4}{14}} = 0.75$$

$$P(\text{No} \mid \text{Cool}) = \frac{P(\text{Cool} \mid \text{No}) \times P(\text{No})}{P(\text{Cool})} = \frac{\frac{1}{5} \times \frac{5}{14}}{\frac{4}{14}} = 0.25$$

For Humidity

$$P(\text{Yes} \mid \text{High}) = \frac{P(\text{High} \mid \text{Yes}) \times P(\text{Yes})}{P(\text{High})} = \frac{\frac{3}{9} \times \frac{9}{14}}{\frac{7}{14}} = 0.43$$

$$P(\text{No} \mid \text{High}) = \frac{P(\text{High} \mid \text{No}) \times P(\text{No})}{P(\text{High})} = \frac{\frac{4}{5} \times \frac{5}{14}}{\frac{7}{14}} = 0.57$$

$$P(\text{Yes} \mid \text{Normal}) = \frac{P(\text{Normal} \mid \text{Yes}) \times P(\text{Yes})}{P(\text{Normal})} = \frac{\frac{6}{9} \times \frac{9}{14}}{\frac{7}{14}} = 0.86$$

$$P(\text{No} \mid \text{Normal}) = \frac{P(\text{Normal} \mid \text{No}) \times P(\text{No})}{P(\text{Normal})} = \frac{\frac{1}{5} \times \frac{5}{14}}{\frac{7}{14}} = 0.14$$

For Wind

$$P(\text{Yes} \mid \text{Strong}) = \frac{P(\text{Strong} \mid \text{Yes}) \times P(\text{Yes})}{P(\text{Strong})} = \frac{\frac{3}{9} \times \frac{9}{14}}{\frac{6}{14}} = 0.5$$

$$P(\text{No} \mid \text{Strong}) = \frac{P(\text{Strong} \mid \text{No}) \times P(\text{No})}{P(\text{Strong})} = \frac{\frac{3}{5} \times \frac{5}{14}}{\frac{6}{14}} = 0.5$$

$$P(\text{Yes} \mid \text{Weak}) = \frac{P(\text{Weak} \mid \text{Yes}) \times P(\text{Yes})}{P(\text{Weak})} = \frac{\frac{6}{9} \times \frac{9}{14}}{\frac{8}{14}} = 0.75$$

$$P(\text{No} \mid \text{Weak}) = \frac{P(\text{Weak} \mid \text{No}) \times P(\text{No})}{P(\text{Weak})} = \frac{\frac{2}{5} \times \frac{5}{14}}{\frac{8}{14}} = 0.25$$

Now, that you have all the probabilities, you can use Naïve Bayes Classifier to make a prediction.

Given the scenario, you need to figure out the probabilities for playing and not playing.

Let's call this scenario as X → Outlook = Rainy, Temperature = Cool, Humidity = High and Wind = Strong

$$P(\text{Yes} \mid X) = P(\text{Yes} \mid \text{Rain}) \times P(\text{Yes} \mid \text{Cool}) \times P(\text{Yes} \mid \text{High}) \times P(\text{Yes} \mid \text{Strong})$$

$$P(\text{Yes} \mid X) = 0.6 \times 0.75 \times 0.43 \times 0.5 = 0.0967$$

$$P(\text{No} \mid X) = P(\text{No} \mid \text{Rain}) \times P(\text{No} \mid \text{Cool}) \times P(\text{No} \mid \text{High}) \times P(\text{No} \mid \text{Strong})$$

$$P(\text{No} \mid X) = 0.4 \times 0.25 \times 0.57 \times 0.5 = 0.0285$$

Now, P(Yes | X) > P(No | X). Hence, Outlook = Rainy, Temperature = Cool, Humidity = High and Wind = Strong is a favourable playing condition.

3.3.2 Smoothing

- To calculate the probability of the overall event, Naïve Bayes classifier multiplies the probabilities of respective data attribute values. A problem with this approach is that if the probability of any data attribute value is 0 (or very close to 0), then the entire probability of the event becomes 0 (or too low) even if there are high probabilities for other data attribute values.

Definition : *Smoothing is a technique that adjusts the probabilities of rare or non-occurring attribute values to avoid overall probability of the event becoming 0 (or too low).*

- There are several smoothing techniques. A common smoothing technique is Laplace smoothing. It adds 1 to all the probabilities before multiplying them for Naïve Bayes classifier. It is represented as

$$P^*(x) = \frac{\text{Count}(x) + 1}{\sum_x (\text{Count}(x) + 1)}$$

Or simply,

$$P(x)_{\text{smooth}} = \frac{x + 1}{N + V}$$

Where,

N is the total number of observations in the particular classification and

V is the total number of distinct classifications

Let's see a few examples of Smoothing.

Practice Questions

Ex. 3.3.5 : Use the following dataset to classify the sentence "a very healthy banana".

Sentence	Classification
A great banana	Fruit
The shirt is nice	Not fruit
Very nice banana	Fruit
A nice banana	Fruit
Nice shirt	Not fruit

Soln. :

You need to find the probabilities P(Fruit | "a very healthy banana") and P(Not Fruit | "a very healthy banana") to classify the sentence "a very healthy banana".

According to Naïve Bayes classifier,

P (Fruit | "a very healthy banana") = P ("a" | Fruit) × P ("very" | Fruit) × P ("healthy" | Fruit) × P ("banana" | Fruit) × P (Fruit)

P (Not Fruit | "a very healthy banana") = P ("a" | Not Fruit) × P ("very" | Not Fruit) × P ("healthy" | Not Fruit) × P ("banana" | Not Fruit) × P (Not Fruit)

The word "heathy" does not appear in the given dataset. Hence, its probability would be 0 and it is not possible to classify the sentence "a very healthy banana" as it would zero out the calculations of both the probabilities P (Fruit | "a very healthy banana") as well as P (Not Fruit | "a very healthy banana")

However, you could use smoothing technique in this scenario to assign a non-zero probability for the word "healthy" that does not appear in the given dataset and continue with the classification based on the other words that appear in the given dataset.

Calculate the probabilities for all the words that appear in the sentence to be classified. The total number of unique words in the given dataset is 8 (a, banana, great, is, nice, shirt, the, very). Totally, 9 words (with repetition) appear in the class Fruit and 6 words (with repetition) appear in the class Not Fruit.

The word *a* appears twice in Fruit class and does not appear in Not Fruit class.

With Laplace Smoothing

$$P(\text{"a"} \mid \text{Fruit}) = \frac{2+1}{9+8} = \frac{3}{17}$$

$$P(\text{"a"} \mid \text{Not Fruit}) = \frac{0+1}{6+8} = \frac{1}{14}$$

The word *very* appears once in Fruit class and does not appear in Not Fruit class.

With Laplace Smoothing

$$P(\text{"very"} \mid \text{Fruit}) = \frac{1+1}{9+8} = \frac{2}{17}$$

$$P(\text{"very"} \mid \text{Not Fruit}) = \frac{0+1}{6+8} = \frac{1}{14}$$

The word *healthy* does not appear in any class.

With Laplace Smoothing

$$P(\text{"healthy"} \mid \text{Fruit}) = \frac{0+1}{9+8} = \frac{1}{17}$$

$$P(\text{"healthy"} \mid \text{Not Fruit}) = \frac{0+1}{6+8} = \frac{1}{14}$$

The word *banana* appears thrice in Fruit class and does not appear in Not Fruit class.

With Laplace Smoothing

$$P(\text{"banana"} \mid \text{Fruit}) = \frac{3+1}{9+8} = \frac{4}{17}$$

$$P(\text{"banana"} \mid \text{Not Fruit}) = \frac{0+1}{6+8} = \frac{1}{14}$$

$$P(\text{Fruit}) = \frac{3}{5}$$

$$P(\text{Not Fruit}) = \frac{2}{5}$$

Now, according to Naïve Bayes classifier with Laplace Smoothing

P (Fruit | "a very healthy banana") = P ("a" | Fruit) × P ("Very" | Fruit) × P ("healthy" | Fruit) × P ("banana" | Fruit) × P (Fruit)

$$P(\text{Fruit} \mid \text{"a very healthy banana"}) = \frac{3}{17} \times \frac{2}{17} \times \frac{1}{17} \times \frac{4}{17} \times \frac{3}{5} = 1.72e^{-4}$$

P (Not Fruit | "a very healthy banana") = P ("a" | Not Fruit) × P ("very" | Not Fruit) × P ("healthy" | Not Fruit) × P ("banana" | Not Fruit) × P (Not Fruit)

$$P(\text{Not Fruit} \mid \text{"a very healthy banana"}) = \frac{1}{14} \times \frac{1}{14} \times \frac{1}{14} \times \frac{1}{14} \times \frac{2}{5} = 1.04e^{-5}$$

P (Fruit | "a very healthy banana") > P (Not Fruit | " a very healthy banana")

Hence, the sentence "a very healthy banana" is classified as Fruit.

Ex. 3.3.6 : A die is thrown 1 times and the outcome is recorded as following. What is the probability of the outcome to be 4 with and without smoothing?

1. Number 1 – once
2. Number 2 – thrice
3. Number 3 – once
4. Number 4 – none
5. Number 5 – thrice
6. Number 6 – twice

Soln. :

Without smoothing, since there is no occurrence of the outcome 4 in the given dataset

$$P(4) = \frac{0}{10} = 0$$

There are 6 possible outcomes and there are 10 observations.

Hence, N = 10 and V = 6

With smoothing,

$$P(4) = \frac{0+1}{10+6} = \frac{1}{16}$$

3.3.3 Advantages of Naïve Bayes Classifier

Naïve Bayes classifier has the following advantages.

1. It can handle missing data attribute values.
2. It can handle irrelevant input variables.
3. It is simple to implement without requiring special software libraries.
4. It is computationally fast.
5. It performs better with categorical variables when compared with decision trees.

3.3.4 Disadvantages of Naïve Bayes Classifier

Naïve Bayes classifier has the following disadvantages.

1. It assumes that the data attributes are conditionally independent.
2. In general, it is not very reliable for probability estimation.
3. Usually, it can be used only with the categorical variables. Continuous variables need to be converted into categorical variables for working with them.
4. If there are a lot of data attributes, probability calculations tend to be very low or near zero. You require additional steps such as taking logarithms or apply smoothing.

3.4 Multi-class Classification Techniques

✍ ***Definition :*** *Multi-class or multinomial classification is the problem of classifying instances (data points) into one of three or more classes.*

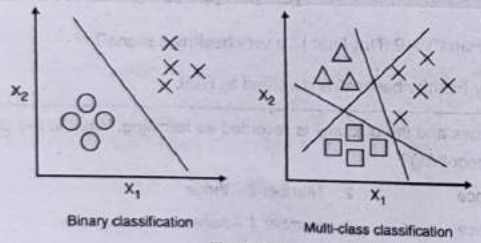


Fig. 3.4.1

Multi-class classification is a task with more than two classes. For example, classifying a set of images of fruits which may be oranges, apples, or pears. Multi-class classification makes the assumption that each sample is assigned to one and only one class (or label). For example, a fruit can either be an apple or a pear, but not both at the same time. Another example could be recognising alphabets in an optical character recognition type of problems where a given alphabet could be one of the 26 alphabets. Let's learn about multi-class classification techniques.

3.4.1 One vs One (OvO)

One vs One (OvO in short) is a heuristic method for using binary classification algorithms for multi-class classification. One vs One technique splits a multi-class classification dataset into binary classification problems. In this approach, the entire dataset is split into one dataset for each class versus every other class.

For example, consider a multi-class classification problem with four classes – 'red', 'blue', 'green', and 'yellow'. This could then be divided into six binary classification datasets as following.

- Binary Classification Problem 1 : red vs blue
- Binary Classification Problem 2 : red vs green
- Binary Classification Problem 3 : red vs yellow
- Binary Classification Problem 4 : blue vs green
- Binary Classification Problem 5 : blue vs yellow
- Binary Classification Problem 6 : green vs yellow

As you see, this is significantly more datasets.

You can calculate the number of binary datasets(or models) that would be created for One vs One approach as following.

$$\text{No. of binary datasets (or models)} = \frac{C \times (C-1)}{2} \text{ where C is the number of classes}$$

In the example given earlier, you have 4 classes. Hence, the number of binary datasets that would be created can be calculated as following.

$$\text{No. of binary datasets for 4 classes} = \frac{4 \times (4-1)}{2} = 6$$

Each binary classification model may predict one class label and the model with the most predictions or votes is predicted. Similarly, if the binary classification models predict a numerical class membership, such as a probability, then the argmax of the sum of the scores is predicted as the class label.

3.4.2 One vs Rest (OvR) (Ove vs All)

One vs Rest (OvR in short, also referred to as One-vs-All or OvA) is a heuristic method for using binary classification algorithms for multi-class classification. It involves splitting the multi-class dataset into multiple binary classification problems. Each binary classifier is then trained on each binary classification problem and predictions are made using the model that is the most confident. Basically, for each split dataset, you take one class as positive and all other classes as negative.

For example, consider a multi-class classification problem with four classes – 'red', 'blue', 'green', and 'yellow'. This could be divided into four binary classification datasets as following.

- Binary Classification Problem 1: red vs [blue, green, yellow]
- Binary Classification Problem 2: blue vs [red, green, yellow]
- Binary Classification Problem 3: green vs [red, blue, yellow]
- Binary Classification Problem 4: yellow vs [red, blue, green]

One machine learning model is created for each class. For example, four classes require four models. This approach requires that each model predicts a class membership probability or a probability-like score. The argmax of these scores (class index with the largest score) is then used to predict a class. This approach is commonly used for algorithms that naturally predict numerical class membership probability or score, such as Logistic Regression and Perceptron. One advantage of this approach is its interpretability. Since each class is represented by one and only one classifier, it is possible to gain knowledge about the class by inspecting its corresponding classifier. This is the most commonly used strategy and is a fair default choice.

3.4.3 Comparison between OvO and OvR

The Table 3.4.1 provides a quick comparison between OvO and OvR multi-class classification techniques.

Table 3.4.1

Comparison Attribute	One vs One (OvO)	One vs Rest (OvR)
Speed	Slower than OvR	Faster than OvO
Computation Complexity	High	Low
Suitable for	Algorithms that don't scale	Algorithms that scale
No. of binary datasets or models for C classes	$\frac{C \times (C-1)}{2}$	C
Interpretability	Low	High
Used	Less commonly	More Commonly

3.5 Balanced and Imbalanced Multi-class Classification Problems

- When observation in one class is way higher than the observation in other classes then there exists a class imbalance problem. For example, suppose that you have a fruit dataset that you are using for training your multi-class classifier having class labels of apples, oranges, and bananas. In that dataset if majority of the data points belong to just apples, then the dataset is said to be skewed and imbalanced.
- So, what exactly is the problem with class imbalance? Most machine learning algorithms work best when the number of samples in each class are about equal. This is because most algorithms are designed to maximise accuracy and reduce errors. However, if the dataset is imbalanced, then in such cases, you get a pretty high accuracy just by predicting the majority class, but you fail to capture the minority class, which is most often the point of creating the model in the first place. For example, if majority of the data points in your fruit dataset belong to apples, then your classification model would just appear to work perfectly when it predicts apples (majority class) most of the times. So, for predicting apples, your model would work around 100% accuracy whereas for other minority classes such as oranges and bananas, the accuracy would be pretty poor and mostly around less than 5%. Such classifiers also fail to predict rare or extreme events.

3.5.1 Causes of Class Imbalance

- The imbalance in the class distribution may have many causes. The two main causes are as following.
  - (1) **Sampling error :** You might have class imbalance due to sampling errors. For example, you could have collected the data points from a very narrowly selected population or there could be bias in sampling along with data collection errors (such as incorrectly labelling samples).
  - (2) **Problem domain :** Based on the domain of your classification problem, there might be natural class imbalance. Suppose, that you are building a model based on weather conditions to predict next earthquake. Now, earthquake events may be very rare whereas non-earthquake events may dominate the majority of weather condition related datapoints. You cannot really do anything about it (natural events, such as an earthquake, are beyond your control) and your overall earthquake dataset might be imbalanced.
- The natural occurrence or presence of one class may dominate other classes. This may be because the process that generates observations in one class is more expensive in time, cost, computation, or other resources. As such, it is often infeasible to simply collect more samples from the domain in order to improve the class distribution.

3.5.2 Techniques to Handle Class Imbalance

Let's learn about a few techniques to handle class imbalance.

3.5.2(A) Random Resampling

- Resampling is a widely used technique for handling imbalanced classes in a dataset. It consists of removing samples from the majority class (under-sampling) and/or adding more samples (artificially) from the minority class (over-sampling). The Fig. 3.5.1 illustrates this concept.

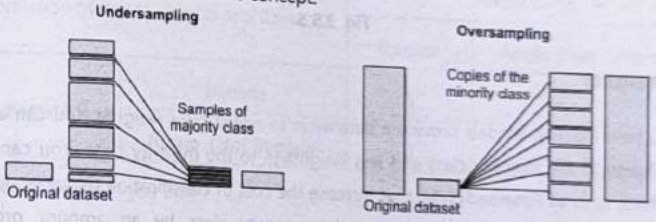


Fig. 3.5.1

- In under-sampling, you remove many observations of the majority class randomly. This is done until the majority and minority classes are balanced out. Under-sampling can be a good choice when you have a lot of majority class data. A drawback of under-sampling is that you randomly remove information which might have been useful.
- In over-sampling, you add more copies of data points to the minority class. Over-sampling can be a good choice when you do not have a lot of data to work with. A drawback of over-sampling is that it can cause overfitting and poor generalisation.

3.5.2(B) Tomek Links

- Tomek links are pairs of very close instances but of opposite classes. Removing the instances of the majority class of each pair increases the space between the two classes and thus facilitating the classification process better. Tomek links exist if the two samples are the nearest neighbours of each other. It is an under-sampling technique to remove datapoints of the majority class. The Fig. 3.5.2 illustrates how Tomek links work.

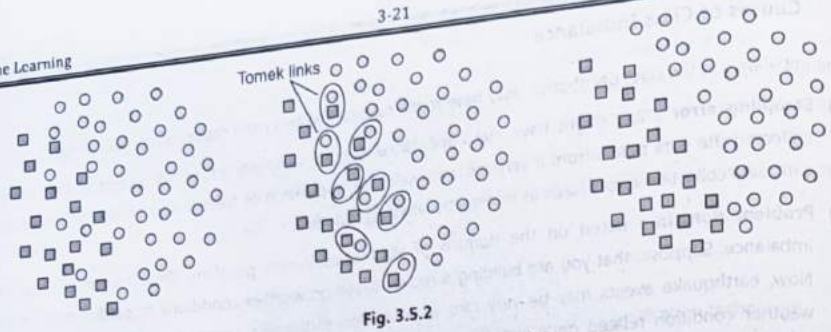


Fig. 3.5.2

3.5.2(C) SMOTE (Synthetic Minority Oversampling Technique)

- SMOTE generates synthetic (artificial) data for the minority class. It is an over-sampling technique. It consists of synthesising elements for the minority class, based on those that already exist. It works by randomly picking a point from the minority class and computing the k-nearest neighbours for this point. The synthetic points are added between the chosen point and its neighbours. The Fig. 3.5.3 illustrates how SMOTE works.

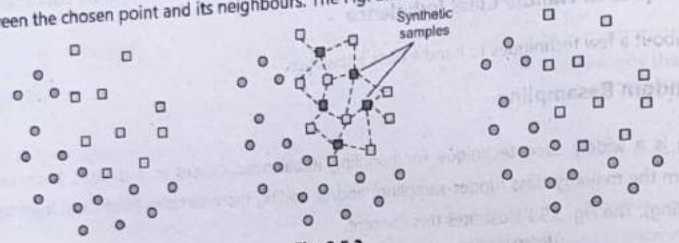


Fig. 3.5.3

3.5.2(D) Class Weights

- Most of the machine learning models provide a parameter to adjust class weights. You can use class weights to add more weightage to the minority class and less weightage to the majority class. You can also use penalised learning algorithms, such as Penalised-SVM, that increase the cost of classification mistakes on the minority class. During training, you can penalise mistakes on the minority class by an amount proportional to how under-represented it is.

3.6 Performance Measures - Measuring Quality of a Model (Diagnostics (Evaluation Measures) of Classifiers)

So far you have learnt about a few classification techniques. When you develop a model based on any of these, you need to know how well your model is working. To help you evaluate your classification model there are a few tools that you could use. Let's learn about them.

3.6.1 Confusion Matrix

✍ ***Definition :*** *A confusion matrix lays out the comparison between the predicted classification and actual classification to provide various performance measures.*

The number of correct and incorrect predictions are written for each class. Confusion matrix provides calculations for various types of errors in prediction and helps you to improve your classification model.

Following is how a confusion matrix looks like for two possible classifications.

Confusion Matrix for Class 1 and Class 2		Actual Class	
		Class 1	**Class 2**
Predicted Class	**Class 1**	**True Positives (TP)**	False Positives (FP)
	Class 2	False Negative (FN)	**True Negative (TN)**

Recall Type I and Type II errors that you learnt earlier in Chapter 2. Confusion matrix provides a similar layout.

- **True Positive** = Correct True Prediction for Class 1. This is desired.
  - Predicted Class = Actual Class : Correct classification
- **True Negative** = Correct False Prediction for Class 1 (True Prediction for Class 2). This is desired.
  - Predicted Class = Actual Class : Correct classification
- **False Positive** = Incorrect True Prediction for Class 2. This is Type I error.
  - Predicted Class ≠ Actual Class : Incorrect classification
- **False Negative** = Incorrect False Prediction for Class 2. This is Type II error.
  - Predicted Class ≠ Actual Class : Incorrect classification

For example, assume that you have built a model for fruit classifier that can classify the pictures into banana and apple. Assume that there were 13 pictures of fruits that were classified using your model. In reality, there were 8 banana pictures and 5 apple pictures. A confusion matrix could look like the following.

Confusion Matrix for Apples and Bananas		Actual Class	
		Banana	**Apple (Not Banana)**
Predicted Class	**Banana**	**5**	2
	Apple (Not Banana)	3	**3**

So, the confusion matrix tells you that

- Your model correctly classified 5 out of 13 pictures as banana. (True Positive)
- Your model correctly classified 3 out of 13 pictures as apple (which were not banana) pictures. (True Negative)
- Your model incorrectly classified 2 out of 13 pictures. They were actually apples which were classified as bananas. (False Positive)
- Your model incorrectly classified 3 out of 13 pictures. They were actually bananas which were classified as apples. (False Negative)

So,

- Overall 8 out of 13 classifications were correct.
- Overall 5 out of 13 classifications were incorrect.

- Based on these values, you can calculate various performance parameters for your classification models.

3.6.2 Accuracy

Accuracy is a performance measure of the rate at which the model has correctly classified the samples. Higher the accuracy, the better is the model

It is calculated as following.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \times 100$$

3.6.3 True Positive Rate (TPR) / Sensitivity / Recall / Hit rate

True Positive Rate (TPR) / Sensitivity / Recall / Hit rate is a performance measure of the rate at which the model has correctly classified the positive outcomes. Higher the True Positive Rate (TPR) / Sensitivity / Recall / Hit rate, the better is the model.

It is calculated as following.

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

3.6.4 False Positive Rate (FPR) / False Alarm Rate / Type I Error Rate / Fall-Out Rate

False Positive Rate (FPR) / False Alarm Rate / Type I error rate / Fall-out rate is a performance measure of the rate at which the model has incorrectly classified the positive outcomes. Lower the False Positive Rate (FPR) / False Alarm Rate / Type I error rate / Fall-out rate, the better is the model.

It is calculated as following.

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

3.6.5 True Negative Rate (TNR) / Specificity / Selectivity

True Negative Rate (TNR) / Specificity / Selectivity is a performance measure of the rate at which the model has correctly classified the negative outcomes. Higher the True Negative Rate (TNR) / Specificity / Selectivity, the better is the model.

It is calculated as following.

$$TNR = \frac{TN}{TN + FP}$$

3.6.6 False Negative Rate (FNR) / Miss rate / Type II error rate

False Negative Rate (FNR) / Miss rate / Type II error rate is a performance measure of the rate at which the model has incorrectly classified the negative outcomes. Lower the False Negative Rate (FNR) / Miss rate / Type II error rate, the better is the model.

It is calculated as following.

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

3.6.7 Precision / Positive Predictive Value (PPV)

Precision / Positive Predictive Value (PPV)is a performance measure based on the percentage of samples marked positive that were actually positive. Higher the Precision / Positive Predictive Value (PPV), the better is the model.

It is calculated as following.

$$\text{Precision} = \frac{TP}{TP + FP}$$

3.6.8 Negative Predictive Value (NPV)

Negative Predictive Value (NPV) is a performance measure based on the percentage of samples marked negative that were actually negative. Higher the Negative Predictive Value (NPV), the better is the model.

It is calculated as following.

$$NPV = \frac{TN}{TN + FN}$$

3.6.9 F1 Score / F-measure

F1 Score / F-measure is a performance measure based on the harmonic mean of the precision and recall. Higher the F1 Score / F-measure, the better is the model.

It is calculated as following.

$$\text{F1 score} = \frac{2TP}{2TP + FP + FN}$$

3.6.10 Matthews Correlation Coefficient (MCC)

Matthews Correlation Coefficient (MCC)is a correlation coefficient between the observed and predicted binary classifications. It returns a value between −1 and +1. A coefficient of +1 represents a perfect prediction, 0 no better than random prediction and −1 indicates total disagreement between prediction and observation.

It is calculated as following.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Let's see a few examples of Confusion Matrix and calculation of performance measures.

Practice Questions

Ex. 3.6.1 : For the following confusion matrix, calculate accuracy, TPR and FPR.

Confusion Matrix for Apples and Bananas		Actual Class	
		Banana	Apple (Not Banana)
Predicted Class	Banana	5	2
	Apple (Not Banana)	3	3

Soln. : Based on the given confusion matrix

- True Positive (TP) = 5

- True Negative (TN) = 3
- False Positive (FP) = 2
- False Negative (FN) = 3

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \times 100$$

$$\text{Accuracy} = \frac{5 + 3}{5 + 3 + 2 + 3} \times 100 = \frac{10}{13} \times 100 = 76.92\ \%$$

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

$$TPR = \frac{5}{5 + 3} = \frac{5}{8} = 0.625$$

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

$$FPR = \frac{2}{2 + 3} = \frac{2}{5} = 0.4$$

Ex. 3.6.2 : For the following confusion matrix, calculate F1 score and MCC. What would you comment about the classifier based on these values?

TP = 100	FP = 10
FN = 5	TN = 50

Soln. :

$$\text{F1 Score} = \frac{2TP}{2TP + FP + FN}$$

$$\text{F1 Score} = \frac{2 \times 100}{2 \times 100 + 10 + 5} = \frac{200}{215} = 0.93$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

$$MCC = \frac{100 \times 50 - 10 \times 5}{\sqrt{(100 + 10)(100 + 5)(50 + 10)(50 + 5)}} = \frac{4950}{6173.73} = 0.80$$

The values of F1 score and MCC are very close to 1. These performance measures suggest that the classifier's accuracy is pretty good.

3.6.11 Micro-Average and Macro-Average Precision, Recall and F1-Score (Per Class Precision and Per Class Recall)

- With binary classification, it is very intuitive to score the model in terms of scoring metrics such as precision, recall and F1-score. However, in case of multi-class classification it becomes tricky. The questions to ask are some of the following.
  - Which metrics to use to score the model trained for multi-class classification?
  - How to calculate precision, recall and f1-score of multi-class classification models?
  - When to use micro-average and macro-averaging scores?

- Micro and Macro averaging methods help you to score the multi-class classification models. Typically, you use micro-averaging score when there is a need to weight each instance or prediction equally and macro-averaging score when all classes need to be treated equally to evaluate the overall performance of the classifier with regard to the most frequent class labels.
- Micro and Macro averages are calculated as following.

$$\text{Micro-average precision} = \frac{TP_1 + TP_2 \ldots TP_n}{TP_1 + TP_2 \ldots + TP_n + FP_1 + FP_2 \ldots + FP_n}$$

$$\text{Micro-average Recall} = \frac{TP_1 + TP_2 \ldots TP_n}{TP_1 + TP_2 \ldots + TP_n + FN_1 + FN_2 \ldots + FN_n}$$

$$\text{Macro-average precision} = \frac{P_1 + P_2 + \ldots P_n}{n}$$

$$\text{Micro-average Recall} = \frac{R_1 + R_2 + \ldots R_n}{n}$$

$$\text{Micro-average F1 score} = \frac{TP_1 + TP_2 \ldots TP_n}{TP_1 + TP_2 + TP_n + \frac{1}{2}(FP_1 + FP_2 + \ldots FP_n + FN_1 + FN_2 + \ldots FN_n)}$$

$$\text{Macro-average F1 score} = \frac{F1score_1 + F1score_2 + \ldots F1score_n}{n}$$

Let's see an example to understand these better.

Practice Questions

Ex. 3.6.3 : Following are the scores for a multi-class classification model. Find out micro and macro precision, recall, and F1-score.

Class	TP	FP	FN	Precision	Recall
A	5	2	1	0.71	0.83
B	10	90	7	0.1	0.58
C	15	11	2	0.57	0.88

Soln. :

$$\text{Micro-average precision} = \frac{TP_1 + TP_2 \ldots TP_n}{TP_1 + TP_2 \ldots + TP_n + FP_1 + FP_2 \ldots + FP_n}$$

$$\text{Micro-average precision} = \frac{5 + 10 + 15}{5 + 10 + 15 + 2 + 90 + 11}$$

$$= \frac{30}{133} = 0.22$$

$$\text{Macro-average precision} = \frac{P_1 + P_2 + \ldots P_n}{n}$$

$$\text{Macro-average precision} = \frac{0.71 + 0.1 + 0.57}{3} = \frac{1.38}{3} = 0.46$$

$$\text{Micro-average recall} = \frac{TP_1 + TP_2 \ldots TP_n}{TP_1 + TP_2 \ldots + TP_n + FN_1 + FN_2 \ldots + FN_n}$$

$$\text{Micro-average Recall} = \frac{5 + 10 + 15}{5 + 10 + 15 + 1 + 7 + 2}$$

$$= \frac{30}{40} = 0.75$$

$$\text{Macro-average Recall} = \frac{R_1 + R_2 + \ldots R_n}{n}$$

$$\text{Macro-average Recall} = \frac{0.83 + 0.58 + 0.88}{3}$$

$$= \frac{2.29}{3} = 0.76$$

$$\text{Micro–average F1 score} = \frac{TP_1 + TP_2 \ldots TP_n}{TP_1 + TP_2 + TP_n + \frac{1}{2}(FP_1 + FP_2 + \ldots FP_n + FN_1 + FN_2 + \ldots FN_n)}$$

$$\text{Micro-average F1 score} = \frac{5 + 10 + 15}{5 + 10 + 15 + \frac{1}{2}(2 + 90 + 11 + 1 + 7 + 2)}$$

$$= \frac{30}{86.5}$$

$$= 0.35$$

$$\text{F1 score} = \frac{2TP}{2TP + FP + FN}$$

$$F1score_1 = \frac{2 * 5}{2 * 5 + 2 + 1}$$

$$= \frac{10}{13}$$

$$= 0.77$$

$$F1score_2 = \frac{2 * 10}{2 * 10 + 90 + 7}$$

$$= \frac{20}{117}$$

$$= 0.17$$

$$F1score_3 = \frac{2 * 15}{2 * 15 + 11 + 2}$$

$$= \frac{30}{43} = 0.70$$

$$\text{Macro-average F1score} = \frac{F1score_1 + F1score_2 + \ldots F1score_n}{n}$$

$$\text{Macro-average F1 score} = \frac{0.77 + 0.17 + 0.70}{3}$$

$$= \frac{1.64}{3}$$

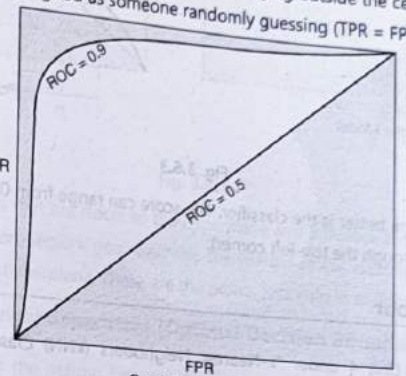
$$= 0.55$$

3.6.12 ROC Curve

- ROC is an acronym for Receiver Operating Characteristic.

Definition : *A ROC curve evaluates the performance of a classifier based on the True Positives (TP) and False Positives (FP) regardless of other factors such as class distribution and error costs.*

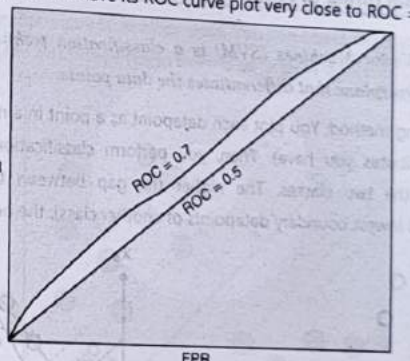
- On the Y-axis, you plot the True Positive Rate (TPR) and on the X-axis you plot the False Positive Rate (FPR). As you adjust your classifier model, a ROC curve can help you to visualise all possible classification thresholds.
- A classifier that does a good job will have ROC curve that is bulging outside the central ROC = 0.5 line. ROC = 0.5 line represents a classifier that is as good as someone randomly guessing (TPR = FPR).



Good Classifier Model

Fig. 3.6.1

- A classifier that does not classify well will have its ROC curve plot very close to ROC = 0.5 line.



Poor Classifier Model

Fig. 3.6.2

3.6.13 Area Under the Curve (AUC)

Definition : *Area Under the Curve (AUC) is calculated by measuring the area under the ROC curve.*

- Imagine an illusory square box around the ROC curve plot. AUC represents the area or the amount of box that the ROC curve fills in.

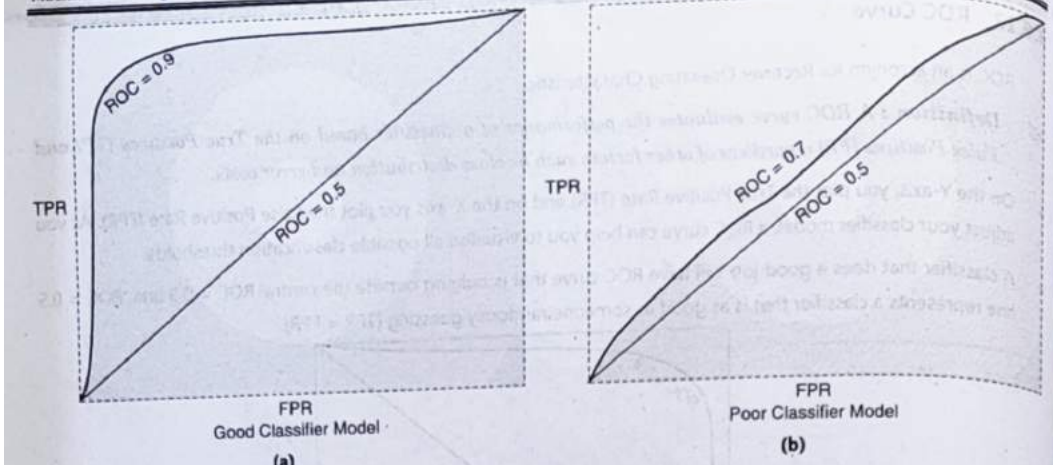


Fig. 3.6.3

- The higher the area filled, the better is the classifier. The score can range from 0.5 (for the diagonal line TPR = FPR) to 1.0 (with ROC passing through the top-left corner).

3.7 k-nearest Neighbour

This topic is covered in Unit 4 under "k-Nearest Neighbours (kNN) Classification Algorithm". Please refer Section 4.2.

3.8 Support Vector Machines (SVM)

***Definition** : Support Vector Machines (SVM) is a classification technique that uses critical boundary datapoints to create a hyperplane that differentiates the data points.*

- SVM is a supervised learning method. You plot each datapoint as a point in a n-dimensional space (where n is the number of datapoint attributes you have). Then, you perform classification by finding the hyperplane that adequately differentiates the two classes. The higher the gap between the datapoints (highest boundary datapoints of one class and lowest boundary datapoints of another class), the better.

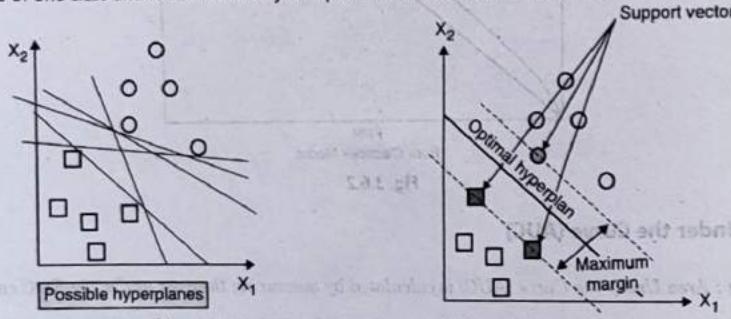


Fig. 3.8.1

- Hyperplanes are decision boundaries that help in classifying the datapoints. Datapoints falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of data attributes. If the number of input data attributes is 2, then the hyperplane is just a line. If the number of input data attributes is 3, then the hyperplane becomes a two-dimensional plane, and likewise.

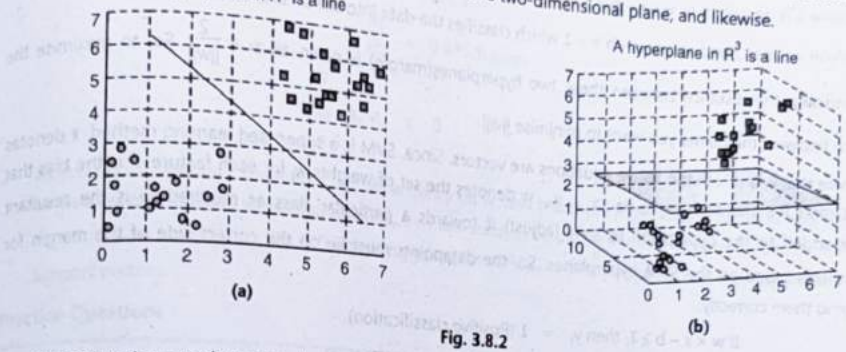


Fig. 3.8.2

- Support vectors are datapoints that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, you maximise the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help in building the SVM model.

3.8.1 Maximum Margin Linear Separators (Optimal Decision Boundary)

- As you understand, the larger the margin between the separating hyperplane the better. Typically, for linearly separable hyperplanes (which is a line), the decision boundary is given as $w \times x - b = 0$.
- So, any positive point above the decision boundary classifies the input as one class and any negative point below the decision boundary classifies the input as another class. You can select two parallel hyperplanes that separate the two classes of data, so that the distance between them is as large as possible.

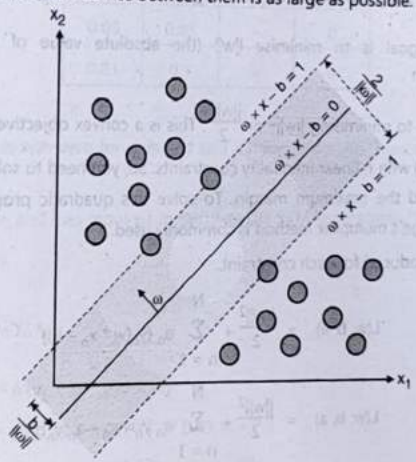


Fig. 3.8.3

- The region bounded by these two hyperplanes is called the "margin", and the maximum-margin hyperplane is the hyperplane that lies halfway between them. With a normalised or standardised dataset, these hyperplanes can be described by the following equations.

  Hyperplane 1 is described as $w \times x - b = 1$ which classifies the data into the first class.

  Hyperplane 2 is described as $w \times x - b = -1$ which classifies the data into the second class.

- Geometrically, the distance between these two hyperplanes(margin) is given as $m = \frac{2}{||w||}$. So, to maximise the distance between the planes you want to minimise $||w||$.

- Note here that w and x in the above equations are vectors. Since, SVM is a supervised learning method, x denotes input sample features such as $x_1, x_2, x_3, \ldots, x_n$. w denotes the set of weights w_i for each feature. b is the bias that can be added to the hyperplane to shift (adjust) it towards a particular class as required. y_i is the resultant classification based on the SVM hyperplanes. So, the datapoints must lie on the correct side of the margin for classifying them correctly.

So, If $w \times x - b \geq 1$, then $y_i = 1$ (Positive classification)

If $w \times x - b \leq -1$, then $y_i = -1$ (Negative classification)

So, you want to minimise $||w||$ (the absolute value of w without sign) subject to $y_i (w \times x_i - b) \geq 1$ for I = 1, 2 ... n

3.9 SVM as Constrained Optimisation Problem

So, you learnt earlier that for SVM, you want to minimise $||w||$ (the absolute value of w without sign) subject to $y_i(w * x_i - b) \geq 1$ for i = 1, 2 ...n. If you look closely, this is a constrained optimisation problem. You are required to minimise $||w||$ given the constraint that $y_i(w * x_i - b) \geq 1$ for i = 1, 2 ...n. In the next section, you would learn how to go about addressing constrained optimisation problem in SVM through quadratic programming.

3.9.1 Quadratic Programming Solution to Finding Maximum Margin Separators

- As you understand, your goal is to minimise $||w||$ (the absolute value of w without sign) subject to $y_i(w * x_i - b) \geq 1$ for i = 1, 2 ...n

- Minimising $||w||$ is equivalent to minimising $||w||^2$ or $\frac{||w||^2}{2}$. This is a convex objective function which is a quadratic program (quadratic equation) with n linear inequality constraints. So, you need to solve a constrained and a convex optimisation problem to find the maximum margin. To solve this quadratic program (or convex optimisation problems in general). Lagrange's multiplier method is commonly used.

- Lagrange multiplier α_n is introduced for each constraint.

$$L(w, b, a) = \frac{||w||^2}{2} + \sum_{n=1}^{N} a_n (y_n (w * x_n - b))$$

$$L(w, b, a) = \frac{||w||^2}{2} + \sum_{n=1}^{N} a_n y_n w x_n - a_n y_n b$$

- To minimise, take partial derivatives of L with respect to w and b and set them to 0.

$$\frac{\partial L}{\partial w} = 0 \text{ which gives}$$

$$w = \sum_{n=1}^{N} a_n y_n x_n$$

$$\frac{\partial L}{\partial b} = 0 \text{ which gives}$$

$$\sum_{n=1}^{N} a_n y_n = 0$$

- As you notice, you have eliminated the dependency on w and b and the minimisation problem now only depends on training examples $(x_n\ y_n)$ and values of a_n. In most cases, a_n would be 0 unless a training instance x_n satisfies the equation $y_n(w * x_i - b) = 1$. Thus, the training examples with $a_n > 0$ lie on the hyperplane margins and hence are support vectors.

Practice Questions

Ex. 3.9.1 : Given the following data, calculate hyperplane. Also, classify (0.6, 0.9) based on the calculated hyperplane.

A1	A2	y	α
0.38	0.47	+	65.52
0.49	0.61	-	65.52
0.92	0.41	-	0
0.74	0.89	-	0
0.18	0.58	+	0
0.41	0.35	+	0
0.93	0.81	-	0
0.21	0.1	+	0

Soln. :

As you see, the value of α is non-zero for only first two training examples. Hence, the first two training examples are support vectors.

Let's calculate the value of w and b as required to calculate the SVM hyperplanes.

Let $w = (w_1, w_2)$

$$w = \sum_{n=1}^{N} a_n y_n x_n$$

$$w_1 = a_1 * y_1 * A1x_1 + a_2 * y_2 * A1x_2$$

$$= 65.52 * 1 * 0.38 + 65.52 * -1 * 0.49 = -7.2$$

$$w_2 = a_1 * y_1 * A2x_1 + a_2 * y_2 * A2x_2$$

$$= 65.52 * 1 * 0.47 + 65.52 * -1 * 0.61 = -9.2$$

The parameter b can be calculated for each support vector as follows.

$$b_1 = 1 - w * x_1$$

$$= 1 - (-7.2) * 0.38 - (-9.2) * 0.47 = 8.06$$

$$b_2 = 1 - w * x_2$$

$$= 1 - (-7.2) * 0.47 - (-9.2) * 0.61 = 10$$

Averaging b_1, b_2 you get b = 9.03

Hence, the hyperplane line is defined as $-7.2\,x_1 - 9.2\,x_2 + 9.03 = 0$

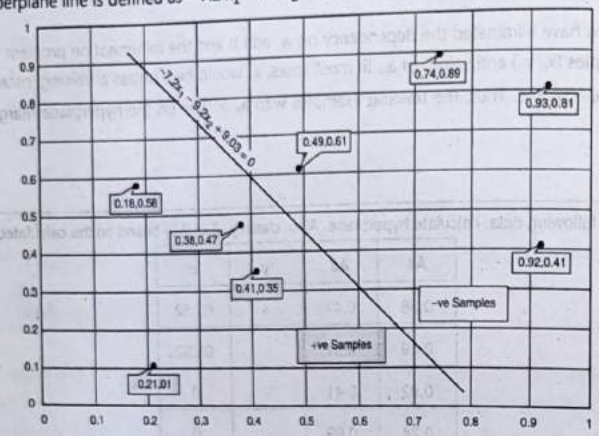


Fig. P. 3.9.1(a)

Now, suppose that there is a new data point (0.6, 0.9) that you want to classify.

Putting the values in the equation $-7.2\,x_1 - 9.2\,x_2 + 9.03$ you get $-7.2 * 0.6 - 9.2 * 0.9 + 9.03 = -3.57$.

Hence, this is classified as negative sample.

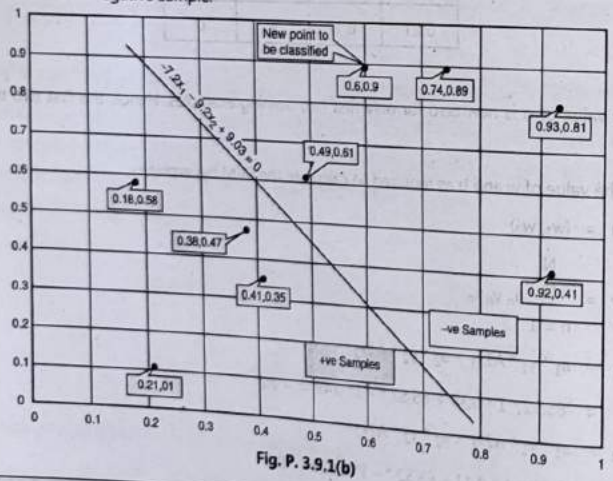


Fig. P. 3.9.1(b)

3.9.2 Kernels for Learning Non-Linear Functions (Kernel Trick)

- So far, you learnt about hyperplanes that could linearly separate the data into two classes using a line very clearly. But what if your dataset looks like the following examples?

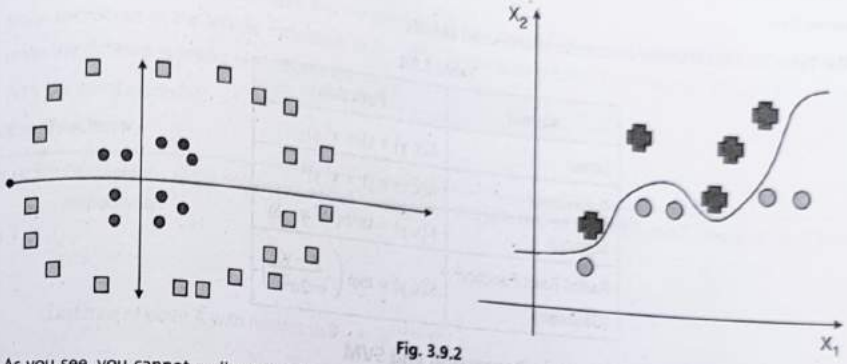


Fig. 3.9.2

- As you see, you cannot really draw a line separating out the classes. In such scenarios, you use a kernel function to map the datapoints such that they can be "uplifted" to higher dimensions and then could possibly be classified.

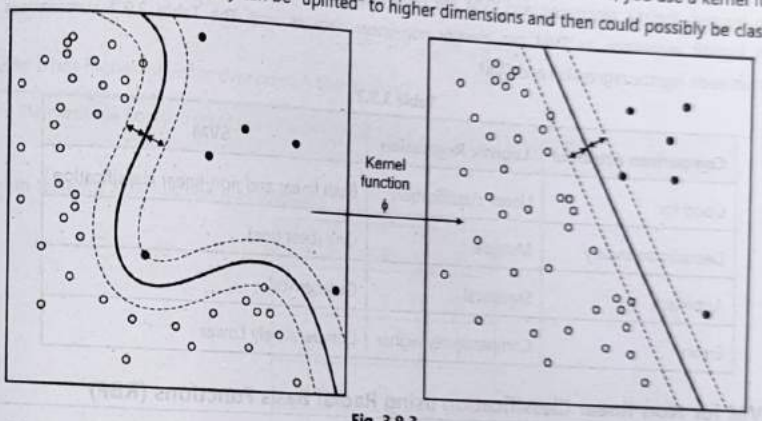


Fig. 3.9.3

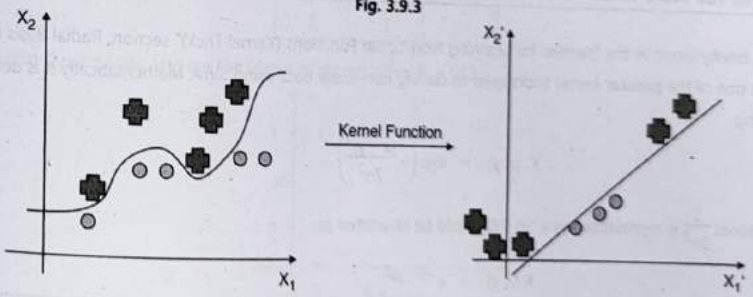


Fig. 3.9.4

- The kernel function is often denoted by $\phi(x)$. You can choose the type of kernel function as appropriate for problem in hand. Some of the commonly used kernel functions are polynomial function, sigmoid function, and radial basis function. This technique of applying a mapping kernel function to adjust the data is also called as kernel trick.
- The Table 3.9.1 summarises various commonly used kernels.

Table 3.9.1

Kernel	Function
Linear	$K(x, y) = (1 + x^T y)$
Polynomial	$K(x, y) = (1 + x^T y)^5$
Sigmoid	$K(x, y) = \tanh(kx^T y - \delta)$
Radial Basis Function (Gaussian)	$K(x, y) = \exp\left(-\frac{(x-y)^2}{2\sigma^2}\right)$

3.9.3 Comparison between Logistic Regression and SVM

As you learned, logistic regression and SVM are both used for data classification. However, SVM is considered better than logistic regression as SVM can classify non-linear data as well. The Table 3.9.2 summarises the key differences between logistic regression and SVM.

Table 3.9.2

Comparison Attribute	Logistic Regression	SVM
Good for	Linear classification	Both linear and non-linear classification
Decision boundary	Multiple	One (best one)
Approach	Statistical	Geometrical
Errors	Comparatively higher	Comparatively Lower

3.10 SVM for Non-linear Classification using Radial Basis Functions (RBF)

- As you briefly learnt in the "Kernels for Learning Non-Linear Functions (Kernel Trick)" section, Radial Basis Function (RBF) is one of the popular kernel tricks used to classify non-linear data using SVM. Mathematically it is denoted as following.

$$K(x, y) = \exp\left(-\frac{(x-y)^2}{2\sigma^2}\right)$$

- Often times $\frac{1}{2\sigma^2}$ is represented as γ. So, RBF could be re-written as

$$K(x, y) = e^{-\gamma(x-y)^2}$$

Note : RBF performs transformation in an infinite dimension. Hence, it is difficult to visualise what it does.

- The way it works is that it finds the influence of the datapoint to be classified with the datapoints that are already classified and assigns the higher-dimensional relationship based on the influence. Higher the influence of the already classified datapoint on the datapoint to be classified, higher the chances of the new datapoint to be classified as the one that influenced the most.
- So, assume that the point x is known, and the point y is to be classified. Then, RBF calculates the distance between those two points in the infinite dimension as $(x - y)^2$ γ is the scaling factor that you may choose as required to make the distance numbers more significant.

Let's see some examples.

Practice Questions

Ex. 3.10.1 : Calculate the radial distance of point A having value of 6 with respect to point B and C having value of 8 and 12 respectively.

Soln. :

Assume $\gamma = 1$

Distance of point A with respect to B $= e^{-\gamma(x-y)^2}$

$= e^{-1(6-8)^2} = e^{-4} = 0.018$

Distance of point A with respect to C $= e^{-\gamma(x-y)^2}$

$= e^{-1(6-12)^2} = e^{-36} = 0$

So, point B has higher influence over point A than point C.

Ex. 3.10.2 : Demonstrate how you could use RBF over XOR function to draw a linear SVM.

Soln. :

The truth table for XOR function is as following.

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

If you plot these points, you get the layout as shown in Fig. P. 3.10.2.

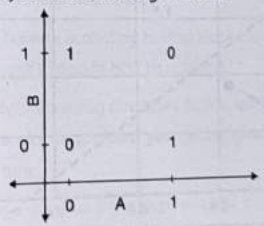


Fig. P. 3.10.2

No matter how hard you try, you would not be able to separate out these points as in the case of a linear SVM. Let's see how RBF helps.

Note that there are two classes here y = {0, 1} and four training samples, two for each class.

{0, 1} and {1, 0} produce 1 (+ ve class) whereas {0, 0} and {1, 1} produce 0 (– ve class). Assume γ = 1.

Let's choose elements {0, 1} and {1, 0} as the basis for RBF.

X (Input)	Basis = {0, 1}	Basis = {1, 0}
{0, 0}	$e^{-\gamma(x-y)^2} = e^{-1(\{0,0\}-\{0,1\})^2} = e^{-1(0-0\,0-1)^2} = e^{-1} = 0.37$	$e^{-\gamma(x-y)^2} = e^{-1(\{0,0\}-\{1,0\})^2} = e^{-1(0-1\,0-0)^2} = e^{-1}$ = 0.37
{0, 1}	$e^{-\gamma(x-y)^2} = e^{-1(\{0,1\}-\{0,1\})^2} = e^{-1(0-0\,1-1)^2} = e^{0} = 1$	$e^{-\gamma(x-y)^2} = e^{-1(\{0,1\}-\{1,0\})^2} = e^{-1(0-1\,1-0)^2} = e^{-4}$ = 0.018
{1, 0}	$e^{-\gamma(x-y)^2} = e^{-1(\{1,0\}-\{0,1\})^2} = e^{-1(1-0\,0-1)^2} = e^{-4} = 0.018$	$e^{-\gamma(x-y)^2} = e^{-1(\{1,0\}-\{1,0\})^2} = e^{-1(1-1\,0-0)^2} = e^{0} = 1$
{1, 1}	$e^{-\gamma(x-y)^2} = e^{-1(\{1,1\}-\{0,1\})^2} = e^{-1(1-0\,1-1)^2} = e^{-1} = 0.37$	$e^{-\gamma(x-y)^2} = e^{-1(\{1,1\}-\{1,0\})^2} = e^{-1(1-1\,1-0)^2} = e^{-1}$ = 0.37

So, RBF does the following transformations to the respective points.

Old Point	New Point
{0, 0}	{0.37, 0.37}
{0, 1}	{1, 0.018}
{1, 0}	{0.018, 1}
{1, 1}	{0.37, 0.37}

If you plot these points, you see that they are now clearly separatable as in the case of a linear SVM.

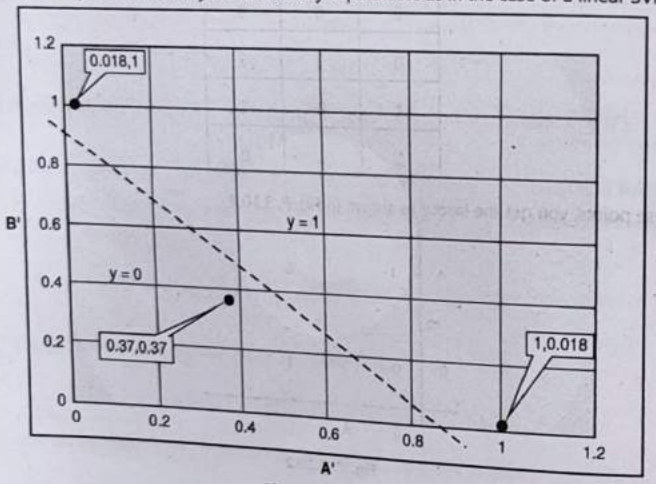


Fig. P. 3.10.2(a)

Points {1, 0.018} and {0.018, 1} give output y = 1.

Points {0.37, 0.37} and {0.37, 0.37} give output y = 0.

Hence, as you see, RBF transformed the XOR points to make them linearly separable.

3.10.1 The Radial Basis Function (RBF) Network

- As you understand, the core concept behind RBF is that inputs that are close together should generate the same output, whereas inputs that are far apart should not.
- Points that are close together exercise more influence over each other than the points that are far apart.
- As you previously learnt about neural networks, for any input that you present to a set of the neurons, some of them will fire strongly, some weakly, and some will not fire at all, depending upon the distance between the weights and the particular input in weight space.
- You can treat these nodes as a hidden layer, just as you did for the Multilayer Perceptrons, and connect up some output nodes in a second layer.
- This simply requires adding weights from each hidden (RBF) neuron to a set of output nodes. This is known as an RBF network.

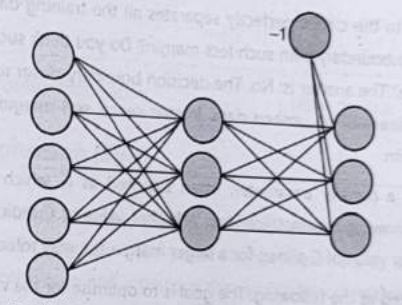


Fig. 3.10.1

- The Radial Basis Function network consists of input nodes connected by weights to a set of RBF neurons, which fire proportionally to the distance between the input and the neuron in weight space.
- The activations of these nodes are used as inputs to the second layer, which consists of linear nodes. RBF networks never have more than one layer of non-linear neurons.
- In an RBF network, input nodes will activate according to how close they are to the input, and the combination of these activations will enable the network to decide how to respond.
- For example, suppose that somebody is signalling directions to you using a torch. If the torch is high (in 12 o'clock position) you go forwards, low (in 6 o'clock position) you go backwards, and left and right (in 9 o'clock and 3 o'clock, respectively) mean that you turn.
- The RBF network would work in such a way that if the torch was at 2 o'clock or so, then you would do some of the 12 o'clock action and a bit more of the 3 o'clock action, but none of the 6 o'clock or 9 o'clock actions. So, you would move forwards and to the right.

3.10.2 Soft Margin SVM

- In real-world scenarios, the data might not be perfectly separable, meaning that there may be overlapping or misclassified points. In such cases, using a hard margin SVM (which requires strict separation) might not be practical. Soft margin SVM allows for some degree of misclassification by allowing data points to fall within the margin or even on the wrong side of the decision boundary.
- For example, in the Fig. 3.10.2, which boundary is better? The one closer to the circles or the one closer to the squares?

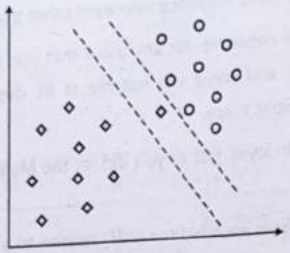


Fig. 3.10.2

- The decision boundary closer to the circles perfectly separates all the training data points. However, is it really a good idea of having a decision boundary with such less margin? Do you think such kind of decision boundary will generalise well on unseen data? The answer is: No. The decision boundary closer to the squares has a wider margin that would allow it to generalise well on unseen data. In that sense, soft margin formulation would also help in avoiding the overfitting problem.
- Soft margin SVM introduces a penalty parameter, often denoted as C, which controls the trade-off between maximising the margin and allowing misclassifications. A higher value of C indicates a smaller margin but fewer misclassifications, while a lower value of C allows for a larger margin but may tolerate more misclassifications.
- Mathematically, it is represented as the following. The goal is to optimise for the value of C.

$$L = \frac{1}{2} \|w\|^2 + C$$

- To illustrate this, consider a scenario where you have two groups of points in a scatter plot, but they are not perfectly separable. With a soft margin SVM, the algorithm might allow a few points from one group to fall within the margin or on the wrong side of the decision boundary if it helps achieve a better overall separation.
- For example, let's say you have a dataset of flowers with two classes: roses and daisies. Some roses and daisies are quite similar, making it challenging to draw a perfectly separating line. A soft margin SVM would find a decision boundary that allows for a few misclassified points, such as a rose falling within the margin on the daisy side or a daisy on the rose side. This flexibility helps accommodate the inherent noise or overlapping nature of the data.
- In summary, soft margin SVM extends the concept of SVM by allowing some degree of misclassification in situations where strict separation is not feasible. It introduces a penalty parameter C to control the trade-off between margin size and misclassification tolerance. This flexibility enables the algorithm to handle real-world datasets with noise and overlapping classes, providing a more practical solution for classification tasks.

Comparison between Hard Margin and Soft Margin SVM

The Table 3.10.1 provides a quick comparison between hard margin and soft margin SVM.

Table 3.10.1

Comparison Attribute	Hard Margin SVM	Soft Margin SVM
Goal	Achieve strict separation	Allow for some misclassifications
Scenario	Data is perfectly separable	Data may have overlapping points
Misclassifications	Not allowed	Allowed within a certain tolerance
Penalty parameter (C)	Not applicable	Determines trade-off between margin size and misclassification
Margin Size	Maximises margin, potentially overfitting the data	Balances margin size and misclassification tolerance
Sensitivity to Outliers	Highly sensitive	More robust due to tolerance for misclassifications
Complexity	Simpler problem formulation	More flexible, handles complex data

3.11 Support Vector Regression (SVR)

- As you understand, Support Vector Machines (SVM) are popularly and widely used for classification problems in machine learning. Support Vector Regression (SVR) uses the same principle as SVM, but for regression problems. The goal of a regression problem is to find a function that approximates mapping from an input domain to real numbers on the basis of a training sample.

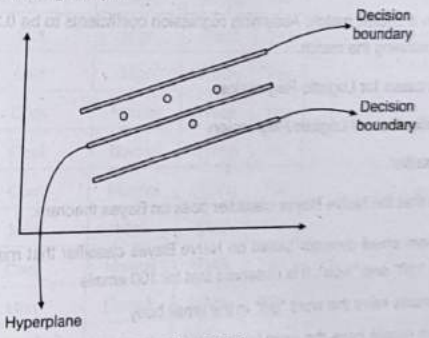


Fig. 3.11.1

- As in case of SVM, consider the two decision boundaries and a hyperplane. Your objective is to consider the points that are within the decision boundary line. The best fit line (or regression line) is the hyperplane that has a maximum number of points. Assume that the decision boundaries are at any distance, say 'a', from the hyperplane.

So, these are the lines that you draw at distance '+a' and '-a' from the hyperplane. Based on SVM, the equation of the hyperplane is as following.

$$Y = wx - b,$$

- The equations of decision boundaries are as following.

$$wx - b = a$$

$$wx - b = -a$$

- Thus, any hyperplane that satisfies SVR should satisfy $-a \leq wx - b \leq a$
- Your goal is to decide a decision boundary at 'a' distance from the original hyperplane such that data points closest to the hyperplane or the support vectors are within that boundary line.

Review Questions

Here are a few review questions to help you gauge your understanding of this chapter. Try to attempt these questions and ensure that you can recall the points mentioned in the chapter.

Classification Model

Q. 1 With an example, explain need for a classification model. **(4 Marks)**

Q. 2 Describe the types of classification. **(4 Marks)**

Q. 3 Compare binary and multiclass classification techniques. **(4 Marks)**

Binary Classification

Q. 4 Explain logistic regression and scenarios where it might be suitable. **(6 Marks)**

Q. 5 Draw Sigmoid curve and explain its significance. **(6 Marks)**

Q. 6 You are applying for a home loan and your credit score is 720. Assuming logistic regression coefficient to be -9.346 and 0.0146 respectively, calculate the probability of your home loan application getting approved. **(4 Marks)**

Q. 7 A team scored 285 runs in a cricket match. Assuming regression coefficients to be 0.3548 and 0.00089 respectively, calculate its probability of winning the match. **(4 Marks)**

Q. 8 Describe some of the use cases for Logistic Regression. **(4 Marks)**

Q. 9 Describe some of the applications of Logistic Regression. **(4 Marks)**

Q. 10 Explain Naïve Bayes Classifier. **(6 Marks)**

Q. 11 Explain the modifications that the Naïve Bayes classifier does on Bayes theorem. **(6 Marks)**

Q. 12 You have designed a spam email detector based on Naïve Bayes classifier that marks emails as spam if the email body has both the words "gift" and "won". It is observed that for 100 emails

- 20 out of 25 spam emails have the word "gift" in the email body
- 5 out of 75 non-spam emails have the word "gift" in the email body
- 15 out of 25 spam emails have the word "won" in the email body
- 10 out of 75 non-spam emails have the word "won" in the email body
- What would be the accuracy of your spam detector? **(6 Marks)**

Q. 13 Given the following dataset, find out if it is likely that the Blue Jeans from Brand X would be on Sale using Naïve Bayes Classifier. **(8 Marks)**

Colour	Cloth Type	Brand	On Sale?
Blue	Jeans	Y	Yes
Blue	Shirt	X	No
Blue	Jeans	Y	Yes
Black	Jeans	Y	Yes
Black	Jeans	X	No
Black	Shirt	X	Yes
Black	Shirt	Y	Yes
Blue	Jeans	Y	No
Black	Jeans	X	Yes
Blue	Shirt	X	Yes

Q. 14 Given the following datapoints, what would be the fruit classification based on Naïve Bayes classifier? **(8 Marks)**

Fruit Name	Yellow	Sweet	Long	Total Samples
Orange	350	450	0	650
Banana	400	300	350	400
Guava	50	100	50	150

Q. 15 Using Naïve Bayes classifier, find out if the following weather condition is favourable for playing. **(8 Marks)**
Outlook = Rainy, Temperature = Cool, Humidity = High and Wind = Strong

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

4 Clustering

Unit 4

Syllabus

At the end of this unit, you should be able to understand and comprehend the following syllabus topics:

- Introduction
  - What is clustering
  - Need of Clustering
  - Types of Clustering
- Hierarchical clustering algorithms (connectivity-based clustering)
  - Agglomerative Hierarchical Clustering (AHC) algorithm
  - Divisive Hierarchical Clustering (DHC) algorithm
- Centroid-based clustering algorithms / Partitioning clustering algorithms
  - K-Means clustering algorithm
    - Advantages and disadvantages of K-Means clustering algorithm
    - Elbow method
    - The Silhouette method
  - K-Medoids
  - K-Prototype
- Density-based clustering algorithms
  - DBSCAN algorithm
    - How it works
    - Advantages and disadvantages of DBSCAN
- Distribution-based clustering algorithms
  - Gaussian mixture model
- Application of Clustering Technique
  - Market Segmentation
  - Statistical data analysis
  - Social network analysis
  - Image segmentation
  - Anomaly detection

4.1 Clustering

- The dictionary meaning of cluster is "a number of similar things that occur together". For example, cluster of stars in the galaxy.

With respect to machine learning,

Definition : *Clustering is a technique in which the data points are arranged in similar groups dynamically without any pre-assignment of groups.*

- It is the task in which the data points are grouped in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups (clusters). It is a data exploration technique commonly used to understand how the data should be interpreted and grouped to be best analysed. There is no prior learning of groupings required. Instead, groups are implicitly arranged (or created) based on the data attributes. How the data is grouped depends on the type of algorithm used. The same dataset can be used to form various clusters depending upon the data attributes and then you can decide which clusters make sense to be used for further data analytics.
- For example, here is a simple plot of data points. As you can see, some set of points are closer to each other when compared with others. These sets could possibly form a group (or a cluster) for further data analytics.

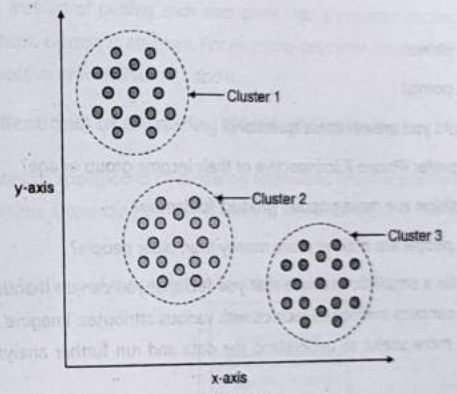


Fig. 4.1.1

- Let's take a simple example to understand the clustering technique.

Customer Name	Region	Monthly Income	Age	Phone Model	Laptop Model
A	Bangalore	50,000	29	iPhone X	MacBook Pro
B	Bangalore	35,000	34	Motorola X	Dell Precision
C	Mumbai	80,000	36	iPhone X	Dell Precision
D	Mumbai	40,000	26	iPhone X	Dell Precision
E	Mumbai	55,000	39	iPhone X	MacBook Pro

- You could build several clusters (groups) for analysis. For example,
- Based on region
  - Bangalore (2 data points)
  - Mumbai (3 data points)
- Based on monthly income
  - More than 50,000 (3 data points)
  - Less than 50,000 (2 data points)
- Based on age
  - Under 30 (2 data points)
  - 30-35 (1 data point)
  - 35 and above (2 data points)
- Based on phone model preference
  - iPhone X (4 data points)
  - Motorola X (1 data point)
- Based on laptop model
  - MacBook Pro (2 data points)
  - Dell Precision (3 data points)
- Based on which cluster could you answer these questions?
  1. Is it true that people prefer iPhone X irrespective of their income group or age?
  2. Is it true that Dell Precision is a more popular product for laptop?
  3. On average, younger people are making more money than older people?
- Note here that this might be a simplistic example that you feel that you can do manually. But, for all practical and useful purposes, a dataset contains millions of records with various attributes. Imagine how hard could it be to find the clusters that could be more useful to understand the data and run further analysis on it. This is not a trivial problem to solve!
- Also, note here that clustering is a data exploration method (or a task where clusters are outcome) and not an algorithm or analytics technique in itself.
- You can use various algorithms such as the following for clustering the data or could do it yourself (if you have patience and time to kill) without an algorithm (like for the simplistic example I gave earlier).
  - Partition clustering (also called K-means clustering)
  - Hierarchical clustering
  - Fuzzy clustering
  - Model based clustering
  - Density-based clustering
- Clustering is used in recommendation and search engines, marketing, economics, and other branches of science.

4.1.1 Properties of a Cluster

Based on the cluster plot and the example given in the previous section, it is evident that typically clusters have the following properties.

1. **All the data points in a cluster should be similar to each other :** By definition, a cluster is a grouping of similar objects. So, a good cluster must have data points that indeed have some convincing similarities on the basis of which they are grouped. For example, phone preference cluster mentioned in the example in the previous section could be a convincing enough cluster.
2. **The data points from different clusters should be as different as possible :** To make clusters distinct enough, the data points from different clusters should be far apart or, in other words, must be clearly distinguishable. For example, a cluster containing all customers whose preference is iPhone Model X is clearly distinguishable from another cluster containing customers preferring Motorola X.

4.1.2 Types of Clustering

At a high-level, there are two types of clustering.

1. **Hard Clustering** : In this, each data point belongs to only one cluster at a time. For example, customer A (in the previously given example) could only be in iPhone X cluster if you clustered based on phone model preference.
2. **Soft Clustering :** In this, instead of putting each data point into a separate cluster, a probability or likelihood of that data point to be in those clusters is assigned. For example, customer A would be in both the clusters -iPhone X and Motorola X with respective probabilities of 1 and 0.

4.1.3 Use Cases (Applications) of Clustering (Need of Clustering)

There are several use cases or applications of clustering technique. These applications are not specific to K-means or any other clustering algorithm. Depending on the dataset to be clustered and the desired clustering outcome any algorithm could be chosen.

Some of the common applications of clustering are as following.

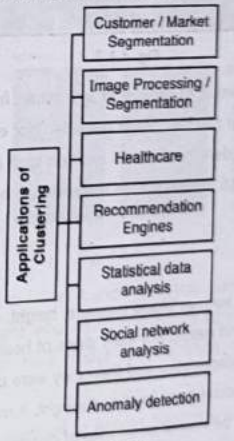


Fig. 4.1.2

1. Customer / Market Segmentation

- One of the most common applications of clustering is customer segmentation. Various companies build customer segment profiles to understand the customer's shopping behaviour and accordingly build strategies for predicting demand and improving sales.
- As you saw in the example given in the previous section, various customers could be clustered to understand their purchase behaviours and appropriate and targeted marketing campaign could be run for different segments. For example, customers who own iPhone Model X could be shown other iPhone related accessories or other products from the same company.
- Another example could be a woman buying clothes for her 3-month old infant. Based on the size chosen, the company could send her promotional offers for next size of clothes she is likely to purchase. For Example, if she bought cloth size fitting 3 months old infant, after 9 months, she could be sent promotional offers on cloth size for 1 year old baby. The chances of her buying clothes for 1 year old baby are quite high. The company could identify such clusters of mothers who would require 1 year old baby clothes and accordingly send them promotional offers.

2. Image Processing / Segmentation

- Using clustering technique, you could identify objects in an image. You can cluster similar pixels in the image together and then match that with known objects to identify the objects in the image.

Fig. 4.1.3

- Additionally, you can use this technique to capture image frames from a video and then determine objects in the video. Based on the objects you can categorise the video (for example, a video having animals), probable location of the video (if you could identify a famous location such as the Gateway of India) or identify people in the video. Image and video based analytics have reached very advanced level these days where you could carry out various useful tasks.

3. Healthcare

- Patient data, containing attributes such as blood pressure, height, weight, cholesterol level and glucose level, could be used to create clusters and detect any early signs of health problems based on historical records of other patients who had similar characteristics and then they were diagnosed with a particular health problem. For example, if someone has high cholesterol level and weight, it might be possible to detect if she is closer to getting a heart attack. There could be several other uses of clustering in the healthcare industry.

- For example, detecting cancer, its type and severity and the possible treatment plan at early stage could have a huge impact on number of years the patient is expected to survive. The shape of the cancerous cells plays a vital role in determining the severity of the cancer. Using clustering, you can determine the shape of the cancerous cells and accordingly create a treatment plan.

Fig. 4.1.4

4. Recommendation Engines

- I am sure that you would have seen video or song recommendations on your favourite media apps or the product recommendations as you browse through an e-commerce portal. Some of the examples are as shown in Fig. 4.1.5.

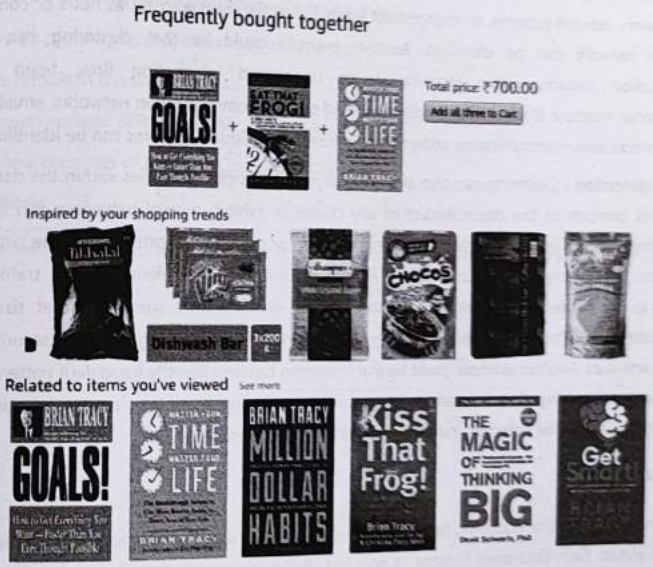


Fig. 4.1.5

- How does recommendation work? How does it know what you may like? Basically, the organisations have the data for you (such as what you bought, what items you looked at, where you clicked, etc.) and several of their other site or app visitors. Based on how similar you are when compared with others (creating clusters), recommendations are made to you.
- The collected data is clustered and analysed to answer questions such as "how likely it is that someone who has bought the product X will also buy product Y?". If the probability is quite high, then the recommendation

is made. For example, if customer A has bought Book 1 and also Book 2 historically and then customer B has just bought Book 1 now, she is recommended that Book 2 might be of interest to her.

5. **Statistical data analysis :** Statistical data analysis refers to the process of analysing and interpreting data to uncover patterns, relationships, and insights using statistical methods. It involves the application of statistical techniques and tools to understand the underlying structure of data, make inferences, and draw conclusions. Clustering is a powerful technique within statistical data analysis that enables the identification of groups or clusters of similar data points. For example, clustering can aid in identifying anomalous patterns in financial transactions to detect fraudulent activities. By clustering transactions based on variables like transaction amount, time, or location, unusual patterns can be identified and flagged for further investigation. Clustering can be used to analyse environmental data, such as air quality measurements or climate variables, to identify regions or time periods with similar patterns. This aids in understanding environmental patterns, detecting pollution sources, or predicting future trends.

6. **Social network analysis :** Clustering can be used to identify communities or groups within social networks based on patterns of connections or interactions. This helps in understanding the structure of social networks, detecting influencers, or targeting specific groups for marketing or intervention. For example, clustering can assist in identifying influential users or opinion leaders in social media platforms. By clustering users based on their followers, retweet patterns, or engagement levels, influential users who act as hubs or connectors within the social network can be identified. Another example could be that clustering can help analyse communication patterns within organisations to understand information flow, team dynamics, or organisational structure. By clustering employees based on their communication networks, email exchanges, or meeting interactions, communication patterns and influential individuals or teams can be identified.

7. **Anomaly detection :** Clustering can also aid in identifying outliers or anomalies within the data. Data points that do not conform to the characteristics of any cluster or exhibit unusual behaviour can be detected as outliers. These anomalies may warrant further investigation or could be indicative of unique patterns or errors in the data. For example, clustering can be employed to detect anomalous network traffic or potential cyberattacks. By clustering network traffic data based on characteristics such as packet size, protocol, or source/destination IP addresses, unusual patterns can be identified, indicating potential security breaches or malicious activities. Another example could be that clustering can help identify fraudulent patterns in insurance claim data. By clustering claims based on features like claim amount, location, or type of damage, unusual clusters or outliers can be detected, indicating potentially fraudulent claims.

4.1.4 K-means

K-means is one of the popular clustering techniques (or algorithm). It helps to form clusters from the given dataset for further data analytics. Each data point belongs to only one cluster. You pre-decide the number of clusters, *k*, that you want to group your datapoints into before executing the algorithm steps. Let's learn about how it works.

Overview of the Method

K-means algorithm is designed to minimise the sum of distances between the data points and their respective cluster centroid. Each cluster is associated with a centroid.

In coordinate geometry,

Definition : *Centroid is a point whose coordinates are the averages of the corresponding coordinates of a given set of points.*

At a high-level, the following steps are taken for clustering the data using K-means clustering algorithm.

1. Decide the number of clusters, *k*, that you desire to group your data points into.
2. Select *k* random data points as centroids.
3. Compute the distance from each data point to each centroid. Assign all the data points to the closest cluster centroid. The distance *d* between any two points, (x_1, y_1) and (x_2, y_2) is calculated as

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

4. Recompute the centroids of newly formed clusters. The centroid (X_c, Y_c) of the *m* data points in a cluster is calculated as

$$(X_c, Y_c) = \left(\frac{\sum_{i=1}^{m} X_i}{m}, \frac{\sum_{i=1}^{m} Y_i}{m} \right)$$

It is a simple arithmetic mean of all X coordinates and Y coordinates of the *m* data points in the cluster.

5. Repeat steps 3 and 4 until any of the following criteria is met
   a. Centroids of newly formed clusters do not change.
   b. Points remain in the same cluster.
   c. Maximum number of iterations are reached as desired.

Let's see a few examples of K-means Clustering Algorithm.

Practice Questions

Ex. 4.1.1 : Use the following data and group them using K-means clustering algorithm. Show calculation of centroids.

Height	Weight
185	72
170	56
168	60
179	68
182	72
188	77
180	71
180	70
183	84
180	88
180	67
177	76

Soln. :

Let's assume height on X-axis and weight on Y-axis. Let's assume that we need two clusters. Hence, k = 2. Let's decide that we would do centroid calculation and data points to cluster assignment twice (number of iterations) in the algorithm. The first plot of the given data points is as shown in Fig. P. 4.1.1.

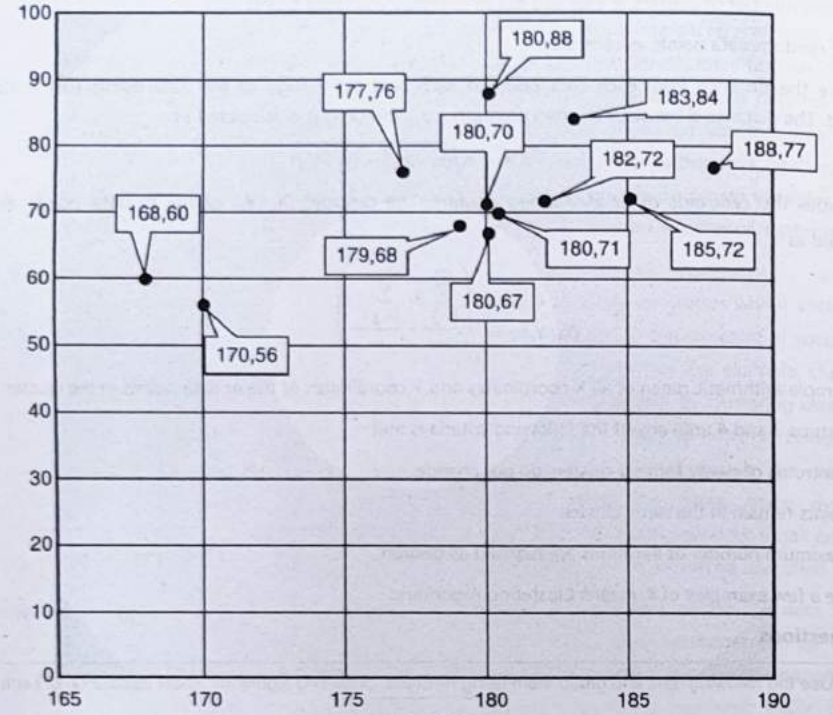


Fig. P. 4.1.1

Iteration 1

Let's randomly choose two centroids.

Centroid 1 = (170, 56) and Centroid 2 = (182, 72).

Now, let's calculate the distance of the data points from the chosen centroids and complete the data points table. The data point is assigned to the cluster based on the closest centroid.

A sample distance calculation is as following for the first data point.

Distance from Centroid 1 (170, 56) for (185, 72) =

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

$$d = \sqrt{(170 - 185)^2 + (56 - 72)^2}$$

$$d = 21.93$$

Height	Weight	Distance from Centroid 1	Distance from Centroid 2	Assigned Cluster
185	72	21.93	3.00	2
170	56	0.00	20.00	1
168	60	4.47	18.44	1
179	68	15.00	5.00	2
182	72	20.00	0.00	2
188	77	27.66	7.81	2
180	71	18.03	2.24	2
180	70	17.20	2.83	2
183	84	30.87	12.04	2
180	88	33.53	16.12	2
180	67	14.87	5.39	2
177	76	21.19	6.40	2

Let's re-calculate the centroids for the next iteration.

For Centroid 1 calculation, you have two data points that fall in cluster 1. They are (170, 56) and (168, 60).

Hence, Centroid 1 is the mean of the data points which is $\left(\frac{170 + 168}{2}, \frac{56 + 60}{2}\right)$ = (169, 58).

For Centroid 2 calculation, take the remaining 10 data points that fall in cluster 2.

Hence, Centroid 2 is the mean of these 10 data points which is

$$\left(\frac{185 + 179 + 182 + 188 + 180 + 180 + 183 + 180 + 180 + 177}{10}, \frac{72 + 68 + 72 + 77 + 71 + 70 + 84 + 88 + 67 + 76}{10}\right)$$

= (181.4, 74.5)

Now, you have both the centroids ready for the next and final iteration.

Iteration 2

Centroid 1 = (169, 58) and Centroid 2 = (181.4, 74.5).

Now, let's calculate the distance of the data points from the centroids and complete the data points table. The data point is assigned to the cluster based on the closest centroid.

Height	Weight	Distance from Centroid 1	Distance from Centroid 2	Assigned Cluster
185	72	21.26	4.38	2
170	56	2.24	21.73	1
168	60	2.24	19.74	1
179	68	14.14	6.93	2
182	72	19.10	2.57	2
188	77	26.87	7.06	2
180	71	17.03	3.77	2
180	70	16.28	4.71	2
183	84	29.53	9.63	2
180	88	31.95	13.57	2
180	67	14.21	7.63	2
177	76	19.70	4.65	2

You stop here because the number of iterations that you decided are completed and also you see that the clusters assigned in the iteration 1 for the data points did not change.

Hence, you got the two clusters as shown in Fig. P. 4.1.1(a).

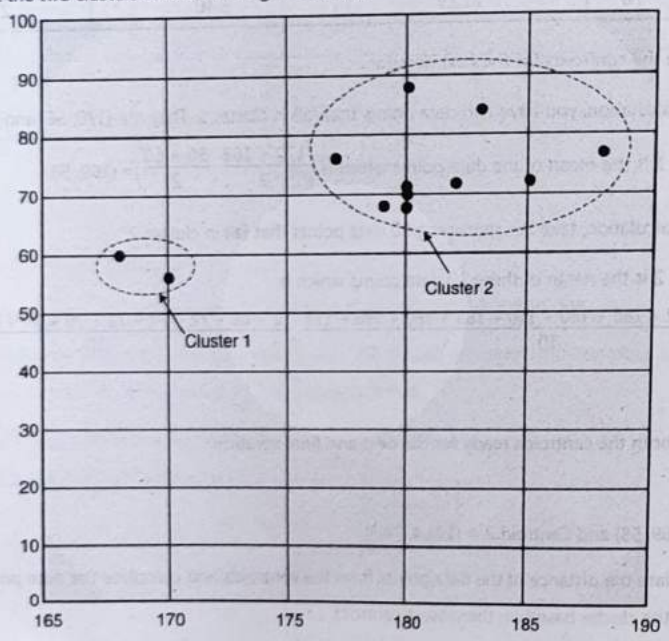


Fig. P. 4.1.1(a)

Ex. 4.1.2 : In a survey, the following data was collected for TV viewership. Is there a group which watches more TV than the other?

Age	Number of TV watching hours
23	3
25	2
28	2
35	4
37	5
40	4
34	3
41	5
39	4
38	3

Soln. :

Let us use K-means clustering for grouping the survey results. Let's choose the desired number of clusters, k = 2 and the number of iterations for centroid calculation and cluster assignment to be 2 as well. Let's put age on X-axis and TV watching hours on Y-axis.

The first plot of data points is as shown in Fig. P. 4.1.2.

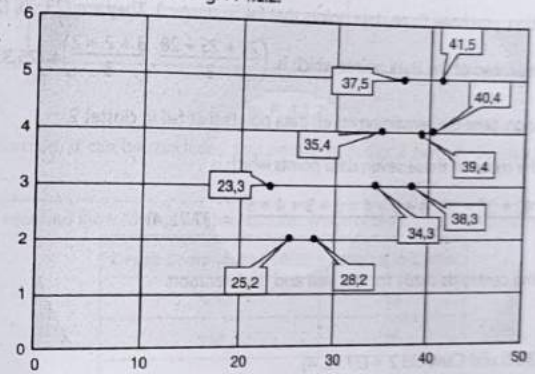


Fig. P. 4.1.2

Iteration 1 : Let's randomly choose two centroids.

Centroid 1 = (25, 2) and Centroid 2 = (35, 4).

Now, let's calculate the distance of the data points from the chosen centroids and complete the data points table. The data point is assigned to the cluster based on the closest centroid.

A sample distance calculation is as following for the first data point.

Distance from Centroid 1 (25, 2) for (23, 3) =

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

$$d = \sqrt{(25-23)^2 + (2-3)^2}$$

$$d = 2.24$$

Age	Hours	Distance from Centroid 1	Distance from Centroid 2	Assigned Cluster
23	3	2.24	12.04	1
25	2	0.00	10.20	1
28	2	3.00	7.28	1
35	4	10.20	0.00	2
37	5	12.37	2.24	2
40	4	15.13	5.00	2
34	3	9.06	1.41	2
41	5	16.28	6.08	2
39	4	14.14	4.00	2
38	3	13.04	3.16	2

Now re-calculate the centroids for the next iteration.

For Centroid 1 calculation, you have three data points that fall in cluster 1. They are (23, 3), (25, 2) and (28, 2).

Hence, Centroid 1 is the mean of the data points which is $\left(\frac{23+25+28}{3}, \frac{3+2+2}{3}\right) = (25.33, 2.33)$.

For Centroid 2 calculation, take the remaining seven data points that fall in cluster 2.

Hence, Centroid 2 is the mean of these seven data points which is

$$\left(\frac{35+37+40+34+41+39+38}{7}, \frac{4+5+4+3+5+4+3}{7}\right) = (37.71, 4)$$

Now, you have both the centroids ready for the next and final iteration.

Iteration 2

Centroid 1 = (25.33, 2.33) and Centroid 2 = (37.71, 4).

Now, let's calculate the distance of the data points from the centroids and complete the data points table. The data point is assigned to the cluster based on the closest centroid.

Age	Hours	Distance from Centroid 1	Distance from Centroid 2	Assigned Cluster
23	3	2.42	14.74	1
25	2	0.47	12.87	1
28	2	2.69	9.91	1
35	4	9.81	2.71	2
37	5	11.97	1.23	2

Age	Hours	Distance from Centroid 1	Distance from Centroid 2	Assigned Cluster
40	4	14.76	2.29	2
34	3	8.70	3.84	2
41	5	15.90	3.44	2
39	4	13.77	1.29	2
38	3	12.69	1.04	2

You stop here because the number of iterations that you decided are completed and also you see that the clusters assigned in the iteration 1 for the data points did not change.

Hence, you got the two clusters as shown in Fig. P. 4.1.2(a).

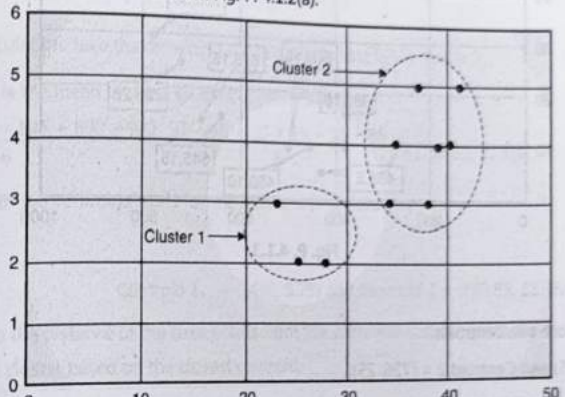


Fig. P. 4.1.2(a)

From the clusters formed, it can be concluded that people in cluster 2 seem to be watching more TV than people in cluster 1.

Ex. 4.1.3 : A bank has received the following loan applications. Which of the applications could be risky to approve?

Credit Score (out of 1000)	Amount in Lakhs
500	10
726	25
430	5
678	15
780	30
380	10
645	15
890	50
900	65
450	10

Soln. :

Let us use K-means clustering for grouping the loan applications. Let's choose the desired number of clusters, k = 2 and the number of iterations for centroid calculation and cluster assignment to be 2 as well. Let's put credit score on X-axis and loan amount on Y-axis.

The first plot of data points is as shown in Fig. P. 4.1.3.

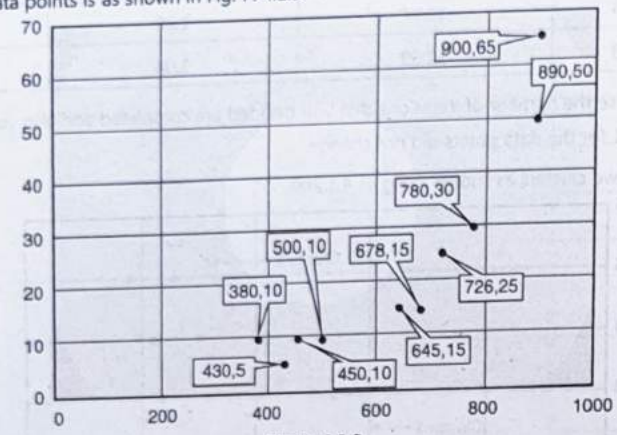


Fig. P. 4.1.3

Iteration 1

Let's randomly choose two centroids.

Centroid 1 = (430, 5) and Centroid 2 = (726, 25).

Now, let's calculate the distance of the data points from the chosen centroids and complete the data points table. The data point is assigned to the cluster based on the closest centroid.

A sample distance calculation is as following for the first data point.

Distance from Centroid 1 (430, 5) for (500, 10) =

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

$$d = \sqrt{(430 - 500)^2 + (5 - 10)^2}$$

$$d = 70.18$$

Credit Score	Amount	Distance from Centroid 1	Distance from Centroid 2	Assigned Cluster
500	10	70.18	226.50	1
726	25	296.67	0.00	2
430	5	0.00	296.67	1
678	15	248.20	49.03	2
780	30	350.89	54.23	2
380	10	50.25	346.32	1
645	15	215.23	81.61	2

Credit Score	Amount	Distance from Centroid 1	Distance from Centroid 2	Assigned Cluster
890	50	462.20	165.89	2
900	65	473.81	178.54	2
450	10	20.62	276.41	1

Now re-calculate the centroids for the next iteration.

For Centroid 1 calculation, you have four data points that fall in cluster 1.

Hence, Centroid 1 is the mean of the data points which is

$$\left(\frac{500 + 430 + 380 + 450}{4}, \frac{10 + 5 + 10 + 10}{4}\right) = (440, 8.75).$$

For Centroid 2 calculation, take the remaining six data points that fall in cluster 2.

Hence, Centroid 2 is the mean of these six data points which is

$$\left(\frac{726 + 678 + 780 + 645 + 890 + 900}{6}, \frac{25 + 15 + 30 + 15 + 50 + 65}{6}\right) = (769.83, 33.33)$$

Now, you have both the centroids ready for the next and final iteration.

Iteration 2

Centroid 1 = (440, 8.75) and Centroid 2 = (769.83, 33.33).

Now, let's calculate the distance of the data points from the centroids and complete the data points table. The data point is assigned to the cluster based on the closest centroid.

Credit Score	Amount	Distance from Centroid 1	Distance from Centroid 2	Assigned Cluster
500	10	60.01	270.84	1
726	25	286.46	44.61	2
430	5	10.68	341.01	1
678	15	238.08	93.64	2
780	30	340.66	10.70	2
380	10	60.01	390.53	1
645	15	205.10	126.17	2
890	50	451.89	121.32	2
900	65	463.43	133.97	2
450	10	10.08	320.68	1

You stop here because the number of iterations that you decided are completed and also you see that the clusters assigned in the iteration 1 for the data points did not change.

Hence, you got the two clusters as shown in Fig. P. 4.1.3(a).

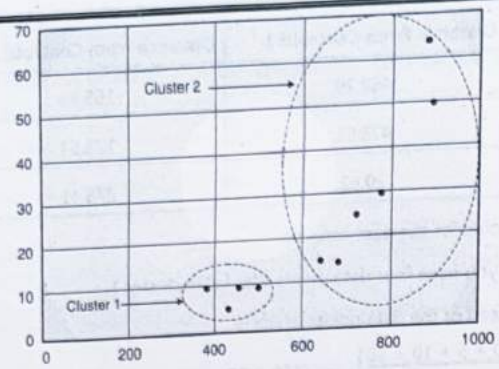


Fig. P. 4.1.3(a)

From the clusters formed, it can be concluded that people in cluster 2 seem to be less risky for granting loan.

4.1.5 Determining the Number of Clusters (Elbow Method)

- To use K-means algorithm you are required to choose the desired number of clusters, the value *k*. But, how do you choose the right number of clusters for a given set of data points?
- Most of the times you choose the value of k by seeing the data plot (guess) or on the basis of your requirements (say you need 3 clusters only).
- But the number you choose in these ways may not be the optimum number to divide your data points into clusters. So, how could you choose the right number? Let's learn about it.
- Do you remember the first property of a cluster? It was "All the data points in a cluster should be similar to each other".
- So, what does this mean? It means that the data points within a good cluster should be as close to each other as possible. In other words, the distance between the data points and the centroid of that cluster should be as minimal as possible.
- This is called Inertia of a cluster.

✍ ***Definition :*** *Inertia is the sum of distances of all the data points within a cluster from the centroid of that cluster.*

- It is also called as intracluster distance. For example, the inertia of the following cluster would be sum of the distances of its data points from its centroid.

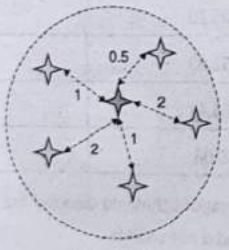


Fig. 4.1.6

This would be 1 + 2 + 1 + 2 + 0.5 = 6.5.

- So, now suppose that you have three clusters having inertia of 5, 10 and 6 respectively. Then, the total inertia of the clusters formed and the data points within them would be sum of individual inertia which is 5 + 10 + 6 = 21.
- Your goal is to ensure that the final clusters created have minimal total inertia to ensure that you have the optimum number of clusters created.
- So, to determine the right number of clusters :
  1. You start with any value of k randomly and perform k-means analysis.
  2. Once the analysis is done, determine the total inertia of the clusters that were created.
  3. Increase the number *k* by 1 and carry out steps 1 and 2 until the change in total inertia is not significant.
- For example, suppose you plot the inertia value against the value of *k*, and it plots as shown in Fig. 4.1.7.

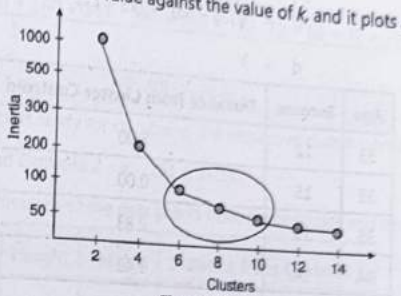


Fig. 4.1.7

- You see that good values of *k* should be 6, 8 or 10 because after 10, the change in the inertia value is very minimal. This type of plot is also called as elbow-plot or knee-curve (because of their shape) where "elbow" or "knee of a curve" serves as a cut-off point beyond which you start to get diminishing returns which are no longer worth the additional cost. In clustering, this means that you should choose an optimum number of clusters so that adding another cluster doesn't give much better modelling of the data.
- Note here that increasing the value of *k* to be too high results in cluster fragmentation. You would not have meaningful and substantial grouping of data points and it would be harder to draw out conclusion and take actions.
- By the way, what is the maximum number of clusters that you can have for a set of data points? Well, it would be the number of data points in the set where each data point is assigned to its own cluster – not very useful right? For example, if you have 1,000 data points in a set and you create 1,000 clusters (one for each data point), what would you do with that cluster? It is as good as treating each data point individually! Do not do that.

Let's see a few examples of cluster inertia calculation.

Practice Questions

Ex. 4.1.4 : A cluster has the following data points. Calculate its inertia. Assume 2nd data point to be the centroid for the cluster.

Age	Income in Thousand
33	12
33	15
35	13
34	14
32	16

Soln. :

To calculate inertia of a cluster, you need to find the distance of all data points from the centroid of the cluster and add them.

Let's assume age on X-axis and income on Y-axis. Now, let's calculate the distance of the data points from the cluster centroid and complete the data points table.

A sample distance calculation is as following for the first data point.

Distance from the Cluster Centroid (33, 15) for (33, 12) =

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

$$d = \sqrt{(33 - 33)^2 + (15 - 12)^2}$$

$$d = 3$$

Age	Income	Distance from Cluster Centroid
33	12	3.00
33	15	0.00
35	13	2.83
34	14	1.41
32	16	1.41
Total		**8.66**

Hence, the cluster inertia is 8.66.

Ex. 4.1.5 : Two clusters have the following data points. Calculate their intracluster distance. Also, calculate total inertia of the data point assignment.

Height	Weight	Assigned Cluster
185	72	2
170	56	1
168	60	1
179	68	2
182	72	2
188	77	2
180	71	2
180	70	2
183	84	2
180	88	2
180	67	2
177	76	2

Soln. :

To calculate inertia (or intracluster distance) of a cluster, you need to find the distance of all data points from the centroid of the cluster and add them. In the question, centroids for the respective clusters are not given. Hence, first you need to calculate the centroids for the respective clusters.

For Centroid 1 calculation, you have two data points that fall in cluster 1. They are (170, 56) and (168, 60).

Hence, Centroid 1 is the mean of the data points which is $\left(\frac{170 + 168}{2}, \frac{56 + 60}{2}\right) = (169, 58)$.

For Centroid 2 calculation, take the remaining 10 data points that fall in cluster 2.

Hence, Centroid 2 is the mean of these 10 data points which is

$$\left(\frac{185 + 179 + 182 + 188 + 180 + 180 + 183 + 180 + 180 + 177}{10}, \frac{72 + 68 + 72 + 77 + 71 + 70 + 84 + 88 + 67 + 76}{10}\right)$$

$$= (181.4, 74.5)$$

Now, you have both the centroids ready for calculating the respective cluster inertia.

Centroid 1 = (169, 58) and Centroid 2 = (181.4, 74.5).

Now, let's calculate the respective distance of the data points from the centroids and complete the data points table.

Height	Weight	Distance from Centroid 1
170	56	2.24
168	60	2.24
Total		**4.48**

Height	Weight	Distance from Centroid 2
185	72	4.38
179	68	6.93
182	72	2.57
188	77	7.06
180	71	3.77
180	70	4.71
183	84	9.63
180	88	13.57
180	67	7.63
177	76	4.65
Total		**64.91**

Hence,

- The inertia for cluster 1 = 4.48
- The inertia for cluster 2 = 64.91
- Total inertia is sum of inertia of all clusters = 4.48 + 64.91 = 69.39

4.1.6 Diagnostics (Performance Measures)

K-means algorithm is usually run through computer programs and the resultant outcome may not always be so obvious. You may require a personal touch and understanding of the data points and the resultant clusters to make the outcome more meaningful. This might require some diagnosis (judgment) of any issues that may occur. When the number of attributes, in a set of data points, is very small, you should plot the data to ensure that

1. Clusters are clearly distinguishable
2. No cluster has only a few data points
3. Centroids are not too close to each other

For example,

The following cluster might be clearly distinguishable and adequately populated with the data points.

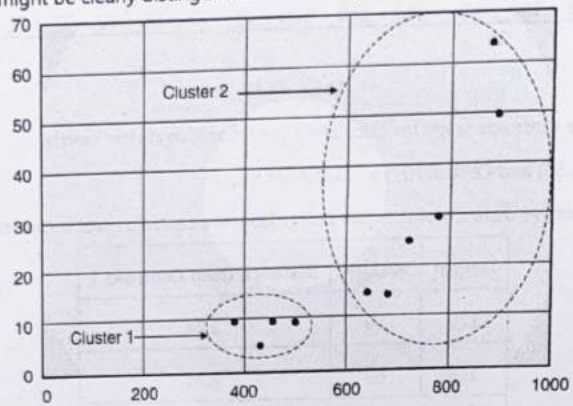


Fig. 4.1.8

But, the following data points on the plot may be too scattered.

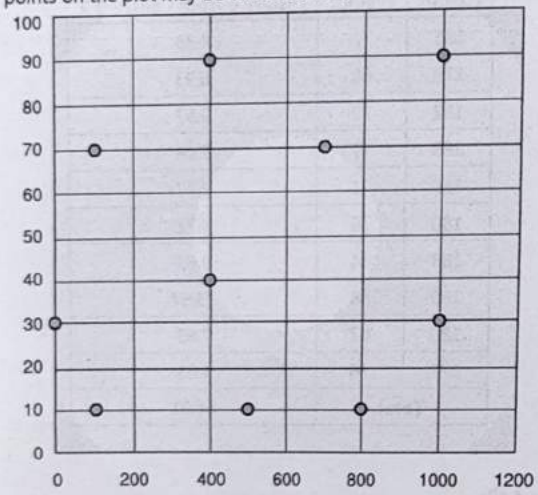


Fig. 4.1.9

You need to apply judgement to see if increasing the number of clusters or changing any other attribute would help to come up with better clusters.

4.1.7 Advantages of K-means Clustering

Some of the major advantages of K-means clustering are as following.

1. **Simplicity :** K-means clustering is relatively easy to understand and implement. It follows a straightforward iterative process, making it accessible to both beginners and experienced data scientists.
2. **Scalability :** K-means is computationally efficient and scales well with large datasets. Its time complexity is linear, which means it can handle substantial amounts of data without significant performance degradation.
3. **Interpretability :** K-means produces easily interpretable results. Each cluster represents a distinct group of data points, and the algorithm assigns each point to the cluster with the closest mean. This can help in understanding the underlying structure or patterns in the data.
4. **Flexibility :** K-means allows the user to specify the number of clusters k in advance. This flexibility enables the algorithm to be tailored to the specific needs of the analysis. However, it's important to note that determining the optimal value of k can also be a challenge.
5. **Speed :** K-means is a fast algorithm, particularly when compared to other clustering techniques. It converges quickly because it minimises the within-cluster sum of squares, leading to faster convergence than algorithms that optimise other criteria.
6. **Robustness :** Despite its simplicity, k-means is robust to noise and outliers. Outliers tend to be assigned to their closest cluster centroid, reducing their influence on the overall clustering result.
7. **Versatility :** K-means can be applied to various types of data, including numerical and continuous variables. It is not limited to any specific data distribution and can be used with both high-dimensional and low-dimensional datasets.

4.1.8 Disadvantages (Challenges) of K-means Clustering

A few things to consider for K-means clustering are as following.

1. **Object attributes :** The data points could have several attributes such as age, weight, height, income, etc. Once your clustering is complete, you need to ensure that the attributes would be available for new data points as and when added for future analysis. For example, if your existing clustering is based on customer ratings on a particular product, such rating may not be immediately available for a customer who has just completed the purchase. It might require a week or a month before the customer is comfortable rating the product judiciously.
2. **Units of measurement and scaling :** You need to be careful in choosing the units for your data point attributes. While it may not impact K-means clustering a lot, it could actually shift a few data points here and there. For example, suppose you have two data points age and income. Age is expressed in double digits. But, if you choose income to be in thousands and someone earning 10,000 is represented as 10, then both age and income are in double digits and equally contribute towards distance calculation. But, suppose the income was not in thousands, then the income data attribute would dominate the distance calculation as it has many more digits than age. This might shift the data points more with respect to income than age and hence skew (bias or distort) the cluster assignment. For example, look at the following distance calculation.

In the first calculation, Centroid 1 = (33, 15) and Centroid 2 = (34, 14)

In the second calculation, Centroid 1 = (33, 15000) and Centroid 2 = (34, 14000)

Age	Income in thousand	Distance from Centroid 1	Distance from Centroid 2	Assigned Cluster
33	12	3.00	2.24	2
33	15	0.00	1.41	1
35	13	2.83	1.41	2
34	14	1.41	0.00	2
32	16	1.41	2.83	1

Age	Income	Distance from Centroid 1	Distance from Centroid 2	Assigned Cluster
33	12000	3000.00	2000.00	2
33	15000	0.00	1000.00	1
35	13000	2000.00	1000.00	2
34	14000	1000.00	0.00	2
32	16000	1000.00	2000.00	1

While you might argue that the cluster assignment has not changed, but if you look closely the distance between the centroids is more noticeable due to income attribute. This could have impact on inertia and might lead you to believe that you need more clusters.

3. **Initial Centroid position :** K-means is sensitive to initial centroid position. Hence, you should run the K-means analysis several times to ensure that clusters created are optimum.
4. **Number of clusters :** As discussed earlier, you must be careful while deciding on the number of clusters required for optimum placement of data points. Cluster Inertia is a key parameter to consider for ensuring that you have the right number of clusters.

4.2 k-Nearest Neighbours (kNN) Classification Algorithm

- One of the popular variations of K-means clustering algorithm is k-Nearest Neighbours (kNN) classification algorithm.
- It works on the same principle as K-means clustering algorithm that a data point is likely to resemble its neighbours and would possibly have the same classification. kNN is a supervised learning method.
- The way it works is simple and straightforward.

1. Assume that you have already assigned labels to the existing data points in a given data set.
2. Then you are given a new data point to classify.
3. You calculate the distance of the new data point with respect to its *k* neighbours. The number *k* is chosen based on your data set and requirements but in general higher the better to reduce the noise and avoid incorrect labelling.
4. The new data point is classified based on the classification of the majority of the surrounding neighbour's classification.

Let's take an example to understand it.

Practice Questions

Ex. 4.2.1 : Following is a dataset for weight of teens and whether they like Pizza.

Weight (Kgs)	Like Pizza?
78	Yes
54	No
69	Yes
73	Yes
59	No
48	No
82	No
65	Yes

Using k-Nearest Neighbours (kNN) Classification Algorithm determine if a teen weighing 63 Kgs likely to like Pizza?

Use k = 3.

Soln. :

Calculate the distance of the new data point (63 Kgs) with respect to other data points in the data set.

A sample distance calculation is as following for the first data point.

$$d = \sqrt{(x_1 - x_2)^2}$$

$$d = \sqrt{(78 - 63)^2} = 15$$

$$d = 15$$

Weight (Kgs)	Distance for 63 Kgs	Like Pizza?
78	15	Yes
54	9	No
69	6	Yes
73	10	Yes
59	4	No
48	15	No
82	19	No
65	2	Yes

Sort the table based on the distance.

Weight (Kgs)	Distance for 63 Kgs	Like Pizza?
65	2	Yes
59	4	No
69	6	Yes
54	9	No
73	10	Yes
78	15	Yes
48	15	No
82	19	No

Now, given that k = 3.

So pick top 3 rows of the sorted table.

Weight (Kgs)	Distance for 63 Kgs	Like Pizza?
65	2	Yes
59	4	No
69	6	Yes

Now, you have got 2 Yes, and 1 No. The majority classification is "Yes". Hence, a teen weighing 63 Kgs is likely to like Pizza.

4.2.1 K-medoids

- The medoid of a cluster is defined as the object (or the data point) in the cluster whose average dissimilarity to all the other objects in the cluster is minimal, that is, it is the most centrally located point in the cluster. The k-medoids algorithm is another approach for clustering similar to the k-means algorithm.
- k-medoids clustering is a partitioning method commonly used in domains that require robustness to outlier data, arbitrary distance metrics, or ones for which the mean or median does not have a clear definition. It is similar to k-means, and the goal of both methods is to divide a set of measurements or observations into k subsets or clusters so that the subsets minimise the sum of distances between a measurement and a center of the measurement's cluster. In the k-means algorithm, the center of the subset is the mean of measurements in the subset, often called a centroid. In the k-medoids algorithm, the center of the subset is a member of the subset, called a medoid.
- The k-medoids algorithm returns medoids which are the actual data points in the data set. This allows you to use the algorithm in situations where the mean of the data does not exist within the data set. This is the main difference between k-medoids and k-means where the centroids returned by k-means may not be within the data set. Hence k-medoids is useful for clustering categorical data where a mean is impossible to define or interpret.

- Both the k-means and k-medoids algorithms are partitional (breaking the dataset up into groups) and attempt to minimise the distance between points labelled to be in a cluster and a point designated as the center of that cluster. In contrast to the k-means algorithm, k-medoids chooses actual input data points as centres (called medoids or exemplars), and thereby allows for greater interpretability of the cluster centres than in k-means, where the center of a cluster is not necessarily one of the input data points (it is the average between the points in the cluster).
- For example, the Fig. 4.2.1 illustrates the difference between the k-means and k-medoids approach.

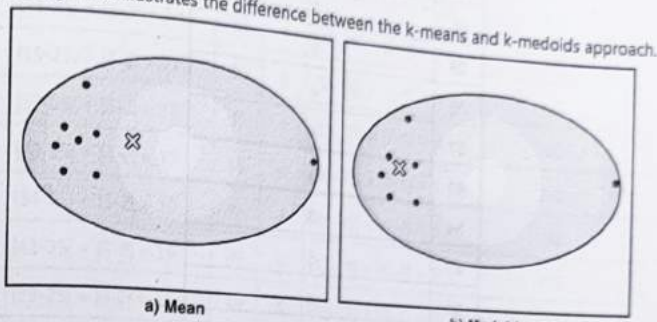


Fig. 4.2.1

- The group of points on the left form a cluster, while the rightmost point is an outlier. Mean is greatly influenced by the outlier and thus cannot represent the correct cluster center, while medoid is robust to the outlier and correctly represents the cluster center.
- k-medoids minimises a sum of pairwise dissimilarities instead of a sum of squared Euclidean distances as in k-means algorithm. It is, thus, more robust to noise and outliers than k-means. k-medoids is a classical partitioning technique of clustering that splits the data set of n objects into k clusters. Similar to k-means, you can decide the number k as per your requirements.
- There are several approaches to implementing k-medoids. However, partitioning around medoids (PAM) is most popularly used. The steps in PAM are as following.
  (1) Build-step : Each of k clusters is associated with a potential medoid. You can randomly choose the medoids based on the number of clusters that you desire. For example, if you desire two clusters, then you choose two data points from the input data set as medoids to start with.
  (2) Swap-step : Within each cluster, each point is tested as a potential medoid by checking if the sum of within-cluster distances gets smaller using that point as the medoid. If so, the point is defined as a new medoid. Every point is then assigned to the cluster with the closest medoid.
- The algorithm iterates the build and swap steps until the medoids do not change, or other termination criteria are met.
- Note here that k-medoids implementation is usually a greedy algorithm. A greedy algorithm follows the problem-solving heuristic of making the locally optimal choice at each stage. In many problems, a greedy strategy does not produce an optimal solution, but a greedy heuristic can yield locally optimal solutions that approximate a globally optimal solution in a reasonable amount of time. So, k-medoids may not give you perfectly clustered data points always but does a great job of quickly clustering the data points.
- Let's see a solved example to learn how k-medoids algorithm (PAM implementation) works.

Practice Question

Ex. 4.2.2 : In a survey, the following data was collected for tv viewership. Use k-medoids method to check if there is a group which watches more tv than the other?

Age	Number of TV watching hours
23	3
25	2
28	2
35	4
37	5
40	4
34	3
41	5
39	4
38	3

Soln. :

- Let us use K-medoids clustering for grouping the survey results. Let's choose the desired number of clusters, k = 2 and the number of iterations for centroid calculation and cluster assignment to be 2 as well. Let's put age on X-axis and Tv watching hours on Y-axis.
- The first plot of data points is as following.

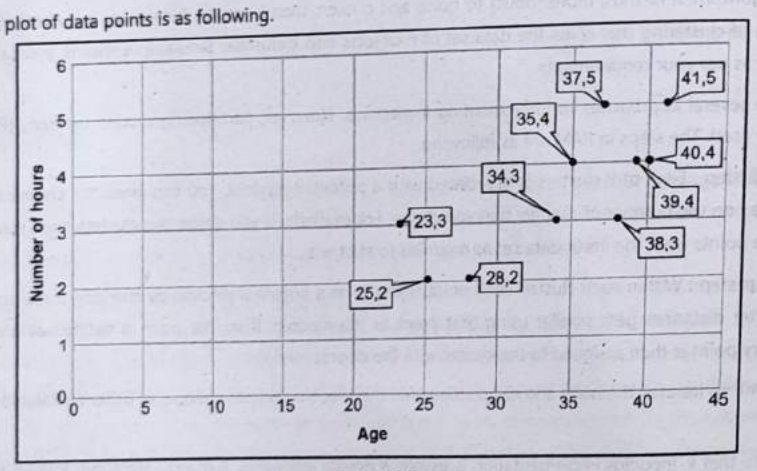


Fig. P. 4.2.2

- Let's randomly choose two medoids M1 (25, 2) and M2 (39, 4).
- For calculating dissimilarity between the data points and chosen medoids, you can use a simple distance calculating formula such as Dissimilarity = $|X_i - X_m| + |Y_i - Y_m|$. The || symbolises that you only take the absolute values without considering sign. X_i and Y_i are respective data points and X_m and Y_m are medoids.

- The following Table P.4.2.2 shows you the calculation for the chosen medoids.

Table P.4.2.2

X	Y	Xm1	Ym1	Dissimilarity from M1	Xm2	Ym2	Dissimilarity from M2	Point Nearest to?	Cluster assigned
23	3	25	2	|23-25| + |3-2| = 3	39	4	|23-39| + |3-4| = 17	M1	1
25	2	25	2	|25-25| + |2-2| = 0	39	4	|25-39| + |2-4| = 16	M1	1
28	2	25	2	|28-25| + |2-2| = 3	39	4	|28-39| + |2-4| = 13	M1	1
35	4	25	2	|35-25| + |4-2| = 12	39	4	|35-39| + |4-4| = 4	M2	2
37	5	25	2	|37-25| + |5-2| = 15	39	4	|37-39| + |5-4| = 3	M2	2
40	4	25	2	|40-25| + |4-2| = 17	39	4	|40-39| + |4-4| = 1	M2	2
34	3	25	2	|34-25| + |3-2| = 10	39	4	|34-39| + |3-4| = 6	M2	2
41	5	25	2	|41-25| + |5-2| = 19	39	4	|41-39| + |5-4| = 3	M2	2
39	4	25	2	|39-25| + |4-2| = 16	39	4	|39-39| + |4-4| = 0	M2	2
38	3	25	2	|38-25| + |3-2| = 14	39	4	|38-39| + |3-4| = 2	M2	2

- Total cost of M1 (25, 2) as medoid is the sum of dissimilarity for the points assigned to cluster for which M1 is the medoid which is 3 + 0 + 3 = 6
- Total cost of M2 (39, 4) as medoid is the sum of dissimilarity for the points assigned to cluster for which M2 is the medoid which is 4 + 3 + 1 + 6 + 3 + 0 + 2 = 19
- Total cost of choosing M1 (25, 2) and M2 (39, 4) as medoids = 6 + 19 = 25.
- Now, for next iteration, let's choose a different point as M2. Let's say it is (41, 5). Keep M1 same as before.
- Now, the calculation Table P.4.2.2(a) looks like the following.

Table P. 4.2.2(a)

X	Y	Xm1	Ym1	Dissimilarity from M1	Xm2	Ym2	Dissimilarity from M2	Point Nearest to?	Cluster assigned
23	3	25	2	|23-25| + |3-2| = 3	41	5	|23-41| + |3-5| = 20	M1	1
25	2	25	2	|25-25| + |2-2| = 0	41	5	|25-41| + |2-5| = 19	M1	1
28	2	25	2	|28-25| + |2-2| = 3	41	5	|28-41| + |2-5| = 16	M1	1
35	4	25	2	|35-25| + |4-2| = 12	41	5	|35-41| + |4-5| = 7	M2	2
37	5	25	2	|37-25| + |5-2| = 15	41	5	|37-41| + |5-5| = 4	M2	2
40	4	25	2	|40-25| + |4-2| = 17	41	5	|40-41| + |4-5| = 2	M2	2
34	3	25	2	|34-25| + |3-2| = 10	41	5	|34-41| + |3-5| = 9	M2	2
41	5	25	2	|41-25| + |5-2| = 19	41	5	|41-41| + |5-5| = 0	M2	2
39	4	25	2	|39-25| + |4-2| = 16	41	5	|39-41| + |4-5| = 3	M2	2
38	3	25	2	|38-25| + |3-2| = 14	41	5	|38-41| + |3-5| = 5	M2	2

- Total cost of M1 (25, 2) as medoid is the sum of dissimilarity for the points assigned to cluster for which M1 is the medoid which is 3 + 0 + 3 = 6
- Total cost of M2 (41, 5) as medoid is the sum of dissimilarity for the points assigned to cluster for which M2 is the medoid which is 7 + 4 + 2 + 9 + 0 + 3 + 5 = 30
- Total cost of choosing M1 (25, 2) and M2 (41, 5) as medoids = 6 + 30 = 36.
- As you see the assignment of (41, 5) as M2 actually increased the overall cost (from 25 to 36).
- As k-medoids is a greedy algorithm, you do no more iterations and stay with previously chosen M1 (25, 2) and M2 (39, 4). Yes, there may be a more optimal solution but that is not what k-medoids aims to find.
- Following plot shows the medoids and the clusters based on those medoids.

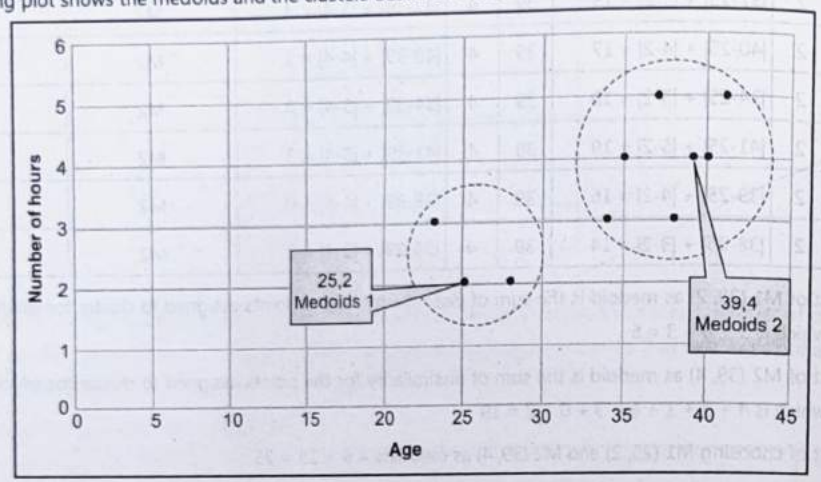


Fig. P. 4.2.2(a)

- From the clusters formed, it can be concluded that people in cluster 2 seem to be watching more tv than people in cluster 1.

4.2.2 Comparison between k-means and k-medoids Algorithms

- The Table 4.2.1 provides a quick comparison between k-means and k-medoids algorithms.

Table 4.2.1

Comparison Attribute	k-means	k-medoids
Center point	Average between the cluster points	Actual input data points
Works on	Minimising Euclidean distance	Minimising dissimilarity
Cluster centres	Less prominent	More prominent
Robustness to outliers and noise	Low	High
Need mean or median	Yes	No

4.2.3 k-prototype

So far you have learnt about examples of clustering where you only had numerical data in your dataset. But what if you have categorical data such as gender, car type, occupation, etc. in your dataset as following?

Customer	Age	Income	Gender	Occupation
Customer 1	35	50000	Male	Engineer
Customer 2	40	60000	Female	Teacher
Customer 3	25	30000	Male	Artist
Customer 4	55	75000	Female	Doctor
Customer 5	30	40000	Male	Engineer
Customer 6	45	80000	Female	Lawyer
Customer 7	28	35000	Female	Artist
Customer 8	50	70000	Male	Doctor

4.2.3(A) How would you go about Clustering them?

- This is exactly where k-prototype algorithm helps. The k-prototype algorithm is an extension of the k-means algorithm that combines categorical and numerical data for clustering. It is specifically designed to handle datasets with a mixture of both categorical and continuous attributes, making it suitable for clustering heterogeneous data.
- The k-prototype algorithm aims to partition the data into k clusters, where each cluster is represented by a centroid that consists of both the numerical mean and the mode of the categorical variables. The algorithm follows an iterative process to optimise the clustering results.

4.2.3(B) How k-prototype Clustering Works?

Following are the high-level steps describing how k-prototype clustering works.

1. **Initialisation**
   (a) Randomly select two initial centroids or use a predefined strategy.
2. **Iterative Update**
   (a) Compute the dissimilarity between each data point and each centroid based on the appropriate dissimilarity measures for numerical and categorical attributes.
   (b) Update the centroids by calculating the mean of the numerical attributes within each cluster and the mode of the categorical attribute.
   (c) Reassign each data point to the nearest centroids based on the updated dissimilarity.
3. **Convergence**
   (a) Repeat the iterative update steps until convergence is reached, i.e., when the centroids no longer change significantly or a maximum number of iterations is reached.

Let's see a simple example to understand this better.

Practice Question

Ex. 4.2.3 : Using k-prototype algorithm, cluster the following dataset. Assume k = 2.

Person	Height	Weight	Size
Person 1	165	60	M
Person 2	170	65	M
Person 3	155	50	L
Person 4	180	75	L
Person 5	160	55	S
Person 6	175	70	L

Soln. :

Let's say you choose Person 1 (165, 60) and Person 4 (180, 75) as the initial centroids.

Iteration 1

So, for the first iteration, the distance table and cluster assignments look like the following.

Person	Height	Weight	Size	Distance from Centroid 1	Distance from Centroid 2	Assigned Cluster
Person 1	165	60	M	0.00	21.21	1
Person 2	170	65	M	7.07	14.14	1
Person 3	155	50	L	14.14	35.36	1
Person 4	180	75	L	21.21	0.00	2
Person 5	160	55	S	7.07	28.28	1
Person 6	175	70	L	14.14	7.07	2

- The mode for Size for Cluster 1 is M, as it is the most frequent size in the resulting cluster.
- The mode for Size for Cluster 2 is L, as it is the most frequent size in the resulting cluster.
- Let's re-calculate the centroids for the next iteration.
- For Centroid 1 calculation, you have four data points that fall in cluster 1.
- Hence, Centroid 1 is the mean of the data points which is

$$\left(\frac{165 + 170 + 155 + 160}{4}, \frac{60 + 65 + 55 + 55}{4}\right) = (162.5, 57.5)$$

- For Centroid 2 calculation, you have two data points that fall in cluster 2.
- Hence, Centroid 2 is the mean of the data points which is

$$\left(\frac{180 + 175}{2}, \frac{75 + 70}{2}\right) = (177.5, 72.5)$$

Iteration 2

Centroid 1 = (162.5, 57.5) and Centroid 2 = (177.5, 72.5).

- Now, let's calculate the distance of the data points from the centroids and complete the data points table. The data point is assigned to the cluster based on the closest centroid.

Person	Height	Weight	Size	Distance from Centroid 1	Distance from Centroid 2	Assigned Cluster
Person 1	165	60	M	3.54	17.68	1
Person 2	170	65	M	10.61	10.61	1
Person 3	155	50	L	10.61	31.82	1
Person 4	180	75	L	24.75	3.54	2
Person 5	160	55	S	3.54	24.75	1
Person 6	175	70	L	17.68	3.54	2

- You stop here you see that the clusters assigned in the iteration 1 for the data points did not change.

Hence,

- Cluster 1 = Person 1, Person 2, Person 3, Person 5. Mode of the cluster for Size is M as M is the most frequent size in the resulting Cluster 1.
- Cluster 2 = Person 4 and Person 6. Mode of the cluster for Size is L as L is the most frequent size in the resulting Cluster 2.

4.3 Hierarchical Clustering

You must be very familiar with how you can divide data into groups and sub-groups within those groups, isn't it? For example, in the Fig. 4.3.1, for some data, the "other" category is further split into sub-groups. Putting data together in this form helps you to quickly understand the relationship within the data and also appropriate group and categorise them.

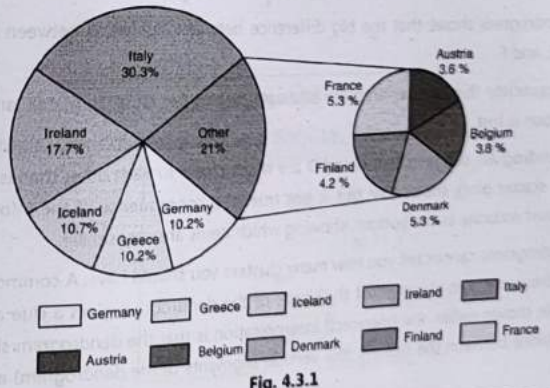


Fig. 4.3.1

Hierarchical clustering is a similar concept.

✍ ***Definition :*** *Hierarchical clustering, also called as hierarchical cluster analysis or HCA, is a method of cluster analysis in which the data points are arranged in a hierarchy of clusters.*

So, you not only cluster the datapoints, but you also have a hierarchy applied on the clusters that can be visually looked at to understand their relationships. The results of hierarchical clustering are usually presented in a dendrogram.

4.3.1 Dendrogram

✍ ***Definition :*** *A dendrogram is a diagram representing a tree or hierarchy.*

- It is a branching diagram representing a hierarchy of categories based on degree of similarity or number of shared characteristics. For example, the Fig. 4.3.2 illustrates a simple dendrogram for six observations.

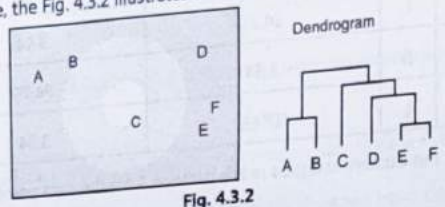


Fig. 4.3.2

- The key to interpreting a dendrogram is to focus on the height at which any two objects are joined together. In the given example, you can see that E and F are most similar, as the height of the link that joins them together is the smallest. The next two most similar objects are A and B.
- The height of the dendrogram indicates the order in which the clusters were joined. A more informative dendrogram can be created where the heights reflect the distance between the clusters as shown in the Fig. 4.3.3.

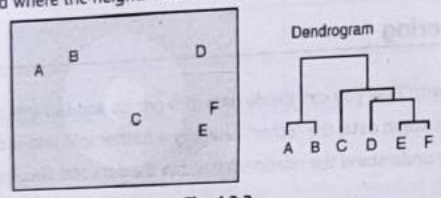


Fig. 4.3.3

- In this case, the dendrogram shows that the big difference between clusters is between the cluster of A and B versus that of C, D, E, and F.
- It is important to appreciate that the dendrogram is a summary of the distance matrix, and, as occurs with most summaries, information is lost.
- For example, the dendrogram suggests that C and D are much closer to each other than is C to B, but the original data (shown in the scatter plot), shows that this is not true. The consequence of the information loss is that the dendrograms are most accurate at the bottom, showing which items are very similar.
- Also, note that dendrograms cannot tell you how many clusters you should have. A common mistake people make when reading dendrograms is to assume that the shape of the dendrogram gives a clue as to how many clusters exist. In the example shown earlier, the (incorrect) interpretation is that the dendrogram shows that there are two clusters, as the distance between the clusters (the vertical segments of the dendrogram) are highest between two and three clusters.

4.3.2 Hierarchical Clustering Strategies (Algorithms)

Strategies for hierarchical clustering generally fall into two types.

1. **Agglomerative :** This is a "bottom-up" approach in which each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
2. **Divisive :** This is a "top-down" approach in which all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

The Fig. 4.3.4 illustrates the difference between agglomerative and divisive strategies.

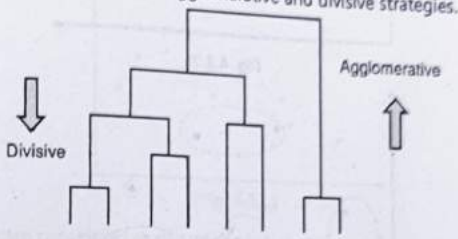


Fig. 4.3.4

In general, the merges and splits are determined in a greedy manner. Let's learn about these strategies in a little more detail.

4.3.2(A) Agglomerative Hierarchical Clustering

Agglomerative Hierarchical Clustering, commonly referred to as AGNES (AGglomerative NESting), works in a bottom-up manner. That is, each observation is initially considered as a single-element cluster (leaf). At each step of the algorithm, the two clusters that are the most similar are combined into a new bigger cluster (nodes). This procedure is iterated until all points are a member of just one single big cluster (root). The result is a tree which can be displayed using a dendrogram. Consider that the data points are plotted as shown in Fig. 4.3.5.

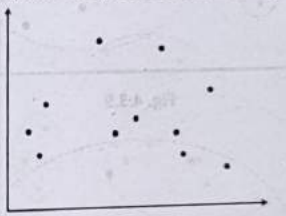


Fig. 4.3.5

In first iteration, individual point clusters are made as Fig. 4.3.6.

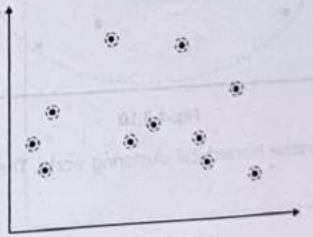


Fig. 4.3.6

In the next few iterations, clusters closer to each other start merging.

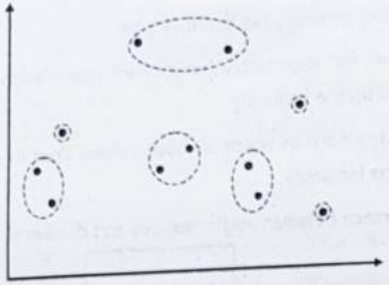


Fig. 4.3.7

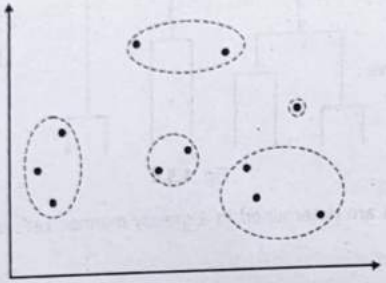


Fig. 4.3.8

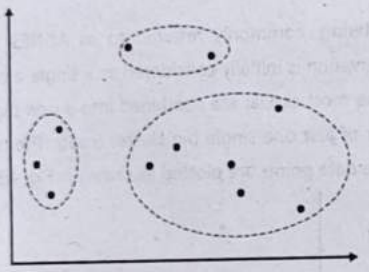


Fig. 4.3.9

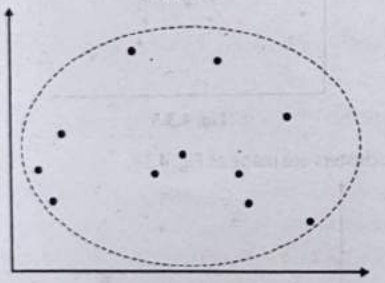


Fig. 4.3.10

In a nut-shell, this is how agglomerative hierarchical clustering works. The result is plotted as dendogram as you learnt earlier.

4.3.2(B) Divisive Hierarchical Clustering

Divisive Hierarchical Clustering, commonly referred to as DIANA (DIvise ANAlysis), works in a top-down manner. DIANA is like the reverse of AGNES. It begins with the root, in which all observations are included in a single cluster. At each step of the algorithm, the current cluster is split into two clusters that are considered most heterogeneous. The process is iterated until all observations are in their own cluster. The result is a tree which can be displayed using a dendrogram. Consider that the data points are plotted as shown in Fig. 4.3.11.

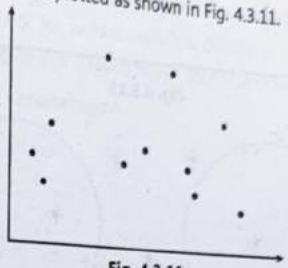


Fig. 4.3.11

In first iteration, all points are considered as part of one big cluster.

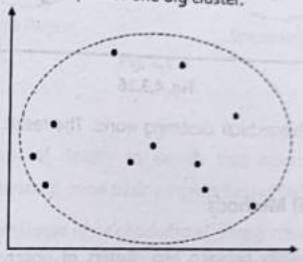


Fig. 4.3.12

In the next few iterations, clusters are further broken up into smaller clusters until all points are in their own cluster.

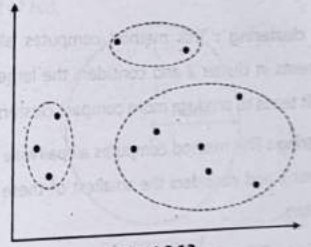


Fig. 4.3.13

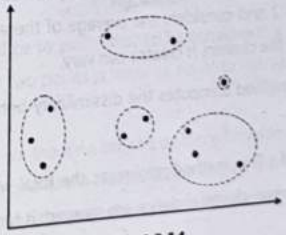


Fig. 4.3.14

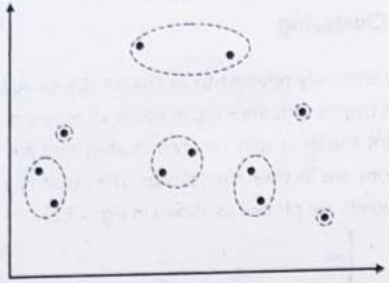


Fig. 4.3.15

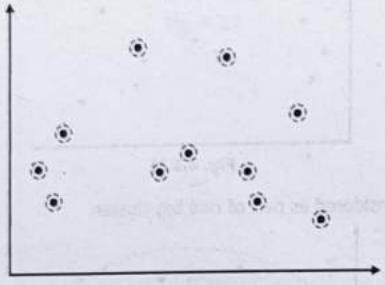


Fig. 4.3.16

In a nut-shell, this is how divisive hierarchical clustering works. The result is plotted as dendogram as you learnt earlier.

4.3.3 Agglomeration (Linkage) Methods

How do you measure the dissimilarity between two clusters of observations? A number of different cluster agglomeration methods (i.e., linkage methods) have been developed to answer this question. The most common methods are as following.

1. **Maximum or complete linkage clustering :** This method computes all pairwise dissimilarities between the elements in cluster 1 and the elements in cluster 2 and considers the largest value of these dissimilarities as the distance between the two clusters. It tends to produce more compact clusters.
2. **Minimum or single linkage clustering :** This method computes all pairwise dissimilarities between the elements in cluster 1 and the elements in cluster 2 and considers the smallest of these dissimilarities as a linkage criterion. It tends to produce long, "loose" clusters.
3. **Mean or average linkage clustering :** This method computes all pairwise dissimilarities between the elements in cluster 1 and the elements in cluster 2 and considers the average of these dissimilarities as the distance between the two clusters. The compactness of the clusters it creates can vary.
4. **Centroid linkage clustering :** This method computes the dissimilarity between the centroid for cluster 1 and the centroid for cluster 2.
5. **Ward's minimum variance method :** This method minimises the total within cluster variance. At each step, the pair of clusters with the smallest between-cluster distance are merged. It tends to produce more compact clusters.

4.4 Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

Definition : *Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is an unsupervised learning method often used for clustering.*

- It is based on the concept of density. As you understand, density provides you with a simple measure of how closely things are packed over a given area. For example, you often say that a country like China is more densely populated than United States. Similarly, for various materials, density is a measure of amount of mass contained per unit volume. For example, density of iron is much higher than that of water.
- The Fig. 4.4.1 illustrates a dense and a sparse region.

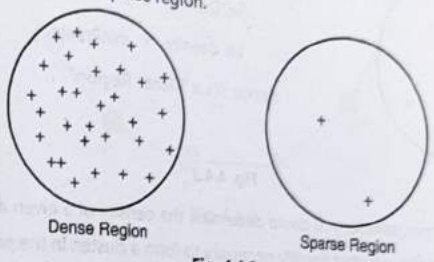


Fig. 4.4.1

4.4.1 Basic Concepts

- DBSCAN similarly uses the concept of density to identify data points that could potentially belong to a neighbourhood (cluster). Let us understand some basic concepts behind DBSCAN.

1. **Epsilon (eps) denoted by ε :** It is a measure of neighbourhood. Simply speaking, it helps you to identify how large the neighbourhood is. For example, if you draw a circle with a centre point C, then - just denotes the radius of the circle. So, for the point C, circle is the neighbourhood, and - is the radius of the circle.

The Fig. 4.4.2 illustrates the concept of eps.

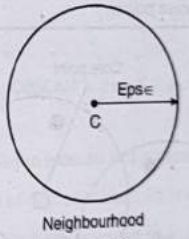


Fig. 4.4.2

Eps specifies how close points should be to each other to be considered a part of a cluster (same neighbourhood). It means that if the distance between two points is lower or equal to this value (eps), these points are considered to be neighbours.

Eps is crucial to choose appropriately for the data set and distance function and usually cannot be left at the default value. It controls the local neighbourhood of the points. When chosen too small, most data will not be clustered at all. If Eps is chosen to be too large, then it causes close clusters to be merged into one cluster, and eventually the entire data set to be returned as a single cluster.

2. **min_sample (also called as minPts) :** min_sample (or minPts) is the measure of minimum number of points you want to consider for identifying a neighbourhood. For example, if you require at least three points and you have 5 points around an area, then that area could be called as a neighbourhood. Another way to look at it is that min_sample tells you how many points you minimally require to consider an area dense. min_sample primarily controls how tolerant the algorithm is towards noise (on noisy and large data sets it may be desirable to increase this parameter).

The Fig. 4.4.3 illustrates the concept of min_sample.

Let the minPoints = 3, and eps=2.5

So, Density = 5

i.e. density > minPoints

Hence, it's a "Dense Region"

Fig. 4.4.3

3. **Density :** Based on ε and min_sample, you could determine the density of a given area or neighbourhood. Higher min_sample or lower eps indicate higher density necessary to form a cluster. In the previous example, density = 5.

4. **Core point :** A point is considered to be a core point if at least a specified number (min_sample) of neighbouring points fall within the specified radius ε.

5. **Border point :** This is a point that has at least one core point at a particular distance but is not surrounded by enough min_sample points. In other words, if the point under observation does not have sufficient neighbours (data points) of its own, but it has at least one core point in its neighbourhood, then that point represents the border point of the cluster. Border points belong to the same cluster of their nearest core point.

6. **Noise points :** All other points that are neither core nor border points are considered as noise points.

The Fig. 4.4.4 illustrates core, border, and noise points.

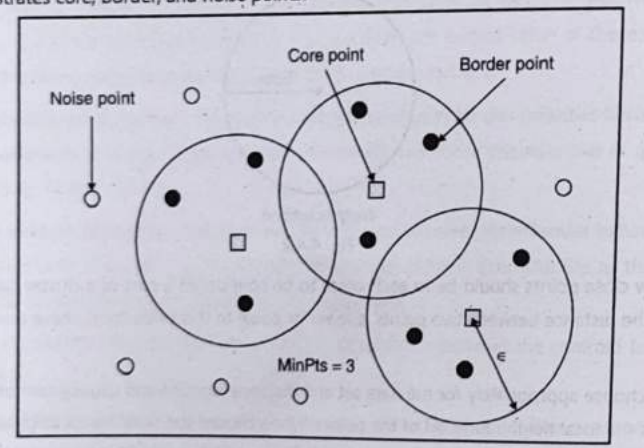


Fig. 4.4.4

7. **Density edge :** Suppose you have two points p and q. If p and q both are core points and the distance between them is less than ε, then you can connect p and q and call it a "density edge".

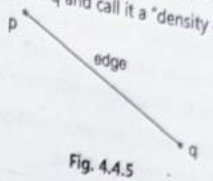


Fig. 4.4.5

8. **Density connected points :** Two points p and q are said to be density connected points if both p and q are core points and there exists a path formed by density edges connecting point p to point q.

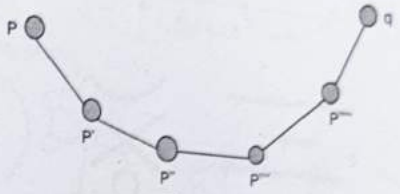


Fig. 4.4.6

4.4.2 DBSCAN Algorithm

- Now that you understand the basic terms and concepts, let's understand the steps in DBSCAN algorithm.
  1. For each point in the dataset, find out the distance to every other point in the dataset.
  2. All points that fall within the neighbourhood (within a radius of ε) are considered as neighbours.
  3. If the minimum point threshold (min_sample) is reached, then the points are grouped together as a cluster or else marked as noise.
  4. This process is repeated until all data points are categorised into clusters or as noise.

4.4.3 Advantages of DBSCAN

- The advantages of DBSCAN are as following.
  1. One of the main advantages of using DBSCAN is that it does not assume that the clusters have a spherical shape as in k-means algorithm.
  2. It does not necessarily assign each point to a cluster but is capable of removing noise points.
  3. You do not need to know the number of desired clusters (k) in advance.
  4. It is highly efficient as once a point has been assigned to a cluster in DBSCAN, it does not change cluster membership.

4.4.4 Challenges / Disadvantages of DBSCAN

- DBSCAN has following challenges.
  1. If your dataset has multiple densities or varying densities, DBSCAN tends to fail.
  2. It is extremely sensitive to the value of eps and min_sample.
  3. It may not work well with very high dimensionality data.

4.5 Spectral Clustering

- As opposed to "traditional clustering algorithms" such as k-means which always result in clusters with convex geometric shape, spectral clustering can solve problems in much more complex scenarios, such as intertwined spirals, or other arbitrary nonlinear shapes, because it does not make assumptions on the shapes of clusters.
- Many data sets can be notoriously difficult to cluster with traditional methods. The Fig. 4.5.1 illustrates a few datasets which are difficult for traditional clustering algorithms.

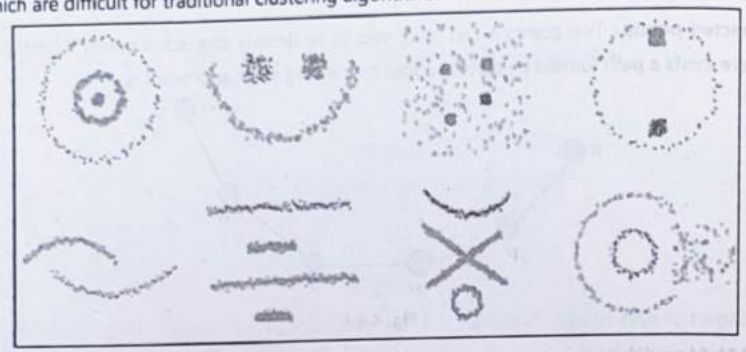


Fig. 4.5.1

- These data sets have been constructed in order to generate clusters of different shapes. On such datasets, algorithms which implicitly assume specific shapes of clusters cannot achieve good results. For example, the Euclidian distance metrics assume a convex shape to the underlying clusters. Obviously such assumptions can impact the quality of the clustering in arbitrary data sets. The spectral clustering method is able to handle such data sets effectively.
- The spectral clustering family can be viewed as a three-step algorithm.
  (1) The first step is to construct a similarity graph for all the data points.
  (2) The data points are embedded in a space, in which the clusters are more "obvious," with the use of the eigenvectors of the graph Laplacian.
  (3) Finally, a classical clustering algorithm such as k-means is applied to partition the embedding.
- The low-dimensional representation obtained in the second step is also referred to as "spectral embedding" and has applications beyond the clustering context, such as dimensionality reduction. The word spectral is used to denote the fact that the clustering results are obtained by analysing the spectrum of the graph Laplacian.
- The Fig. 4.5.2 shows the application of the spectral clustering on an example data set. The figure in the middle is a K-nearest neighbour (KNN) graph built in term of the Euclidean distance. The clustering result is illustrated in the adjacent figure on the right.

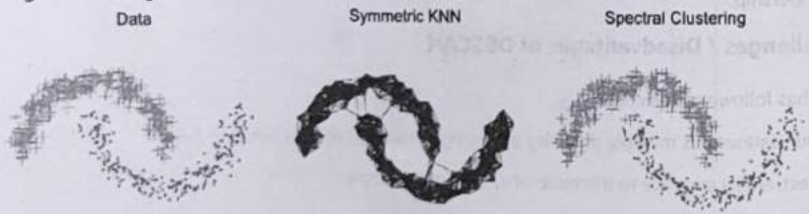


Fig. 4.5.2

- So, to understand the steps in a more simplified way, you start with a set of data points, for example, as shown in the following Fig. 4.5.3.

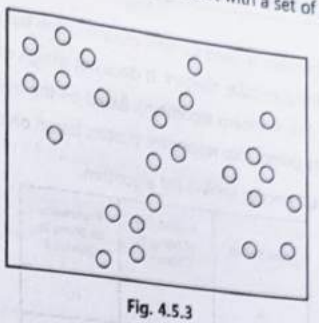


Fig. 4.5.3

- You then construct a similarity graph.

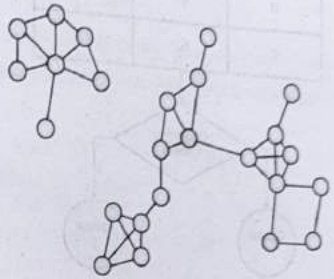


Fig. 4.5.4

- Finally, you partition the similarity graph into respective clusters.

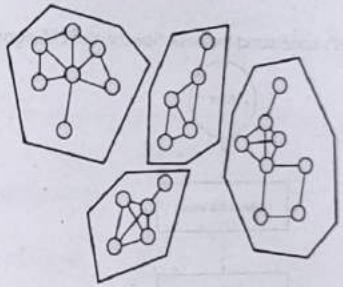


Fig. 4.5.5

4.6 Distribution-based Clustering Algorithm (Probabilistic Clustering Algorithms)

- Distribution-based clustering algorithms, also known as probabilistic clustering algorithms, are a family of clustering methods that aim to partition data points into distinct groups based on the underlying probability distribution of the data. These algorithms assume that the data points are generated from a mixture of probability distributions and try to estimate the parameters of these distributions to determine the clustering structure.
- One of the popular distribution-based clustering algorithms is Expectation-Maximisation algorithm that is based on Gaussian Mixture Model (GMM). Let's learn about it in brief.

4.7 Expectation-Maximisation (EM) Algorithm (Gaussian Mixture Model)

Expectation-Maximisation (EM) algorithm is used for soft-clustering the data points, i.e. assigning a probability or likelihood of the data points to be in the particular clusters. It does not assign data points to one of the clusters of its own i.e. it does not do hard clustering (like K-means algorithm). Based on the assigned probabilities to the data points, you can then choose to place those data points into respective clusters based on certain clustering criteria.

The Fig. 4.7.1 outlines the high-level concept behind EM algorithm.

Data Points	Probability of being in Cluster 1	Probability of being in Cluster 2
A	0.3	0.7
B	0.1	0.9
C	0.7	0.3

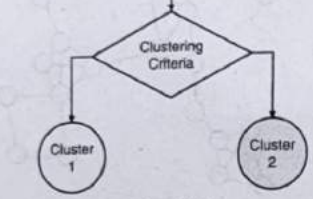


Fig. 4.7.1

4.7.1 How EM Algorithm Works ?

Before you dive into the details, let's understand the basic flowchart of EM algorithm.

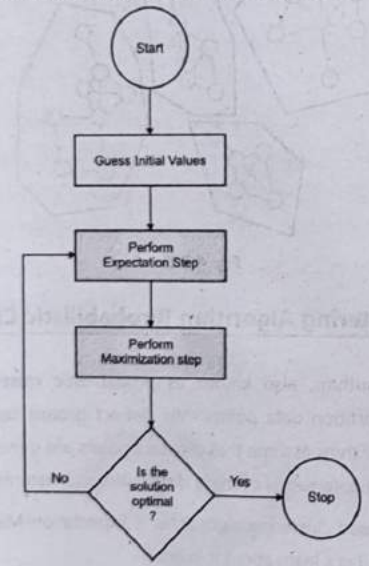


Fig. 4.7.2

- The EM algorithm involves three steps.
  1. Guess Initialise values (also called as initial hypothesis)
  2. Expectation Step (E-Step)
  3. Maximisation Step (M-Step)

  Until an optimal solution is found, E-step and M-step are iterated.
- EM algorithm assumes that the classes (or clusters) are laid out as Gaussian distributions (the same kind of bell-curve or distribution as you learnt in "Difference of Means" section). A distribution is characterised by two parameters :
  1. Mean of the distribution (denoted by μ). A smaller value of moves the distribution to the left and a larger value of moves the distribution to the right.
  2. Standard deviation of the distribution (denoted by σ). It determines the width of the distribution. A larger value of σ makes the distribution shorter and wider and smaller value of σ makes the distribution taller and narrower.

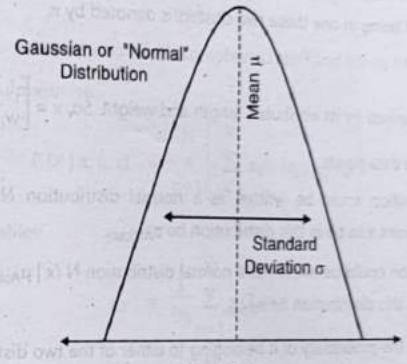


Fig. 4.7.3

The equation for the normal curve is given as

$$p(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- This equation helps you to determine the probability of the data point x to belong to the distribution given the values of μ and σ.
- For example, assume that for a distribution $\mu = 30$ and $\sigma = 2$. So, the probability of the data point $x = 32$ for belonging to this distribution could be calculated as

$$p(x = 32 \mid (\mu = 30, \sigma = 2)) = \frac{1}{\sqrt{2\pi 2^2}} e^{-\frac{(32-30)^2}{2 \cdot 2^2}}$$

$$= 0.12$$

- Now, with this background, let's understand the EM algorithm through an example.

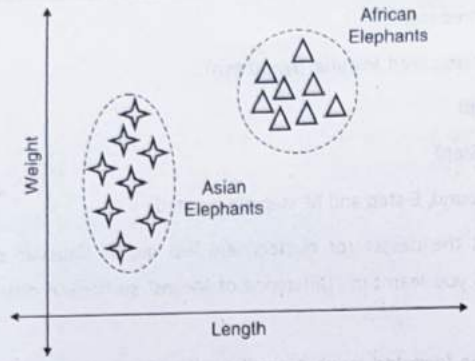


Fig. 4.7.4

- Assume that you have two distributions based on length and weight of elephants that cluster them into either African elephants or Asian elephants. Each of these normal distributions is characterised by mean μ and a co-variance matrix denoted by ε (note here that since this distribution is two-dimensional having length and weight, you take a co-variance matrix instead of standard deviation as in the case of a normal distribution in one dimension). The probability of being in one these two clusters is denoted by π.

  So, to summarise,

  - A data point x is characterised by its attributes, length and weight. So, $x = \begin{bmatrix} l_i \\ w_i \end{bmatrix}$ where l_i and w_i are respective length and weight of the data points.
  - African Elephant Distribution could be written as a normal distribution $N(x \mid \mu_{African}, \varepsilon_{African})$ and the probability of the data point x to be in this distribution be $\pi_{African}$.
  - Asian Elephant Distribution could be written as a normal distribution $N(x \mid \mu_{Asian}, \varepsilon_{Asian})$ and the probability of the data point x to be in this distribution be π_{Asian}.

- So, overall, for any random x, the probability of it belonging to either of the two distributions would be

$$p(x) = \pi_{African} \times N(x \mid \mu_{African}, \varepsilon_{African}) + \pi_{Asian} \times N(x \mid \mu_{Asian}, \varepsilon_{Asian})$$

- This equation can be written generically as

$$P(X \mid \pi, \mu, \varepsilon) = \prod_{n=1}^{N} \left[\sum_{k=1}^{k} \pi_k N(x_k \mid \mu_k, \varepsilon_k) \right]$$

  π = Denotes multiplication

  k = number of clusters

  N = total number of data points in the distribution

- So, the equation multiplies the respective probabilities of each data point to be in the respective cluster to give the probability of the entire distribution. So, now your goal is to maximize μ, Σ, π for the entire distribution having N points.
- To solve this maximisation problem, you use the EM algorithm. EM algorithm inserts some hidden (auxiliary, latent, or extra) variables in this equation to make it easier to solve this maximisation problem.

Assume that

$$Z_{nk} = \begin{cases} 1, & \text{if } x_n \text{ is in class } k \\ 0, & \text{otherwise} \end{cases}$$

- Probability of Z_{nk} to be either 1 or 0 is denoted by $\gamma(Z_{nk})$ and it can be written as $\gamma(Z_{nk}) = p(Z_{nk} = 1 \mid x_n)$. which means the probability of $Z_{nk} = 1$ given the data point x_n.

  This could be re-written as;

$$\gamma(Z_{nk}) = \sum_{j=1}^{k} \frac{P(Z_k = 1) * P(x_n \mid Z_{nk} = 1)}{P(Z_{nj} = 1) * P(x_n \mid Z_{nj} = 1)}$$

$$\gamma(Z_{nk}) = \sum_{j=1}^{k} \frac{\pi_k * N(x_n \mid \mu_k, \varepsilon_k)}{\pi_j * N(x_n \mid \mu_j, \varepsilon_j)}$$

Also,

$$N_k = \sum_{n=1}^{N} \gamma(Z_{nk})$$

- So, now you may have a question, why do all this complicated stuff and define these extra variables?

  The reason is that when you maximise.

$$P(X \mid \pi, \mu, \varepsilon) = \prod_{n=1}^{N} \left[\sum_{k=1}^{k} \pi_k N(x_k \mid \mu_k, \varepsilon_k) \right]$$

You would need those variables.

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(Z_{nk}) \times x_n$$

$$\varepsilon_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(Z_{nk}) (x_n - \mu_k) (x_n - \mu_k)^T$$

$$\pi_k = \frac{N_k}{N}$$

So,

- $\mu_k, \varepsilon_k, \pi_k$ depend upon $\gamma(Z_{nk})$
- $\gamma(Z_{nk})$ depend upon $\mu_k, \varepsilon_k, \pi_k$

EM algorithm has the following steps to iterate over various values of μ_k ε and find an optimal solution.

1. Initialise $\mu_k, \varepsilon_k, \pi_k$
2. E-step : Compute $\gamma(Z_{nk})$ based on $\mu_k, \varepsilon_k, \pi_k$.
3. M-step : Recompute $\mu_k, \varepsilon_k, \pi_k$ based on $\gamma(Z_{nk})$.
4. Go to step 2 if not optimal.

4.8 Measuring Clustering Quality

- Suppose you have assessed the clustering tendency of a given data set. You may have also tried to pre-determine the number of clusters in the set. You can now apply one or multiple clustering methods to obtain clustering of the data set. "How good is the clustering generated by a method, and how can you compare the clustering generated by different methods?"
- You have a few methods to choose from for measuring the quality of a clustering. In general, these methods can be categorised into two groups according to whether ground truth is available. Here, ground truth is the ideal clustering that is often built using human experts.
- If ground truth is available, it can be used by extrinsic methods, which compare the clustering against the group truth and measure.
- If the ground truth is unavailable, you can use intrinsic methods, which evaluate the goodness of a clustering by considering how well the clusters are separated.
- Ground truth can be considered as supervision in the form of "cluster labels". Hence, extrinsic methods are also known as supervised methods, while intrinsic methods are unsupervised methods.
- Let's have a look at simple methods from each category.

4.8.1 Extrinsic Methods

- When the ground truth is available, you can compare it with a clustering to assess the clustering. Thus, the core task in extrinsic methods is to assign a score, $Q(C, C_g)$, to a clustering C given the ground truth C_g.
- Whether an extrinsic method is effective largely depends on the measure, Q, it uses. In general, a measure Q on clustering quality is effective if it satisfies the following four essential criteria.

1. Cluster homogeneity

This requires that the purer the clusters in a clustering are, the better the clustering. Suppose that ground truth says that the objects in a data set, D, can belong to categories L1, ..., Ln. Consider clustering, C_1, wherein a cluster $C \in C_1$ contains objects from two categories Li, Lj ($1 \le I < j \le n$). Also consider clustering C_2, which is identical to C_1 except that C_2 is split into two clusters containing the objects in Li and Lj, respectively. A clustering quality measure, Q, respecting cluster homogeneity should give a higher score to C_2 than C_1, that is, $Q(C_2, C_g) > Q(C_2, C_g)$ as C_2 is better clustered (having homogenous items instead of mix) than C_1.

2. Cluster completeness

This is the complement of cluster homogeneity. Cluster completeness requires that for a clustering, if any two objects belong to the same category according to ground truth, then they should be assigned to the same cluster. Cluster completeness requires that a clustering should assign objects belonging to the same category (according to ground truth) to the same cluster.

Consider clustering C_1, which contains clusters A_1 and A_2, of which the members belong to the same category according to ground truth. Let clustering C_2 be identical to C_1 except that A_1 and A_2 are merged into one cluster in C_2. Then, a clustering quality measure, Q, respecting cluster completeness should give a higher score to C_2, that is, $Q(C_2, C_g) > Q(C_1, C_g)$ as C_2 has one complete cluster (instead of two) containing both A_1 and A_2.

3. Rag bag

In many practical scenarios, there is often a "rag bag" category containing objects that cannot be merged with other objects. Such a category is often called "miscellaneous," "other," and so on. The rag bag criterion states that putting a heterogeneous object into a pure cluster should be penalised more than putting it into a rag bag.

Consider a clustering C_1 and a cluster $C \in C_1$ such that all objects in C except for one, denoted by o, belong to the same category according to ground truth. Consider a clustering C_2 identical to C_1 except that o is assigned to a cluster $C' \neq C$ in C_2 such that C' contains objects from various categories according to ground truth, and thus is noisy. In other words, C' in C_2 is a rag bag. Then, a clustering quality measure, Q, respecting the rag bag criterion should give a higher score to C_2, that is, $Q(C_2, C_g) > Q(C_1, C_g)$ as it correctly puts the object o in a separate noise category instead of mixing it with a "good" cluster.

4. Small cluster preservation

If a small category is split into small pieces in a clustering, those small pieces may likely become noise and thus the small category cannot be discovered from the clustering. The small cluster preservation criterion states that splitting a small category into pieces is more harmful than splitting a large category into pieces.

Consider an extreme case. Let D be a data set of $n + 2$ objects such that, according to ground truth, n objects, denoted by $o_1, ..., o_n$, belong to one category and the other two objects, denoted by o_{n+1}, o_{n+2} belong to another category.

Suppose clustering C_1 has three clusters,

$$A_1 = \{o_1, ..., o_n\}$$
$$A_2 = \{o_{n+1}\}$$
$$A_3 = \{o_{n+2}\}$$

Let clustering C_2 have three clusters, too, namely

$$A_1 = \{o_1, ..., o_{n-1}\}$$
$$A_2 = \{o_n\}$$
$$A_3 = \{o_{n+1}, o_{n+2}\}$$

- In other words, C_1 splits the small category and C_1 splits the big category. A clustering quality measure, Q, preserving small clusters should give a higher score to C_2, that is, $Q(C_2, C_g) > Q(C_1, C_g)$.
- Many clustering quality measures satisfy some of these four criteria. BCubed precision and recall metrics satisfy all four criteria. BCubed evaluates the precision and recall for every object in a clustering on a given data set according to ground truth. The precision of an object indicates how many other objects in the same cluster belong to the same category as the object. The recall of an object reflects how many objects of the same category are assigned to the same cluster.

4.8.2 Intrinsic Methods

- When the ground truth of a data set is not available, you have to use an intrinsic method to assess the clustering quality. In general, intrinsic methods evaluate a clustering by examining how well the clusters are separated and how compact the clusters are. Many intrinsic methods have the advantage of a similarity metric between objects in the data set.

Silhouette Coefficient

- The silhouette coefficient is an intrinsic measure of cluster quality. For a data set, D, of n objects, suppose D is partitioned into k clusters, $C_1, \ldots, C_k$. For each object $o \in D$, you calculate a(o) as the average distance between o and all other objects in the cluster to which o belongs. Similarly, b(o) is the minimum average distance from o to all clusters to which o does not belong. Formally, suppose $o \in C_i$ $(1 \le i \le k)$; then

$$a(o) = \frac{\Sigma o' \in C_i, o \neq o' \text{ dist } (o, o')}{|C_i| - 1}$$

$$b(o) = \min_{C_j : 1 \le j \le k, i = j} \frac{\Sigma o' \in C_j \text{ dist } (o, o')}{|C_j|}$$

- The silhouette coefficient of o is then defined as

$$s(o) = \frac{b(o) - a(o)}{\max \{a(o), b(o)\}}$$

- The value of the silhouette coefficient is between −1 and 1.
  - 1 means clusters are well apart from each other and clearly distinguished.
  - 0 means clusters are indifferent, or you can say that the distance between clusters is not significant
  - −1 means clusters are assigned incorrectly
- The value of a(o) reflects the compactness of the cluster to which o belongs. The smaller the value, the more compact the cluster. The value of b(o) captures the degree to which o is separated from other clusters. The larger b(o) is, the more separated o is from other clusters. Therefore, when the silhouette coefficient value of o approaches 1, the cluster containing o is compact and o is far away from other clusters, which is the preferable case. However, when the silhouette coefficient value is negative, i.e., b(o) < a(o), then this means that, in expectation, o is closer to the objects in another cluster than to the objects in the same cluster as o. In many cases, this is a bad situation and should be avoided.
- To measure a cluster's fitness within a clustering, you can compute the average silhouette coefficient value of all objects in the cluster. To measure the quality of a clustering, you can use the average silhouette coefficient value of all objects in the data set. The silhouette coefficient and other intrinsic measures can also be used in the elbow method to heuristically derive the number of clusters in a data set by replacing the sum of within-cluster variances.

Review Questions

Here are a few review questions to help you gauge your understanding of this chapter. Try to attempt these questions and ensure that you can recall the points mentioned in the chapter.

Clustering

Q. 1 Write a short note on clustering. **(4 Marks)**

Q. 2 Why do you use clustering? Give an example. **(4 Marks)**

Q. 3 Clustering is a very useful algorithm in itself. Comment. **(4 Marks)**

Q. 4 Explain the properties of a cluster. **(4 Marks)**

Q. 5 Explain types of clustering. **(4 Marks)**

Q. 6 Explain 3 Use Cases of Clustering. **(6 Marks)**

Q. 7 Explain 3 Applications of Clustering. **(6 Marks)**

Q. 8 With example, describe customer segmentation using clustering. **(4 Marks)**

Q. 9 With example, describe image processing using clustering. **(4 Marks)**

Q. 10 With example, describe the use of clustering in the healthcare industry. **(4 Marks)**

Q. 11 With example, describe the use of clustering in recommendation engines. **(4 Marks)**

Q. 12 With example, describe the use of clustering in statistical data analysis. **(4 Marks)**

Q. 13 With example, describe the use of clustering in social network analysis. **(4 Marks)**

Q. 14 With example, describe the use of clustering in anomaly detection. **(4 Marks)**

Q. 15 Explain the K-means algorithm. **(5 Marks)**

Q. 16 Explain the steps involved in the K-means algorithm. **(6 Marks)**

Q. 17 With a suitable example, explain the steps involved in the K-means algorithm. **(10 Marks)**

Q. 18 Use the following data and group them using K-means clustering algorithm. Show calculation of centroids. **(10 Marks)**

Height	Weight
185	72
170	56
168	60
179	68
182	72
188	77
180	71
180	70
183	84
180	88
180	67
177	76

Q. 19 In a survey, the following data was collected for tv viewership. Is there a group which watches more tv than the other? **(10 Marks)**

Age	Number of TV watching hours
23	3
25	2
28	2
35	4
37	5
40	4
34	3
41	5
39	4
38	3

5 Ensemble Learning

Unit 5

Syllabus

At the end of this unit, you should be able to understand and comprehend the following syllabus topics:

- Ensemble Learning
  - Introduction to Ensemble Learning
  - Need of Ensemble Learning
  - Homogeneous and Heterogeneous ensemble methods
  - Advantages and Limitations of Ensemble methods
  - Applications of Ensemble Learning
- Basic Ensemble Learning Techniques
  - Voting Ensemble
  - Types of Voting
    - Max Voting
    - Averaging
    - Weighted Average
- Advanced Ensemble Learning Techniques
  - Bagging
    - Bootstrapping
    - Aggregation
  - Boosting
    - Adaptive Boosting (AdaBoost)
    - Gradient Boosting
    - XGBoost
  - Stacking
    - Variance Reduction
    - Blending
    - Random Forest Ensemble
    - Advantages of Random Forest

5.1 Introduction to Ensemble Learning

- The dictionary meaning of the word ensemble is "a group producing a single effect". For example, a musical orchestra producing some awesome music. Ensemble methods involve group of predictive models to achieve better accuracy and model stability. Ensemble methods reduce bias and variance from the models.
- Ensemble learning is a machine learning technique that involves combining multiple models, known as "base models" or "weak learners," to create a more powerful and accurate model, known as an "ensemble model" or "strong learner." The idea behind ensemble learning is that by aggregating the predictions or decisions of multiple models, the ensemble model can make better predictions than any individual model.
- Each base model in an ensemble is trained on a different subset of the training data or using a different algorithm. This diversity in training allows the base models to capture different patterns and relationships in the data. When it comes to making predictions, the ensemble model combines the predictions from all the base models, typically using a voting or averaging approach. The collective wisdom of the base models helps to reduce errors, improve generalisation, and enhance overall performance.
- Ensemble learning can be compared to a group decision-making process, where each base model represents a member with a different perspective or expertise. By combining the opinions of the group members, the ensemble model can make a more informed and accurate decision. This approach is especially effective when individual models have varying strengths and weaknesses, as the ensemble can compensate for their shortcomings and leverage their strengths.
- Overall, ensemble learning is a powerful technique that harnesses the collective knowledge of multiple models to improve predictions and achieve better performance in machine learning tasks.

5.1.1 Need for Ensemble Learning

Some of the key reasons for using ensemble learning are as following.

1. **Improved accuracy :** Ensemble learning can often produce more accurate predictions compared to individual models. By combining the predictions of multiple models, the ensemble can effectively reduce errors and increase overall accuracy. This is particularly useful when dealing with complex or noisy datasets where a single model might struggle to capture all the underlying patterns.
2. **Increased robustness :** Ensemble learning enhances the robustness of the model. If a base model in the ensemble makes a mistake or performs poorly on certain instances, other models can compensate for it. By aggregating the predictions of multiple models, the ensemble becomes more resilient to outliers, noisy data, or biases in individual models.
3. **Reduction of overfitting :** Overfitting occurs when a model becomes too specialised in learning from the training data and fails to generalise well on unseen data. Ensemble learning can mitigate overfitting by introducing diversity among the base models. Each model is trained on a different subset of data or using a different algorithm, reducing the likelihood of all models making the same overfitting errors.
4. **Handling model bias :** Models can have inherent biases based on the training data or algorithmic limitations. Ensemble learning allows combining models with different biases, which can lead to a more balanced and unbiased prediction. The ensemble can consider different perspectives and mitigate the impact of individual model biases.

5. **Exploration of solution space :** Ensemble learning can explore a broader solution space by combining models with different approaches or algorithms. Each base model might specialise in capturing specific aspects or patterns of the data, enabling the ensemble to leverage diverse strategies. This can lead to more comprehensive coverage of the problem domain and potentially discover better solutions.

6. **Improved stability :** Ensemble learning can enhance the stability of the model's performance over time. Since the ensemble relies on the collective decisions of multiple models, minor fluctuations or changes in individual models' performance have a limited impact on the overall ensemble's predictions. This stability is desirable when dealing with dynamic or evolving datasets.

5.1.2 Advantages of Ensemble Learning

Some of the advantages of ensemble learning are as following.

1. **Improved predictive performance :** Ensemble learning has the potential to achieve higher predictive accuracy compared to individual models. By combining the predictions of multiple models, the ensemble can capture different patterns and relationships in the data, leading to more accurate predictions. This can be particularly beneficial in tasks where accuracy is crucial, such as in medical diagnosis or financial forecasting.

2. **Robustness and error reduction :** Ensemble learning enhances the robustness of the model by reducing the impact of errors or biases in individual models. If a base model makes a mistake on a particular instance, other models in the ensemble can provide corrective predictions. This collective decision-making helps to mitigate errors and improve overall performance, making the ensemble more reliable and resilient.

3. **Generalisation and variance reduction :** Ensemble learning helps in reducing overfitting and improving generalisation. Overfitting occurs when a model becomes too specialised in learning from the training data and fails to generalise well to new data. By combining multiple models trained on different subsets of data or using different algorithms, ensemble learning reduces the risk of overfitting and produces a model with better generalisation capabilities.

4. **Handling complex and noisy data :** Ensemble learning can effectively handle complex datasets with noisy or conflicting information. The diversity among the base models allows the ensemble to capture different aspects of the data and extract meaningful patterns. It helps to smooth out noisy data points, reduce the impact of outliers, and make more reliable predictions even in challenging scenarios.

5. **Bias reduction :** Ensemble learning can help in reducing biases present in individual models. Different models may have different biases due to the data they were trained on or the algorithms used. By combining models with diverse biases, the ensemble can balance out these biases and produce more fair and unbiased predictions. This is particularly important in applications like hiring processes or loan approvals, where avoiding bias is crucial.

6. **Model combination and interpretation :** Ensemble learning allows for combining the strengths of different models and leveraging their individual expertise. Each base model may excel at capturing specific aspects of the data or using different algorithms. By combining these models, the ensemble can provide a more comprehensive understanding of the problem domain and deliver superior predictions. Additionally, ensemble models can also provide insights into the importance of different features or patterns, aiding interpretability.

7. **Flexibility and adaptability :** Ensemble learning is a flexible approach that can accommodate various types of models and algorithms. It can be applied to different machine learning techniques, including decision trees, neural networks, support vector machines, and more. This flexibility allows for experimentation and adaptation to different problem domains and data characteristics.

5.1.3 Disadvantages (Limitations) of Ensemble Learning

Some of the disadvantages (limitation) of ensemble learning are as following.

1. **Increased complexity :** Ensemble learning introduces additional complexity compared to using a single model. Managing and training multiple models require more computational resources, memory, and time. The process of building an ensemble can be more intricate, involving techniques such as model selection, model combination, and tuning ensemble parameters. This complexity can make ensemble learning more challenging to implement and maintain.

2. **Higher computational requirements :** As ensemble learning involves training and combining multiple models, it typically requires more computational resources and time compared to training a single model. The training time and memory requirements increase linearly with the number of base models in the ensemble. This can be a limitation when dealing with large datasets or limited computational resources.

3. **Decreased model interpretability :** Ensemble models tend to be less interpretable compared to individual models. While individual models, such as decision trees or linear regression, can provide clear insights into feature importance and decision-making processes, the combination of multiple models in an ensemble can make it harder to interpret and understand how predictions are made. This reduced interpretability may be a concern in domains where explainability is essential.

4. **Overfitting risks :** Although ensemble learning can help mitigate overfitting, there is still a risk of overfitting if the ensemble is too complex or if the individual models are highly correlated. If the base models are too similar, they may make similar errors, reducing the ensemble's ability to generalise. Careful selection of diverse and independent base models is necessary to avoid overfitting and ensure the ensemble's effectiveness.

5. **Increased training time :** Training an ensemble of models typically requires more time compared to training a single model. Each base model needs to be trained independently, which can significantly increase the overall training time, especially for large ensembles or complex models. Additionally, the process of combining predictions or decisions from the base models during inference also adds computational overhead.

5.1.4 Applications of Ensemble Learning

Some of the common applications of ensemble learning are as following.

1. **Classification :** Ensemble learning is widely used in classification tasks, where the goal is to assign a class label to a given input. Ensemble methods like Random Forests, Gradient Boosting Machines (GBMs), and AdaBoost are popular in classification problems. They combine multiple classifiers to improve accuracy and handle complex decision boundaries.

2. **Regression :** Ensemble learning techniques can also be applied to regression problems, where the goal is to predict a continuous numerical value. Ensemble regression methods, such as Boosted Regression Trees (BRTs) and Stacked Generalisation (stacking), combine multiple regression models to enhance predictive accuracy and handle non-linear relationships.

3. **Anomaly detection :** Ensemble learning can be utilised for anomaly or outlier detection tasks. By combining multiple anomaly detection models, an ensemble can identify unusual patterns or instances that deviate significantly from the norm. Ensemble-based anomaly detection methods can improve detection accuracy and reduce false positives.

4. **Recommender systems :** Ensemble learning is employed in recommender systems to provide personalised recommendations to users. Collaborative filtering techniques, such as ensemble-based matrix factorization or hybrid ensemble methods, combine the predictions of multiple recommendation algorithms to enhance the quality and diversity of recommendations.
5. **Image and object recognition :** Ensemble learning is used in computer vision tasks such as image classification and object recognition. Ensembles of convolutional neural networks (CNNs) or deep learning models can improve accuracy by combining the predictions of multiple models or performing model averaging.
6. **Natural language processing :** In natural language processing (NLP), ensemble learning is applied to tasks like sentiment analysis, text classification, and named entity recognition. By combining the outputs of multiple NLP models, ensemble methods can improve performance, handle linguistic variations, and capture different aspects of language semantics.
7. **Time series forecasting :** Ensemble learning techniques can be employed in time series forecasting to enhance prediction accuracy. By combining the forecasts of multiple models, ensemble methods can capture different temporal patterns, reduce forecasting errors, and handle uncertainties in the data.
8. **Medical diagnosis :** Ensemble learning is valuable in medical diagnosis and disease prediction. Ensembles of classifiers or decision support systems can combine different diagnostic criteria, clinical data, or medical imaging features to improve accuracy and provide more reliable diagnoses.
9. **Fraud detection :** Ensemble learning is effective in fraud detection and anomaly identification in financial transactions. By combining the outputs of multiple fraud detection models, ensemble methods can enhance detection rates, reduce false positives, and adapt to evolving fraudulent behaviours.
10. **Portfolio optimisation :** Ensemble learning techniques are utilised in financial applications, such as portfolio optimisation. Ensemble-based models can combine the predictions of multiple trading strategies or investment models to construct portfolios that balance risk and return and improve investment performance.

5.1.5 Homogeneous and Heterogeneous Ensemble Methods

- Ensemble learning is broadly categorised into homogeneous and heterogeneous methods based on the approaches they take for combining base models.
- The Fig. 5.1.1 illustrates the difference between how homogeneous and heterogeneous ensemble methods work.

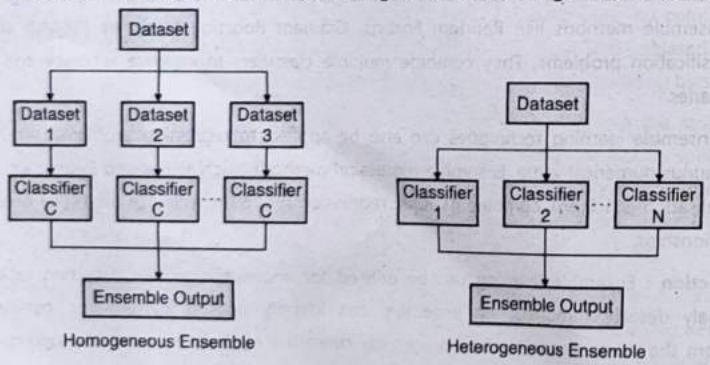


Fig. 5.1.1

Let's learn about them briefly.

5.1.5(A) Homogeneous Ensemble Methods

- Homogeneous ensemble methods consist of multiple base models that are of the same type or belong to the same family. These models are often referred to as "weak learners" or "base classifiers/regressors". Homogeneous ensembles leverage the diversity created by training multiple instances of the same model with different subsets of the training data or different initialisations.
- Some popular homogeneous ensemble methods are as following.

1. **Bagging (Bootstrap Aggregating) :** Bagging combines multiple independently trained base models, such as decision trees, by training them on different bootstrap samples of the training data. The ensemble combines their predictions, typically using majority voting for classification or averaging for regression.
2. **Random Forest :** Random Forest is a variant of bagging that uses decision trees as the base models. Each tree is trained on a bootstrap sample, and at each split, a random subset of features is considered. The final prediction is obtained by aggregating the predictions of all the trees.
3. **AdaBoost (Adaptive Boosting) :** AdaBoost is an iterative ensemble method that assigns weights to training instances and focuses on the instances that are difficult to classify correctly. Base models are trained sequentially, with each subsequent model giving more weight to the misclassified instances of the previous models. The final prediction is obtained by combining the weighted predictions of all the base models.

5.1.5(B) Heterogeneous Ensemble Methods

Heterogeneous ensemble methods involve combining base models that are different in terms of their type, algorithm, or characteristics. These models may have complementary strengths and weaknesses, and by combining them, the ensemble aims to leverage their diverse perspectives and improve overall performance. Some common heterogeneous ensemble methods are as following.

1. **Stacking**

Stacking combines multiple base models of different types by training a meta-model on the predictions of the base models. The meta-model learns to combine the base models' predictions and make the final prediction. The base models can be diverse, including decision trees, support vector machines, neural networks, or any other machine learning algorithm.

2. **Boosting**

Boosting is a sequential ensemble method that combines weak learners by emphasizing the misclassified instances in each iteration. Different boosting algorithms, such as AdaBoost, Gradient Boosting Machines (GBMs), or XGBoost, use different strategies to assign weights to instances or optimise the model's parameters during training.

3. **Voting**

Voting ensembles combine the predictions of multiple base models using a voting mechanism. There are different types of voting, such as majority voting (for classification), weighted voting (where each base model has a different weight), or soft voting (taking into account the probabilities/confidences of the predictions). Voting can involve any combination of base models, including decision trees, SVMs, neural networks, etc.

5.1.5(C) Comparison between Homogeneous and Heterogenous Ensemble Methods

The Table 5.1.1 provides a quick comparison between homogeneous and heterogenous ensemble methods.

Table 5.1.1

Comparison Attributes	Homogeneous Ensemble Methods	Heterogeneous Ensemble Methods
Composition	Multiple base models of the same type	Multiple base models of different types
Base Models	Same or similar algorithms/family (e.g., decision trees, SVMs)	Different algorithms/types (e.g., decision trees, SVMs, neural networks)
Diversity	Diversity achieved by training on different subsets of data or different initialisations	Diversity achieved by combining different modelling techniques
Strengths	Simpler to implement and train	Potential for higher performance improvement
Interpretability	Base models may be more interpretable	Base models may be less interpretable
Complexity	Lower complexity and computational requirements	Higher complexity and computational requirements
Performance	Can still improve performance through aggregation of base models	Can potentially provide greater performance improvement due to diverse modelling techniques
Example Methods	Bagging, Random Forests, AdaBoost	Stacking, Boosting, Voting

5.2 Basic Ensemble Learning Techniques

Let's learn about a basic ensemble learning technique.

5.2.1 Voting

Voting averages the probability estimates or numeric predictions for classification or regression. In the case of regression, this involves calculating the average of the predictions from the models. In the case of classification, the predictions for each label are summed and the label with the majority vote is predicted. Fig. 5.2.1 is a conceptual diagram for how voting works.

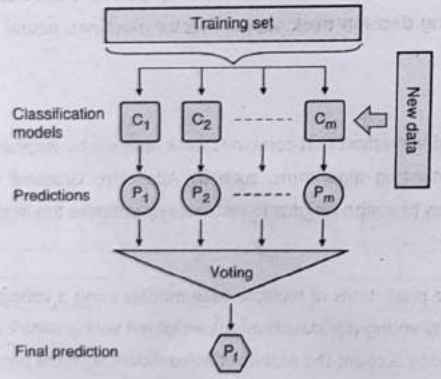


Fig. 5.2.1

5.2.2 Types of Voting

Voting could be carried out in several ways. Let's learn about a few of them.

1. **Majority Voting / Max Voting / Hard Voting**
   - Majority Voting / Max Voting / Hard Voting involves summing the predictions for each class label and predicting the class label with the most votes.
   - For example, assume that there are three classifiers which predicted the class labels as following.
     - Classifier 1 → class A
     - Classifier 2 → class B
     - Classifier 3 → class A
   - Now, since majority voting (A, B, A) = A, you overall predict the class label as A.
2. **Soft Voting**
   - Soft voting involves summing the predicted probabilities (or probability-like scores) for each class label and predicting the class label with the largest probability. For example, assume that the classifiers classified the labels as following along with the respective probabilities.
     - Classifier 1 → class A, probability = 0.9
     - Classifier 2 → class B, probability = 0.6
     - Classifier 3 → class A, probability = 0.4
   - You then calculate the average of the probabilities for each class label.

Average probability of predicting class A $= \frac{0.9 + 0.4 + 0.4}{3} = 0.57$

Average probability of predicting Class B $= \frac{0.1 + 0.6 + 0.6}{3} = 0.43$

As the average probability of predicting class A is higher than the average probability of predicting class B, you overall predict the class label as A.

3. **Averaging**

Averaging is a simple and commonly used method in ensemble learning, where the predictions of multiple models are combined by taking the average. In this approach, each model in the ensemble independently makes predictions, and the final prediction is obtained by averaging the individual predictions.

4. **Weighted Average Voting**

Weighted voting is similar to majority voting, but each model's prediction is given a weight, and the final prediction is determined by summing the weighted votes. The weights can be assigned based on the performance or expertise of the individual models.

5. **Stacked Voting**

Stacked voting, also known as stacked generalisation, involves training multiple models on the same dataset. The predictions of these models are then used as features for a meta-model, which makes the final prediction. The meta-model is trained to learn the optimal combination of the base models' predictions.

6. Conditional Voting

In conditional voting, the models in the ensemble are divided into different groups based on some criteria, such as their performance on specific subsets of the data. Each group of models votes independently, and the final prediction is determined based on the voting results of the different groups.

7. Hierarchical Voting

Hierarchical voting involves creating a hierarchy among the models in the ensemble. The models are organised in a tree-like structure, where the predictions of higher-level models serve as inputs to lower-level models. The final prediction is obtained by aggregating the predictions from the lowest-level models.

5.3 Advanced Ensemble Learning Technique

Now that you understand what ensemble learning is, let's learn about a few ensemble learning techniques

5.3.1 Bagging (Bootstrapping and Aggregation)

✍ ***Definition :*** *Bagging is an ensemble technique designed to improve the stability and accuracy of classification algorithms.*

- It is a short form for Bootstrap aggregating. It reduces bias and variance from the predictive models.
- In bagging, two key concepts are used: bootstrapping and aggregation.
  1. **Bootstrapping :** Bootstrapping is a statistical resampling technique that involves creating multiple subsets of the original dataset by randomly sampling with replacement. Each subset is the same size as the original dataset, but some instances may be duplicated while others may be left out. Bootstrapping allows you to generate diverse subsets that capture different variations of the data.
  2. **Aggregation :** Aggregation refers to combining the predictions of multiple models to arrive at a final prediction. In bagging, aggregation is typically done by taking a majority vote (for classification tasks) or averaging (for regression tasks) across the predictions of all the models in the ensemble. The idea behind aggregation is that by combining the predictions of multiple models, we can reduce the variance and improve the overall predictive performance.
- The way bootstrap method (bagging) works is as following.
- Let's assume that you have a sample of 100 values (say distribution *d*), and you would like to get an estimate of the mean of the sample. You could simply calculate the mean directly from the sample as

$$\text{Mean (d)} = \frac{\text{Sum of values in d}}{100}$$

- Now suppose that your samples have very low as well as very high values. Since the sample size is small it could be that your mean calculation might not be an accurate representation of the distribution. You could improve the estimate of the mean using the bootstrap procedure by
  1. Creating many (say 1000) random sub-datasets of your original dataset. You may select a datapoint in various sub-datasets multiple times.
  2. Calculating the mean of each sub-dataset
  3. Calculating the average of all the means for sub-datasets

- Let's take a simple example. Assume that you have the following sample values, and you want to know the mean representation of the sample.

Sample Values
5
8
9
12
45
2
0
11
3
4

- If you calculate a simple mean of this dataset you would get 9.9. Let's break this dataset into 5 sub-datasets.
- Average of all the means from the 5 sub-dataset

Sub-dataset 1
5
8
4
Mean = 5.66

Sub-dataset 2
45
0
11
Mean = 18.66

Sub-dataset 3
45
2
8
Mean = 18.33

Sub-dataset 4
12
9
4
Mean = 8.33

Sub-dataset 5
2
0
3
Mean = 1.66

Samples is $\frac{5.66 + 18.66 + 18.33 + 8.33 + 1.66}{5} = 10.52$

A simple block diagram for bagging as shown in Fig. 5.3.1.

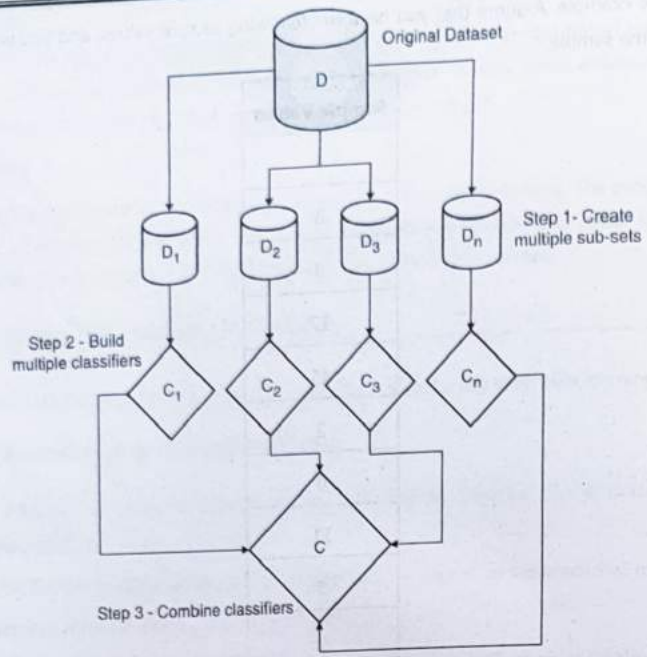


Fig. 5.3.1

Bagging involves three steps

1. **Create multiple sub-sets :** From the original dataset, multiple sub-sets are created by randomly drawing sample datapoints.
2. **Build multiple classifiers :** Classification models are built for each sub-set. Generally, the same classifier model is used for each sub-set.
3. **Combine classifier :** The classification prediction from each classifier is combined using statistical methods as appropriate.

5.3.2 Subagging

The word Subagging is derived from a combination of 'subsample' and 'bagging'. It is very similar to how bagging is performed. Unlike bagging, instead of producing samples that are the same size as original data and using same datapoints multiple times, in subagging you make smaller datasets and do not reuse the same datapoints. It is common to use a dataset size that is half that of the original data, and the results of this can often be comparable to a full bagging simulation.

Boosting

✍ ***Definition :*** *Boosting is an ensemble technique designed to combine several weak classifiers into a strong classifier.*

- A weak classifier is the one that is only slightly related to the final classification whereas a strong classifier is well-correlated with the final classification. Let's take an example.

- Suppose that you want to identify a received SMS as fraud or not fraud. What are some of the criteria that you would look for in the SMS to classify it? Some of the criteria could be
  1. If the SMS is from a known number – Not Fraud
  2. If the SMS is from an official provider such as a bank – Not Fraud
  3. If the SMS contains a link – Fraud
  4. If the SMS announces that you won something – Fraud
  5. If the SMS comes from an unidentified source – Fraud
  6. If the SMS has some sort of urgency – Fraud
  7. If the SMS is asking for a quick action – Fraud
- Now, look at these classification rules (or say models) individually. Each one of these do not strongly stand on its own but when you combine them together and run a SMS through these models you end up in strongly classifying a SMS as fraud or not fraud. That is precisely what boosting does.

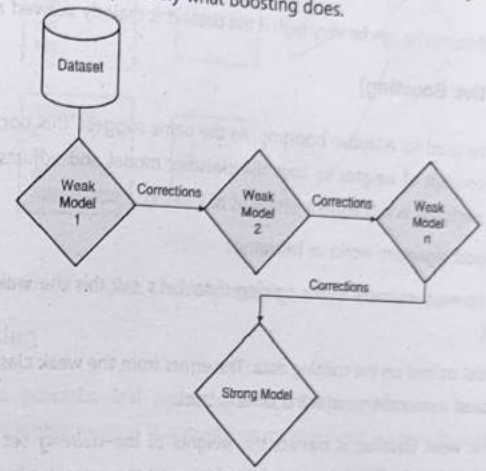


Fig. 5.3.2

- At a high-level, boosting involves two stages :
  1. Iteratively build models each time assigning weight and correcting errors from the previous model
  2. Combine the results from the several models to get a strong model
- Boosting is an iterative process that is continued until you have found a sufficiently strong model. Boosting pays higher focus on datapoints which are mis-classified or have higher errors by the preceding classifier model. In each iteration, it identifies the mis-classified datapoints, increases their weights and decreases the weights of correctly classified points, so that the next classifier will have higher chances of getting mis-classified datapoints right.

5.3.3 Stumping (Decision Stump)

- There is a very extreme form of boosting that is applied to trees. In real world, the stump of a tree is the tiny piece that is left over when you chop off the rest of the tree.
- Similarly, in machine learning, stumping consists of simply taking the root of the tree and using that as the decision maker. So, for each classifier, you use the very first question that makes up the root of the tree, and that is it. The decision tree is not further grown. It is also called as a decision stump.

Definition : *A decision stump is a machine learning model consisting of a one-level decision tree.*

- It is a decision tree with one internal node (the root) which is immediately connected to the terminal nodes (its leaves). A decision stump makes a prediction based on the value of just a single input feature.
- Following is an example of a decision stump.

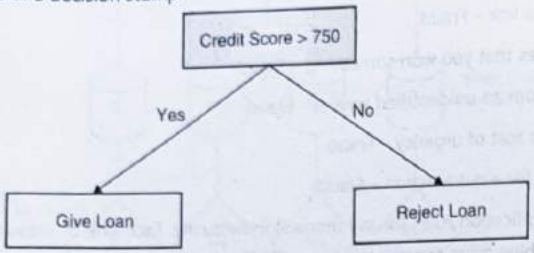


Fig. 5.3.3

- The overall performance of stumping can be very high if the dataset is majorly skewed around a particular feature.

5.3.4 AdaBoost (Adaptive Boosting)

- AdaBoost is the short name used for adaptive boosting. As the name suggest, this boosting method is adaptive in the sense that it uses a concept of weights to train the classifier model and adjusts these weights dynamically based on how the model performs in subsequent iterations of training.
- At a high-level, the AdaBoost algorithm works as following.

(1) A weight is applied to every example in the training data. Let's call this the weight vector D. Initially, these weights are all equal.

(2) A weak classifier is first trained on the training data. The errors from the weak classifier are calculated, and the weak classifier is trained a second time with the same dataset.

(3) The second time the weak classifier is trained, the weights of the training set are adjusted such that the examples properly classified the first time are weighted less and the examples incorrectly classified in the first iteration are weighted more.

(4) To get uniform output from all of these weak classifiers, AdaBoost assigns α values to each of the classifiers. The α values are based on the error of each weak classifier. The error ϵ is given as

$$\epsilon = \frac{\text{number of incrorrectly classified examples}}{\text{total number of examples}}$$

α is calculated as

$$\alpha = \frac{1}{2} \ln\left(\frac{1-\epsilon}{\epsilon}\right)$$

(5) After you calculate α, you can update the weight vector D so that the examples that are correctly classified will decrease in weight and the misclassified examples will increase in weight.

If the classification was correct, then D is calculated as

$$D_i^{t+1} = \left(D_i^t \frac{e^{-\alpha}}{\text{Sum (D)}}\right)$$

Else if the classification was incorrect, then D is calculated as

$$D_i^{t+1} = \left(D_i^t \frac{e^{\alpha}}{\text{Sum (D)}}\right)$$

(6) After D is calculated, AdaBoost starts on the next iteration. The AdaBoost algorithm repeats the training and weight-adjusting iterations until the training error is 0or until the number of weak classifiers reaches a user-defined value.

The AdaBoost algorithm is illustrated as following.

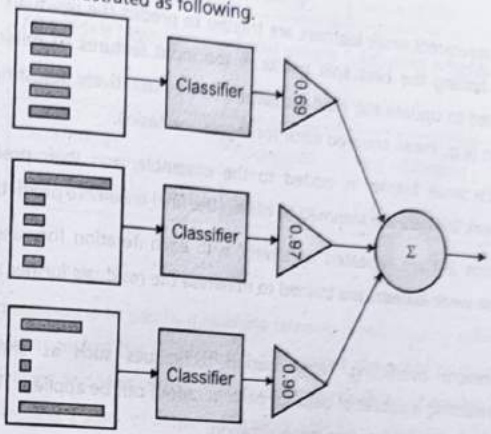


Fig. 5.3.4

As you see, you start with equal weights, and then keep adjusting the weights as you proceed.

5.3.5 Gradient Boosting

Gradient boosting is a powerful and popular machine learning technique used for both regression and classification tasks. It is an ensemble method that combines multiple weak learners, typically decision trees, to create a strong predictive model.

The basic idea behind gradient boosting is to sequentially train new models that focus on reducing the errors made by the previous models in the ensemble. This iterative process results in an ensemble that can effectively capture complex relationships in the data and make accurate predictions. The Fig. 5.3.5 illustrates gradient boosting.

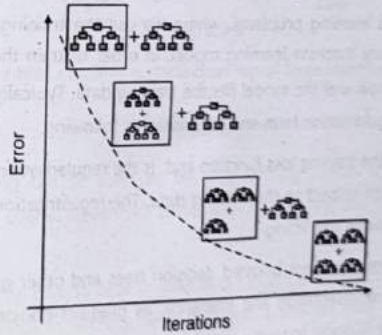


Fig. 5.3.5

The key components and steps involved in gradient boosting are as following.

1. **Weak Learners :** Gradient boosting starts with a weak learner, often a decision tree with a shallow depth (also known as a "weak learner"). The weak learner is trained to predict the target variable based on the input features.
2. **Residual Calculation :** The difference between the actual target values and the predictions of the current weak learner is calculated. These differences, known as residuals or errors, represent the "unexplained" portion of the target variable.
3. **Gradient Descent :** The subsequent weak learners are trained to predict the residuals from the previous step. The learning process involves finding the best split points in the input features to minimise the residuals. Gradient descent optimisation is used to update the model's parameters by iteratively adjusting them in the direction that minimises the loss function (e.g., mean squared error for regression tasks).
4. **Ensemble Building :** Each weak learner is added to the ensemble, and their predictions are combined. The predictions from all the weak learners are summed to obtain the final ensemble prediction.
5. **Boosting Iterations :** Steps 2-4 are repeated iteratively, with each iteration focusing on the residuals from the previous iteration. The new weak learners are trained to minimise the residuals further, leading to a reduction in the overall prediction errors.
6. **Regularisation :** To prevent overfitting, regularisation techniques such as shrinkage (learning rate) and subsampling (randomly selecting a subset of data for each iteration) can be applied. These techniques help control the complexity of the final model and improve generalization.

Gradient boosting is known for its ability to handle complex and nonlinear relationships in the data, and it often achieves state-of-the-art performance on various machine learning tasks. Popular implementations of gradient boosting include XGBoost, LightGBM, and CatBoost, which offer additional optimisations and features to improve training efficiency and model performance.

5.3.6 XGBoost

- XGBoost (eXtreme Gradient Boosting) is an open-source software library which provides a regularising gradient boosting framework for various languages such as C++, Java, Python, etc. It is highly efficient, flexible and portable. It provides a parallel tree boosting (also known as GBDT, GBM) that solves many data science problems in a fast and accurate way. The same code runs on major distributed environment such as Hadoop, SGE, and MPI.
- XGBoost is used for supervised learning problems, where we use the training data (with multiple features) to predict a target variable y_i. In any machine learning model, in order to train the model, you need to define the objective function to measure how well the model fits the training data. Typically, an objective function consist of two parts – training loss and regularisation term and is denoted as following.
- $obj(\theta) = L(\theta) + \Omega(\theta)$ where L is the training loss function and is the regularisation term. The training loss measures how predictive is the model with respect to the training data. The regularisation term controls the complexity of the model, which helps you to avoid overfitting.
- XGBoost is normally used to train gradient-boosted decision trees and other gradient boosted models. Random Forests use the same model representation and inference, as gradient-boosted decision trees, but a different training algorithm. You can use XGBoost to train a standalone random forest or use random forest as a base model for gradient boosting.

5.3.7 Comparison between Bagging and Boosting

The Table 5.3.1 provides a quick comparison between Bagging and Boosting.

Table 5.3.1

Comparison Attribute	Bagging	Boosting
Weak models work	Parallelly	Sequentially
All models have	Equal weight	Adjusted weights
Reduces	Variance	Bias
Overfitting	Addressed	Not addressed
Merge predictions of	Same type	Different types

5.3.8 Stacking

The idea behind stacking method is to handle a machine learning problem using different types of models that are capable of learning a particular aspect of the problem but not explore the entire problem space. Using stacking, you can make intermediate predictions and then add a new model that can learn using the intermediate predictions and gradually progress towards building a stack of models that work with each other. Fig. 5.3.6 is a conceptual diagram for how stacking works.

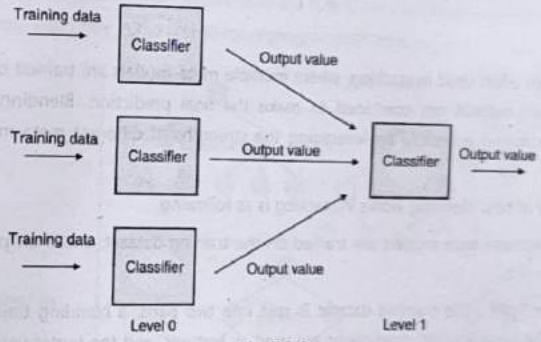


Fig. 5.3.6

The Fig. 5.3.6 represents that a final classifier is stacked on top of three intermediate classifiers.

5.3.9 Variance Reduction

Variance reduction in stacking can be achieved by incorporating various techniques to reduce the individual model's variance and improve the overall ensemble's performance. Some approaches for variance reduction in stacking are as following.

1. **Diverse Base Models :** Selecting diverse base models in the stacking ensemble is crucial. By incorporating models with different underlying algorithms, architectures, or hyperparameters, you can reduce the risk of all models making similar errors. Diversity in the base models helps to decrease variance and improve the overall ensemble's robustness.

2. **Regularisation :** Applying regularisation techniques to the base models can help reduce their individual variance. Regularisation methods like L1 or L2 regularisation, dropout, or early stopping can prevent overfitting and encourage the models to generalise better to unseen data. Regularisation helps to control the complexity of the individual models and reduce their variance.
3. **Ensemble Size :** Increasing the number of base models in the stacking ensemble can help reduce the variance. As the ensemble size grows, the predictions of individual models get averaged out, leading to more stable and reliable predictions. However, there is a trade-off between ensemble size and computational complexity, so the number of models should be chosen wisely.
4. **Meta-Model Regularisation :** Just as regularisation is applied to the base models, it can also be applied to the meta-model in stacking. Regularisation techniques like L1 or L2 regularisation or early stopping can help control the complexity of the meta-model and prevent overfitting. Regularising the meta-model reduces its variance and improves the overall ensemble's performance.
5. **Cross-Validation :** Proper cross-validation techniques can help estimate the performance of the stacking ensemble and reduce the variance. By using techniques like k-fold cross-validation, you can evaluate the ensemble's performance on multiple validation folds and obtain a more robust estimate of its generalization performance. Cross-validation provides a better understanding of the ensemble's variance and helps in model selection.
6. **Model Blending :** Instead of using a single meta-model, model blending can be employed, where multiple meta-models are trained and their predictions are combined. By blending predictions from different meta-models, you can further reduce variance and improve the ensemble's stability.

5.3.10 Blending

- Blending is a technique often used in stacking, where multiple meta-models are trained on the predictions of the base models and their outputs are combined to make the final prediction. Blending can help improve the performance of the stacking ensemble by leveraging the strengths of different meta-models and reducing the overall variance.
- A high-level overview of how blending works in stacking is as following.

1. **Base Models :** Multiple base models are trained on the training dataset, and their predictions for the target variable are obtained.
2. **Train-Validation Split :** The training dataset is split into two parts: a blending training set and a blending validation set. The base models' predictions are used as features, and the target variable is used for training the blending meta-models.
3. **Blending Meta-Models :** Multiple meta-models, also known as blending models, are trained on the blending training set. Each blending model takes the base models' predictions as inputs and learns to make predictions for the target variable.
4. **Blending Predictions :** The blending models are used to make predictions on the blending validation set, which was not used during their training. The blending predictions serve as a measure of the blending models' performance and are combined to obtain the final blending ensemble prediction.
5. ***Final Prediction*** **:** The base models' predictions are combined using appropriate aggregation techniques, such as averaging or majority voting, to obtain the final ensemble prediction. The blending ensemble prediction may be combined with the predictions of the individual base models as well.

- Blending allows for combining different meta-models that may have different strengths or capture different aspects of the data. Each blending model may focus on different patterns or exploit different modelling techniques, leading to a more robust and accurate final prediction. By blending the predictions of multiple meta-models, the stacking ensemble can achieve better generalisation performance and reduce variance.
- It's worth noting that blending introduces an additional layer of complexity to the stacking ensemble and requires careful model selection and tuning. The choice of blending models, the number of blending models, and the aggregation technique for combining their predictions should be carefully considered based on the problem domain and the characteristics of the base models and data.

5.4 Random Forests

Definition : *Random Forests is an ensemble technique designed to combine several decision trees to reduce errors and to build a more accurate prediction model.*

In random forest, instead of building one decision tree, multiple decision trees are built.

1. Each tree in the forest is built using random datapoints drawn from the dataset.
2. The trees in the forest are split such that a fixed subset of input variables (attributes) are taken each time and the best possible split is performed. For example, if there are 5 attributes in the dataset, you could choose to take 2 attributes at a time and split based on those attributes to create a tree in the forest.
3. Finally, the result from each tree is aggregated to give the outcome. Usually, you count the total number of votes (based on the classification outcome from each tree in the forest) for classification problems and average (based on the outcome value from each tree in the forest) for regression problems.

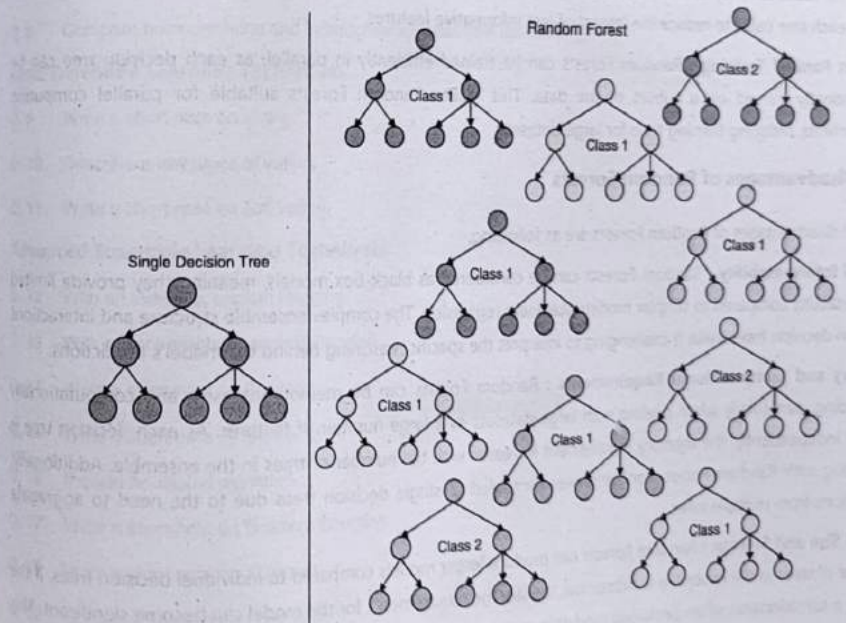


Fig. 5.4.1

5.4.1 Advantages of Random Forests

Some of the major advantages of random forests are as following.

1. **Robustness to Overfitting :** Random Forests are less prone to overfitting compared to individual decision trees. By combining multiple decision trees, each trained on a random subset of features and data, Random Forests reduce the risk of overfitting and improve generalisation performance.
2. **High Predictive Accuracy :** Random Forests tend to provide high predictive accuracy due to the ensemble's ability to capture complex relationships in the data. The aggregation of multiple decision trees allows for more robust and accurate predictions, especially in the presence of noisy or incomplete data.
3. **Outlier and Noise Handling :** Random Forests are robust to outliers and noisy data points. The majority voting or averaging mechanism used in Random Forests helps mitigate the impact of individual noisy predictions, reducing their influence on the final prediction.
4. **Feature Importance :** Random Forests provide a measure of feature importance. By examining the average decrease in impurity across all decision trees, Random Forests can rank the features based on their predictive power. This feature importance information can aid in feature selection, identifying the most relevant features for the task at hand.
5. **Nonlinear Relationships :** Random Forests can capture nonlinear relationships between features and the target variable. By recursively partitioning the feature space, decision trees can capture intricate patterns and interactions, allowing Random Forests to model complex relationships effectively.
6. **Handling Large Feature Spaces :** Random Forests can handle high-dimensional feature spaces with a large number of features. They are often robust to irrelevant or redundant features, as the random subset of features used in each tree helps to reduce the impact of less informative features.
7. **Efficient Parallel Training :** Random Forests can be trained efficiently in parallel, as each decision tree can be independently trained on a subset of the data. This makes Random Forests suitable for parallel computing environments, reducing training time for large datasets.

5.4.2 Disadvantages of Random Forests

Some of the disadvantages of Random Forests are as following.

1. **Lack of Interpretability :** Random Forests can be considered as black-box models, meaning they provide limited interpretability compared to simpler models like linear regression. The complex ensemble structure and interactions between decision trees make it challenging to interpret the specific reasoning behind the model's predictions.
2. **Memory and Computational Requirements :** Random Forests can be memory-intensive and computationally demanding, particularly when dealing with large datasets or a large number of features. As each decision tree is trained independently, the memory requirement increases with the number of trees in the ensemble. Additionally, predicting with Random Forests can be slower compared to single decision trees due to the need to aggregate predictions from multiple trees.
3. **Model Size and Storage :** Random Forests can produce larger models compared to individual decision trees. If the number of trees in the ensemble is substantial, the storage requirements for the model can become significant. This can be a consideration when deploying models in memory-constrained or resource-limited environments.

4. **Biased Predictions for Imbalanced Data :** Random Forests tend to favour majority classes when dealing with imbalanced datasets. The majority class tends to have more influence on the predictions, potentially leading to biased results and lower performance for minority classes.
5. **Hyperparameter Tuning :** Random Forests have several hyperparameters that require tuning, such as the number of trees, maximum tree depth, and the number of features to consider at each split. Finding the optimal combination of hyperparameters can be time-consuming and computationally expensive.

Review Questions

Here are a few review questions to help you gauge your understanding of this chapter. Try to attempt these questions and ensure that you can recall the points mentioned in the chapter.

Introduction to Ensemble Learning

Q. 1	What is ensemble learning?	
Q. 2	Why do we need ensemble learning?	(4 Marks)
Q. 3	Describe the advantages of ensemble learning.	(6 Marks)
Q. 4	Describe the disadvantages of ensemble learning.	(4 Marks)
Q. 5	Describe a few applications of ensemble learning.	(4 Marks)
Q. 6	Write a short note on homogenous ensemble methods.	(6 Marks)
Q. 7	Write a short note on heterogeneous ensemble methods.	(4 Marks)
Q. 8	Compare homogeneous and heterogeneous ensemble methods.	(4 Marks)
		(6 Marks)

Basic Ensemble Learning Techniques

Q. 9	Write a short note on voting.	(4 Marks)
Q. 10	Describe a few types of voting.	(6 Marks)
Q. 11	Write a short note on soft voting.	(4 Marks)

Advanced Ensemble Learning Techniques

Q. 12	With an example, explain Bagging.	(6 Marks)
Q. 13	With a block diagram, explain Bagging.	(6 Marks)
Q. 14	Write a short note on Boosting.	(4 Marks)
Q. 15	Write a short note on Stumping.	(4 Marks)
Q. 16	Explain AdaBoost algorithm.	(6 Marks)
Q. 17	Write a short note on Gradient Boosting.	(4 Marks)
Q. 18	Write a short note on XGBoost.	(4 Marks)
Q. 19	Compare bagging and boosting.	(4 Marks)
Q. 20	With a block diagram, explain Boosting.	(6 Marks)

Q. 21	Write a short note on stacking.	(4 Marks)
Q. 22	Describe a few ways for reducing variance in stacking ensemble.	(4 Marks)
Q. 23	Write a short note on blending.	(4 Marks)
Q. 24	Write a short note on Random Forests.	(4 Marks)
Q. 25	Explain advantages of Random Forests.	(4 Marks)
Q. 26	Describe disadvantages of Random Forests.	(4 Marks)

□□□

6 Reinforcement Learning

Unit 6

Syllabus

At the end of this unit, you should be able to understand and comprehend the following syllabus topics:

- Reinforcement learning
  - What is Reinforcement Learning
  - Need for Reinforcement Learning
  - Supervised vs Unsupervised vs Reinforcement Learning
  - Types of Reinforcement
  - Elements of Reinforcement Learning
  - Real life applications of Reinforcement learning
  - Bellman equation
- Markov's Decision Process
  - Markov property
  - Markov chain/process
  - Markov reward process (MRP)
  - Markov decision process (MDP)
  - Reward, Policy, Value functions
- Q Learning
  - Introduction of Q-Learning
  - Important terms in Q learning
  - table
  - Q functions
  - Q learning algorithm

6.1 Reinforcement Learning

You learnt about reinforcement learning in Unit 1 earlier.

Definition : *Reinforcement machine algorithms improves upon themselves and learn from new situations using a trial-and-error method.*

The favourable outputs are encouraged, or 'reinforced', and non-favourable outputs are discouraged or 'punished'.

The Fig. 6.1.1 outlines how reinforcement learning works at a high-level.

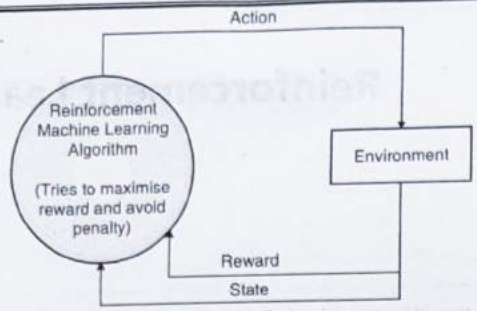


Fig. 6.1.1

Let's extend the discussion further.

6.1.1 Need for Reinforcement Learning

So, why do we need reinforcement learning? Unlike supervised and unsupervised machine learning tasks, reinforcement learning is suited for different kind of learning requirements. The need for reinforcement learning arises from several key factors as following.

1. **Sequential Decision-Making :** Reinforcement learning is designed to tackle problems that involve sequential decision-making, where actions have consequences and outcomes unfold over time. In such scenarios, traditional machine learning approaches that rely on static datasets or supervised learning may not be suitable. Reinforcement learning allows agents to learn and adapt their behaviour based on feedback received from the environment on random as-you-go basis.
2. **Limited or Unavailable Labelled Data :** In many real-world scenarios, obtaining labelled data for training machine learning models can be challenging, time-consuming, or even impossible. Reinforcement learning provides a framework where agents can learn from interactions with the environment, acquiring knowledge and improving performance through trial and error, without relying on explicitly labelled data.
3. **Dynamic and Changing Environments :** Reinforcement learning is particularly useful in dynamic environments where the optimal course of action may change over time. By continuously interacting with the environment, reinforcement learning agents can adapt their policies and strategies to changing conditions, ensuring robust and flexible decision-making.
4. **Exploration and Exploitation Trade-off :** Reinforcement learning enables agents to balance exploration (trying out new actions to gather information) and exploitation (leveraging known knowledge to maximise rewards). This trade-off is critical when the agent needs to explore uncertain or unknown areas of the environment to discover optimal strategies while simultaneously exploiting already learned knowledge for maximising rewards.
5. **Reward Optimisation :** Reinforcement learning is driven by the optimisation of a reward signal. By defining a reward function that quantifies the desirability of different outcomes, agents can learn to maximise this reward over time. This ability is crucial in various domains, including games, robotics, and resource management, where agents aim to achieve specific objectives or goals.
6. **Adaptive and Autonomous Systems :** Reinforcement learning allows systems to adapt and improve their behaviour autonomously. Once trained, reinforcement learning agents can operate independently in their respective environments, continuously learning and updating their policies based on real-time feedback. This autonomy is crucial in domains like robotics, autonomous driving, and intelligent control systems.

6.1.2 Real Life Applications of Reinforcement Learning

Some of the common real life applications of reinforcement learning are as following.

1. **Game Playing :** Reinforcement learning has been instrumental in achieving significant breakthroughs in game playing. For instance, DeepMind's AlphaGo and AlphaZero used reinforcement learning techniques to master the complex games of Go and Chess, respectively, by learning from millions of game simulations. These systems achieved superhuman performance, showcasing the power of reinforcement learning in strategic decision-making.
2. **Robotics :** Reinforcement learning is essential for training robots to perform complex tasks. By providing a reward signal based on task completion or progress, robots can learn to navigate in dynamic environments, manipulate objects, or even perform delicate surgical procedures. Reinforcement learning allows robots to adapt and improve their behaviour over time through trial and error, making them more capable and autonomous.
3. **Autonomous Driving :** Developing self-driving cars requires agents to learn from real-world experiences. Reinforcement learning can train autonomous vehicles to navigate traffic, make decisions at intersections, and respond to various road conditions. By optimising for safety and efficiency, reinforcement learning algorithms enable autonomous vehicles to learn from real-time data and improve their driving skills continually.
4. **Resource Management :** Reinforcement learning is valuable in domains where optimal resource allocation is crucial. For instance, in energy management, reinforcement learning algorithms can learn to control and optimise the operation of renewable energy sources, storage systems, and demand-response mechanisms. This approach maximises energy efficiency, reduces costs, and minimizes environmental impact.
5. **Personalised Recommendations :** Reinforcement learning plays a role in creating personalised recommendation systems. By using reinforcement learning, these systems can adapt their recommendations based on user feedback and engagement. For example, streaming platforms like Netflix or music platforms like Spotify use reinforcement learning to learn users' preferences and suggest relevant movies or songs, enhancing the overall user experience.
6. **Financial Trading :** Reinforcement learning algorithms can be employed to make informed trading decisions in complex financial markets. By learning from historical data and market trends, these algorithms can optimise investment strategies to maximise profits while managing risk.
7. **Healthcare Treatment Optimisation :** Reinforcement learning can assist in optimising treatment plans for patients. By considering various factors such as medical history, symptoms, and response to previous treatments, reinforcement learning algorithms can recommend personalised treatment options that improve patient outcomes.
8. **Natural Language Processing :** Reinforcement learning can enhance natural language processing systems by enabling dialogue agents to learn and improve their conversational abilities. These agents can be trained to engage in meaningful conversations, provide relevant information, and respond effectively to user queries.
9. **Supply Chain Management :** Optimising supply chain operations is a challenging task due to various uncertainties and constraints. Reinforcement learning can be utilised to manage inventory levels, delivery schedules, and logistics by learning from past performance, demand patterns, and market dynamics.
10. **Autonomous Drones :** Reinforcement learning is instrumental in training autonomous drones to perform complex tasks, such as search and rescue operations, package delivery, or monitoring and inspection of infrastructure. These drones can learn to navigate obstacles, plan routes, and adapt to changing environments.

6.1.3 Characteristics (Elements) of Reinforcement Learning

- As you understand, Reinforcement Learning (RL) tells you how to make the best decisions, sequentially, within a context, to maximise a real-life measure of success. The decision-making entity learns this through trial and error. It is not told which decisions to make, but instead it must learn by itself, by trying them.
- At a high-level, there are four components of RL – Agent, Action, Environment, and Reward. The Fig. 6.1.2 illustrates what they are.

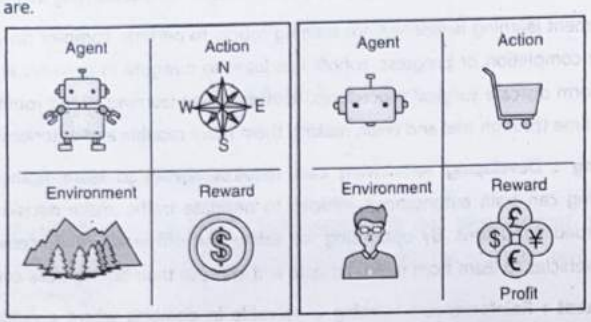


Fig. 6.1.2

1. Agent

- An agent is someone or something that interacts with the environment by executing certain actions, making observations, and receiving eventual rewards for this. The agent is the entity that makes decisions. In most practical RL scenarios, the agent is the piece of computer software or program that is supposed to solve some problem in a more-or-less efficient way.
- In real-life scenarios, some of the examples of agents could be
  - **Financial trading :** A trading system or a trader making decisions about order execution
  - **Chess :** A player or a computer program
  - **Dopamine system :** The brain itself, which, according to sensory data, decides whether it was a good experience
  - **Computer games :** The player who enjoys the game or the computer program
  - **Machine operator :** The person or program that operates the machine optimally
  - **Robotic arms :** The program that appropriates handles / works with various objects

2. Action

- Actions are things that an agent can do in the environment. Each decision is an action. For example, when you ride a cycle, the various actions are steering directions, how much to pedal, and where to brake or slow down. Similar, when you go out shopping, then you decide which products to add and in how much quantity to your cart.
- In RL, there are two types of actions discrete or continuous. Discrete actions form the finite set of mutually exclusive things an agent can do, such as move left or right. Continuous actions have some value attached to them, such as a car's action to turn the wheel having an angle and direction of steering. Different angles could lead to a different scenario a second later, isn't it? So, just turn the wheel is definitely not enough!

3. Environment

- The environment is everything outside of an agent on which the agent has no control over as such. The agent usually works in an operating environment. For example, a robotic arm may be placed near a car assembly and may be purposed to lift and shift spare parts. The agent's communication with the environment is limited to reward (obtained from the environment), actions (executed by the agent in the given environment), and observations (some information besides the reward that the agent receives from the environment).
- Imagine you are training a robot to pick up objects. The objects to be picked up, the tray where the objects lay, the wind, and everything outside the decision maker are part of the environment. That means the robot arm is also part of the environment because it isn't part of the agent. Even though the agent can decide to move the arm, the actual arm movement is noisy, and thus the arm is part of the environment. The Fig. 6.1.3 helps you to understand the difference between agent and environment.

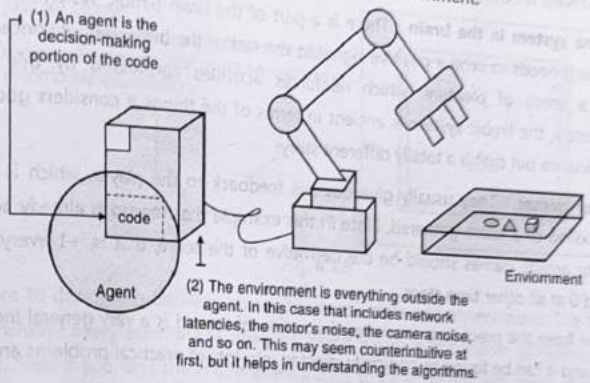


Fig. 6.1.3

- The environment is represented by a set of variables related to the problem. For instance, in the robotic arm example, the location and velocities of the arm would be part of the variables that make up the environment. This set of variables and all the possible values that they can take are referred to as the state space. A state is an instantiation of the state space, a set of values the variables take.

4. Reward

- The reward encodes the challenge. This feedback mechanism tells the agent which actions led to success (or failure). The reward signal is typically numeric but only needs to reinforce the desired behaviour. For example, you could reward a robot for reaching a goal or the agent for adding the right product to a basket. Simple, right? But what if the robot takes three days to solve a simple maze because it spends most of the time spinning in circles? Or what if the agent starts adding all products to the basket? You need to appropriately build a reward system to reinforce only the positive behaviour.
- Reward can be positive or negative, large or small, but it is just a number. The purpose of reward is to tell the agent how well it has behaved. You don't define how frequently the agent receives this reward; it can be every second or once in an agent's lifetime, although it's common practice to receive rewards every fixed timestamp or at every environment interaction, just for convenience. In the case of once-in-a-lifetime reward systems, all rewards except the last one will be zero.

- As I stated, the purpose of reward is to give an agent feedback about its success, and it is the main idea behind RL. Basically, the term reinforcement comes from the fact that reward obtained by an agent should reinforce its behaviour in a positive or negative way. Reward is local, meaning that it reflects the success of the agent's recent activity and not all the successes achieved by the agent so far. Of course, getting a large reward for some action doesn't mean that a second later you won't face dramatic consequences as a result of your previous decisions. It's like robbing a bank it could look like a good idea until you think about the consequences.
- What an agent is trying to achieve is the largest accumulated reward over its sequence of actions. To give you a better understanding of reward, here is a list of some concrete examples with their rewards.
  - **Financial trading** : An amount of profit is a reward for a trader buying and selling stocks.
  - **Chess :** Reward is obtained at the end of the game as a win, lose, or draw.
  - **Dopamine system in the brain :** There is a part of the brain (limbic system) that produces dopamine every time it needs to send a positive signal to the rest of the brain. Higher concentrations of dopamine lead to a sense of pleasure, which reinforces activities considered by this system to be good. Unfortunately, the limbic system is ancient in terms of the things it considers good food, reproduction, and dominance but that is a totally different story!
  - **Computer games :** They usually give obvious feedback to the player, which is either the number of enemies killed or a score gathered. Note in this example that reward is already accumulated, so the RL reward for arcade games should be the derivative of the score, that is, +1 every time a new enemy is killed and 0 at all other time steps.
- As you can see from the preceding examples, the notion of reward is a very general indication of the agent's performance, and it can be found or artificially injected into lots of practical problems around us.

5. Observation

- Interestingly, often, agents don't have access to the actual full state of the environment. The part of a state that the agent can observe is called an Observation. Observations are pieces of information that the environment provides the agent with that say what is going on around the agent. The Fig. 6.1.4 illustrates the concept of observation.

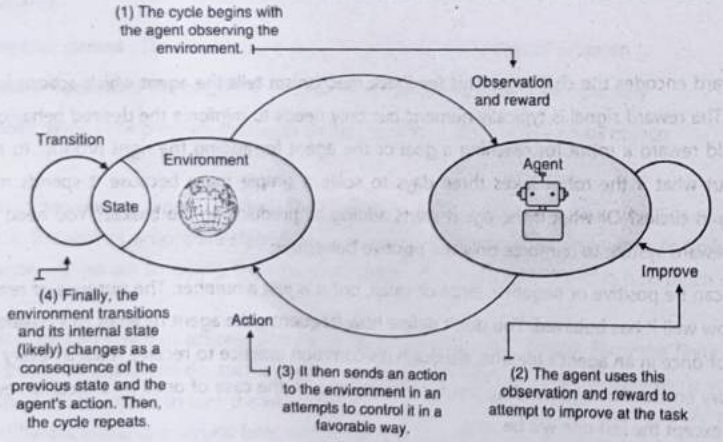


Fig. 6.1.4

- Observations depend on states but are what the agent can see. For instance, in the robotic arm example, the agent may only have access to camera images. While an exact location of each object exists, the agent doesn't have access to this specific state. Instead, the observations the agent perceives are derived from the states.
- The Fig. 6.1.5 illustrates the difference between state and observation.

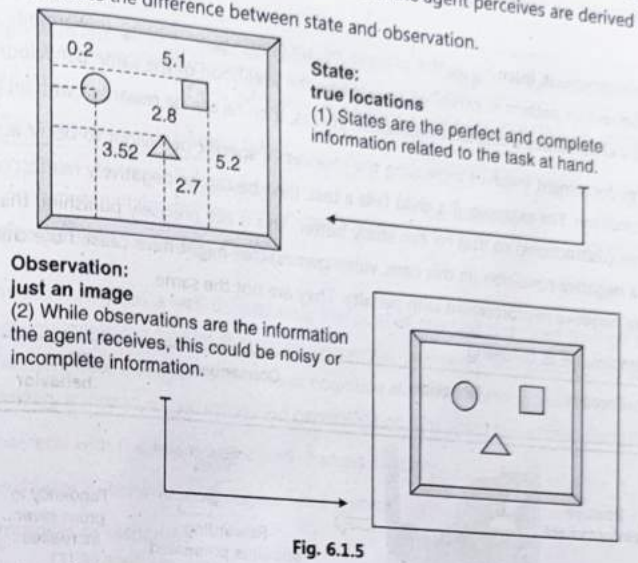


Fig. 6.1.5

- It is important to distinguish between an environment's state and observations. The state of an environment potentially includes every atom in the universe, which makes it impossible to measure everything about the environment. Even if you limit the environment's state to be small enough, most of the time, it will be either not possible to get full information about it or your measurements will contain noise. This is completely fine, though, and RL was created to support such cases. A few real-life examples of observations are as following.
  - **Financial trading :** Here, the environment is the whole financial market and everything that influences it. This is a huge list of things, such as the latest news, economic and political conditions, weather, food supplies, and Twitter trends. However, your observations are limited to stock prices, news, and so on. You don't have access to most of the environment's state, which makes trading such a nontrivial thing.
  - **Chess :** The environment here is your board plus your opponent, which includes their chess skills, mood, brain state, chosen tactics, and so on. Observations are what you see (your current chess position), but, at some levels of play, knowledge of psychology and the ability to read an opponent's mood could increase your chances.
  - **Dopamine system :** The environment here is your brain plus your nervous system and your organs' states plus the whole world you can perceive. Observations are the inner brain state and signals coming from your senses.
  - **Computer game :** Here, the environment is your computer's state, including all memory and disk data. For networked games, you need to include other computers plus all Internet infrastructure between them and your machine. Observations are a screen's pixels and sound only.

6.1.4 Positive vs Negative Reinforcement Learning (Types of Reinforcement)

- So, as you understand, an agent's reinforcement learning could be based off positive reinforcement or negative reinforcement.
  - Positive reinforcement learning is the process of encouraging or adding (rewarding) something when an expected behaviour pattern is exhibited to increase the likelihood of the same behaviour being repeated. For example, if a child passes a test with impressive grades, then he can be rewarded with an ice cream.
  - Negative reinforcement involves increasing the chances of specific behaviour to occur again by removing the negative condition. For example, if a child fails a test, then he can be negatively reinforced by taking away his video games (distractions) so that he can study better. This is not precisely punishing the child for failing, but removing a negative condition (in this case, video games) that might have caused the child to fail the test. Do not confuse negative reinforcement with penalty. They are not the same.
- A few other examples are as following.

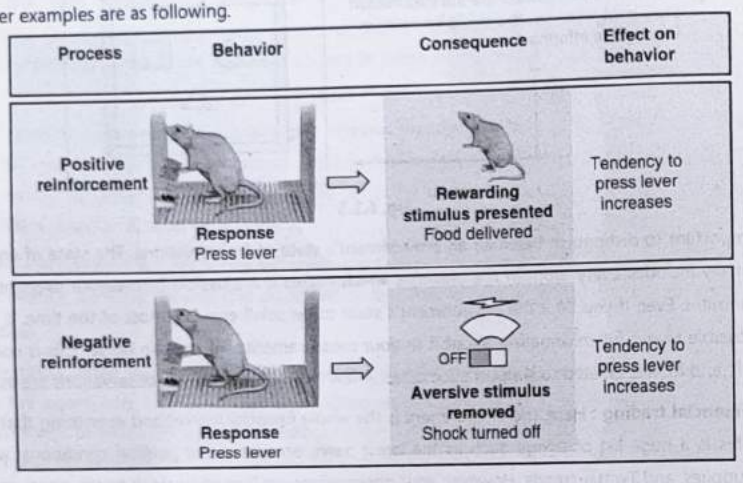


Fig. 6.1.6

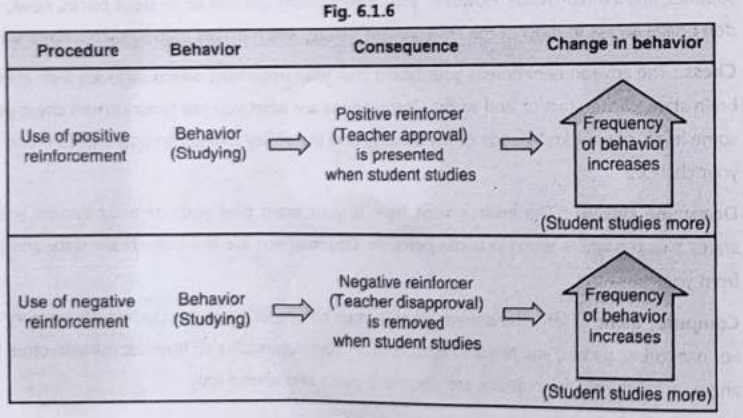


Fig. 6.1.7

- So, as you notice, negative reinforcement takes away "bad occurrences or experiences" instead of adding "good or positive experiences". Overall the agent is "better". Negative reinforcement is same as you wearing a seat belt to avoid fatal injury. The Table 6.1.1 summarises the key difference between positive and negative reinforcement.

Table 6.1.1

Comparison Attribute	Positive Reinforcement	Negative Reinforcement
Stimulus	Favourable stimulus added	Unfavourable stimulus removed
Good Behaviour	Strengthens	Strengthens
Practised	Commonly	Occasionally

6.1.5 The Reinforcement Learning Cycle

- The environment commonly has a well-defined task. The goal of this task is defined through the reward function. The reward-function signals can be simultaneously sequential, evaluative, and sampled. To achieve the goal, the agent needs to demonstrate intelligence, or at least cognitive abilities commonly associated with intelligence, such as long-term thinking, information gathering, and generalisation. The agent has a three-step process.
  1. The agent interacts with the environment (and changes its state)
  2. The agent evaluates its behaviour (through reward process)
  3. The agent improves its responses

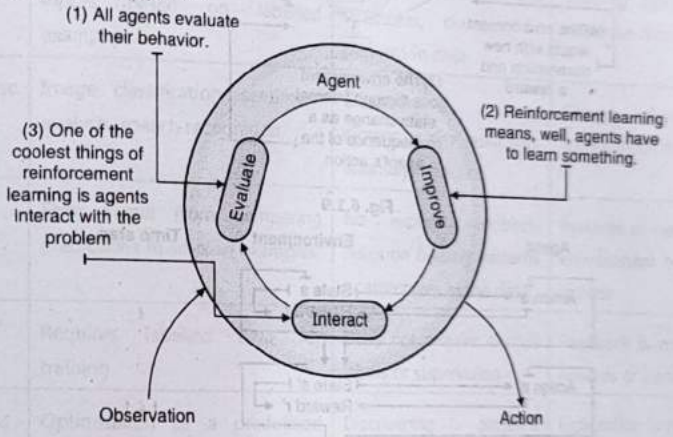


Fig. 6.1.8

- **Policy :** The agent can be designed to learn mappings from observations to actions called policies. You can think of policies as the strategy that the agent uses to determine the next action based on the current state (observation). In other words, it maps states (observations) to actions so that the agent can choose the action with the highest reward.
- **Model :** The agent can be designed to learn the model of the environment on mappings called models. Model is the agent's view of the environment. It maps the state-action pairs to the probability distributions over states. However, not every RL agent uses a model of its environment.

- **Value Functions :** The agent can be designed to learn to estimate the reward-to-go on mappings called value functions. In simple terms, the value function represents how favourable a state is for the agent. The state's value represents the long-term reward the agent will receive starting from that particular state to executing a specific policy.

- The interactions between the agent and the environment go on for several cycles. Each cycle is called a time step. At each time step, the agent observes the environment, takes action, and receives a new observation and reward. The set of the state, the action, the reward, and the new state is called an experience. Every experience has an opportunity for learning and improving performance.

- The Fig. 6.1.9 illustrate the agent's interaction with the environment and the reward process.

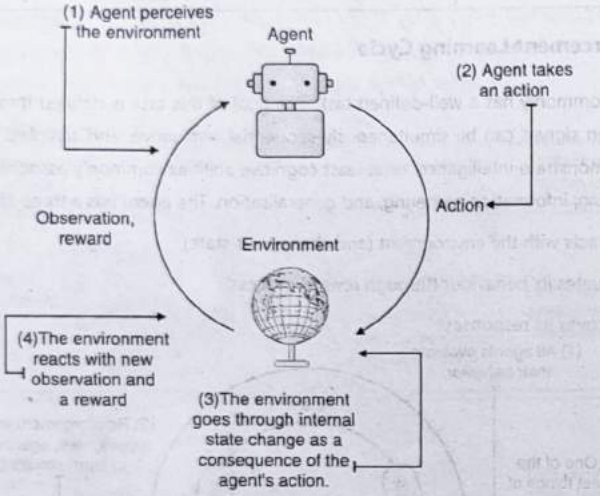


Fig. 6.1.9

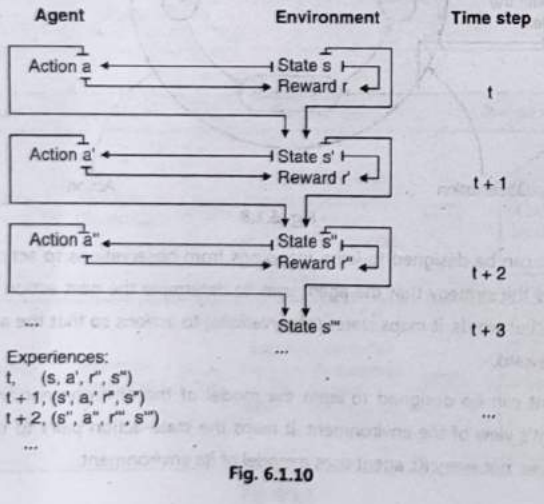


Fig. 6.1.10

- The task the agent is trying to solve may or may not have a natural ending. Tasks that have a natural ending, such as a game, are called episodic tasks. Conversely, tasks that don't end are called continuing tasks, such as learning forward motion. The sequence of time steps from the beginning to the end of an episodic task is called an episode. Agents may take several time steps and episodes to learn to solve a task. Agents learn through trial and error: they try something, observe, learn, try something else, and so on.

6.1.6 Supervised vs Unsupervised vs Reinforcement Learning

While there can be overlaps between the various machine learning approaches, the Table 6.1.2 provides a quick comparison between the various approaches.

Table 6.1.2

Comparison Attributes	Supervised Learning	Unsupervised Learning	Reinforcement Learning
Training Data	Labelled data	Unlabelled data	Feedback from environment
Learning Approach	Learning from labelled examples with input-output pairs	Discovering patterns and relationships in unlabelled data	Learning from interactions with the environment to maximise rewards
Objective	Predicting or classifying new inputs based on labelled examples	Finding hidden structures, clusters, or patterns in data	Learning optimal actions to maximise cumulative rewards
Example Use Cases	Image classification, sentiment analysis, speech recognition	Clustering, dimensionality reduction, anomaly detection	Game playing, robotics, autonomous driving
Feedback	Error signal from comparing predictions to labelled examples	No explicit feedback, relies on finding patterns or structures in the data	Rewards or penalties from the environment based on agent's actions
Supervision	Requires labelled data for training	Does not require explicit labels or supervision	Feedback is provided through rewards or penalties
Training Process	Optimisation of a predefined objective function	Discovering patterns without a predefined objective	Exploration and exploitation to find optimal actions based on rewards
Evaluation Metrics	Accuracy, precision, recall, F1-score	Quality of clustering, reconstruction error, perplexity	Cumulative rewards, success rate, convergence rate
Generalisation	Able to generalise well to unseen data with labelled examples	Generalisation is challenging as there is no explicit supervision	Generalises well to similar but unseen environments based on reinforcement history

6.2 Markov Models

Markov Models is one of the most commonly used techniques in machine learning. It is commonly used for speech analysis, facial recognition, speech tagging, and gesture recognition. Let's dive deeper into understanding it.

Markov Process

✍ **Definition :** *A Markov process is a chain of events that is memoryless.*

- It means that what is going to happen next depends only on the current state and not states previous to the current state.

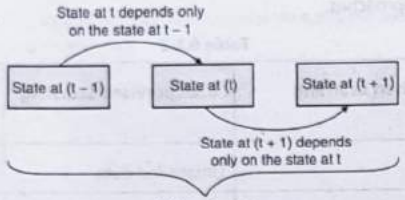


Fig. 6.2.1 : Markov Process

- For example, you decide your next move on a chess board based on the current state of the board and not based on the previous states of the board. Another example could be what you eat in your current meal depends on what you ate in your previous meal. So, if your previous meal was heavy, probably you would like to eat something light as the next meal.

Markov Assumptions

Markov processes (and models based on these processes) make the following assumptions.

1. The number of possible outcomes or states is finite.
2. The outcome at any stage depends only on the outcome of the previous stage.
3. The probabilities of transitioning from one state to another state are constant over time.

Markov Chain

✍ ***Definition :*** *A Markov chain is a mathematical model that represents the state transitioning probabilities of a Markov process.*

- As you understand, the next state of a Markov process depends only upon the current state. Markov chain assigns state transitioning probabilities for a Markov process.
- For example, the Fig. 6.2.2 illustrates a simple Markov chain for weather.

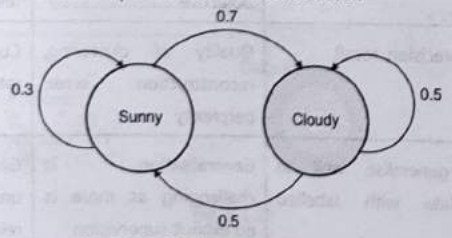


Fig. 6.2.2 : Markov chain for weather

- Assume that weather could only have two states – sunny or cloudy. Markov chain represents the state transitioning probabilities between the states.
- So, for the given Markov chain,
  - Probability of sunny remaining sunny = P (Sunny | Sunny) = 0.3
  - Probability of sunny becoming cloudy = P (Cloudy | Sunny) = 0.7
  - Probability of cloudy remaining cloudy = P (Cloudy | Cloudy) = 0.5
  - Probability of cloudy becoming sunny = P (Sunny | Cloudy) = 0.5
- As you know, note that the sum of probabilities of all outgoing arrows from a state should be 1. It cannot be higher than 1 (sure to happen).
- You can represent the four state transitioning probabilities as a matrix.

	To Sunny	To Cloudy	
From Sunny	0.3	0.7	= 1
From Cloudy	0.5	0.5	= 1

State Transitioning probabilities = $\begin{bmatrix} 0.3 & 0.7 \\ 0.5 & 0.5 \end{bmatrix}$

Initial State

- Initial state or the starting state just represents the beginning of a Markov process. For example, the way you arrange your chess board at first remains the same and then once the game proceeds, you can have various states of the board.
- Extending the previous example from Markov chain, assume that the current day is sunny. So, what is the probability of the next day to be sunny or cloudy?

Let's solve it out.

Practice Questions

Ex. 6.2.1 : For the given Markov chain, predict the respective probabilities of weather for next 3 days assuming that the current day is Sunny.

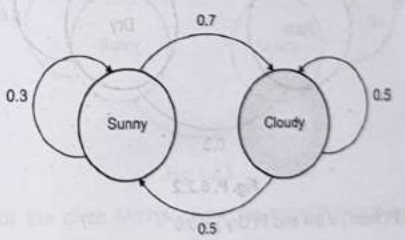


Fig. P. 6.2.1

Soln. :

- For the given Markov chain,
  - Probability of sunny remaining sunny = P (Sunny | Sunny) = 0.3

- Probability of sunny becoming cloudy = P (Cloudy | Sunny) = 0.7
- Probability of cloudy remaining cloudy = P (Cloudy | Cloudy) = 0.5
- Probability of cloudy becoming sunny = P (Sunny | Cloudy) = 0.5

State Transitioning Probabilities = $\begin{bmatrix} 0.3 & 0.7 \\ 0.5 & 0.5 \end{bmatrix}$

- Given that the current day is Sunny and not cloudy. So, it can be written as [1 0], where 1 is the probability of it being a sunny day (given in the question) and 0 is the probability of it being a cloudy day.
- As per Markov process, the next state depends only upon the current state P ($Weather_t$ | $Weather_{t-1}$).
- Next state probability could be calculated by multiplying state transition matrix with the current state.

$$P(W_1 \mid W_0) = [1 \quad 0] \times \begin{bmatrix} 0.3 & 0.7 \\ 0.5 & 0.5 \end{bmatrix} = [0.3 \quad 0.7]$$

- So, given that today is sunny, there is 0.3 probability that the next day would be sunny and 0.7 probability that the next day would be cloudy. This is also evident from the given Markov chain.
- Let's calculate the probabilities for the next two days.

$$P(W_2 \mid W_1) = [0.3 \quad 0.7] \times \begin{bmatrix} 0.3 & 0.7 \\ 0.5 & 0.5 \end{bmatrix} = [0.44 \quad 0.56]$$

$$P(W_3 \mid W_2) = [0.44 \quad 0.56] \times \begin{bmatrix} 0.3 & 0.7 \\ 0.5 & 0.5 \end{bmatrix} = [0.41 \quad 0.59]$$

Hence, the weather probabilities for next 3 days are as following.

	Day 1	Day 2	Day 3
Sunny	0.3	0.44	0.41
Cloudy	0.7	0.56	0.59

Ex. 6.2.2 : Consider Markov chain model for 'Rain' and 'Dry' is shown in following Fig. P. 6.2.2.

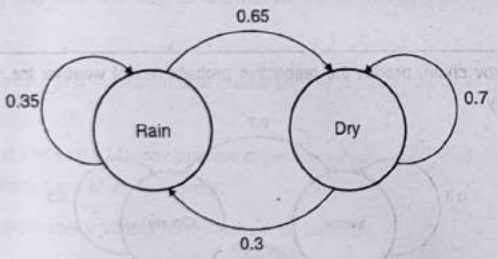


Fig. P. 6.2.2

Initial Probabilities are P('Rain') = 0.4 and P('Dry') = 0.6

Calculate the probability for a sequence of states {'Dry', 'Rain', 'Rain', 'Dry'}

Soln. :

For the given Markov chain,

$$P(\text{Rain} \mid \text{Rain}) = 0.35$$

$$P(\text{Dry} \mid \text{Rain}) = 0.65$$

$$P(\text{Rain} \mid \text{Dry}) = 0.3$$

$$P(\text{Dry} \mid \text{Dry}) = 0.7$$

State Transitioning Probabilities = $\begin{bmatrix} 0.35 & 0.65 \\ 0.3 & 0.7 \end{bmatrix}$

Initial State = [0.4 0.6]

$$P(W_1 \mid W_0) = [0.4 \quad 0.6] \times \begin{bmatrix} 0.35 & 0.65 \\ 0.3 & 0.7 \end{bmatrix} = [0.32 \quad 0.68]$$

So, for state 1, P_1 (Rain) = 0.32 and P_1 (Dry) = 0.68

$$P(W_2 \mid W_1) = [0.32 \quad 0.68] \times \begin{bmatrix} 0.35 & 0.65 \\ 0.3 & 0.7 \end{bmatrix} = [0.316 \quad 0.684]$$

So, for state 2, P_2 (Rain) = 0.316 and P_2 (Dry) = 0.684

$$P(W_3 \mid W_2) = [0.316 \quad 0.684] \times \begin{bmatrix} 0.35 & 0.65 \\ 0.3 & 0.7 \end{bmatrix} = [0.3158 \quad 0.6842]$$

So, for state 3, P_3 (Rain) = 0.3158 and P_3 (Dry) = 0.6842

$$P(W_4 \mid W_3) = [0.3158 \quad 0.6842] \times \begin{bmatrix} 0.35 & 0.65 \\ 0.3 & 0.7 \end{bmatrix} = [0.316 \quad 0.684]$$

So, for state 4, P_4 (Rain) = 0.316 and P_4 (Dry) = 0.684

You need probability for state transition as {'Dry', 'Rain', 'Rain', 'Dry'}. Multiply the respective probability of the 4 states.

$$P(\text{'Dry', 'Rain', 'Rain', 'Dry'}) = P_1(\text{Dry}) \times P_2(\text{Rain}) \times P_3(\text{Rain}) \times P_4(\text{Dry})$$

$$= 0.68 \times 0.316 \times 0.3158 \times 0.684 = 0.05$$

6.2.1 Steady State

- Markov chains are also useful for determining the chances of reaching a steady state where the probabilities for reaching a particular state is more or less the same over a longer period of time. It helps you to determine and answer questions like what percentage of times a particular state would be reached.
- Let's continue the previous example.

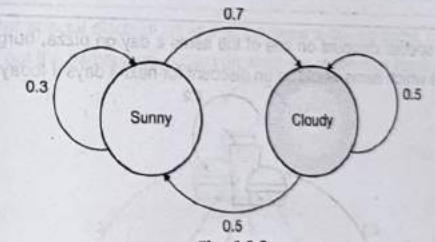


Fig. 6.2.3

- Earlier you found out that for the given Markov chain, the respective probabilities for various states are as following.

	Day 1	Day 2	Day 3
Sunny	0.3	0.44	0.41
Cloudy	0.7	0.56	0.59

- If you keep on doing this exercise, you will find that the probabilities would more or less remain steady for days after days. That is when you say that you have reached the steady state (or say known routine or schedule). Let's call those probabilities be π_1 probability of being sunny in the steady state) and π_2 (probability of being cloudy in the steady state).

  So,

	Day 1	Day 2	Day 3	Day...	Day n
Sunny	0.3	0.44	0.41	...	π_1
Cloudy	0.7	0.56	0.59	...	π_2

- Assume that you want to calculate the steady state probability that the next day would be sunny.

  You can write it as

$$\pi_1 = \pi_1 \times 0.3 + \pi_2 \times 0.5$$

[as 0.3 is the probability of a sunny day remaining sunny and 0.5 is the probability of a cloudy day becoming sunny the next day].

Also, $\pi_1 + \pi_2 = 1$ [as sum of probabilities is 1]

$$\pi_2 = 1 - \pi_1$$

- Put the value of π_2 in the first equation you get,

$$\pi_1 = \pi_1 \times 0.3 + (1 - \pi_1) \times 0.5$$

$$\pi_1 = \frac{5}{12} = 0.42$$

Hence, $\pi_2 = \frac{7}{12} = 0.58$

- So, in the steady state there is 0.42 (or 42%) chance that the day would be sunny and 0.58 (58%) chance that the day would be cloudy. You could also say that 42% of all days would be sunny and 58% of all days would be cloudy. Very powerful derivation, isn't it?

Practice Questions

Ex. 6.2.3 : A restaurant provides special discount on one of the items a day on pizza, burger, and hotdog as per the given Markov chain. Find out which items would be on discount for next 3 days if today the discount is on pizza.

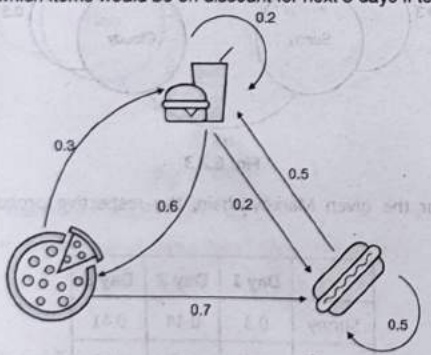


Fig. P. 6.2.3

Soln. : First, arrange the state transition probabilities as a matrix.

States	Burger	Pizza	Hotdog
Burger	0.2	0.6	0.2
Pizza	0.3	0	0.7
Hotdog	0.5	0	0.5

State Transitioning Probabilities $= \begin{bmatrix} 0.2 & 0.6 & 0.2 \\ 0.3 & 0 & 0.7 \\ 0.5 & 0 & 0.5 \end{bmatrix}$

Today, the discount is on Pizza.

Hence, Initial state = [0 1 0]

Let's calculate the probabilities for next 3 days.

$$P(D_1 \mid D_0) = [0 \quad 1 \quad 0] \times \begin{bmatrix} 0.2 & 0.6 & 0.2 \\ 0.3 & 0 & 0.7 \\ 0.5 & 0 & 0.5 \end{bmatrix} = [0.3 \quad 0 \quad 0.7]$$

$$P(D_2 \mid D_1) = [0.3 \quad 0 \quad 0.7] \times \begin{bmatrix} 0.2 & 0.6 & 0.2 \\ 0.3 & 0 & 0.7 \\ 0.5 & 0 & 0.5 \end{bmatrix} = [0.41 \quad 0.18 \quad 0.41]$$

$$P(D_3 \mid D_2) = [0.41 \quad 0.18 \quad 0.41] \times \begin{bmatrix} 0.2 & 0.6 & 0.2 \\ 0.3 & 0 & 0.7 \\ 0.5 & 0 & 0.5 \end{bmatrix} = [0.34 \quad 0.25 \quad 0.41]$$

Hence, the probabilities for discounts on respective items for next 3 days are as following.

	Day 1	Day 2	Day 3
Burger	0.3	0.41	0.34
Pizza	0	0.18	0.25
Hotdog	0.7	0.41	0.41

Ex. 6.2.4 : A restaurant provides special discount on one of the items a day on pizza, burger, and hotdog as per the given Markov chain. Find out the percentage of days on which the respective items are on discount.

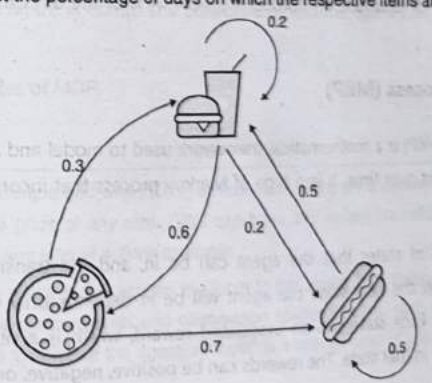


Fig. P. 6.2.4

Soln. :

Assume that :

π_B = Steady state probability for discount on burger

π_P = Steady state probability for discount on pizza

π_H = Steady state probability for discount on hotdog

You can write steady state equations as following.

$$\pi_B = 0.2\pi_B + 0.3\pi_P + 0.5\pi_H$$

$$\pi_P = 0.6\pi_B$$

$$\pi_H = 0.2\pi_B + 0.7\pi_P + 0.5\pi_H$$

$$\pi_B + \pi_P + \pi_H = 1$$

Solve these linear equations.

$$\pi_B = 0.2\pi_B + 0.3\,(0.6\pi_B) + 0.5\,(1 - \pi_B - \pi_P)$$

$$= 0.2\pi_B + 0.18\pi_B + 0.5 - 0.5\pi_B - 0.5 \times 0.6\pi_B$$

$$\pi_B = 0.5 - 0.42\pi_B$$

$$\pi_B = 0.35$$

$$\pi_P = 0.6\pi_B$$

$$= 0.6 \times 0.35 = 0.21$$

$$\pi_H = 1 - \pi_B - \pi_P$$

$$= 1 - 0.35 - 0.21 = 0.44$$

Hence,

- The steady state probability for discount on burger = 0.35 = 35% of the days the restaurant is likely to provide discount on burger
- The steady state probability for discount on pizza = 0.21 = 21% of the days the restaurant is likely to provide discount on pizza
- The steady state probability for discount on hotdog = 0.44 = 44% of the days the restaurant is likely to provide discount on hotdog

6.2.2 Markov Reward Process (MRP)

- A Markov Reward Process (MRP) is a mathematical framework used to model and analyse systems where an agent interacts with an environment over time. It is a type of Markov process that incorporates rewards associated with each state transition.
- In an MRP, you have a set of states that the agent can be in, and the transitions between these states are probabilistic. This means that the next state the agent will be in depends only on its current state and has no memory of previous states. Each state has an associated reward, which is a numerical value representing the desirability or utility of being in that state. The rewards can be positive, negative, or zero.
- The Fig. 6.2.3 illustrates MRP. As you see, rewards are assigned to different states in a Markov chain.

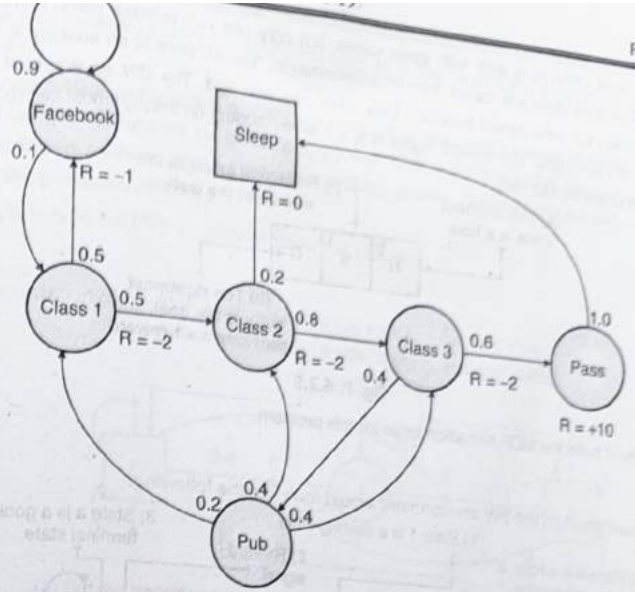


Fig. 6.2.3

6.2.3 Markov Decision Process (MDP)

- In a Markov decision process, the transition model describes the dynamics of the environment. The next state and reward depend only on the previous state and the action. The only difference between an MDP and a traditional Markov chain, which is also called a Markov process, is the addition of the action and reward parameters. The action is a decision, which is the "D" in MDP.
- So, the transition model of a MDP can be given as p (s′, r | s, a) which is read as the probability that the agent leaves the environment in state s′ (after carrying out the action) and gets a reward r given the previous state s and action a.
- You can think of MDP as an interface. It is a contract between the agent and the environment. The agent can only observe the states and suggest an action. In return, the environment gives the agent a new state and some value that represents how well the agent is doing. The environment and the agent can be as complex as they needs to be.
- Let's see a few simple examples of MDP.

Practice Question

Ex. 6.2.5 : Bandit Walk (BW) is a simple grid-world (GW) environment. GWs are a common type of environment for studying RL algorithms that are grids of any size. GWs can have any model (transition and reward functions) you can think of and can make any kind of actions available.

But they all commonly make move actions available to the agent: Left, Down, Right, Up. Also, they all tend to have a fully observable discrete state and observation spaces (that is, state equals observation) with integers representing the cell id location of the agent. A "walk" is a special case of grid-world environments with a single row.

The bandit walk (BW) is a walk with three states, but only one non-terminal state. Environments that have a single non-terminal state are called "bandit" environments. "Bandit" here is an analogy to slot machines, which are also known as "one-armed bandits"; they have one arm and. The BW environment has just two actions available – left and right. The reward signal is a + 1 when landing on the rightmost cell (goal), 0 otherwise. The agent starts in the middle cell.

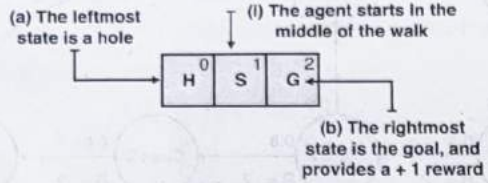


Fig. P. 6.2.5

Your aim is to build the MDP transition table for this problem.

Soln. :

A graphical representation of the BW environment would look like the following.

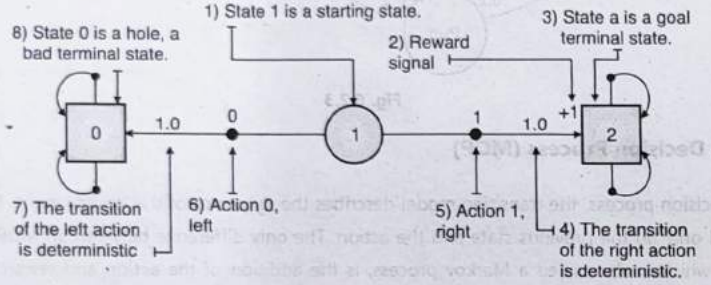


Fig. P. 6.2.5(a)

As you see, the agent can move left or right. The reward signal of +1 is provided only then the agent reaches the goal (state 2). Also, once the agent reaches the terminal state (state 0 or state 2), no further transitions are possible and hence the reward continues to be 0.

Let's put this information in a MDP transition table.

State	Action	Next state	Transition probability	Reward signal
0 (Hole)	0 (Left)	0 (Hole)	1.0	0
0 (Hole)	1 (Right)	0 (Hole)	1.0	0
1 (Start)	0 (Left)	0 (Hole)	1.0	0
1 (Start)	1 (Right)	2 (Goal)	1.0	+ 1
2 (Goal)	0 (Left)	2 (Goal)	1.0	0
2 (Goal)	1 (Right)	2 (Goal)	1.0	0

Simple, isn't it?

Ex. 6.2.6 : In the previous example, transition probabilities were all 1. But, what if the transition probabilities are not 1 and are random?

Let's say the surface of the walk is slippery and each action has a 20% chance of sending the agent backwards. You can call this environment the bandit slippery walk (BSW). BSW is still a one-row-grid world with only Left and Right actions available. Again, three states and two actions. The reward is the same as before that is +1 for reaching the goal and 0 otherwise.

Your aim is to build the MDP transition table for this problem.

Soln. :

A graphical representation of the BSW environment would look like the following.

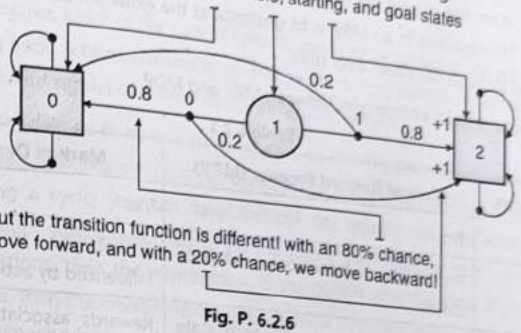


Fig. P. 6.2.6

As you see, the agent can move left or right. The reward signal of +1 is provided only then the agent reaches the goal (state 2). Also, once the agent reaches the terminal state (state 0 or state 2), no further transitions are possible and hence the reward continues to be 0. Also, notice that transition function has randomness now. Each action has a 20% chance of sending the agent backwards which is to say that even if the agent took left from state 1, there is 20% chance that it would reach goal.

Let's put this information in a MDP transition table.

State	Action	Next state	Transition probability	Reward signal
0 (Hole)	0 (Left)	0 (Hole)	1.0	0
0 (Hole)	1 (Right)	0 (Hole)	1.0	0
1 (Start)	0 (Left)	0 (Hole)	0.8	0
1 (Start)	0 (Left)	2 (Goal)	0.2	+1
1 (Start)	1 (Right)	2 (Goal)	0.8	+1
1 (Start)	1 (Right)	0 (Hole)	0.2	0
2 (Goal)	0 (Left)	2 (Goal)	1.0	0
2 (Goal)	1 (Right)	2 (Goal)	1.0	0

6.2.4 Difference between Markov Reward Process (MRP) and Markov Decision Process (MDP)

- The concept of "Markov" in both MRP and MDP refers to the Markov property, which states that the future is independent of the past given the present. In other words, in both processes, the probability distribution of transitioning to the next state only depends on the current state.
- The key difference between MRP and MDP lies in the presence of actions and decision-making. In an MDP, the agent can choose actions at each state, which can influence the next state and the resulting reward. The goal in an MDP is to find an optimal policy, which is a set of actions that maximises the cumulative rewards over time.
- In contrast, an MRP does not involve decision-making or actions. It focuses solely on modelling the dynamics of states and rewards. The objective in an MRP is to understand the expected cumulative rewards that an agent will receive while moving through the states over time.
- The Table 6.2.1 provides a quick comparison between MRP and MDP.

Table 6.2.1

Comparison Attributes	Markov Reward Process (MRP)	Markov Decision Process (MDP)
Involves Actions	No	Yes
Transitions	Probabilistic transitions between states	Probabilistic transitions between states, influenced by actions
Rewards	Rewards associated with each state transition	Rewards associated with each state-action transition
Involves Decision-making	No	Yes
Objective	Understand expected cumulative rewards over time	Find optimal policy to maximise cumulative rewards
Gives Optimal Policy	No	Yes
Application	Modelling and analysis of state dynamics and rewards	Reinforcement learning, decision-making problems

6.3 Reinforcement Learning Algorithms

You learnt about policy, models, and value functions earlier.

As a refresher,

- **Policy :** The agent can be designed to learn mappings from observations to actions called policies. You can think of policies as the strategy that the agent uses to determine the next action based on the current state (observation). In other words, it maps states (observations) to actions so that the agent can choose the action with the highest reward.
- **Model :** The agent can be designed to learn the model of the environment on mappings called models. Model is the agent's view of the environment. It maps the state-action pairs to the probability distributions over states. However, not every RL agent uses a model of its environment.

- **Value Functions :** The agent can be designed to learn to estimate the **reward-to-go** on mappings called value functions. In simple terms, the value function represents how favourable a state is for the agent. The state's value represents the long-term reward the agent will receive starting from that particular state to executing a specific policy.

There are various reinforcement learning algorithms that train the agent based on policies, models, and value functions respectively. Let's learn about them.

6.3.1 Mathematical Foundation for Reinforcement Learning Algorithms

- Before you learn about various reinforcement learning algorithms, let's learn about a few foundational mathematical concepts behind these algorithms.
- RL problems have an objective, which is the sum of rewards received by an agent. An agent's goal is to maximise the objective by selecting good actions. It learns to do this by interacting with the environment in a process of trial and error, and uses the reward signals it receives to reinforce good actions.
- Agent and environment are defined to be mutually exclusive, so that the boundaries between the exchange of the state, action, and reward are unambiguous. You can consider the environment to be anything that is not the agent. For example, when riding a cycle, you can have multiple but equally valid definitions of an agent and an environment. If you consider your entire body to be the agent that observes your surroundings and produces muscle movements as actions, then the environment is the bicycle and the road. If you consider your mental processes to be the agent, then the environment is your physical body, the cycle, and the road, with actions being the neural signals sent from your brain to the muscles and states being the sensory inputs sent back to your brain. Essentially, a reinforcement learning system is a feedback control loop where an agent and an environment interact and exchange signals, while the agent tries to maximise the objective.
  - The signals exchanged are (s_t, a_t, r_t), which stand for state, action, and reward, respectively, and t denotes the time step in which these signals occurred.
  - The (s_t, a_t, r_t) tuple is called an experience.
  - The control loop can repeat forever or terminate by reaching either a terminal state or a maximum time step t = T.
  - The time horizon from t = 0 to when the environment terminates is called an episode.
  - A trajectory is a sequence of experiences over an episode, $\tau = (s_0, a_0, r_0), (s_1, a_1, r_1), \ldots, (s_T, a_T, r_T)$. An agent typically needs many episodes to learn a good policy, ranging from hundreds to millions depending on the complexity of the problem.
  - So, $s_t \in S$ is the state, S is the state space. The state space S is the set of all possible states in an environment. Depending on the environment, it can be defined in many different ways as integers, real numbers, vectors, matrices, structured or unstructured data.
  - $a_t \in A$ is the action, A is the action space. The action space A is the set of all possible actions defined by an environment. It can also take many forms but is commonly defined as either a scalar or a vector.
  - $r_t \in R\ (s_t, a_t, s_{t+1})$ is the reward R is the reward function. The reward function assigns a positive, negative, or zero scalar to each transition (s_t, a_t, s_{t+1}). The state space, action space, and reward function are specified by the environment. Together, they define the (s, a, r) tuples which are the basic unit of information describing a reinforcement learning system.

- The agent tries to maximise the objective, which is the sum of all rewards received by an agent over time. Let's call it return and denote it by R. So, return over a trajectory τ is given as the following.

$$R(\tau) = r_0 + \gamma r_1 + \gamma^2 r_2 \ldots \gamma^T r_T = \sum_{t=0}^{T} \gamma^t r_t$$

- This return function defines the return as a discounted sum of the rewards in a trajectory, where $\gamma \in [0, 1]$ is the discount factor. As you understand, the terminal state may not ever be reached, and in such a scenario, the time period could really be infinite. In such a case, the expected return could also be infinite. To lessen against the explosive power of infinity, you add a discounting factor that exponentially reduces future rewards. It is called the discount rate or factor and is denoted by γ. It controls how soon future rewards are ignored and should be a value between 0 and 1, both inclusive. γ is an important variable which changes the way future rewards are valued.
- The smaller the γ, the less weightage is given to rewards in future time steps, making the objective "short-sighted". In the extreme case with $\gamma = 0$, the objective only considers the initial reward. In other words, if $\gamma = 0$, then only the present (initial) reward (r_0) is taken into consideration.
- The larger the γ, the more weightage is given to rewards in future time steps. The objective becomes more "far-sighted". If $\gamma = 1$, then rewards from every time step are weighted equally (infinite rewards).
- Typically you want to take the future rewards into account to be able to solve problems where the reward is far into the future. But precise values will differ depending on the problem at hand. In most examples, the discount rate is set between 0.9 and 0.99. For problems with infinite time horizon, you need to set $\gamma < 1$ to prevent the objective from becoming unbounded. For finite time horizon problems, γ is an important parameter as a problem may become more or less difficult to solve depending on the discount factor that you use.
- The objective $J(\tau)$ is simply the expectation of the returns over many trajectories and is given as the following.

$$J(\tau) = E_\tau [R(\tau)] = E_\tau \left[\sum_{t=0}^{T} \gamma^t r_t \right]$$

- The objective $J(\tau)$ is the return averaged over many episodes. The expectation accounts for stochasticity (randomness) in the actions and the environment that is, in repeated runs, the return may not always end up the same. Maximising the objective is the same as maximising the return.

6.3.2 Learnable Functions in Reinforcement Learning

With reinforcement learning formulated as an MDP, the natural question to ask is, what should an agent learn?

You have seen that an agent can learn an action-producing function known as a policy. However, there are other properties of an environment that can be useful to an agent. In particular, there are three primary functions to learn in reinforcement learning as following.

1. A policy, π, which maps state to action $a \sim \pi(s)$
2. A value function, $V^\pi(s)$ or $Q^\pi(s, a)$, to estimate the expected return $E_\tau [R(\tau)]$
3. The environment model $P(s' \mid s, a)$

Note : To make notation more compact, it is customary to write a successive pair of tuples (s_t, a_t, r_t), $(s_{t+1}, a_{t+1}, r_{t+1})$ as (s, a, r), (s', a', r'), where the prime symbol ' represents the next time step. So, don't get confused if you see use of ' or ''. They just mean subsequent time steps.

- A policy π is how an agent produces actions in the environment to maximise the objective. Given the reinforcement learning control loop, an agent must produce an action at every time step after observing a state s. A policy is fundamental to this control loop, since it generates the actions to make it run. A policy can be stochastic (random). That is, it may probabilistically output different actions for the same state. You can write this as $\pi(a \mid s)$ to denote the probability of an action a given a state s. An action sampled from a policy is written as $a \sim \pi(s)$.
- The value functions provide information about the objective. They help an agent understand how good the states and available actions are in terms of the expected future return. They come in two forms the $V^\pi(s)$ and $Q^\pi(s, a)$ functions.

$$V^\pi(s) = E_{s_0 = s, \tau \sim \pi} \left[\sum_{t=0}^{T} \gamma^t r_t \right]$$

$$Q^\pi(s, a) = E_{s_0 = s, a_0 = a, \tau \sim \pi} \left[\sum_{t=0}^{T} \gamma^t r_t \right]$$

- The value function V^π evaluates how good or bad a state is. V^π measures the expected return from being in state s, assuming the agent continues to act according to its current policy π. The return $R(\tau) = \sum_{t=0}^{T} \gamma^t r_t$ is measured from the current state s to the end of an episode. It is a forward-looking measure, since all rewards received before state s are ignored. Let's take a simple example to understand this better.
- The Fig. 6.3.1 depicts a grid-world environment in which an agent can move from cell to cell vertically or horizontally. Each cell is a state with an associated reward, as shown on the left of the Fig. 6.3.1. The environment terminates when the agent reaches the goal state with reward r = + 1.

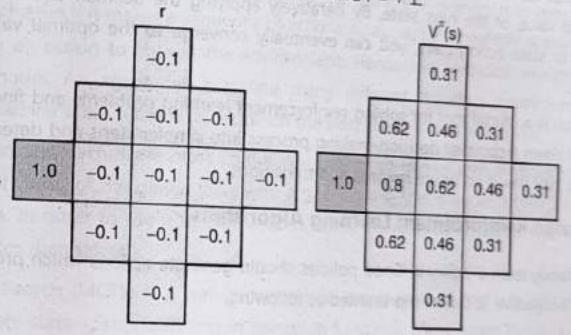


Fig. 6.3.1

- The value of a state is calculated from the rewards using with $\gamma = 0.9$ while using a policy π that always takes the shortest path to the goal state with r = + 1. If you had chosen another policy, for example, one that always moves right, then the values would be different. Here you can see the forward-looking property of the value function and its ability to help an agent differentiate between states that give the same reward. The closer an agent is to the goal state, the higher the value.
- The Q-value function Q^π evaluates how good or bad a state-action pair is. Q^π measures the expected return from taking action a in state s assuming that the agent continues to act according to its current policy, π. In the same manner as V^π, the return is measured from the current state s to the end of an episode. It is also a forward-looking measure, since all rewards received before state s are ignored.

- The transition function P (s′ | s, a) provides information about the environment. If an agent learns this function, it is able to predict the next state s′ that the environment will transition into after taking action a in state s. By applying the learned transition function, an agent can "imagine" the consequences of its actions without actually touching the environment. It can then use this information to plan good actions.

6.3.2(A) Bellman Equation

- The Bellman equation is a fundamental concept in dynamic programming and reinforcement learning. It describes the relationship between the value of a state or state-action pair and the values of its neighbouring states.
- In simple terms, the Bellman equation states that the value of being in a particular state (or taking a specific action in a state) is equal to the immediate reward obtained from that state (or state-action pair) plus the expected value of being in the next state (or the expected value of taking the next action in the next state).
- Mathematically, the Bellman equation can be written as follows:

$$V(s) = R(s) + \gamma * \Sigma[P(s, a, s') * V(s')]$$

In this equation:

- V(s) represents the value of state s, which indicates how desirable or valuable it is to be in that state.
- R(s) is the immediate reward obtained when transitioning to state s.
- γ (gamma) is a discount factor that determines the importance of future rewards compared to immediate rewards. It is a value between 0 and 1.
- P(s, a, s') is the probability of transitioning from state s to state s' when taking action a.
- Essentially, the Bellman equation tells you that the value of a state is the sum of the immediate reward and the discounted expected value of the next state. By iteratively applying the Bellman equation to update the value estimates of states or state-action pairs, you can eventually converge to the optimal values that maximise the cumulative rewards.
- The Bellman equation is a crucial tool for solving reinforcement learning problems and finding optimal policies. It allows you to break down a complex decision-making process into simpler steps and determine the value of each state or state-action pair based on future rewards and transitions.

6.3.2(B) Policy-Based Reinforcement Learning Algorithms

- Algorithms in this family learn a policy π. Good policies should generate actions which produce trajectories τ that maximise an agent's objective. It can be represented as following.

$$J(\tau) = E_{\tau \sim \pi}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$$

- This approach is quite easy to understand. If an agent needs to act in an environment, it makes sense to learn a policy. What constitutes a good action at a given moment depends on the state, so a policy function π takes a state s as input to produce an action $a \sim \pi(s)$. This means that an agent can make good decisions in different contexts. REINFORCE is the most well-known policy-based algorithm.
- A major advantage of policy-based algorithms is that they are a very general class of optimisation methods. They can be applied to problems with any type of actions discrete, continuous, or a mixture (multi-actions). They also directly optimise for the thing an agent cares most about the objective j (τ). Additionally, this class of methods is guaranteed to converge to a locally optimal policy. One disadvantage of these methods is that they have high variance and are sample-inefficient.

6.3.2(C) Value-Based Reinforcement Learning Algorithms

- An agent learns either $V^{\pi}(s)$ or $Q^{\pi}(s, a)$ value functions. It uses the learned value function to evaluate (s, a) pairs and generate a policy. For example, an agent's policy could be to always select the action a in state s with the highest estimated $Q^{\pi}(s, a)$. Learning $Q^{\pi}(s, a)$ is far more common than $V^{\pi}(s)$ for pure value-based approaches because it is easier to convert into a policy. This is because $Q^{\pi}(s, a)$ contains information about paired states and actions whereas $V^{\pi}(s)$ just contains information about states.
- SARSA is one of the older reinforcement learning algorithms. Despite its simplicity, SARSA incorporates many of the core ideas of value-based methods, so it is a good algorithm to study first in this family. However, it is not commonly used today due to its high variance and sample inefficiency during training. Deep Q-Networks (DQN) and its descendants, such as Double DQN and DQN with Prioritised Experience Replay (PER), are much more popular and effective algorithms.
- Value-based algorithms are typically more sample-efficient than policy-based algorithms. This is because they have lower variance and make better use of data gathered from the environment. However, there are no guarantees that these algorithms will converge to an optimum. In their standard formulation, they are also only applicable to environments with discrete action spaces. This has historically been a major limitation, but with more recent advances, such as QT-OPT, they can be effectively applied to environments with continuous action spaces.

6.3.2(D) Model-Based Reinforcement Learning Algorithms

- Algorithms in this family either learn a model of an environment's transition dynamics or make use of a known dynamics model. Once an agent has a model of the environment, P (s′ | s, a), it can "imagine" what will happen in the future by predicting the trajectory for a few time steps. If the environment is in state s, an agent can estimate how the state will change if it makes a sequence of actions $a_1, a_2, \ldots a_n$ by repeatedly applying P (s′ | s, a), all without actually producing an action to change the environment. Hence, the predicted trajectory occurs in the agent's "head" using a model. An agent can complete many different trajectory predictions with different actions sequences, then examine these options to decide on the best action a to actually take.
- Purely model-based approaches are most commonly applied to games with a target state, such as winning or losing in a game of chess, or navigation tasks with a goal state s. This is because their transition functions do not model any rewards. In order to use it to plan actions, some information about an agent's objective needs to be encoded in the states themselves.
- Monte Carlo Tree Search (MCTS) is a well-known model-based method that can be applied to problems with deterministic discrete state spaces with known transition functions. Many board games such as chess and Go fall into this category, and until recently MCTS powered many computer Go programs. It does not use any machine learning but randomly samples sequences of actions, known as Monte Carlo rollouts, to explore a game's states and estimate their value. There have been a number of improvements to this algorithm, but this is the essential idea.
- Other methods, such as iterative Linear Quadratic Regulators or Model Predictive Control (MPC), involve learning the transition dynamics, often under quite restrictive assumptions. To learn the dynamics, an agent will need to act in an environment to gather examples of actual transitions (s, a, r, s′).
- Model-based algorithms are very appealing because a perfect model endows an agent with foresight. It can play out scenarios and understand the consequences of its actions without having to actually act in an environment. This can be a significant advantage in situations where it is very time-consuming or expensive to gather

experiences from the environment such as robotics. Compared to policy-based or value-based methods, these algorithms also tend to require many fewer samples of data to learn good policies since having a model enables an agent to supplement its actual experiences with imagined ones.

- However, for most problems, models are hard to come by. Many environments are stochastic, and their transition dynamics are not known. In these cases, the model must be learned. This approach is still in early development, and it faces a number of challenges. First, an environment with a large state space and action space can be very difficult to model; doing so may even be intractable, especially if the transitions are extremely complex. Second, models are only useful when they can accurately predict the transitions of an environment many steps into the future. Depending on the accuracy of the model, prediction errors may compound for every time step and quickly grow to make the model unreliable.
- The lack of good models is currently a major limitation for the applicability of model-based approaches. However, model-based methods can be very powerful. When they work, they are often 1 or 2 orders of magnitude more sample-efficient than model-free methods.
- The distinction between model-based and model-free is also used to classify reinforcement learning algorithms. A model-based algorithm is simply any algorithm that makes use of the transition dynamics of an environment, whether learned or known in advance. Model-free algorithms are those that don't explicitly make use of the environment transition dynamics.

6.3.2(E) Comparison between Policy-Based, Value-Based, and Model-Based Algorithms

The Table 6.3.1 provides a quick comparison between policy-based, value-based, and model-based algorithms for reinforcement learning.

Table 6.3.1

Comparison Attribute	Policy-Based	Value-Based	Model-Based
Variance	High	Low	Low
Sample efficiency	Low	High	High
Possible actions	Discrete or continuous	Discrete	Discrete or continuous
Objective Optimisation	Yes	May be	May be
Complexity	Low	Medium	Very high

6.3.3 On-Policy and Off-Policy Algorithms

- An important distinction between deep reinforcement learning algorithms is whether they are on-policy or off-policy. This affects how training iterations make use of data.
- An algorithm is on-policy if it learns on the policy, that is, training can only utilise data generated from the current policy π. This implies that as training iterates through versions of policies, $\pi_1, \pi_2, \pi_3, \ldots \pi_n$, each training iteration only uses the current policy at that time to generate training data. As a result, all the data must be discarded after training, as it becomes unusable. This makes on-policy methods sample-inefficient (they require more training data). Examples of on-policy reinforcement learning algorithms are REINFORCE and SARSA.
- In contrast, an algorithm is off-policy if it does not have this requirement. Any data collected can be reused in training. Consequently, off-policy methods are more sample-efficient, but this may require much more memory to store the data. One of the common off-policy reinforcement learning algorithm is DQN.

6.3.4 Q-Learning (TD Learning)

- So far, you have learnt about the $Q^{\pi}(s, a)$ value function. The Q-value function Q^{π} evaluates how good or bad a state-action pair is. Q^{π} measures the expected return from taking action a in state s assuming that the agent continues to act according to its current policy, π. Q-Learning is a model-free reinforcement learning algorithm to learn the value of an action in a particular state. It chooses this action at random and aims to maximise the reward. Q-Learning is also called temporal difference (TD) learning as it uses the temporal difference formula for finding the Q-Value by using the value of current state and action and previous state and action. Let's take a simple example to understand this.

6.3.5 Important Terms in Q learning

- Suppose an agent is learning to play the toy environment as shown in the Fig. 6.3.2.

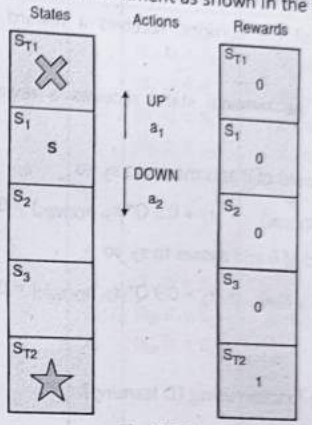


Fig. 6.3.2

- It is an environment having five states and two actions per state. This is essentially a corridor and the agent has to learn to navigate to the end of the corridor to the good terminal state s_{T2}, denoted with a star. There are five states in total – s_1, s_2, s_3 and two terminal states s_{T1} and s_{T2}. There are only two actions, a_{UP} and a_{DOWN}. Choosing a_{UP} moves the agent one cell up the corridor; choosing a_{DOWN} moves the agent one cell down the corridor. The agent always starts the game in state s_1, denoted s, and the game ends if the agent reaches either of the terminal states. s_{T2} is the goal state the agent receives a reward of 1 if it reaches this state. The agent receives rewards of 0 in all other states. The agent's discount rate γ is 0.9. The game is therefore optimally solved by a policy which reaches s_{T2} in the smallest number of steps because an agent values rewards received sooner more than rewards received later in time. In this case, the smallest number of steps an agent can take to optimally solve the environment is 3.

6.3.6 Q-Table

- For this very simple environment, the Q-function can be represented by a table, known at a tabular Q-function or Q-table, with one cell per (s, a) pair. This makes six pairs in total since the agent cannot act once it has moved to the terminal states. The optimal Q-function is defined as the expected cumulative discounted rewards from taking action a in state s and thereafter following the optimal policy. For this environment, the optimal policy is to select the action a_{DOWN} in all states since this is the quickest way to reach s_{T2} from any of the other states. The optimal Q-function for this environment is shown in the following Q-table.

Q*(s,a)	UP	DOWN
S_1	0	0.81
S_2	0.73	0.9
S_3	0.81	1.0

Fig. 6.3.3

6.3.7 Q-Functions

- The optimal Q-values are derived from the definition of the Q-function. Let's consider some examples.
- (s_0, a_{up}) : The agent moves out of the corridor, receives a reward of 0, and the episode terminates, so $Q^*(s_0, a_{up}) = 0$.
- (s_3, a_{DOWN}) : The agent reaches the terminal state, receives a reward of 1, and the episode terminates, so $Q^*(s_3, a_{DOWN}) = 1$.
- (s_2, a_{DOWN}) : The agent receives a reward of 0 and moves to s_3, so

$$Q^*(s_2, a_{DOWN}) = r_2 + 0.9\, Q^*(s_3, a_{DOWN}) = 0 + 0.9 * 1.0 = 0.9$$

- (s_3, a_{up}) : The agent receives a reward of 0 and moves to s_2, so

$$Q^*(s_3, a_{UP}) = r_3 + 0.9\, Q^*(s_2, a_{DOWN}) = 0 + 0.9 * (0 + 0.9 * 1.0) = 0.81$$

6.3.8 Q-Learning Algorithm

- So, how can you learn the optimal Q-function using TD learning?
  1. You initialise the Q-value table to 0 for every cell.
  2. The process of learning the Q-function involves randomly sampling a set of trajectories and, every time the agent experiences a (s, a, r, s') tuple, updating the Q-value table using the Bellman equation. The Bellman equation is given as following.

$$Q^*(s, a) = r + \gamma Q^*(s', a')$$

  So, the new Q-values can be derived using this equation and you can build the Q-value table.
- The following figure illustrates the process of learning the optimal Q-function in tabular form using a set of five trajectories from the environment. The diagram is split into five blocks from top to bottom. Each block corresponds to a single episode of experiences in the environment; the first block corresponds to the first episode, the second block the second episode, and so on. Each block contains a number of columns.
- They are interpreted from left to right as follows:
  - **Q-function episode start :** The value of the Q-function at the start of the episode. At the beginning of episode 1, all of the values are initialised to 0 since you have no information yet about the function.
  - **Episode :** The episode number.
  - **Time step :** Each block contains a variable number of experiences. For example, there are three experiences in episode 2 and seven experiences in episode 4. The time index of each experience within the block is indicated by the time step.

- **Action :** The action the agent took at each time step.
- **(s, a, r, s') :** The agent's experience at each time step. This consists of the current state s, the action the agent took a, the reward received r, and the next state the environment transitioned into s'.
- **$r + \gamma Q^*(s', a)$:** The target value (i.e., the right-hand side of the equation) to use in the Bellman update:

$$Q^*(s, a) = r + \gamma Q^*(s', a')$$

- **Q-function episode end :** The value of Q-function at the end of the episode. The Bellman update has been applied for each experience of the episode in time step order. This means that the Bellman update was applied first for the experience corresponding to time step 1, then time step 2, and so on. The table shows the final result after all of the Bellman updates have been applied for the episode.

Q-function episode start	Episode	Time step	Action	(s, a, r, s')	$r + \gamma Q^*(s', a)$	Q-function episode end
Up Down S_1 0 0 S_2 0 0 S_3 0 0	1	1	↓	$(S_1, D, 0, S_2)$	$0 + 0.9 \times 0 = 0$	Up Down S_1 0 0 S_2 0 0 S_3 0 1
	1	2	↑	$(S_2, U, 0, S_1)$	$0 + 0.9 \times 0 = 0$	
	1	3	↓	$(S_1, D, 0, S_2)$	$0 + 0.9 \times 0 = 0$	
	1	4	↓	$(S_2, D, 0, S_3)$	$0 + 0.9 \times 0 = 0$	
	1	5	↓	$(S_3, D, 1, S_{T2})$	1	
Up Down S_1 0 0 S_2 0 0 S_3 0 1	2	1	↓	$(S_1, D, 0, S_2)$	$0 + 0.9 \times 0 = 0$	Up Down S_1 0 0 S_2 0 0.9 S_3 0 1
	2	2	↓	$(S_2, D, 0, S_3)$	$0 + 0.9 \times 1 = 0.9$	
	2	3	↓	$(S_3, D, 1, S_{T2})$	1	
Up Down S_1 0 0 S_2 0 0.9 S_3 0 1	3	1	↓	$(S_1, D, 0, S_2)$	$0 + 0.9 \times 0.9 = 0.81$	Up Down S_1 0 0.81 S_2 0 0.9 S_3 0.81 1
	3	2	↓	$(S_2, D, 0, S_3)$	$0 + 0.9 \times 1 = 0.9$	
	3	3	↑	$(S_3, U, 0, S_2)$	$0 + 0.9 \times 0.9 = 0.81$	
	3	4	↓	$(S_2, D, 0, S_3)$	$0 + 0.9 \times 1 = 0.9$	
	3	5	↓	$(S_3, D, 1, S_{T2})$	1	
Up Down S_1 0 0.81 S_2 0 0.9 S_3 0.81 1	4	1	↓	$(S_1, D, 0, S_2)$	$0 + 0.9 \times 0.9 = 0.81$	Up Down S_1 0 0.81 S_2 0.73 0.9 S_3 0.81 1
	4	2	↑	$(S_2, U, 0, S_1)$	$0 + 0.9 \times 0.81 = 0.73$	
	4	3	↓	$(S_1, D, 0, S_2)$	$0 + 0.9 \times 0.9 = 0.81$	
	4	4	↑	$(S_2, U, 0, S_1)$	$0 + 0.9 \times 0.81 = 0.73$	
	4	5	↓	$(S_1, D, 0, S_2)$	$0 + 0.9 + 0.9 = 0.81$	
	4	6	↓	$(S_2, D, 0, S_3)$	$0 + 0.9 \times 1 = 0.9$	
	4	7	↓	$(S_3, D, 1, S_{T2})$	1	
Up Down S_1 0 0.81 S_2 0.73 0.9 S_3 0.81 1	5	1	↑	$(S_1, U, 0, S_{T1})$	0	Up Down S_1 0 0.81 S_2 0.73 0.9 S_3 0.81 1

Fig. 6.3.4

- After five trajectories consisting of 21 time steps in total, the tabular Q-function has the same values as the optimal Q-function table shown earlier. That is, it has converged to the optimal Q-function.
- The TD update makes use of the actual reward received in the following state. This has the effect of gradually backing up reward signals from future time steps to earlier time steps by one time step each update. Each time there is a TD backup, the Q-function incorporates information about an additional time step in the future. Over the course of many backups, more information about the future is incorporated into the Q-function estimate at time t. This mechanism makes it possible for the Q-function to incorporate long-term information.
- The number of time steps that the Q-function will take to converge depends on both the environment and the actions an agent takes, as they both affect the experiences that are used to learn the Q-function. If, for example, the agent didn't select the UP action until the sixth episode, then the Q-function would have taken longer (in terms of episodes) to converge because we would not have any information about the left-hand side of the Q-table until episode 6.
- The value of the discount factor γ affects the optimal Q-values. The Fig. 6.3.5 shows the optimal Q-values for the extreme values of γ, 0 or 1.

$Q^*(s,a)$ $\gamma = 0$

	Up	Down
S_1	0	0
S_2	0	0
S_3	0	1.0

$Q^*(s,a)$ $\gamma = 1$

	Up	Down
S_1	0	1.0
S_2	1.0	1.0
S_3	1.0	1.0

Fig. 6.3.5

- If y = 0, the agent becomes myopic (short-sighted), only caring about the reward it receives in the current state. In this case, the only (s, a) pair with a nonzero Q-value is (s_3, a_{DOWN}). This is not very useful because information about the reward in future time steps is no longer incorporated in the Q-values. Consequently, the agent doesn't get any clues about how to act in s_1 and s_2. If $\gamma = 1$, then all (s, a) pairs except (s1, a_{up}) (which leads to the game ending with no reward) have a Q-value of 1 because the agent does not care about receiving the reward for reaching s_{T2} sooner rather than later.
- Another way to think about γ is to consider how it affects the speed at which the Q-function can be learned. Small values of γ correspond to the agent having a short time horizon, it only cares about a few time steps into the future, so the Q-values will only need to be backed up for a few steps. It will likely be faster to learn the Q-function under these circumstances. However, there may be important information about rewards further into the future that is not captured by this Q-function, thus negatively affecting an agent's performance. If γ is large, then the Q-function may take much longer to learn because there are many time steps of information to back up, but the resulting function may be much more useful to the agent. Consequently, the optimal value of γ depends on the environment.

Review Questions

Here are a few review questions to help you gauge your understanding of this chapter. Try to attempt these questions and ensure that you can recall the points mentioned in the chapter.

Reinforcement Learning

Q. 1 Describe the need for reinforcement learning. (6 Marks)

Q. 2 Describe some real life applications of reinforcement learning. (5 Marks)

Q. 3 Explain characteristics of reinforcement learning. (6 Marks)

Q. 4 With respect to reinforcement learning, describe agent. (4 Marks)

Q. 5 With respect to reinforcement learning, describe action. (4 Marks)

Q. 6 With respect to reinforcement learning, describe environment. (4 Marks)

Q. 7 With respect to reinforcement learning, describe reward. (4 Marks)

Q. 8 With respect to reinforcement learning, describe observation. (4 Marks)

Q. 9 With examples, explain positive and negative reinforcement learning. (4 Marks)

Q. 10 Explain reinforcement learning cycle. (4 Marks)

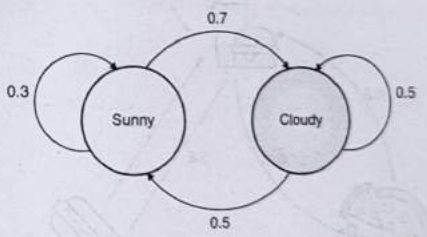
Q. 11 Compare supervised, unsupervised, and reinforcement learning approaches. (5 Marks)

Q. 12 Explain Markov Process. (4 Marks)

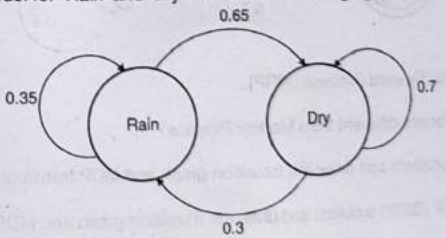
Q. 13 Describe Markov assumptions. (4 Marks)

Q. 14 With a suitable example, explain Markov chain. (5 Marks)

Q. 15 For the given Markov chain, predict the respective probabilities of weather for next 3 days assuming that the current day is Sunny. (6 Marks)



Q. 16 Consider Markov chain model for 'Rain' and 'Dry' is shown in following figure.



Initial Probabilities are P('Rain')=0.4 and P('Dry') = 0.6
Calculate the probability for a sequence of states {'Dry', 'Rain', 'Rain', 'Dry'}. (6 Marks)