5B17ICSC
Introduction to Machine Learning

Lecture Notes

Mithun A V

Contents

Unit I

1. Introduction to Machine Learning

1.1 What is Machine Learning?

Machine learning is the science (and art) of programming computers so they can learn from data.

Definition 1.1. *Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed.*

—Arthur Samuel, 1959

Definition 1.2. *A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.*

—Tom Mitchell, 1997

The examples that the system uses to learn are called the **training set**. Each training example is called a **training instance** (or **sample**). The part of a machine learning system that learns and makes predictions is called a **model**.

For example, spam filter is one of the ML applications where

- the task T is to flag spam for new emails
- the experience E is the training data
- the performance measure P can be the ratio of correctly classified emails, known as *accuracy*.

1.2 Machine Learning Vs. Traditional Programming

If a program is written using traditional programming for spam filter, it will check for each suspicious patterns, and would flag emails as spam if a number of these patterns were detected (Figure 1.1). Since the problem is difficult, the traditional program will likely become a long list of complex rules and will be difficult to maintain.

In contrast, a spam filter based on machine learning techniques automatically learns which words and phrases are good predictors of spam by detecting unusually frequent patterns of words in the spam examples. The program is much shorter and easier to maintain.

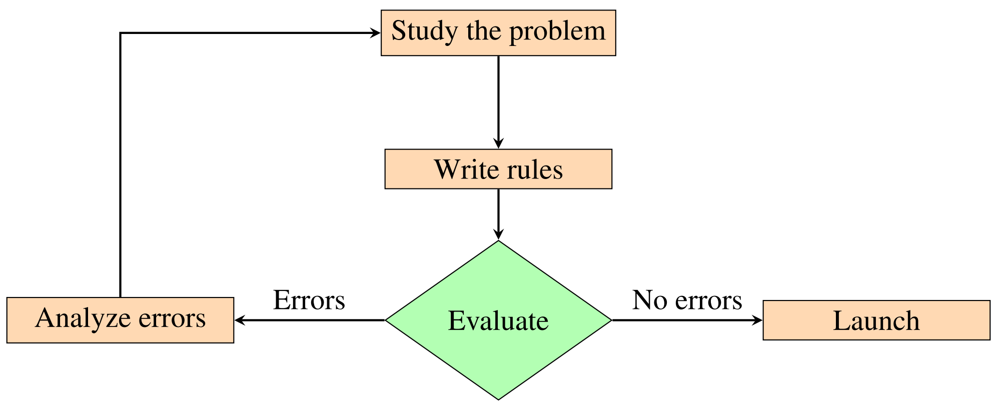


Figure 1.1: Traditional programming approach

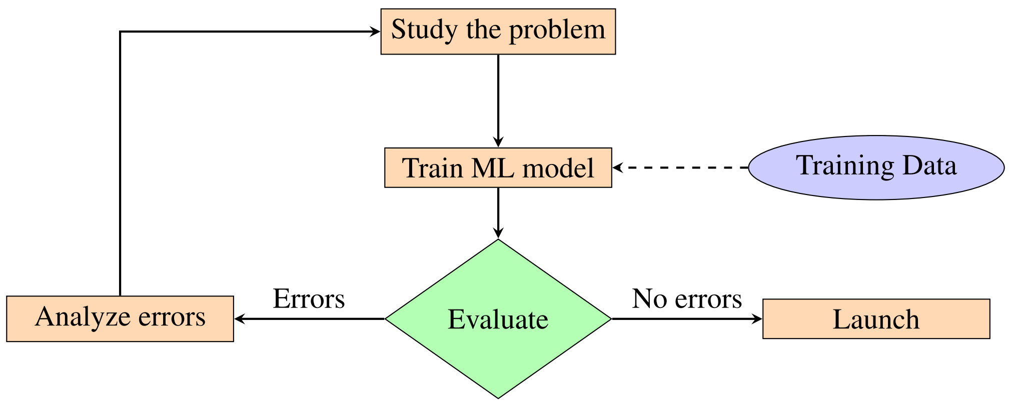


Figure 1.2: Machine learning approach

Machine learning are more suitable than the traditional programming approach for problems that either are too complex for traditional approaches or have no known algorithm. For example, speech recognition.

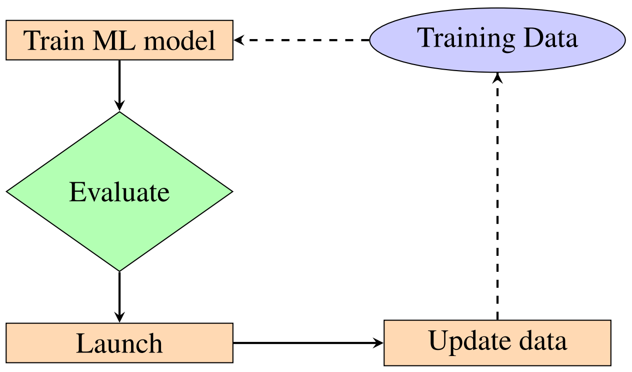


Figure 1.3: Automatically adopting to change

1.3 Applications of Machine Learning

Some of the example applications are

1. **Analyzing images of products to automatically classify them**
   This is known as image classification, typically performed using *convolutional neural networks (CNNs)* or *transformers*.

2. **Detecting tumors in brain scans**
   This is known as semantic image segmentation, where each pixel in the image is classified, typically using CNNs or transformers.

3. **Automatically classifying news articles**
   This is natural language processing (NLP), using *recurrent neural networks* (RNNs), transformers and CNNs.

4. **Automatically flagging offensive comments on discussion forums**
   This is text classification, using NLP tools.

5. **Summarizing long documents automatically**

This is a branch of NLP called text summarization.

6. **Creating a chatbot or a personal assistant**
   This involves many NLP components, including natural language understanding (NLU) and question-answering modules.

7. **Forecasting a company's revenue next year, based on many performance metrics**
   This is a regression task (i.e., predicting values) which may be tackled using any regression model, such as a linear regression or polynomial regression model, a regression support vector machine.

8. **Making an app react to voice commands metrics.**
   This is speech recognition, which requires processing audio samples, using RNNs, CNNs, or transformers

9. **Detecting credit card fraud**
   This is anomaly detection, which can be tackled using isolation forests, Gaussian mixture models, or autoencoders

10. **Segmenting clients based on their purchases so that different marketing strategy can be designed for each segment**
    This is clustering, which can be achieved using k-means, DBSCAN etc.

11. **Representing a complex, high-dimensional dataset in a clear and insightful diagram**
    This is data visualization, often involving dimensionality reduction techniques

12. **Recommending a product that a client may be interested in, based on past purchases**
    This is a recommender system.

13. **Building an intelligent bot for a game**
    This is often tackled using reinforcement learning (RL), which is a branch of machine learning that trains agents (such as bots) to pick the actions that will maximize their rewards over time, within a given environment.

1.4 Types of Machine Learning Systems

Machine learning systems can be classified based on the following criteria:

- How they are supervised during training (supervised, unsupervised, semi-supervised)
- Whether or not they can learn incrementally on the fly (online versus batch learning)
- Whether they work by simply comparing new data points to known data points, or instead by detecting patterns in the training data and building a predictive model(instance-based versus model-based learning)

1.4.1 Training Supervision

ML systems can be classified according to the amount and type of supervision they get during training.

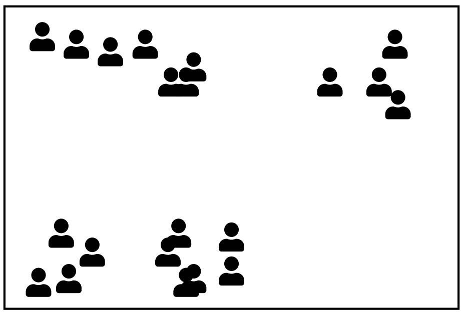


Figure 1.5: An unlabelled training set for unsupervised learning

Supervised Learning

In supervised learning, the training data set given to the algorithm includes the desired solutions, called **labels**. Some of the supervised learning tasks are

- **Classification**: Classification is used to predict the categorical class labels of new instances based on past observations. The spam filter is a good example of this: it is trained with many example emails along with their class (spam or ham).

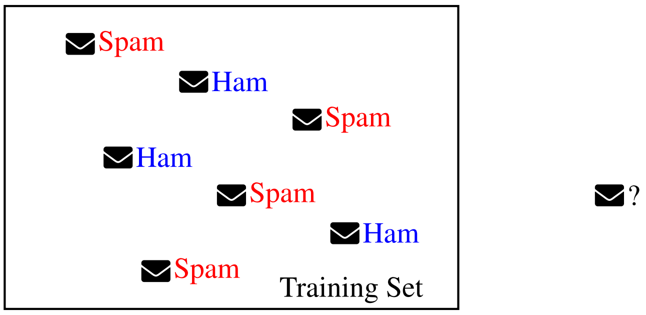


Figure 1.4: A labeled training set for spam classification

- **Regression:** Regression is used to predict a continuous numerical value based on input variables (features). Unlike classification, which predicts discrete class labels, regression models predict values that can take on any number within a range. For example, predicting the price of a car, given a set of features (mileage, age, brand, type of fuel etc.). To train the system, the system must be provided with a training set containing many examples of cars, including both their features and their **target** (price of the car).
- Some of the other supervised learning algorithms are
  - k-nearest Neighbours
  - Logistic Regression
  - Support Vector Machines (SVMs)
  - Decision Trees and Random Forests
  - Neural Networks

Unsupervised Learning

In unsupervised learning, the training data is unlabelled.

Some of the unsupervised learning mechanisms are

- **Clustering**: Clustering is used to group similar data points into clusters based on certain characteristics. For example, clustering algorithm can be used to detect groups of similar visitors from a dataset containing details about visitors of a blog. Some of the common algorithms used for clustering are

  - K-Means
  - DBSCAN
  - Hierarchical Cluster Analysis (HCA)
- **Anomaly detection and novelty detection:** Detecting unusual credit card transactions to prevent fraud, catching manufacturing defects, automatically removing outliers from a dataset before feeding it to another learning algorithm are examples for this. Some of the common algorithms for anomaly detection are
  - One-class SVM
  - Isolation Forest
- **Visualization and dimensionality reduction:** Visualization techniques can be used to output 2D or 3D representations of large amounts of complex and unlabeled data. These algorithms try to preserve as much structure as possible so that one can understand how the data is organized and perhaps identify unexpected patterns. In dimensionality reduction, the goal is to simplify the data without losing too much information.This can be done by merging several correlated features into one. For example, a car's mileage may be strongly correlated with its age, so the dimensionality reduction algorithm will merge them into one feature that represents the car's wear and tear. This is called *feature extraction*. Some of the algorithms for visualization and dimensionality reduction are
  - Principal Component Analysis (PCA)
  - Kernel PCA
  - Locally Linear Embedding (LLE)
  - t-Distributed Stochastic Neighbor Embedding (t-SNE)
- **Association rule learning:** In association rule learning, the goal is to discover interesting relationships between attributes in large datasets. For example, in a supermarket, running an association rule on sales logs may reveal that people who purchase barbecue sauce and potato chips also tend to buy steak. Thus, these items can be placed close to each other. Some of the algorithms for association rule learning are
  - Apriori
  - Eclat

Semisupervised Learning

Semisupervised Learning algorithms deal with data that is partially labelled. Most semisupervised learning algorithms are combinations of unsupervised and supervised algorithms.

For example, Google Photos uses semi-supervised learning. Once all the family photos are uploaded to the service, it automatically recognizes that the same person, A, appears in several photos. This clustering is the unsupervised part of the algorithm. If all the individuals in the photos are labeled once, the system will then be able to label the people in all the photos.

Reinforcement Learning

In reinforcement learning, the system is called an **agent**. The agent can observe the environment, select and perform actions, and get rewards in return or penalties in the form of negative rewards. It must then learn by itself what is the best strategy, called a *policy*, to get the most reward over time. A policy defines what action the agent should choose when it is in a given situation. For example, many robots implement Reinforcement Learning algorithms to learn how to walk.

1.4.2 Batch and Online Learning

Machine learning systems can be classified based on whether or not the system can learn incrementally from a stream of incoming data.

Batch learning

In batch learning, the system is does not learn incrementally. It must be trained using all the available data. This will generally take a lot of time and computing resources, so it is typically done offline. After training, the system is launched into production and runs without learning anymore. This is called **offline learning**.

To learn about new data, the system must be trained from scratch with full dataset including the new data. The old system must then be replaced with the new one.

Batch learning system is simple and often works fine, but the disadvantages are

- Training using the full set of data can take many hours. Hence this system cannot be used for rapidly changing data.
- Training on the full set of data requires a lot of computing resources (CPU, memory space, disk space, disk I/O, network I/O, etc.)

Online Learning

In online learning, the system is trained incrementally by feeding it data instances sequentially, individually or in small groups called *mini-batches*. Each learning step is fast and cheap, so the system can learn about new data on the fly, as it arrives. Online learning this systems can be used for rapidly changing data.

One important parameter called the **learning rate** determines how fast the system should adapt to changing data. A high learning rate means that the system will rapidly adapt to new data, but it will also tend to quickly forget the old data. Conversely, a low learning rate means that the system will learn more slowly, but it will also be less sensitive to noise in the new data.

1.4.3 Instance-Based Versus Model-Based Learning

Machine Learning systems can also be classified based on how they *generalize*.

Instance-based learning

The most basic form of learning is simply to memorize. For example, in the case of a spam filter, an email can be flagged as spam by measuring its similarity to a known spam email. One way to measure similarity between two emails is to count the number of words they have in common. The system would then flag an email as spam if it shares many words with a known spam email.

This approach is known as instance-based learning. In this method, the system memorizes examples and then generalizes to new cases by using a similarity measure to compare them to the learned examples.

Model-based learning

In model-based learning, a model is built from examples and then that model is used to make predictions.

For example, consider the sample data given in Table 1.1. Plot for this data is shown in Figure 1.6.

Based on the table, there is a noticeable trend where life satisfaction increases nearly linearly with a country's GDP per capita. Therefore, it suggests that life satisfaction could potentially be modeled as a linear function of GDP per capita. This step in data analysis is commonly referred to as *model selection*. The model can be written as

$$\text{life satisfaction} = \theta_0 + (\theta_1 \times \text{GDP per capita})$$

This model has two parameters θ_0 and θ_1.Varying the values of these parameters will represent different linear functions, as illustrated in Figure 1.7. A performance measure can be used to determine the parameter values that make the model perform best. For linear regression problems, a cost function that measures the distance between the linear model's predictions and the training

Country	**GDP per capita (USD)**	**Life satisfaction**
Hungary	12,240	4.9
Korea	27,195	5.8
France	37,675	6.5
Australia	50,962	7.3
United States	55,805	7.2

Table 1.1: Extract from the data obtained by combining Better Life Index data from the OECD's website and stats about gross domestic product (GDP) per capita from the IMF's website

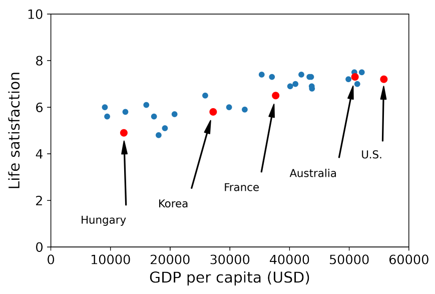


Figure 1.6: GDP per capita Vs Life satisfaction

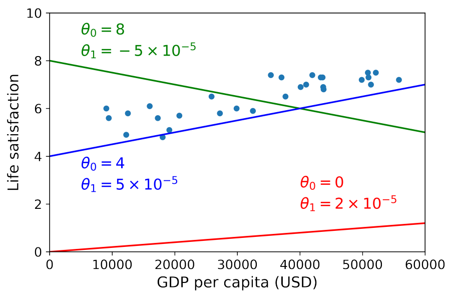


Figure 1.7: GDP per capita Vs Life satisfaction - Example models

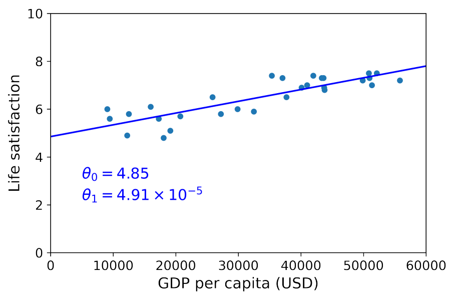


Figure 1.8: GDP per capita Vs Life satisfaction - Best fit Model

examples serves as the performance measure. The objective is to minimize this distance, thereby optimizing the model's accuracy.

A Linear Regression algorithm finds the parameters that make the linear model fit best to the training examples. This is called *training* the model. For example, the optimal parameter values for the data in Table 1.1 is $\theta_0 = 4.85$ and $\theta_1 = 4.91 \times 10^{-5}$ (Figure 1.8).

1.5 Challenges of Machine Learning

Insufficient Quantity of Training Data

Machine learning algorithms typically need thousands of examples for training even for simple problems. For complex problems such as image or speech recognition millions of examples may be needed.

Nonrepresentative Training Data

Machine learning systems make accurate predictions when the training data is representative of the new cases they will encounter. However, if the sample size is too small, it can introduce sampling noise, meaning the data may not accurately reflect the overall population due to random chance. Even with large samples, if the sampling method is flawed, the data can still be non-representative. This issue is known as sampling bias.

To mitigate these problems, ensure that the training data is sufficiently large and collected using unbiased sampling methods. This will help the machine learning model generalize well to new, unseen data.

Poor-Quality Data

Identifying patterns in training data becomes challenging when the data contains errors, outliers, and noise, which reduces the likelihood of the system performing well. Therefore, it is crucial to clean the training data first.

- If certain instances are clear outliers, they can be discarded or manually corrected.
- If some instances are missing a few features, several options are available:
  - Ignore the attribute entirely
  - Disregard these instances
  - Fill in missing values
  - Train one model with the feature and another model without it

Irrelevant Features

For a machine learning system to perform well, it must be trained with a dataset containing a good set of features. This process is called *feature engineering*. The steps involved in feature engineering are:

- Feature selection (selecting the most useful features to train on among existing features)
- Feature extraction (combining existing features to produce a more useful one)
- Creating new features by gathering new data

Overfitting the Training Data

In machine learning overfitting means that the model performs well on the training data, but it does not generalize well. Complex models like deep neural networks are capable of detecting subtle patterns in data. However, if the training data is noisy or too small, the model might end up detecting patterns within the noise itself. These patterns, based on noise rather than meaningful data, will not generalize well to new instances.

Constraining a model to make it simpler and reduce the risk of overfitting is called *regularization*. The amount of regularization to apply during learning can be controlled by a *hyperparameter*. A hyperparameter is a parameter of a learning algorithm (not of the model). In linear regression, if regularization hyperparameter is set to a very large value, it will result in an almost flat model (a slope close to zero). In this case, the learning algorithm will almost certainly not overfit the training data, but it will be less likely to find a good solution.

Underfitting the Training Data

Underfitting is the opposite of overfitting: it occurs when the model is too simple to learn the underlying structure of the data. For example, a linear model of life satisfaction is prone to underfit. Some options for fixing this problem are

- Select a more powerful model, with more parameters.
- Feed better features to the learning algorithm (feature engineering).
- Reduce the constraints on the model.

1.6 Variables

Variables are the properties or characteristics of an object or some events, which can take different values. Two types of variables are

- **Dependent variables:** A dependent variable is a variable whose value depends upon some independent variables. The dependent variable is what is being measured in an experiment or evaluated in a mathematical equation.
- **Independent variables:** The independent variable is the variable whose change is not affected by any other variable in the experiment.

Independent variables are categorized into different levels.

- **Quantitative and Qualitative Variables:** Qualitative variables are those that express a qualitative attribute such as gender, religion, etc.. These variables does not need any numerical ordering. Qualitative variables are also known as *categorical variables*. Quantitative variables are those variables that are measured in terms of numbers such as height, weight, length etc..
- **Discrete and Continuous Variables:** Discrete variables are variables that can take on a finite number of values. For example, number of students in a class is a discrete variable. Continuous variables are variables that can take on an infinite number of values within a given range. These values are not countable, meaning that between any two values, there can always be another value. For example, "time to respond to a question" is a continuous variable since the scale is continuous and not made up of discrete steps.

1.7 Understanding Data

Data is a collection of values that can be either qualitative (descriptive) or quantitative (numerical). In simpler terms, data is raw information that has not yet been processed or organized. This raw data, sometimes called source data or atomic data, is the basic building block for further analysis and interpretation. This data is collected, measured, reported, and analyzed. It can be represented using graphs, images, and various data analysis tools.

Data analysis is the process of inspecting, cleansing, transforming, and modelling data. The purpose of data analysis is to extract relevant information from the data. This useful information can reveal patterns and insights about the data's behavior, helping to draw conclusions and support decision-making.

1.7.1 Types of Data

Data can be categorized on the basis of size, structure, origin, usefulness etc.

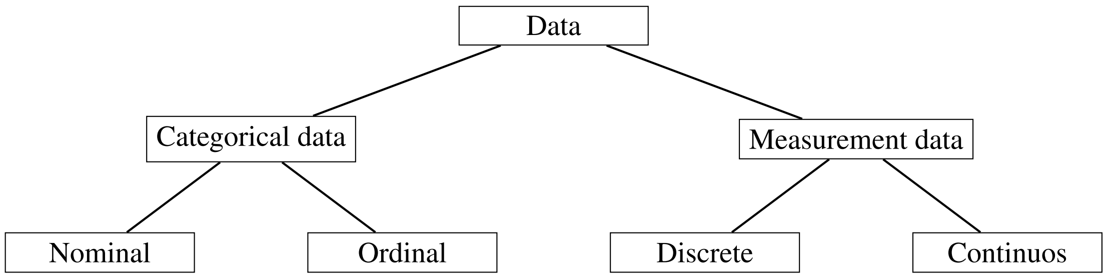


Figure 1.9: Types of data

Categorical Data

In this type, data can be grouped into categories, based on some qualitative feature. They can be classified as

- **Nominal:** A nominal variable can have two or more categories, but these categories have no special ordering. For example, consider the variable gender. It is categorical and has two categories - male and female. But these categories are not ordered.
- **Ordinal:** An ordinal variable is similar to a nominal variable but with a specific order to each of its categories. For example, T-shirt size will have different categories which are ordered.

Measurement Data

Measurement is the process which evaluates a feature. The objects being studied are “measured" based on some numerical characteristic. Measurement data can be classified as

- **Discrete:** Discrete data are finite values which can be counted. For example, number of students in a class is a discrete variable.
- **Continuos:** Continuous data will have infinte number of values. These values are not countable, meaning that between any two values, there can always be another value. For example, “time to respond to a question” is a continuous variable since the scale is continuous and not made up of discrete steps.

1.8 Measures of Central Tendency

Measures of central tendency are a class of statistics used to identify a value that falls in the central position or middle of a set of data. In other words, measure of central tendency attempts to describe a whole set of data using a single value.Three main measures of central tendency are **mean**, **median** and **mode**.

1.8.1 Mean

The mean can be calculated by dividing the sum of the value of each observation by the number of observations. It is also known as the arithmetic average.

Mean of ungrouped data

If $x_1, x_2, \ldots, x_n$ are n observations, then

$$\text{Mean } \bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

■ **Example 1.1** Find the mean age of a particular club member from the following data set: 54, 54, 55, 56, 57, 57, 58, 59, 60, and 60.

$$\begin{aligned}\bar{x} &= \frac{54+54+55+56+57+57+58+59+60+60}{10} \\ &= 57\end{aligned}$$

■

Mean for a frequency distribution

If $x_1, x_2, \ldots x_n$ are n observations and $f_1, f_2, \ldots, f_n$ are there respective frequencies, then

$$\text{Mean } \bar{x} = \frac{\sum_{i=1}^{n} x_i f_i}{\sum_{i=1}^{n} f_i}$$

■ **Example 1.2** Find the mean for the data given below,

x_i	f_i	$x_i f_i$
5	4	20
10	10	100
15	20	300
20	6	120
Total	40	540

$$\begin{aligned}\bar{x} &= \frac{540}{40} \\ &= 13.5\end{aligned}$$

■

Mean for continuous frequency distribution

Let $x_1, x_2, x_3, \ldots, x_n$ be the middle values of the given classes and let $f_1, f_2, f_3, \ldots, f_n$ be the corresponding frequencies then

$$\text{Mean } \bar{x} = \frac{\sum_{i=1}^{n} x_i f_i}{\sum_{i=1}^{n} f_i}$$

■ **Example 1.3** Find the mean for the data given below,

Marks	f_i	x_i	$x_i f_i$
0-10	4	5	20
10-20	6	15	90
20-30	10	25	250
30-40	20	35	700
40-50	6	45	270
50-60	4	55	220
Total	50		1550

$$\bar{x} = \frac{1550}{50}$$
$$= 31$$

■

Merits of Mean

- It is rigidly defined.
- It is simple to calculate.
- It is easy to understand.
- It depends on magnitude of all observations.
- It is capable of further algebraic treatment.
- Arrangement of data is not required.
- It is used in clustering process.

Demerits of Mean

- Mean can be an impossible value.
- If one observation is missing or its exact magnitude is not known, mean cannot be calculated.
- Arithmetic mean cannot be calculated for open end classes.
- Arithmetic mean cannot be located graphically.

1.8.2 Median

The median is the middle value of an ordered distribution. To find the median, the values should be arranged in ascending or descending order. The median divides the distribution into 2 equal parts.

Median for ungrouped data

In an ordered distribution, if the number of observations is odd, the median value is the middle value. When the number of observations is even, the median value is the mean of the two middle values.

■ **Example 1.4** The median of the values 54, 54, 55, 56, 57, 57, 58, 59, 60 and 60 is

$$\text{Median} = \frac{57+57}{2} = 57$$

■

Median for a frequency distribution

The steps for finding median for a frequency distribution are

1. Sort the data in ascending order.
2. Prepare a Less Than Cumulative Frequency Table. This table has two columns
   - One column lists the distinct values from the sorted dataset.
   - Another column lists the cumulative frequency of each value, which is the total count of all values less than or equal to that value.
3. Find Cumulative Frequency just higher than $\frac{N}{2}$.
   (a) Calculate N, the total number observations
   (b) Computer $\frac{N}{2}$ to find the position of the median
   (c) Look for the cumulative frequency in the table that is just greater than $\frac{N}{2}$
4. Identify the Median. Once the cumulative frequency just higher than $\frac{N}{2}$ is found, the corresponding value in the dataset will give the median.

■ **Example 1.5** Find the median for the following data

Data	150	100	120	80	200	250
Frequency	24	60	32	40	15	8

Cumulative frequencies of the data is given below.

Data	80	100	120	150	200	250
Frequency	40	60	32	24	15	8
Cumulative Frequency	40	100	132	156	171	179

$$N = 179$$
$$\text{Median frequency} = \frac{N}{2} = \frac{179}{2} = 89.5$$

Cumulative frequency just higher than 89.5 is 100 and the corresponding observation is 100. Hence the median is 100. ■

Median for continuous frequency distribution

The steps for finding median for continuous frequency distribution are

1. Calculate Cumulative Frequencies:
   (a) Arrange the data into a frequency distribution table or cumulative frequency distribution table.
   (b) Compute the cumulative frequency for each class. The cumulative frequency is the sum of frequencies up to and including that class.
2. Find $\frac{N}{2}$
   - N is the total number of observations in the dataset.
   - Calculate $\frac{N}{2}$ to determine the position of the median.

3. Identify Median Class
   (a) Locate the cumulative frequency that is just higher than $\frac{N}{2}$
   (b) This cumulative frequency corresponds to the median class.
4. Calculate Median using

$$\text{Median } = l + \frac{(\frac{N}{2} - m)c}{f}$$

   where
   - l is the lower boundary of the median class
   - N is the total frequency
   - m is the cumulative frequency just before the median class
   - c is the class width of the median class
   - f is the frequency of the median class

■ **Example 1.6** Calculate the median for the following data

Class	Frequency	Cumulative Frequency
0-100	26	26
100-200	32	58
200-300	65	123
300-400	75	198
400-500	60	258
500-600	42	300

■

Here, $\frac{N}{2} = \frac{300}{2} = 150$. The cumulative frequency just greater than or equal to 150 is 198. From the table, the median class is 300 - 400. Hence the median can be calculated as

$$\text{Median} = 300 + \frac{(\frac{300}{2} - 123)100}{75} = 336$$

Merits of Median

- It is easy to understand and simple to calculate
- It depends on magnitude of all observations
- Outliers do not considerably affect it
- Even when some of the observations are unknown the median can sometimes be calculated provided their relative position is known
- It can be determined when data is in the form of frequency table with open classes at beginning or end.
- Median is especially useful in qualitative phenomenon like intelligence, beauty, efficiency, health etc.
- It can be determined graphically.

Demerits of Median

- If number of observations is very large the determination of the median is very difficult as the observations are to be arranged in order of magnitude.
- It is not capable of further algebraic treatments.
- It is less stable than mean.

1.8.3 Mode

The mode of a distribution is the most frequently occurring value or the value with the highest frequency. A set of data can have more than 1 modal value.

Mode for ungrouped or raw data

Consider a set of data showing the ages of a group of peopls: 54, 54, 55, 56, 57, 57, 57, 58, 59, 60 and 60. Here the value 57 occurs more than any other values. Hence the mode is 57.

Mode for a frequency distribution

For a frequency distribution, the modal value is the observation corresponding to the highest value of frequency.

■ **Example 1.7** Find the mode for the following data

Age	Frequency
14	3
15	2
16	1
13	1

Since the frequency corresponding to age 14 is the highest, it is the value of mode for the data. ■

Mode of a continuous frequency distribution

The class which corresponds to the maximum frequency is known as the **modal class**. Then the mode can be calculated using the formula

$$\text{Mode} = l + \frac{c(f_1 - f_0)}{2f_1 - f_0 - f_2}$$

Where

- l is the lower bound of the modal class
- c is the class width of the modal class
- f_1 is the frequency of the modal class
- f_0 is the frequency before the modal class
- f_2 is the frequency after the modal class

■ **Example 1.8** Find the mode for the following data

Age	Frequency
0-5	15
5-10	10
10-15	15
15-20	30
20-25	20

$$\text{Mode} = 15 + \frac{(30-15)5}{2 \times 30 - 15 - 20} = 18$$

■

1.8.4 Outliers

An outlier is a value that differs extremely from other values in the same set. Outliers may significantly affect the measure of central tendency.

■ **Example 1.9** Consider the sample dataset 10, 10, 20, 20, 20, 20, 30, 30, 30, 40. The measures of the central tendency for this data set are

$$\begin{aligned} \text{Mean} &= 23 \\ \text{Median} &= 20 \\ \text{Mode} &= 20 \end{aligned}$$

If an outlier with value 400 is added to the data set: 10, 10, 20, 20, 20, 20, 30, 30, 30, 40, 400 then

$$\begin{aligned} \text{Mean} &= 57.27 \\ \text{Median} &= 20 \\ \text{Mode} &= 20 \end{aligned}$$

Mean has a significant effect when there is an outlier. ■

1.9 Measures of Spread (Dispersion)

Measures of central tendency indicate the central position of a series. They do not give details about how much the data is spread out, or how much does each value differ with others. Measures of dispersion may be defined as either as a measure of scatter of observations among themselves or a measure of a scatter of items from an average. Variability is said to exist if values differ from the mean. If all values are the same, then there is no variability. Different measures of dispersion are

- Range
- Quartile Deviations
- Mean deviations
- Standard deviations
- Variance

1.9.1 Range

The range is a measure that gives the measure of how much is the total span of the data. It can be calculated as the difference between the highest and the lowest data values. If the range is high (large), it means the data points are spread over a wide area. The values are more dispersed from each other. If the range is low (small), it means the data points are closer to each other and, possibly, to the mean.

$$\text{Range} = X_L - X_S$$

Where, X_L is the largest value and X_S is the smallest value.

1.9.2 Quartile Deviation

Quartile Deviation (Q_D) is the inter-quartile distance divided by 2.

$$\text{Quartile Deviation Q.D.} = \frac{Q_3 - Q_1}{2}$$

$$Q_3 = l_3 + \frac{(\frac{3N}{4} - m_3)c_3}{f_3} \qquad Q_1 = l_1 + \frac{(\frac{N}{2} - m_1)c_1}{f_1}$$

■ **Example 1.10** Find the Quartile Deviation for the following set of data: 490, 540, 595, 595, 620, 650, 680, 770, 832, 838, 890, 900.

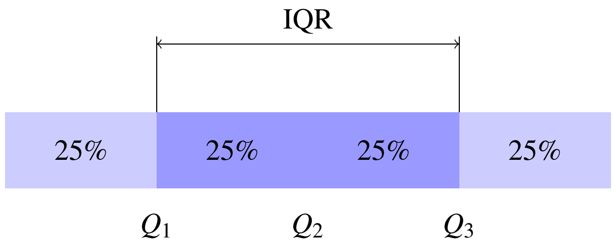
$$Q_1 = \frac{595+595}{2} = 595$$

$$Q_3 = \frac{832+838}{2} = 835$$

$$Q.D. = \frac{835-595}{2} = 120$$

■

The interquartile range (IQR) is a measure of distance between the third quartile and first quartile ($Q_3 - Q_1$) of the data.



1.9.3 Mean Deviation

The Mean Absolute Deviation (M.A.D.) of a dataset is the average distance between each data point and the mean. It gives an idea about the variability in a dataset.

$$\text{M.A.D.} = \frac{\sum_{i=1}^{n} |x_i - \bar{x}|}{n}$$

■ **Example 1.11** Find the mean deviation of the dataset 3, 5, 7, 7, 8, 12.

$$\bar{x} = \frac{3+5+7+7+8+12}{6} = 7$$

x_i	$x_i - \bar{x}$
3	4
5	2
7	0
7	0
8	1
12	5

$$\text{M.A.D} = \frac{12}{6} = 2$$

■

1.9.4 Standard Deviation

The standard deviation gives an idea about how much the data varies from the mean. It is the most commonly used measure of dispersion. It is the square of the mean of squares of deviations of observations from their arithmetic mean. Let $x_1, x_2, \ldots, x_n$ be n observations then their standard deviation is

$$\text{S.D.} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

If $x_1, x_2, \ldots, x_n$ are n observations and $f_1, f_2, \ldots f_n$ are the corresponding frequencies, then their standard deviation is

$$\text{S.D.} = \sqrt{\frac{1}{N}\sum f_i(x_i - \bar{x})^2}$$

where N is the total number of population.

When samples are taken, then the equation is,

$$\text{S.D.} = \sqrt{\frac{1}{n-1}\sum(x_i - \bar{x})^2}$$

■ **Example 1.12** Find the standard deviation of the dataset 3, 5, 7, 7, 8, 12.

$$\bar{x} = \frac{3+5+7+7+8+12}{6} = 7$$

x_i	$(x_i - \bar{x})^2$
3	16
5	4
7	0
7	0
8	1
12	25

$$\text{S.D.} = \sqrt{\frac{1}{5}} \times 46 = 3.033$$

■

1.9.5 Variance

Variance also gives a measure of dispersion. It is the square of standard deviation. It is calculated as the average of the squared deviations from the mean.

$$\text{Population Variance, } \sigma^2 = \frac{\sum(x_i - \mu)^2}{n}$$

where μ is the population mean.

$$\text{Sample Variance, } s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

1.9.6 Box Plot

A Box and Whisker Plot (or Box Plot) uses the help of quartiles to represent the data distribution visually. There is a rectangular box with lines (called "whiskers") extending from it. They indicate variability outside the upper and lower quartiles. Outliers are plotted as individual dots that are in-line with whiskers. It may be drawn vertically or horizontally. Box plots takes up less space, which is advantageous when comparing distributions between many groups or datasets.

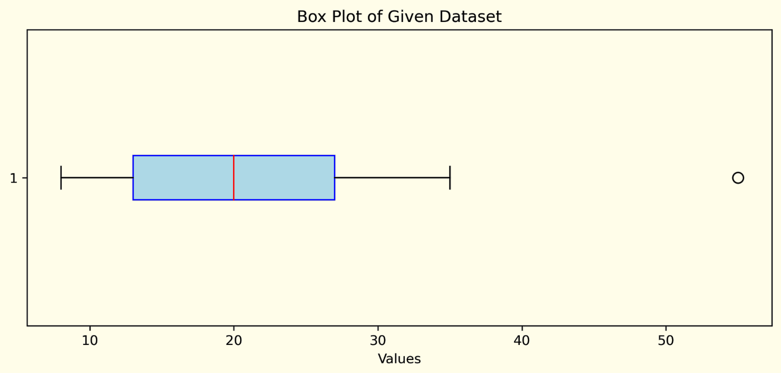
■ **Example 1.13** Draw the box plot for the dataset 11, 22, 20, 14, 29, 8, 35, 27, 13, 55, 10, 24, 17.

The data set can be arranged in ascending order to find the median and quartiles as

$$8, 10, 11, 13, 14, 17, 20, 22, 24, 27, 29, 35, 55$$

$$\begin{aligned} Q_2 &= 20 \\ Q_1 &= \frac{11+13}{2} = 12 \\ Q_3 &= \frac{27+29}{2} = 28 \\ min &= 8 \\ max &= 55 \end{aligned}$$

In a box plot, the values outside the range $[Q_1 - 1.5 \times IQR, Q_3 + 1.5 \times IQR]$ are considered outliers. Hence, in this example 55 is an outlier. So the *max* value is 35. The box plot for this dataset is given below.



■

Unit II

2. Regression

2.1 Introduction

Regression analysis is a statistical method used to model the relationship between one or more independent variables (predictors/inputs) and a dependent variable (output/target), where the dependent variable takes on a continuous numerical value. IN other words, regression is a supervised learning technique used to predict the continuous output variable based on the one or more predictor variables. Some examples of regression are

- Predicting the price of a house based on features such as size, number of bedrooms, and other characteristics.
- Predicting exam scores based on the number of study hours, class attendance, and prior grades.
- Estimating a car's fuel efficiency based on engine size, vehicle weight, and speed.
- Forecasting sales revenue based on advertising expenditure, product price, and market size.

2.2 Terminology and Notations

- **Model:** A model in machine learning is a mathematical representation that is trained on data to make predictions or decisions without being explicitly programmed to perform the task.
- **Training set:** The dataset used to train the model.
- **Input variable/Input feature:** The feature/variable in the training data set, used to predict output. Input variable is usually denoted by letter x.
- **Output variable/target variable:** The feature/variable whose value is predicted by the model. Output variable is usually denoted by letter y.

x	Input feature
y	Output feature
m	Number of training examples in the training set
$(x^{(i)}, y^{(i)})$	i^{th} training example

Table 2.1: Notations - 1

■ **Example 2.1** Consider the following dataset for housing price.

Sl. No.	size (sq. ft.)	price (Rs. 1000's)
1	2000	3000
2	1000	1800
3	1500	2100
⋮	⋮	⋮
120	800	1200

If the price of a house is to be predicted based on its size, then the input feature x represents the size of the house, while the output feature y represents the price.

$$\begin{aligned} m &= 120 \\ x^{(1)} = 2000 &\quad y^{(1)} = 3000 \\ x^{(2)} = 1000 &\quad y^{(2)} = 1800 \\ &\vdots \\ x^{(120)} = 800 &\quad y^{(120)} = 1200 \end{aligned}$$

2.3 Linear Regression Model

In supervised learning like linear regression, a learning algorithm takes a training dataset as input and outputs a function f. This function f is known as a **hypothesis** and it represents a **model**. Given a value for input feature x, f predicts an output denoted as $\hat{y}$. $\hat{y}$ is known as an **estimate** of y and it represents the value predicted by the model, while y is the actual observed value corresponding to the input x.

$$\begin{tikzcd} & \boxed{\text{Training Set}} \arrow[d] & \\ & \boxed{\text{Learning Algorithm}} \arrow[d] & \\ x \arrow[r] & \boxed{f} \arrow[r] & \hat{y} \end{tikzcd}$$

If the output is predicted based on a single input feature, then it is known as **simple linear regression** or **univariate linear regression**. In simple linear regression, the function f is a straight line which is of the form

$$f_{w,b}(x) = wx + b$$

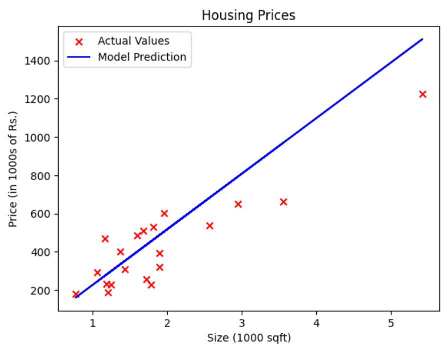


Figure 2.1: Simple Linear Regression Model example

In Figure 2.1, the blue line is the function f which represents the model $wx+b$.

In the model $wx+b$, w and b are called the **parameters** of the model. In machine learning, parameters of a model are the variables that can be adjusted during training in order to improve the model. They are also known as **coefficients** or **weights**. Depending on the values chosen for w and b, the function $f_{w,b}(x)$ will vary, producing different lines on the graph (Figure 2.2).

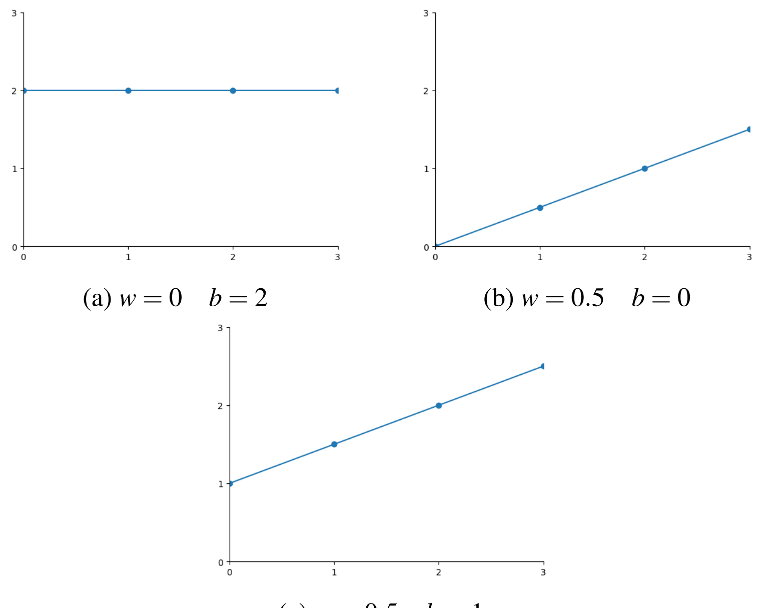


Figure 2.2: Different plots for different values of parameters w and b

Linear regression aims to find values for the parameters w and b so that the straight line defined by the function f fits the data points as closely as possible. This means that the line should be near the training examples compared to other lines that may not fit as well.

2.4 Cost Function

To find values for w and b such that the predicted values $\hat{y}^{(i)}$ are close to the true targets $y^{(i)}$ for many or ideally all training examples $(x^{(i)}, y^{(i)})$, it is essential to have a mechanism for measuring the goodness of fit of the regression line to the data. A cost function J can be used to measure how well the model fits the training data. One of the most commonly used cost function is **squared error cost function** given below.

$$
\begin{aligned}
J(w,b) &= \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})^2 \\
&= \frac{1}{2m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})^2
\end{aligned}
$$

The goal of the learning algorithm is to find optimal values for w and b that minimize the cost function. By minimizing the cost function, the algorithm aims to improve the accuracy of the model's predictions, ensuring that the regression line closely fits the data points.

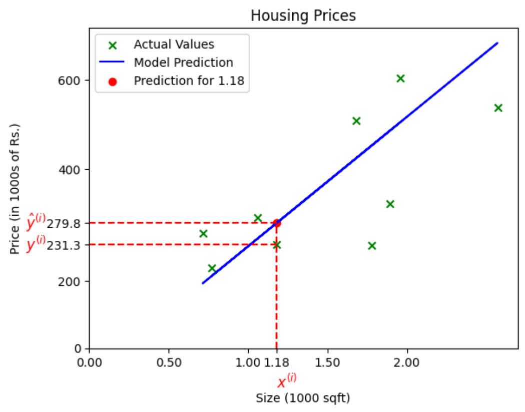


Figure 2.3: Regression Analysis: The plot illustrates the relationship between actual and predicted values. The green crosses represent the observed data points, while the blue line indicates the regression model's predictions. The red dot highlights the predicted value for an input of 1.18.

In univariate linear regression

Model	: $f_{w,b}(x) = wx + b$
Parameters	: w, b
Cost function	: $J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})^2$
Goal	: Minimize $J(w,b)$

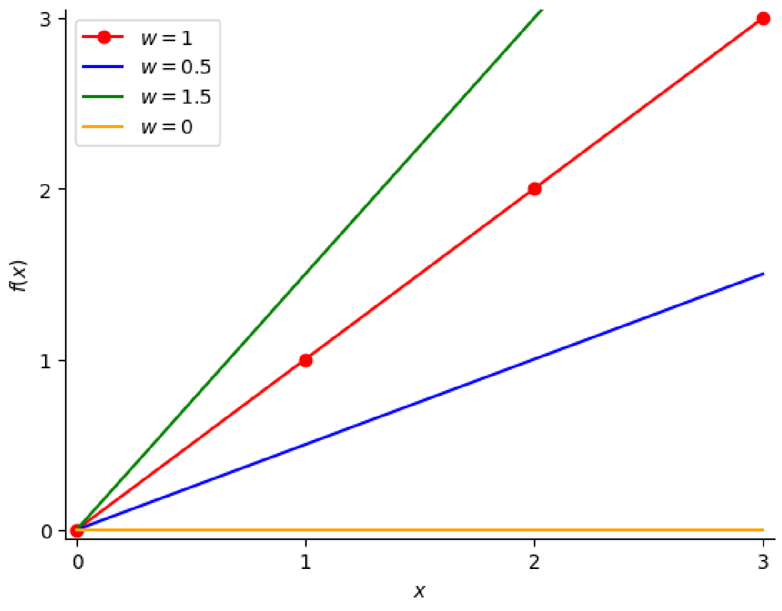
2.4.1 Cost Function of a Simplified Model

To visualize the cost function, a simplified model can be considered where $b = 0$.

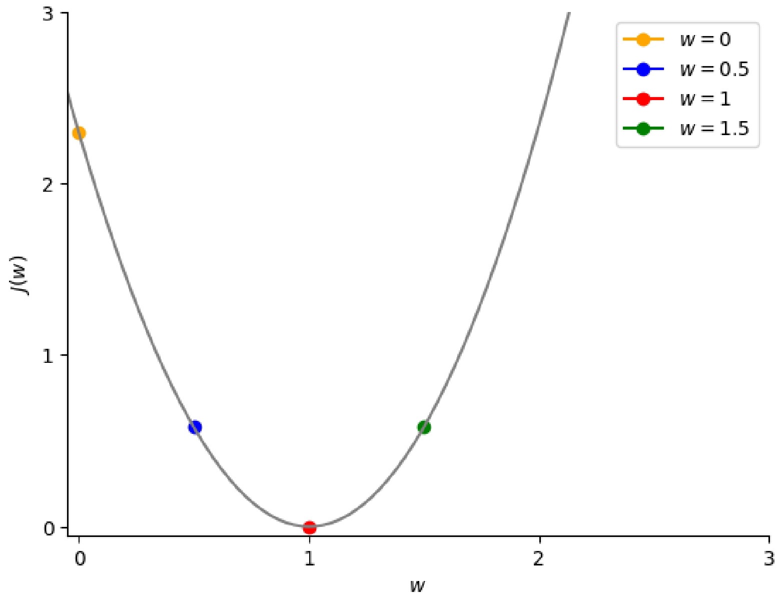
Model	: $f_w(x) = wx$
Parameters	: w
Cost function	: $J(w) = \frac{1}{2m}\sum_{i=1}^{m}(f_w(x^{(i)}) - y^{(i)})^2$
Goal	: Minimize $J(w)$

For example, given three training data points $(1,1),(2,2),(3,3)$, different choices of the parameter w will yield distinct linear regression models, each with its own associated cost.

$w = 0$	$w = 0.5$
$J(0) = \frac{1}{2\times3}[(0-1)^2 + (0-2)^2 + (0-3)^2)]$ $= \frac{1}{6}\times 14 \approx 2.3$	$J(0.5) = \frac{1}{2\times3}[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2)]$ $= \frac{1}{6}\times 3.5 \approx 0.58$
$w = 1$	$w = 1.5$
$J(1) = \frac{1}{2\times3}[(1-1)^2 + (2-2)^2 + (3-3)^2)]$ $= \frac{1}{6}\times 0 = 0$	$J(1.5) = \frac{1}{2\times3}[(0-1)^2 + (1.5-2)^2 + (4.5-3)^2)]$ $= \frac{1}{6}\times 14 \approx 0.58$



(a) Different linear regression models for different values of w



(b) Cost function values $J(w)$ at different weights w. The points indicate the calculated values, with the connecting line illustrating the trend.

Figure 2.4: Intuition of how cost function value changes with different values of w

2.4.2 Cost Function of the Original Univariate Linear Regression Model

In case of the original univariate linear regression, the model is

$$f_{w,b}(x) = wx + b$$

This model has 2 parameters w and b. Figure 2.5 shows a 3D plot and contour map depicting the cost for various combinations of w and b. Any single point on this surface represents some particular choice of w and b. In a contour plot, the ovals or circles indicate points on the 3D surface that are at the same height, meaning they share the same cost function value $J(w,b)$. In a contour plot, the minimum value of the cost function J is located at the center of the concentric ovals, which represents the lowest point of the bowl-shaped surface.

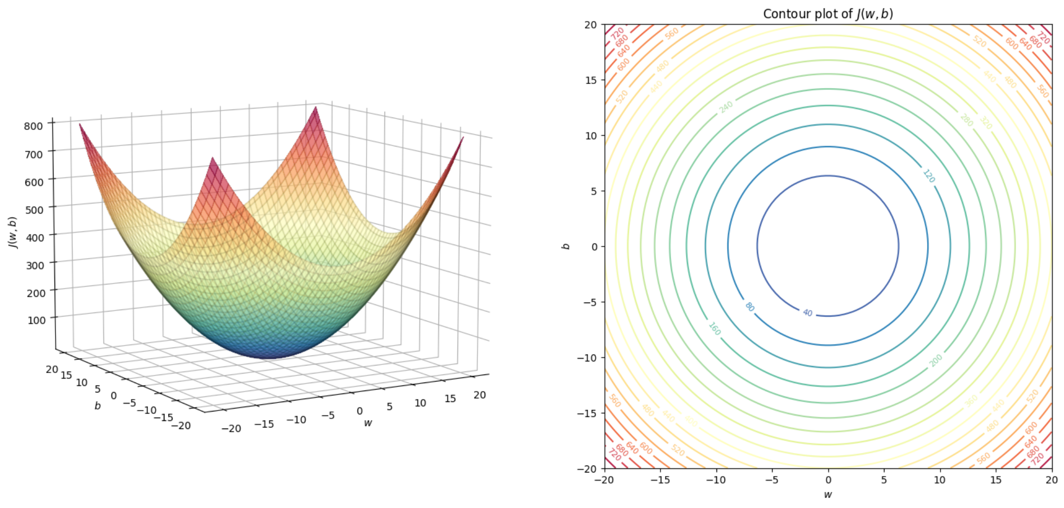


Figure 2.5: 3D plot and Contour plot illustrating the cost function $J(w,b)$ for different combinations of parameters w and b

For the sample dataset of housing prices, two models and their corresponding cost functions are illustrated in the contour plot shown in Figure 2.6. The blue line appears to provide a better fit for the dataset, and the contour plot indicates that the blue dot is positioned close to the center of the concentric ovals, suggesting a lower cost function value.

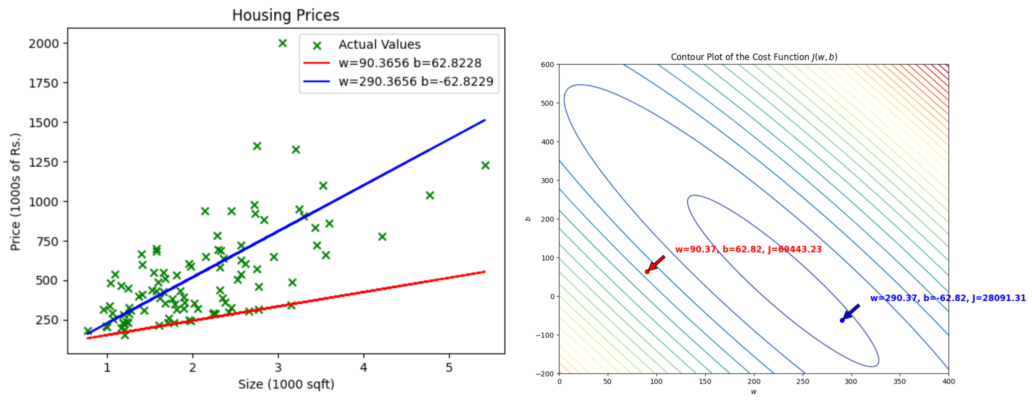


Figure 2.6: Contour plot illustrating the cost functions of two models applied to the housing prices dataset. The blue line represents the model that provides a superior fit, as indicated by its proximity to the center of the concentric ovals, which correspond to lower cost function values. This suggests that the blue model is more effective in minimizing the cost associated with predicting housing prices.

2.5 Gradient Descent Algorithm

Gradient Descent Algorithm is a systematic way to find the values of w and b, that results in the smallest possible cost, $J(w,b)$. Gradient descent is used not just for linear regression, but for training some of the most advanced neural network models, also called deep learning models.

Algorithm 1 Gradient Descent Algorithm

```
1: Choose an initial guess for the $w$ and $b$
2: Choose learning rate $\alpha$
3: repeat
4:     $w = w - \alpha \frac{\partial}{\partial w} J(w,b)$
5:     $b = b - \alpha \frac{\partial}{\partial b} J(w,b)$
6: until convergence
```

Note: w and b should be updated simultaneously.

The gradient descent algorithm continues to iterate until a stopping criterion is met. This stopping criterion could be based on reaching a maximum number of iterations or achieving a threshold where the change in the cost function value $J(w,b)$ between iterations falls below a certain value. The parameter α is called learning rate. The learning rate α determines the step size in each iteration of the gradient descent algorithm. By adjusting the learning rate, the size of the step taken can be controlled. A smaller learning rate results in smaller steps, which may improve convergence but may also slow down the optimization process. Conversely, a larger learning rate may lead to faster convergence but could result in overshooting or oscillations.

In gradient descent algorithm w and b should be updated simultaneously. The correct way to implement gradient descent update is given below.

```
1: $tempw = w - \alpha \frac{\partial}{\partial w} J(w,b)$
2: $tempb = b - \alpha \frac{\partial}{\partial b} J(w,b)$
3: $w = tempw$
4: $b = tempb$
```

2.5.1 Gradient Descent Intuition

To develop an intuition for gradient descent, consider a simpler model with just one parameter w. In this case, the gradient descent algorithm has only one update step, as shown below:

$$w = w - \alpha \frac{\partial}{\partial w} J(w)$$

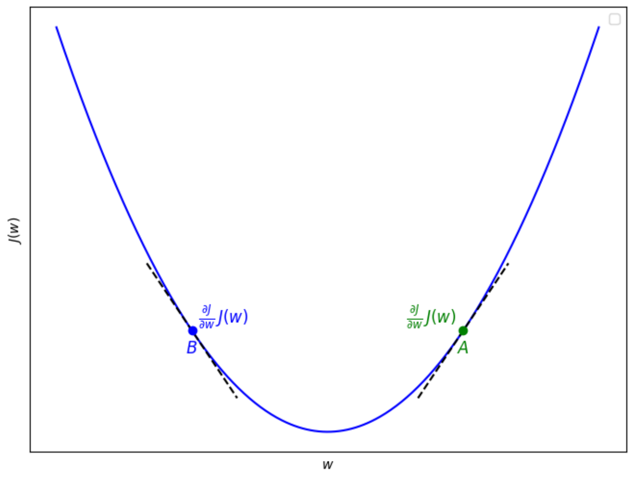


Figure 2.7: Gradient Descent intuition

Figure 2.7 illustrates how the cost function $J(w)$ changes with different values of the parameter

w. If the current value of w in the gradient descent algorithm is at point A, the partial derivative of $J(w)$ at point A is represented by the slope of the tangent at that point. Since the slope of the tangent at point A is positive, the next update in the gradient descent algorithm will be:

$$\begin{aligned} w &= w - \alpha \frac{\partial}{\partial w} J(w) \\ &= w - \alpha(\text{positive number}) \end{aligned}$$

This update will decrease the value of w, moving it closer to the optimal value that minimizes $J(w)$.

Similarly, if the current value of w in the gradient descent algorithm is at point B, the partial derivative of $J(w)$ at point B is represented by the slope of the tangent at that point. Since the slope of the tangent at point B is negative, the next update in the gradient descent algorithm will be:

$$\begin{aligned} w &= w - \alpha \frac{\partial}{\partial w} J(w) \\ &= w - \alpha(\text{negative number}) \end{aligned}$$

This update will increase the value of w, moving it closer to the optimal value that minimizes $J(w)$.

2.5.2 Gradient Descent for Linear Regression

In univariate linear regression

$$\begin{aligned} &\text{Model} && : f_{w,b}(x) = wx + b \\ &\text{Cost function} && : J(w,b) = \tfrac{1}{2m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})^2 \end{aligned}$$

To implement gradient descent algorithm, the partial derivatives of $J(w,b)$ with respect to w and b are to be calculated.

$$\begin{aligned} \frac{\partial}{\partial w} J(w,b) &= \frac{\partial}{\partial w} \frac{1}{2m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})^2 \\ &= \frac{\partial}{\partial w} \frac{1}{2m} \sum_{i=1}^{m} (wx^{(i)} + b - y^{(i)})^2 \\ &= \frac{1}{2m} \times 2 \sum_{i=1}^{m} (wx^{(i)} + b - y^{(i)}) x^{(i)} \\ &= \frac{1}{m} \sum_{i=1}^{m} (wx^{(i)} + b - y^{(i)}) x^{(i)} \\ &= \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)} \end{aligned}$$

$$
\begin{aligned}
\frac{\partial}{\partial b} J(w,b) &= \frac{\partial}{\partial b} \frac{1}{2m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})^2 \\
&= \frac{\partial}{\partial b} \frac{1}{2m} \sum_{i=1}^{m} (wx^{(i)} + b - y^{(i)})^2 \\
&= \frac{1}{2m} \times 2 \sum_{i=1}^{m} (wx^{(i)} + b - y^{(i)}) \\
&= \frac{1}{m} \sum_{i=1}^{m} (wx^{(i)} + b - y^{(i)}) \\
&= \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})
\end{aligned}
$$

Therefore the gradient descent algorithm of univariate linear regression is given below.

Algorithm 2 Gradient Descent Algorithm for Univariate Linear Regression

```
1: Choose an initial guess for the $w$ and $b$
2: Choose learning rate $\alpha$
3: repeat
4:     $w = w - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)}$
5:     $b = b - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})$
6: until convergence
```

Note: w and b should be updated simultaneously.

2.6 Multiple Linear Regression

If the training set has more than one feature that can improve prediction, these features can be utilized in various ways to enhance the model's performance. In machine learning, leveraging multiple features, especially those that provide additional information, can significantly improve prediction accuracy. For example, if training dataset has feature like size, number of floors, age etc., then the values of these features can improve the prediction.

x_j	j^{th} input feature
y	Output feature
n	Number of input features
m	Number of training examples in the training set
$\vec{x}^{(i)}$	Features of i^{th} training example.
$\vec{x}_j^{(i)}$	Value of j^{th} feature in the i^{th} training example.

Table 2.2: Notations - 2

For example, in the training set given below, $\vec{x}^{(2)} = [100, 2, 12, 1800]$, $\vec{x}_2^{(2)} = 2$.

Sl. No.	size (sq. ft.)	no. of floors	age	price
1	2000	1	10	3000
2	1000	2	12	1800
3	1500	2	5	2100
⋮	⋮	⋮	⋮	⋮
120	800	1	10	1200

Linear regression with multiple input features is known as **multiple linear regression**. In multiple linear regression, the model can be written as

$$\begin{aligned} f_{\vec{w},b}(\vec{x}) &= w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_nx_n + b \\ &= \vec{w}.\vec{x} + b \end{aligned}$$

where $\vec{w} = [w_1, w_2, \ldots, w_n]$ and $\vec{x} = [x_1, x_2, \ldots, x_n]$.

In multiple linear regression

Model	: $f_{\vec{w},b}(\vec{x}) = \vec{w}.\vec{x} + b$
Parameters	: $\vec{w}$, b
Cost function	: $J(\vec{w},b) = \frac{1}{2m}\sum_{i=1}^{m}(f_{\vec{w},b}(x^{(i)}) - y^{(i)})^2$
Goal	: Minimize $J(\vec{w},b)$

2.6.1 Gradient Descent for Multiple Linear Regression

The gradient descent algorithm for multiple linear regression is given below.

Algorithm 3 Gradient Descent Algorithm for Multiple Linear Regression

```
Choose an initial guess for $w_1$, $w_2$, $w_3$, $\ldots w_n$ and $b$
Choose learning rate $\alpha$
repeat
    $w_1 = w_1 - \alpha\frac{1}{m}\sum_{i=1}^{m}(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})x_1^{(i)}$
    $w_2 = w_2 - \alpha\frac{1}{m}\sum_{i=1}^{m}(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})x_2^{(i)}$
        ⋮
    $w_n = w_n - \alpha\frac{1}{m}\sum_{i=1}^{m}(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})x_n^{(i)}$
    $b = b - \alpha\frac{1}{m}\sum_{i=1}^{m}(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})$
until convergence
Note: $w_1$, $w_2$, $w_3$, $\ldots w_n$ and $b$ should be updated simultaneously.
```

2.7 Feature Scaling

When the different features have very different ranges of values, it can cause gradient descent to converge slowly. The algorithm may run longer to find the optimal parameters if features with larger values have more influence on the updates than smaller ones. Rescaling the features so they all take on comparable ranges can significantly improve the speed of gradient descent. Techniques such as normalization or standardization are commonly used to transform features to a similar scale, leading to faster and more efficient convergence during the optimization process.

2.7.1 Feature Scaling Techniques

- To scale features that differ largely in range, one approach is to rescale the values by dividing each feature by its maximum value. This brings the range of each feature between 0 and 1. For instance, if a house size feature ranges from 500 to 2000, after rescaling, the new values will fall between 0 and 1.
- **Mean normalization** uses the following equation to rescale the values.

$$x_{scaled} = \frac{x - \mu}{x_{max} - x_{min}}$$

  where
  - μ is the mean of the feature values,
  - x is the original value of the feature,
  - x_{max} is the maximum value of the feature,
  - x_{min} is the minimum value of the feature

  Mean normalization often scales the values between -1 and +1.
- **Z-score normalization** uses the following equation to rescale the values.

$$x_{scaled} = \frac{x - \mu}{\sigma}$$

  where
  - μ is the mean of the feature values
  - x is the original value of the feature
  - σ is the standard deviation

2.7.2 When to normalize?

Ideally, each feature may be rescaled in such a way that, the value ranges between -1 and 1, i.e. $-1 \leq x_j \leq 1$ for each feature x_j. he following table shows some of the ranges that require scaling and those that do not.

Range	**Requires scaling?**
$-3 \leq x_j \leq 3$	No
$-3 \leq x_j \leq 3$	No
$-0.3 \leq x_j \leq 0.3$	No
$0 \leq x_j \leq 3$	No
$-2 \leq x_j \leq 0.5$	No
$-100 \leq x_j \leq 100$	Yes
$-0.001 \leq x_j \leq 0.001$	Yes
$99.25 \leq x_j \leq 105$	Yes

2.8 Choosing the Learning Rate

A well-chosen learning rate is essential for an algorithm's performance. If the learning rate is too small, the algorithm will run very slowly, and if it's too large, it might not converge at all. To find a good learning rate, one can plot the cost over a number of iterations. If the cost fluctuates, going up and down, it indicates that gradient descent isn't working properly. This could be due to a bug in the code, or a high learning rate. Sometimes the cost may consistently increase after each iteration, which is likely due to a learning rate that is too large. This issue can usually be fixed by selecting a smaller learning rate.

A key trade-off in selecting the learning rate is that if it's too small, gradient descent can take many iterations to converge. To find a good learning rate, it's common to test a range of values for alpha. One might start with 0.001, then try larger values like 0.01 or 0.1. For each value, gradient descent can be run for a few iterations and plot the cost function J over time. After testing different values, choose the learning rate that decreases the cost steadily and quickly. Another common approach is to increase the learning rate in smaller steps, like starting with 0.001, then increasing it threefold to 0.003, then to 0.01, and so on.

2.9 Feature Engineering

Feature engineering is the process of transforming raw data into meaningful inputs that help improve the performance of machine learning models. It involves creating new features or modifying existing ones to better represent the underlying patterns in the data. For example, when working with a dataset about houses, a new feature like *area of the plot* can be created from the existing *frontage* and *depth* features. The goal of feature engineering is to make the data more informative and useful for the model, which can lead to better predictions.

2.10 Polynomial Regression

Polynomial regression is used to fit curves or non-linear functions to data. For example, in the housing price dataset, as shown in the figure 2.8, a straight line may not provide a good fit.

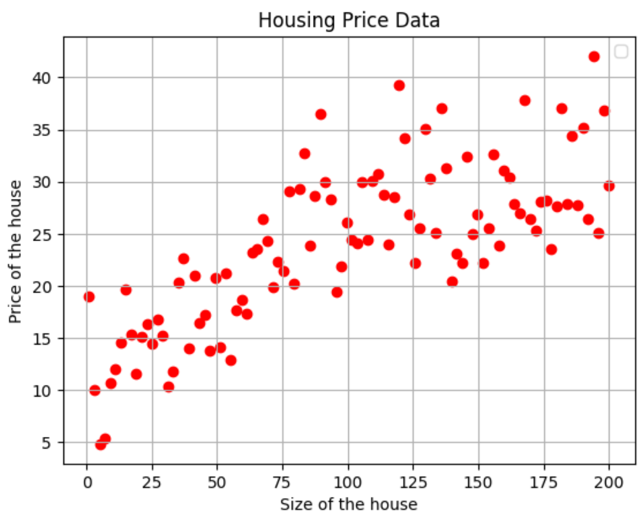


Figure 2.8: Training dataset for housing price

A **quadratic function** could be applied to the data, incorporating both the size x and x^2, which represents size raised to the power of two.

$$f_{\vec{w},b} = w_1x + w_2x^2 + b$$

This might provide a better fit to the data. However, it may be determined that the quadratic model doesn't make sense in this context, as a quadratic function eventually curves downward (Figure 2.9). In reality, housing prices are not expected to decrease as the size of the house increases.

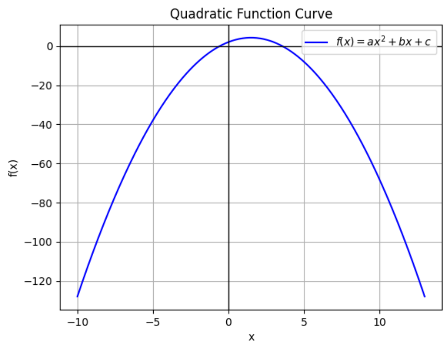


Figure 2.9: Quadratic function curve

So, instead of quadratic function, a **cubic function**, $f_{\vec{w},b} = w_1x + w_2x^2 + w_3x^3 + b$ (Figure 2.10) or a **square root function**, $f_{\vec{w},b} = w_1x + w_2\sqrt{x} + b$ (Figure 2.11) may be applied.

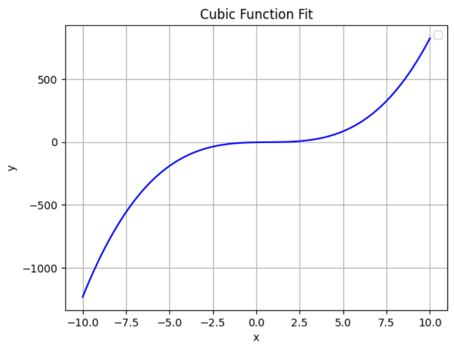


Figure 2.10: Cubic function curve

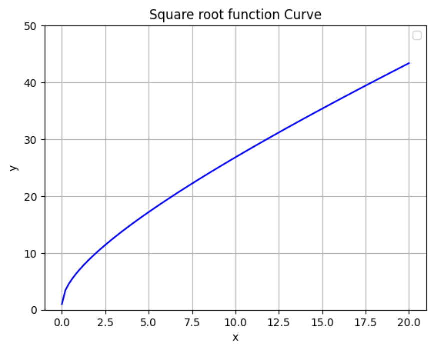


Figure 2.11: Cubic function curve

3. Classification

3.1 Introduction

Classification is used to predict the categorical class labels of new instances based on past observations. Some of the examples for classification problem are

- Classify whether an email is spam or not.
- Detect whether a financial transaction is fraudulent or not.
- Given the features of a tumour, predict whether it is malignant or not.

These types of classification problems where there are only two possible outputs are known as **binary classification** problem. The two classes are commonly represented as 0 for 'False' (No) and 1 for 'True' (Yes). One common convention in binary classification is to refer to the false or zero class as the negative class, and the true or one class as the positive class. For example, in spam classification, an email that is not spam is called a negative example. This is because the answer to the question "Is it spam?" is no, or 0. In contrast, an email that is spam is referred to as a positive example, as the answer to "Is it spam?" is yes, or 1.

3.2 k Nearest Neighbour Classification

The k-Nearest Neighbors (k-NN) algorithm is used mostly for solving classification problems. The k-NN algorithm compares a new data entry to the values in a given data set (with different classes or categories). Based on its closeness or similarities in a given range (K) of neighbours, the algorithm assigns the new data to a class or category in the data set. k-NN algorithm is described below.

Algorithm 4 k-NN Algorithm

```
1: Assign a value to $k$.
2: Calculate the distance between the new data entry and all other existing data entries using any
   one of the distance measures. Arrange them in ascending order.
3: Find the $k$ nearest neighbours to the new entry based on the calculated distances.
4: Assign the new data entry to the majority class in the nearest neighbours.
```

The distance between data points can be measured using Euclidean distance or Manhattan Distance. The Euclidean distance is given be the equation

$$d = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

■ **Example 3.1** The table below provides a training data set containing six observations, three predictors, and one qualitative response variable. Suppose we wish to use this data set to make a prediction for Y when $X_1 = X_2 = X_3 = 0$ using K-nearest neighbors.

	X_1	X_2	X_3	Y
1	0	3	0	Red
2	2	0	0	Red
3	0	1	3	Red
4	0	1	2	Green
5	-1	0	1	Green
6	1	1	1	Red

Euclidean distance between each observation and the test point, $Z = (X_1, X_2, X_3) = (0,0,0)$ can be computed as follows:

	X_1	X_2	X_3	Y	$d(X,Z)$
1	0	3	0	Red	3.000
2	2	0	0	Red	2.000
3	0	1	3	Red	3.162
4	0	1	2	Green	2.236
5	-1	0	1	Green	1.414
6	1	1	1	Red	1.732

- **When $K = 1$**
  The single nearest neighbour of Z is observation 5. Hence Z is classified as Green.
- **When $K = 3$**
  The three nearest neighbours of Z are observations 5 (Green), 6 (Red) and 2 (Red). Out of three neighbours two are Red. Hence Z is classified as Red.

■

3.3 Naïve Bayes Classification

The Naïve Bayes algorithm is a supervised learning technique based on Bayes' Theorem, and it is used for classification tasks. It is especially useful for text classification, where the training data involves many features. It is called Naïve because it assumes that each feature is independent of the others when making predictions.

Naïve Bayes is a simple yet powerful classifier that is known for building fast machine learning models capable of making quick predictions. As a probabilistic classifier, it makes predictions based on the likelihood or probability of an event occurring.

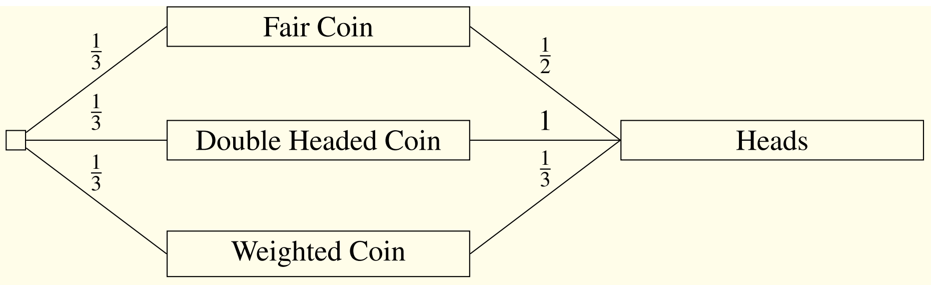
Some common examples of where Naïve Bayes is used include spam filtering, sentiment analysis, and classifying news articles.

3.3.1 Total Probability Theorem

Definition 3.1. *If $E_1, E_2, E_3, \ldots$ are mutually exclusive and exhaustive events, and A is an event that can occur with one of $E_1, E_2, E_3, \ldots$, then*

$$P(A) = \sum_{i=1}^{n} P(E_i).P(A/E_i)$$

■ **Example 3.2** A box contains 3 coins: one fair coin, one double headed and one weighted coin $P(H) = \frac{1}{3}$. A coin is selected at random and tossed. Find the probability that head appears.



$$P(H) = \left(\frac{1}{3}.\frac{1}{2}\right) + \left(\frac{1}{3}.1\right) + \left(\frac{1}{3}.\frac{1}{3}\right) = \frac{11}{18}$$

■

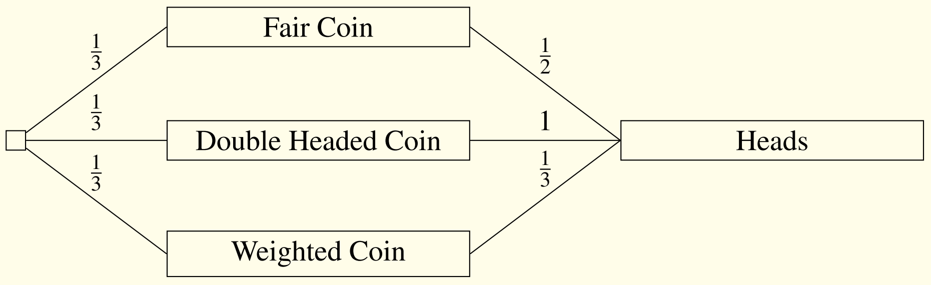
3.3.2 Bayes' Theorem

Definition 3.2.

$$P(Y/X) = \frac{P(X/Y) * P(Y)}{P(X)}$$

$$P(Y/X_1, X_2, \dots X_n) = \frac{P(X_1/Y)P(X_2/Y) \dots * P(Y)}{P(X_1) * P(X_2) * \dots P(X_n)}$$

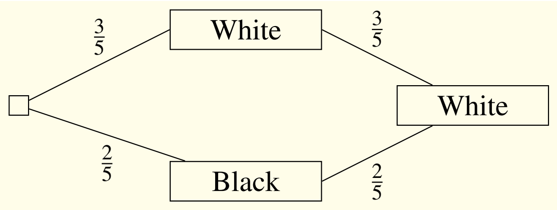
■ **Example 3.3** A box contains 3 coins: one fair coin, one double headed and one weighted coin $P(H) = \frac{1}{3}$. A coin is selected at random and tossed. Find the probability that a fair coin was selected given that a head is obtained.



$$P(\text{Fair Coin}/H) = \frac{\frac{1}{3}.\frac{1}{2}}{\left(\frac{1}{3}.\frac{1}{2}\right) + \left(\frac{1}{3}.1\right) + \left(\frac{1}{3}.\frac{1}{3}\right)} = \frac{3}{11}$$

■

■ **Example 3.4** Bag A contains 3 white and 2 black balls. Bag B contains 2 white and 2 black balls. One ball is drawn at random from A and transferred to B. One ball is drawn at random from B and is found to be white. Find the probability that the transferred ball is white.

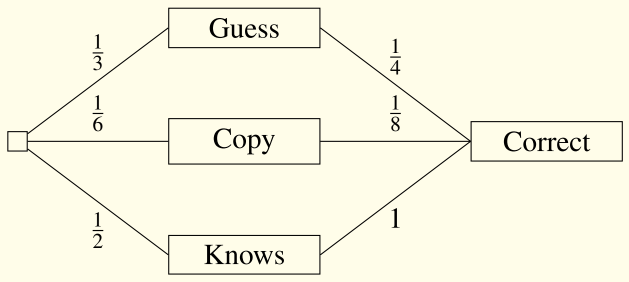


Let E be the event that first ball transferred from bag A is white and W be an event that the ball finally obtained is white.

$$P(E/W) = \frac{\frac{3}{5} \cdot \frac{3}{5}}{\left(\frac{3}{5} \cdot \frac{3}{5}\right) + \left(\frac{2}{5} \cdot \frac{2}{5}\right)} = \frac{9}{13}$$

Exercise 3.1 In a factory which manufactures bolts, machines A, B and C manufactures 25%, 35% and 40% of the bolts respectively. Of their outputs, 5, 4 and 2 percent are respectively defective bolts. A bolt is drawn at random from the products and is found to be defective. What is the probability that it is manufactured by Machine B? (Ans: $\frac{140}{345}$)

■ **Example 3.5** In a test, an examinee either guesses or copies or knows the correct answer for a multiple choice question having 4 choices of which exactly one is correct. The probability that he makes a guess is $\frac{1}{3}$ and the probability for copying is $\frac{1}{6}$. The probability that the answer is correct, given he copied it is $\frac{1}{8}$. Find the probability that he knows the answer given that his answer is correct.



$$P(\text{Knows}/\text{Correct}) = \frac{\frac{1}{2}}{\left(\frac{1}{3} \cdot \frac{1}{4}\right) + \left(\frac{1}{6} \cdot \frac{1}{8}\right) + \left(\frac{1}{2} .1\right)} = \frac{24}{29}$$

3.3.3 Applying Bayes' Theorem for Classification

Naïve Bayes' is algorithm is trained on a labelled dataset, where it learns the probability of each class and the probability of each feature given a class. When a new data point is given, the algorithm calculates the probability for each class using Bayes' theorem. It then chooses the class with the highest probability.

Description 3.3.1 The denominator in Bayes' Theorem can be ignored when applying it for classification because it is the same for all classes. Since the goal is to compare probabilities across classes, the denominator doesn't affect which class has the highest probability. Therefore, it can be omitted.

Example 3.6 Consider the following dataset of movies and predict whether a movie with genres comedy and action will be a success or not.

Movie	**Action**	**Comedy**	**Success**
1	Yes	No	Yes
2	No	Yes	Yes
3	Yes	Yes	Yes
4	No	No	No
5	Yes	No	Yes
6	No	No	Yes
7	Yes	No	Yes
8	Yes	No	No
9	No	Yes	Yes
10	No	Yes	No

From the dataset given

$$P(\text{Success} = \text{Yes}) = \frac{7}{10}$$
$$P(\text{Success} = \text{No}) = \frac{3}{10}$$

The conditional probabilities for genres are given below.

	Success	
	Yes	No
Action	$\frac{4}{7}$	$\frac{2}{3}$
Comedy	$\frac{3}{7}$	$\frac{2}{3}$

$$\begin{aligned} P(\text{Success=Yes}/\text{Action=Yes, Comedy=Yes}) &= \frac{4}{7} \times \frac{3}{7} * \frac{7}{10} \\ &= 0.17 \\ P(\text{Success=No}/\text{Action=Yes, Comedy=Yes}) &= \frac{2}{3} \times \frac{2}{3} * \frac{3}{10} \\ &= 0.13 \end{aligned}$$

Hence, according to the Naïve Bayes' classification, the new movie will be a success. ■

3.4 Decision Trees

A decision tree is a straightforward model used for supervised classification tasks. It classifies a single discrete target feature by making decisions at each internal node based on a Boolean test of an input feature. The edges of the tree represent the possible values of that input feature, while the leaf nodes provide the final classification. To classify an example, one starts at the root of the

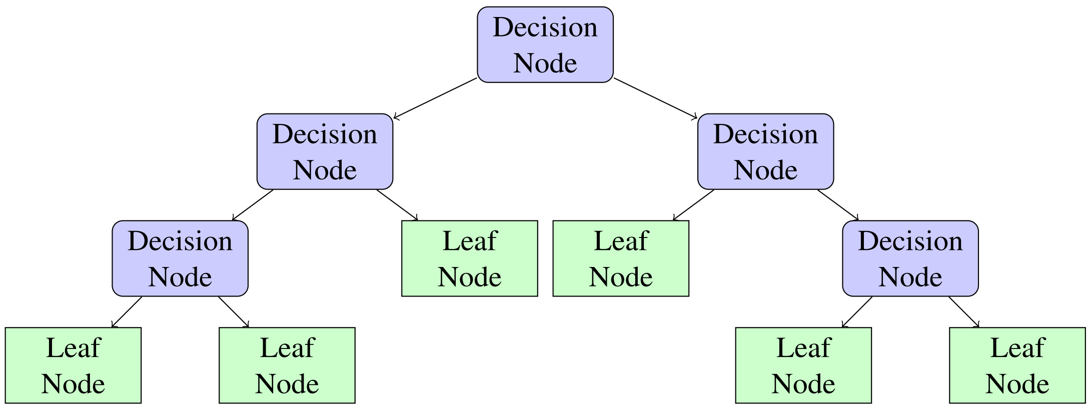


Figure 3.1: Decision Tree Structure

tree, moves down through the nodes by evaluating the tests and following the appropriate edges, and finally reaches a leaf node, which gives the classification result. Structure of a decision tree is shown in Figure 3.1.

3.4.1 Decision Tree Terminology

- **Root Node:** Root node is the node from where the decision tree starts.
- **Leaf Node:** Leaf nodes are the final output node.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.

3.4.2 ID3 (Iterative Dichotomiser 3) Algorithm

The ID3 algorithm (Iterative Dichotomiser 3) is a popular method used to create decision trees in machine learning. It works by selecting the feature that best splits the data based on information gain, a measure of how well a feature separates the data into distinct classes. The algorithm starts with the entire dataset and repeatedly splits it into smaller subsets using the best feature until all the data points in a subset belong to the same class or no more useful features are left. ID3 is commonly used for classification tasks, like predicting whether an email is spam or not, and is known for producing simple, easy-to-understand decision trees. The steps for ID3 algorithm are

1. If all examples in the current subset belong to the same class, the tree stops growing, and a leaf node is returned.
2. If there are no more features to split on, return a leaf node with the most frequent class.
3. Calculate Information Gain for each feature and choose the feature that best splits the data.
4. For each value of the chosen feature, recursively split the data and build the tree until the base cases are reached (pure class or no features).

Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as

$$Entropy(S) = \sum_{i}^{n} -p_i \log_2 p_i$$

where p_i is the probability of class i and n is the total number of classes.

Information gain calculates how much information a feature provides about a class. According to the value of information gain, the node is split and the decision tree is built. A node/attribute having the highest information gain is split first. It can be calculated using the below formula:

$$IG(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where S is the dataset, $|S_v|$ is the set of rows in S for which the feature column A has value v.

■ **Example 3.7** Consider the following dataset. Construct a decision tree to determine whether a boy is allowed to play cricket or not.

Day	Weather	Temperature	Humidity	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Cloudy	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Cloudy	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Cloudy	Mild	High	Strong	Yes
13	Cloudy	Hot	Normal	Weak	Yes
14	Rain	Hot	High	Strong	No

- Calculate Information Gain of weather
  1. Entropy of the entire dataset

$$E(S)\{+9,-5\} = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94$$

  2. Entropy of all attributes

$$E(\text{Sunny})\{+2,-3\} = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.97$$
$$E(\text{Cloudy})\{+4,-0\} = -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4} = 0$$
$$E(\text{Rain})\{+3,-2\} = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.97$$

$$IG(\text{Weather}) = E(S) - \frac{5}{14}E(\text{Sunny}) - \frac{4}{14}E(\text{Cloudy}) - \frac{5}{14}E(\text{Rain}) = 0.246$$

- Similarly the information gains of other attributes are

$$\begin{aligned} IG(\text{Temperature}) &= 0.029 \\ IG(\text{Humidity}) &= 0.15 \\ IG(\text{Wind}) &= 0.0478 \end{aligned}$$

Out of the four attributes, Weather has the highest value for information gain. Hence, Weather is selected as the root node for the decision tree.

Repeat the same procedure on the subset of the dataset where Weather is Sunny.

Day	Weather	Temperature	Humidity	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

- Calculate Information Gain of Temperature
  1. Entropy of the entire dataset

$$E(\text{Sunny})\{+2,-3\} = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.97$$

  2. Entropy of all attributes

$$\begin{aligned} E(\text{Hot})\{+0,-2\} &= -\frac{0}{2}\log_2\frac{0}{2} - \frac{2}{2}\log_2\frac{2}{2} = 0 \\ E(\text{Mild})\{+1,-1\} &= -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1 \\ E(\text{Cool})\{+1,-0\} &= -\frac{1}{1}\log_2\frac{1}{1} - \frac{0}{1}\log_2\frac{0}{1} = 0 \end{aligned}$$

$$IG(\text{Temperature}) = E(\text{Sunny}) - \frac{2}{5}E(\text{Hot}) - \frac{2}{5}E(\text{Mild}) - \frac{1}{5}E(\text{Cool}) = 0.57$$

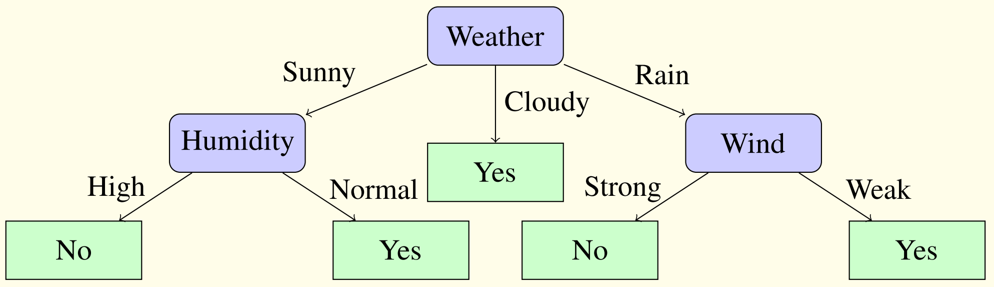
- Similarly the information gains of other attributes are

$$\begin{aligned} IG(\text{Humidity}) &= 0.97 \\ IG(\text{Wind}) &= 0.019 \end{aligned}$$

Out of the three attributes, Humidity has the highest value for information gain. Hence, Humidity is selected as the root node for the decision tree.

Repeat the same procedure on the subset of the dataset where Weather is Rain.

$$IG(\text{Temperature}) = 0.019$$
$$IG(\text{Humidity}) = 0.019$$
$$IG(\text{Wind}) = 0.97$$



3.5 Random Forest

Random Forest is a popular machine learning algorithm that falls under supervised learning. It can be used for both classification and regression tasks. The algorithm is based on ensemble learning, which combines multiple classifiers to tackle complex problems. This approach helps improve the model's performance by making more accurate predictions. Instead of relying on a single decision tree, the random forest algorithm aggregates predictions from multiple trees. It uses the majority vote from these predictions to determine the final output. A greater number of trees in the forest enhances accuracy and helps mitigate the risk of overfitting.

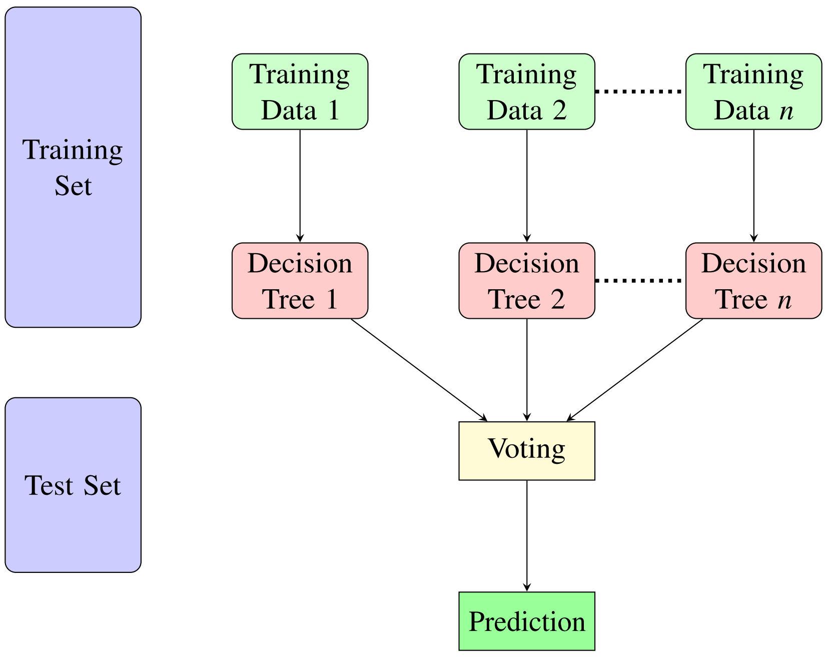


Figure 3.2: Random Forest

Advantages of random forest are

1. It requires less training time compared to many other algorithms.
2. It delivers highly accurate predictions and runs efficiently, even on large datasets.

3. It maintains accuracy even when a significant portion of the data is missing.

3.5.1 Steps in Random Forest Algorithm

Step 1 Create a random bootstrap dataset with k data points, from the original dataset.
Step 2 Create a decision tree for this random bootstrap dataset.
Step 3 Decide on the number of decision trees N, to be created.
Step 4 Repeat Step 1 and 2.
Step 5 For new data points, obtain predictions from each decision tree, assign the data points to the category that receives the majority of votes.

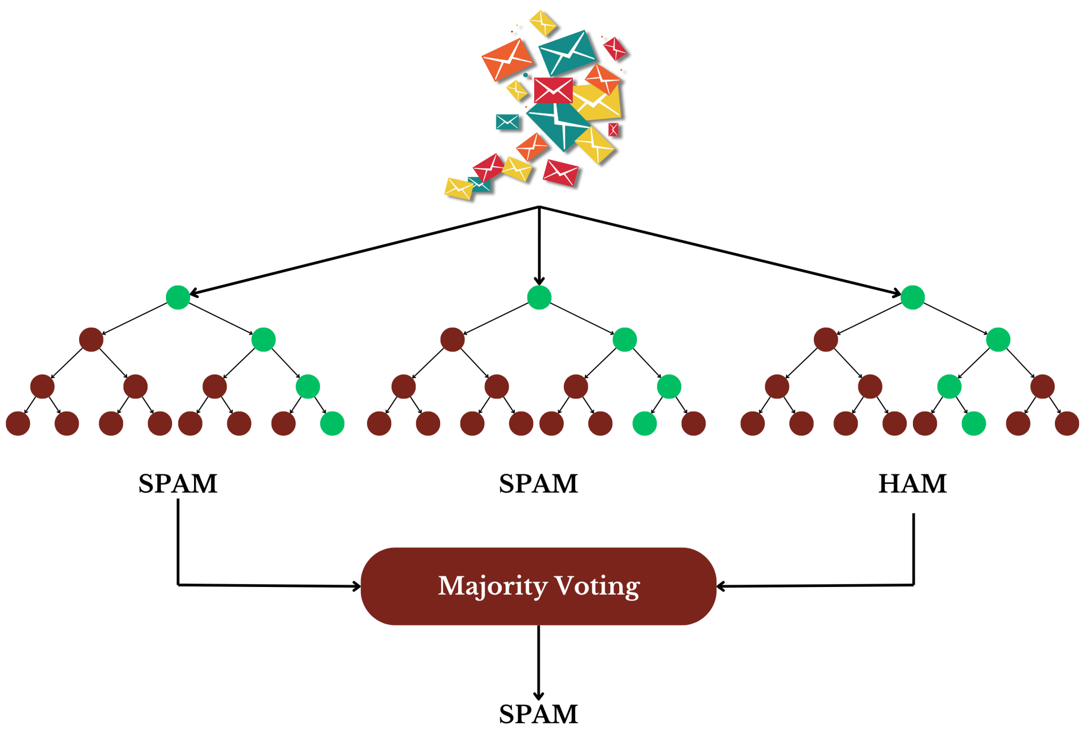


Figure 3.3: Example for Random forest classification

3.6 Support Vector Machines

3.6.1 Mathematics for SVM

The general equation of a line is

$$Ax + By + C = 0$$

where A, B and C are real numbers and $A, B \neq 0$.

The slope intercept form of a straight line is given by the equation

$$y = mx + b$$

where m is the slope and b is the y intercept.

So the general equation can be converted to slope intercept form as

$$y = -\frac{A}{B}x - \frac{C}{B}$$

Hence, the slope is $-\frac{A}{B}$ and the y intercept is $-\frac{C}{B}$.

Figure 3.4 shows a line $y = 0.5x + 1$, the general form of which is given as

$$-2x + 4y - 4 = 0$$

.

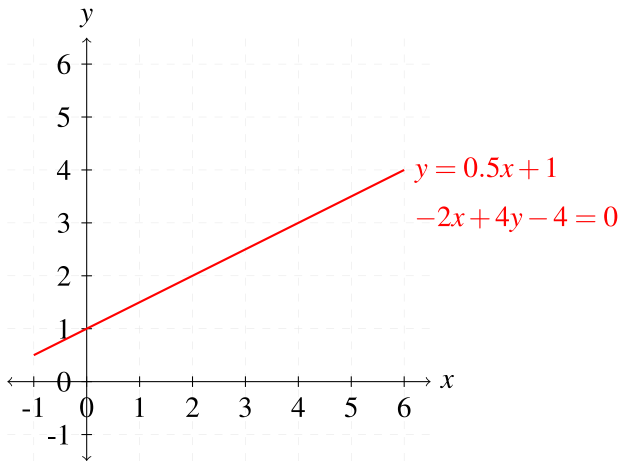


Figure 3.4: Plot representing the line $y = 0.5x + 1$

In this example, $A = -2$, $B = 4$ and $C = -4$. Changing only the value of A will change the slope of the line and the line will rotate around the point $(0, 1)$. Changing only the value of C will change the y-intercept of the line and the line will shift above or below the point $(0, 1)$. Changing only the value of B will change the slope and intercept of the line.

A feature of the general equation for the line is that, it can tell whether a data point is located above or below the line. If the values for x and y coordinates of the point are substituted in the equation, it will result in a positive value if the point is above the line and in a negative value if the point is below the line. For example, the point (5,4) is located above the line and the value for the equation is $-2 \times 5 + 4 \times 4 - 4 = 2$. Similarly, the point (5,3) is located below the line and the value for the equation is $-2 \times 5 + 4 \times 3 - 4 = -2$.

If 1 is added to the right hand side of the equation, the general equation becomes

$$\begin{aligned} -2x + 4y - 4 &= 1 \\ \text{i.e.} \ -2x + 4y - 5 &= 0 \end{aligned}$$

The slope of the new line remains the same, while the y-intercept is shifted to 1.25.

Similarly, If -1 is added to the right hand side of the equation, the general equation becomes

$$\begin{aligned} -2x + 4y - 4 &= -1 \\ \text{i.e.} \ -2x + 4y - 3 &= 0 \end{aligned}$$

The slope of the new line remains the same, while the y-intercept is shifted to 0.75 (Figure 3.5).

The distance between a point (x_0, y_0) and a line $Ax + By + C = 0$ is given by the equation

$$d = \frac{|Ax_0 + By_0 + C|}{\sqrt{A^2 + B^2}}$$

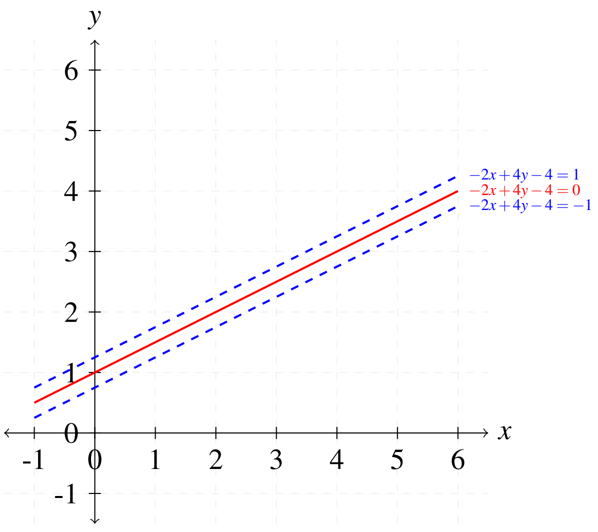


Figure 3.5: Lines with adjusted C values

The distance between two parallel lines $Ax+By+C_1=0$ and $Ax+By+C_2=0$ is given by the equation

$$d = \frac{|C_2 - C_1|}{\sqrt{A^2+B^2}}$$

The distance between the two lines $Ax+By+C=1$ and $Ax+By+C=-1$ is

$$d = \frac{2}{\sqrt{A^2+B^2}}$$

3.6.2 SVM Working Principle

Support Vector Machine (SVM) is a supervised learning algorithm used for classification and regression tasks. SVM tries to find a **hyperplane** that best separates the data points into different classes. A hyperplane is a boundary that separates different classes in the feature space. In 2D, the hyperplane is a line. In higher dimensions (3D, 4D, etc.), it becomes a plane or hyperplane. The data points from each class, that are closest to the hyperplane are known as **Support Vectors**. These points define the margin and influence the position and orientation of the hyperplane. The hyperplane is adjusted to be equidistant from the support vectors of each class.

The **margin** is the distance between the hyperplane and the support vectors. The objective of SVM is to maximize the margin between the classes while ensuring the data points are correctly classified. This is solved through convex optimization techniques, like quadratic programming.

For example, consider the following dataset.

Class	x_1	x_2
A(+1)	1	4
A(+1)	2	5
A(+1)	3	5
A(+1)	3	4
B(-1)	6	1
B(-1)	4	0
B(-1)	5	2
B(-1)	5	1

In this dataset, there are two classes, A and B. In Support Vector Machines (SVM), one class is typically referred to as the positive class (+1), and the other as the negative class (-1). The best hyperplane for this dataset and the support vectors are shown in Figure 3.6.

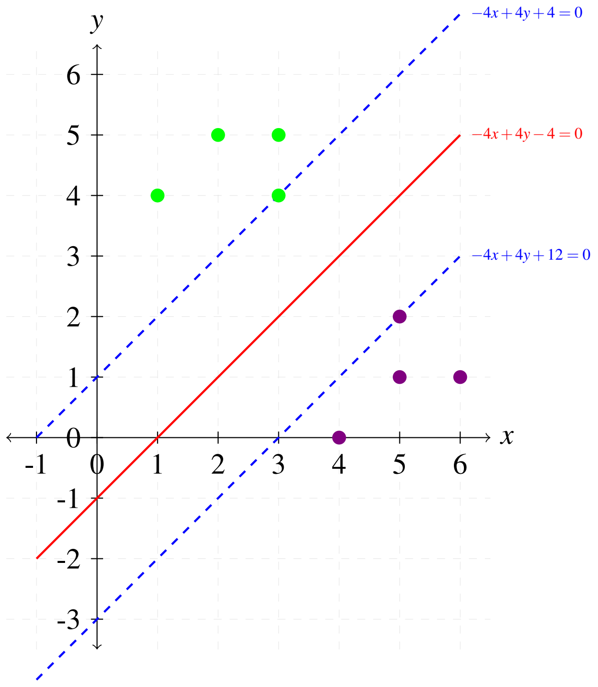


Figure 3.6: Hyperplane and the support vectors for the given dataset

The equation of the hyperplane is

$$-4x+4y+4=0$$

The equation for the line passing through the support vector in the positive class is

$$-4x+4y-4=0$$

The equation for the line passing through the support vector in the negative class is

$$-4x+4y+12=0$$

The equation for the hyperplane can be normalized in such a way that the left hand side of the equation for the line passing through the support vector in the negative class is equal to -1 and left hand side of the equation for the line passing through the support vector in the positive class is equal to 1 with constants for the three equations being the same. This can be achieved by finding the value for k, such that

$$k(-4x+4y+4)=1$$

The value of k can be found by substituting the values for the x and y coordinates of the support vector in the positive class.

$$\begin{aligned} k(-4x+4y+4) &= 1 \\ k(-4\times 3+4\times 4+4) &= 1 \\ k\times 8 &= 1 \\ \therefore k &= \frac{1}{8} \end{aligned}$$

The equations for the three lines can now be written as

$$\begin{aligned} -0.5x+0.5y+0.5 &= 1 \\ -0.5x+0.5y+0.5 &= 0 \\ -0.5x+0.5y+0.5 &= -1 \end{aligned}$$

The general form of the three lines is

$$\begin{aligned} &\vec{w}\ .\ \vec{x}+b=1 \\ &\vec{w}\ .\ \vec{x}+b=0 \text{ (hyperplane)} \\ &\vec{w}\ .\ \vec{x}+b=-1 \end{aligned}$$

where w is the weight vector, x is the input data and b is the bias.

The class of a new data point can be predicted by substituting its x and y coordinates into the equation of the hyperplane. If the result is less than zero, the point is classified as belonging to the negative class (e.g., -1). If the result is greater than zero, the point is classified as belonging to the positive class (e.g., +1). If the result is exactly zero, the point lies on the decision boundary between the two classes.

In SVM, Y_i usually defines the class of the training data as 1 and -1. For example, the given data set is represented as

$$Y_i=[1,1,1,1,-1,-1,-1,-1]$$

The classification in SVM is defined as

$$Y=\begin{cases} +1, & \text{if } \vec{w}\ .\ \vec{x}+b\geq 0 \\ -1, & \text{if } \vec{w}\ .\ \vec{x}+b<0 \end{cases}$$

The margin of the SVM is

$$d=\frac{2}{\sqrt{A^2+B^2}}=\frac{2}{||w||}$$

The SVM algorithm tries to find the values of w and b such that it maximizes the margin.

$$\max\frac{2}{||w||} \text{ such that } \vec{w}\ .\ \vec{x}_i+b\begin{cases} \geq 1, & \text{if } Y_i=+1 \\ \leq 1, & \text{if } Y_i=-1 \end{cases}$$

for all data points in the training dataset.

This equation can be simplified as given below.

$$\max \frac{2}{||w||} \text{ such that } Y_i(\vec{w}\ .\ \vec{x}_i + b) \geq 1$$

If the data is linearly separable, SVM finds a hyperplane that perfectly divides the data into two classes. For separable data, the support vectors lie exactly on the margin boundaries. If the data is not linearly separable, SVM uses a technique called the **kernel trick** to map the data into a higher-dimensional space where it becomes linearly separable. Common kernels include linear, polynomial, radial basis function (RBF), and sigmoid.

In some cases, the data may not be perfectly separable, and SVM allows for some misclassifications using a **soft margin**. This is controlled by a parameter C, which balances between making the margin larger and reducing the classification errors.

Unit III

4. Clustering

4.1 Introduction

Clustering is an unsupervised machine learning technique. It is used to group similar data points together based on the values of certain features. Since clusteirng is an unsupervised learning technique, it does not rely on labeled data; instead, it attempts to uncover the inherent structure of the data by identifying clusters, or groups, of similar objects. The goal of clustering is to organize a dataset into meaningful groups, such that objects in the same cluster are more similar to each other than to those in other clusters. The core of clustering is measuring similarity or distance between data points. Commonly used metrics are Euclidean distance, Manhattan distance, cosine similarity.

Clustering is used in many areas such as

- Data mining: Identifying patterns and grouping data in large datasets.
- Image segmentation: Grouping pixels or features in an image for analysis or processing.
- Market segmentation: Grouping customers with similar behaviors or characteristics.
- Anomaly detection: Identifying unusual patterns or outliers in data.

4.2 Requirements of Clustering

- **Scalability:** Clustering algorithms should be able to handle large datasets efficiently. As the size of the dataset increases, the algorithm's complexity and computational requirements should scale appropriately. For example, algorithms like k-means and DBSCAN are often chosen because they can scale to large datasets.

- **Ability to Handle Different Data Types:** Clustering algorithms should support various types of data, such as numerical, categorical, or a combination of both.

- **Ability to deal with noisy data:** The clustering algorithm should be robust to noise and outliers, as real-world data often contains noise. For exmaple, density-based methods like DBSCAN can handle noise effectively, as they can separate outliers from clusters.

- **High Dimensionality:** In many cases, data might have a high number of features (dimensions), which can make clustering more difficult due to the "curse of dimensionality". Effective clustering algorithms need to work well even in high-dimensional spaces or reduce dimensions as a pre-processing step. Algorithms like spectral clustering or Principal Component Analysis (PCA) combined with k-means can help with high-dimensional data.

- **Interpretability and Usability:** The clustering results should be easy to interpret and usable by non-experts. k-means is popular for its simplicity and interpretability

- **Ability to Handle Arbitrary Shaped Clusters:** The algorithm should be flexible enough to identify clusters of arbitrary shapes, not just spherical clusters. DBSCAN is effective for detecting clusters of arbitrary shapes, unlike k-means, which tends to find spherical clusters.

- **Minimal Input Parameters:** Clustering algorithms should require a minimal number of input parameters. The choice of parameters should not overly influence the results. For example, k-means requires the number of clusters k.

- **Ability to Handle Large Differences in Cluster Sizes:** Clustering algorithms should be able to handle data where clusters vary in size and density. Algorithms like hierarchical clustering can handle clusters of varying sizes better than k-means, which tends to equalize cluster sizes.

- **Efficiency in Memory Usage:** Efficient memory usage is critical when clustering large datasets. Algorithms that require large amounts of memory to store intermediate results or large distance matrices may not be practical for very large datasets. Hierarchical clustering can require significant memory for large datasets, whereas k-means can be more memory-efficient.

- **Handling Incremental Data:** Some clustering problems involve continuously incoming data (streaming data). The clustering algorithm should be able to update clusters incrementally as new data points arrive.

4.3 Types of Data in Cluster Analysis

4.3.1 Interval-Scaled Variables

Interval-scaled variables are quantitative data types. The characteristics of interval-scaled variables are

- **Ordered Values:** Interval-scaled variables have a defined order. They are ranked according to their values.
- **Equal Intervals:** The differences between values of interval-scaled variables are consistent and meaningful.
- **No True Zero:** Interval-scaled variables do not have an absolute zero point. For example, 0 degrees Celsius does not indicate the absence of temperature; it is just another point on the scale.
- **Arithmetic Operations:** Addition and subtraction are allowed on interval-scaled data, but multiplication and division are not applicable due to the lack of a true zero.

Examples for interval-scaled variables are temperature, dates

4.3.2 Binary Variables

Binary variables are a type of categorical variable that can take on only two possible values. These values typically represent two distinct categories or outcomes. They are often denoted as 0 and 1, or “True” and “False”. The characteristics of binary variables are

- **Two Categories:** Binary variables represent two mutually exclusive categories. Each observation in the dataset can belong to only one of these two categories.
- **Simplicity:** Binary variables simplify data representation and analysis.
- **Quantitative Representation:** Since binary variables can be represented numerically (e.g., 0 for one category and 1 for the other), they can be easily integrated into mathematical models.

Some examples for binary variables are

- Yes/No questions
- Presence/Absence indicators
- Success/Failure outcomes

4.3.3 Categorical Variables

Categorical variables represent distinct categories or groups. They can take on a fixed number of possible values, which are qualitative in nature. Categorical variables can be further classified into two main types: **nominal** and **ordinal**.

- In case of **nominal** categorical variables, categories do not have an intrinsic order or ranking among the categories. e.g. colours, gender,
- In case of **ordinal** categorical variables, categories have a meaningful order or ranking. e.g. educational qualifications, customer satisfaction ratings.

4.3.4 Ratio-scaled Variables

Ratio-scaled variables possess all the properties of interval-scaled variables with a meaningful zero point. This true zero allows for meaningful comparisons and interpretations of ratios between values. The characteristics of ratio-scaled variables are

- **Ordered Values:** ratio-scaled variables have a defined order. They are ranked according to their values.
- **Equal Intervals:** The differences between values of ratio-scaled variables are consistent and meaningful.
- **True Zero Point:** The presence of a true zero indicates the absence of the quantity being measured. This enables meaningful comparisons.
- **Arithmetic Operations:** Addition, subtraction, multiplication and division are allowed on ratio-scaled data.

Examples for ratio-scaled variable are height, weight, distance etc.

4.3.5 Variables of Mixed Types

Variables of mixed types refer to datasets that contain a combination of different types of variables, including numerical (interval, ratio), categorical (nominal, ordinal), binary, and sometimes even text or date variables. Characteristics of variables of mixed types are

- **Diverse Data Representation:** Mixed-type datasets can represent complex real-world date.
- **Complexity in Analysis:** The presence of different variable types can complicate data preprocessing, analysis, and the choice of appropriate algorithms.
- **Need for Specialized Techniques:** Mixed-type data often requires specific methods for analysis and modeling to handle each type of variable appropriately.

An example for mixed type variables is Customer Data which is a dataset containing:

- Age (ratio-scaled)
- Gender (categorical)
- Income (ratio-scaled)
- Satisfaction rating (ordinal)
- Membership status (binary: yes/no)

4.4 Types of Clustering Methods

Clustering methods can be broadly categorized into three main types:

- **Partitioning Methods:** Partitioning methods divide the dataset into distinct non-overlapping clusters such that each data point belongs to exactly one cluster. These methods often use a central point (centroid) to represent each cluster. In this method, the number of clusters must be predefined. Examples are **K-Means** clustering, **K-Medoids** clustering.

- **Hierarchical Methods:** Hierarchical methods create a hierarchy of clusters, which can be represented as a tree (dendrogram). Examples are **agglomerative** and **divisive**.

- **Density-Based methods:** Density-based methods identify clusters based on the density of data points in a given region. They can discover clusters of arbitrary shapes and sizes. Examples are **DBSCAN** (Density-Based Spatial Clustering of Applications with Noise), **OPTICS** (Ordering Points To Identify the Clustering Structure).

4.5 Clustering Using Partitioning Method

Partitioning methods divide the dataset into distinct non-overlapping clusters such that each data point belongs to exactly one cluster. These methods often use a central point (centroid) to represent each cluster. In this method, the number of clusters must be predefined. Examples are **K-Means** clustering, **K-Medoids** clustering.

Advantages of partitioning methods are
- Simple and easy to implement.
- Efficient for large datasets.

Limitations of portioning methods are
- Sensitive to the choice of initial centroids.
- Assumes spherical clusters and equal cluster sizes.

4.5.1 K-Means Clustering

K-Means clustering works by assigning data points to the nearest centroid and updating the centroids based on the mean of assigned points.

Algorithm 5 K-Means Clustering Algorithm

```
1: Initialize $K$ centroids randomly from the dataset
2: repeat
3:     For each data point $x_i$, calculate the distance between $x_i$ and each cluster's centroid (the
       mean of the points in the cluster) and assign $x_i$ to the cluster with the closest centroid.
4:     Update the cluster means for each cluster.
5: until no change
```

■ **Example 4.1** Form three clusters from the following data points: $P_1(2,10)$, $P_2(2,5)$, $P_3(8,4)$, $P_4(5,8)$, $P_5(7,5)$, $P_6(6,4)$, $P_7(1,2)$, $P_8(4,9)$.

- **Pass 1:**
  Initial Centroids (selected randomly):

$$C_1 = (2,10),\ C_2 = (5,8),\ C_3 = (1,2)$$

Data Points	**Distance to**			**Cluster**	**New Cluster**
	$(\mathbf{2,10})$	$(\mathbf{5,8})$	$(\mathbf{1,2})$		
$P_1(2,10)$	0.00	3.61	8.06	1	
$P_2(2,5)$	5.00	4.24	3.16	3	
$P_3(8,4)$	8.49	5.00	7.28	2	
$P_4(5,8)$	3.61	0.00	7.21	2	
$P_5(7,5)$	7.07	3.61	6.71	2	
$P_6(6,4)$	7.21	4.12	5.39	2	
$P_7(1,2)$	8.06	7.21	0.00	3	
$P_8(4,9)$	2.24	1.41	7.62	2	

Euclidean distance is used to measure the distance between data point and the centroid.

$$d = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

- **Pass 2:**
  Current Centroids:
  $C_1 = (2,10), C_2 = (\frac{8+5+7+6+4}{5}, \frac{4+8+5+4+9}{5}) = (6,6), C_3 = (\frac{2+1}{2}, \frac{5+2}{2}) = (1.5,3.5)$

Data Points	Distance to			Cluster	New Cluster
	$(\mathbf{2,10})$	$(\mathbf{6,6})$	$(\mathbf{1.5,3.5})$		
$P_1(2,10)$	0.00	5.66	6.52	1	1
$P_2(2,5)$	5.00	4.12	1.58	3	3
$P_3(8,4)$	8.49	2.83	6.52	2	2
$P_4(5,8)$	3.61	2.24	5.70	2	2
$P_5(7,5)$	7.07	1.41	5.70	2	2
$P_6(6,4)$	7.21	2.00	4.53	2	2
$P_7(1,2)$	8.06	6.40	1.58	3	3
$P_8(4,9)$	2.24	3.61	6.04	2	1

- **Pass 3:**
  Current Centroids:

$$C_1 = \left(\frac{2+4}{2}, \frac{10+9}{2}\right) = (3, 9.5)$$
$$C_2 = \left(\frac{8+5+7+6}{4}, \frac{4+8+5+4}{4}\right) = (6.5, 5.25)$$
$$C_3 = \left(\frac{2+1}{2}, \frac{5+2}{2}\right) = (1.5, 3.5)$$

Data Points	Distance to			Cluster	New Cluster
	$(\mathbf{3,9.5})$	$(\mathbf{6.5,5.25})$	$(\mathbf{1.5,3.5})$		
$P_1(2,10)$	1.12	6.54	6.52	1	1
$P_2(2,5)$	4.61	4.51	1.58	3	3
$P_3(8,4)$	7.43	1.95	6.52	2	2
$P_4(5,8)$	2.50	3.13	5.70	2	1
$P_5(7,5)$	6.02	0.56	5.70	2	2
$P_6(6,4)$	6.26	1.35	4.53	2	2
$P_7(1,2)$	7.76	6.39	1.58	3	3
$P_8(4,9)$	1.12	4.51	6.04	1	1

- **Pass 4:**
  Current Centroids:

$$C_1 = \left(\frac{2+5+4}{3}, \frac{10+8+9}{3}\right) = (3.67, 9)$$

$$C_2 = \left(\frac{8+7+6}{3}, \frac{4+5+4}{4}\right) = (7, 4.33)$$

$$C_3 = \left(\frac{2+1}{2}, \frac{5+2}{2}\right) = (1.5, 3.5)$$

Data Points	Distance to			Cluster	New Cluster
	$(\mathbf{3.67}, \mathbf{9})$	$(\mathbf{7}, \mathbf{4.33})$	$(\mathbf{1.5}, \mathbf{3.5})$		
$P_1(2,10)$	1.94	7.56	6.52	1	1
$P_2(2,5)$	4.33	5.04	1.58	3	3
$P_3(8,4)$	6.62	1.05	6.52	2	2
$P_4(5,8)$	1.67	4.18	5.70	1	1
$P_5(7,5)$	5.21	0.67	5.70	2	2
$P_6(6,4)$	5.52	1.05	4.53	2	2
$P_7(1,2)$	7.49	6.44	1.58	3	3
$P_8(4,9)$	0.33	5.55	6.04	1	1

The clusters are
Cluster 1 (2,10), (5,8), (4,9)
Cluster 2 (8,4), (7,5), (6,4)
Cluster 3 (2,5), (1,2)

■

4.5.2 K-Medoids Clustering

K-Means algorithm is sensitive to outliers since a large value may distort the distribution of data. K-Medoids algorithms take a representative object in a cluster called **medoid** which is the most centrally located point in a cluster. K-medoids algorithm is also known as **Partitioning Around Medoid (PAM)**. In K-Medoids, the most commonly used distance metric is Manhattan distance.

$$d = \sum_{i=1}^{n} |p_i - q_i|$$

Algorithm 6 K-Medoids Clustering Algorithm

```
1: Initialize $K$ medoids randomly from the dataset
2: repeat
3:     Assign each data point to the cluster corresponding to the nearest medoid.
4:     Calculate the objective function which is the sum of the dissimilarities of all the objects to
       their nearest medoid.
5:     Swap the medoid $x$ by an object $y$ if such a swap reduces the objective function.
6: until no change
```

■ **Example 4.2** Form two clusters from the following data points: $P_1(2,6)$, $P_2(3,4)$, $P_3(3,8)$, $P_4(4,7)$, $P_5(6,2)$, $P_6(6,4)$, $P_7(7,3)$, $P_8(7,4)$, $P_9(8,5)$, $P_{10}(7,6)$.

- **Pass 1:**
  Select two medoids randomly:

$$M_1 = (3,4),\ M_2 = (7,4)$$

Data point	**Distance to**		**Cluster**
	$\mathbf{M_1(3,4)}$	$\mathbf{M_2(7,4)}$	
$P_1(2,6)$	3	7	M_1
$P_2(3,4)$	0	4	M_1
$P_3(3,8)$	4	8	M_1
$P_4(4,7)$	4	6	M_1
$P_5(6,2)$	5	3	M_2
$P_6(6,4)$	3	1	M_2
$P_7(7,3)$	5	1	M_2
$P_8(7,4)$	4	0	M_2
$P_9(8,5)$	6	2	M_2
$P_{10}(7,6)$	6	2	M_2

$$\begin{aligned}\text{Objective function (Total dissimilarity)} &= 3+0+4+4+3+1+1+0+2+2\\ &= 20\end{aligned}$$

- **Pass 2**:
  Randomly select another medoid randomly ($M_0 = (7,3)$)from the data points and replace it with one of the previously selected medoids ($M_2 = (7,4)$). The current medoids are

$$M_1 = (3,4),\ M_0 = (7,3)$$

Data point	**Distance to**		**Cluster**
	$\mathbf{M_1(3,4)}$	$\mathbf{M_0(7,3)}$	
$P_1(2,6)$	3	8	M_1
$P_2(3,4)$	0	5	M_1
$P_3(3,8)$	4	9	M_1
$P_4(4,7)$	4	7	M_1
$P_5(6,2)$	5	2	M_0
$P_6(6,4)$	3	2	M_0
$P_7(7,3)$	5	0	M_0
$P_8(7,4)$	4	1	M_0
$P_9(8,5)$	6	3	M_0
$P_{10}(7,6)$	6	3	M_0

$$\begin{aligned}\text{Objective function (Total dissimilarity)} &= 3+0+4+4+2+2+0+1+3+3\\ &= 22\end{aligned}$$

Since the new cost is greater than the previous cost, the previous medoid is a better choice. Hence the clusters are
Cluster 1 $P_1(2,6), P_2(3,4), P_3(3,8), P_4(4,7)$
Cluster 2 $P_5(6,2), P_6(6,4), P_7(7,3), P_8(7,4), P_9(8,5), P_{10}(7,6)$

■

4.5.3 CLARA (Clustering LARge Applications) Algorithm

K-Medoids algorithm can be computationally expensive for large datasets due to the repeated calculations of pairwise distances. CLARA (Clustering LARge Applications) is an extension of the K-Medoids algorithm, designed specifically for clustering large datasets. CLARA uses a **sampling-based approach**.

The steps involved in CLARA

1. Take multiple samples from the dataset. Each sample is significantly smaller than the entire dataset but is representative of the overall structure.
2. For each sample, apply the K-Medoids algorithm to find the optimal medoids for that sample.
3. Test the medoids obtained from each sample against the entire dataset to compute the clustering quality. The quality is measured by calculating the total cost (sum of the distances between points and their nearest medoid) for the entire dataset.
4. The set of medoids that yields the lowest total cost on the full dataset is selected as the final solution.

Limitations:

- **Sampling Bias:** If the samples do not adequately represent the entire dataset, the resulting clusters may not be optimal.
- **Number of Samples:** The quality of clustering depends on the number of samples and the size of each sample. Too few samples or too small sample sizes may lead to poor performance.

4.5.4 CLARANS (Clustering Large Applications based on RANdomized Search)

CLARANS improves upon K-Medoids and CLARA by combining the advantages of both while introducing randomized search. It is designed for large datasets and aims to avoid the limitations of purely sampling-based algorithms like CLARA.

The steps involved in CLARANS

1. Randomly select an initial set of medoids.
2. For each medoid, **randomly** choose a non-medoid point and perform a swap.
   (a) Calculate the new clustering cost (sum of distances between points and their nearest medoid).
   (b) If the new configuration has a lower cost, accept the swap and update the medoid.
3. Do a **local search** by repeating the swapping process for a specified number of iterations or until no further improvements are found in the local search space.
4. If the local search does not result in an improvement, restart the process with a different random initial set of medoids (**global search**).
5. Terminates when the algorithm has explored a predefined number of local and global search iterations or when no further improvement is found in the overall clustering cost.

The parameters of CLARANS

- **Maxneighbor**: The number of neighbors (data points to swap) to explore for each medoid in the local search.
- **Numlocal:** The number of restarts or local minima to explore before the algorithm terminates.

Advantages

- **Scalability**: It can handle very large datasets more efficiently than K-Medoids and CLARA.
- **Flexibility**: It avoids the fixed-sample limitation of CLARA, allowing it to explore a larger portion of the dataset.
- Better Global Search: By performing both local and global searches, CLARANS is more likely to find an optimal or near-optimal solution.

Disadvantages

- **Randomization:** Since it is based on random sampling, different runs of CLARANS may result in different clusters.
- **Parameter Sensitivity:** The performance of CLARANS can be sensitive to the choice of parameters like the number of neighbors to explore or the number of restarts.

Example 4.3 Form 2 clusters using CLARANS for the data points (2,6), (3,4), (3,8), (4,7), (6,2), (7,3), (7,4),

- Select two medoids randomly:

$$M_1 = (2,6),\ M_2 = (7,3)$$

Data point	Distance to		Cluster
	$\mathbf{M_1(2,6)}$	$\mathbf{M_2(7,3)}$	
(2,6)	0	8	M_1
(3,4)	3	5	M_1
(3,8)	3	9	M_1
(4,7)	3	7	M_1
(6,2)	8	2	M_2
(7,3)	8	0	M_2
(7,4)	7	1	M_2

$$\begin{aligned}\text{Objective function (Total dissimilarity)} &= 0+3+3+3+2+0+1\\ &= 12\end{aligned}$$

- Perform random swap of one of the medoids and evaluate the clusters. For example, swap the medoid (2,6) with (3,8).

$$M_1 = (3,8),\ M_2 = (7,3)$$

Data point	Distance to		Cluster
	$\mathbf{M_1(3,8)}$	$\mathbf{M_2(7,3)}$	
(2,6)	3	8	M_1
(3,4)	4	5	M_1
(3,8)	0	9	M_1
(4,7)	2	7	M_1
(6,2)	9	2	M_2
(7,3)	9	0	M_2
(7,4)	8	1	M_2

$$\begin{aligned}\text{Objective function (Total dissimilarity)} &= 3+4+0+2+2+0+1\\ &= 12\end{aligned}$$

- CLARANS will continue performing random swaps of medoids and reassign points, calculating the cost after each swap. If the new medoid configuration reduces the total cost, it is accepted, and CLARANS proceeds with the new medoids. This process repeats for a specified number of iterations or until no further improvement is found.

- After several iterations, CLARANS selects the medoid configuration with the lowest total cost.

■

4.6 Hierarchical Methods

Hierarchical methods create a hierarchy of clusters, which can be represented as a tree (dendrogram). They can be either agglomerative (bottom-up) or divisive (top-down).

The characteristics of hierarchical clustering are

- No Predefined Number of Clusters: The number of clusters is not required beforehand; it can be determined by cutting the dendrogram.
- Nested Clustering: Provides a multi-level clustering structure.

Common Algorithms using hierarchical methods are

- **Agglomerative Clustering:**
  Starts with each data point as an individual cluster and iteratively merges the closest clusters based on a distance metric.
- **Divisive Clustering:**
  Begins with a single cluster containing all data points and recursively splits it into smaller clusters.

4.6.1 Agglomerative Clustering

Agglomerative clustering is a type of hierarchical clustering technique that works by recursively merging the closest pairs of clusters. It starts by treating each data point as its own individual cluster and successively merges clusters based on their similarity. This merging ends up with a single cluster that contains all data points.

Steps in Agglomerative Clustering

1. Start with each data point in its own cluster.
2. Compute the distance (or similarity) between all clusters (or individual points) using any of the distance metrics such as euclidean distance or manhattan distance.
3. Find the two clusters that are closest to each other and merge them to a single cluster.
4. Update the distance matrix to reflect the new cluster.
5. Continue merging the two closest clusters until there is only one cluster left (or until a stopping criterion is reached).

The hierarchical process forms a dendrogram, a tree-like structure where the root represents the single cluster that contains all the data points, and the leaves represent individual data points.

Linkage Criteria

The key to agglomerative clustering is the distance between clusters is measured. There are several types of linkage criteria that define how the distance between clusters is computed:

- **Single Linkage (Minimum Linkage):**
  The distance between two clusters is the minimum distance between any two points in the clusters. Disadvantage of this is that it is sensitive to noise and outliers, and can result in "chaining" where clusters form elongated shapes.
- **Complete Linkage (Maximum Linkage):**
  The distance between two clusters is the maximum distance between any two points in the clusters. Advantage of this is that it tends to produce more compact, spherical clusters.
- **Average Linkage:**
  The distance between two clusters is the average distance between all pairs of points in the two clusters.

Advantages of Agglomerative Clustering

- **No need to specify the number of clusters:** Unlike K-means, which requires the number of clusters to be predefined, agglomerative clustering creates a hierarchy that allows the user to choose the number of clusters by cutting the dendrogram.
- **Versatility with different distance metrics:** You can use different distance metrics and linkage criteria, providing flexibility for different types of data.
- **Hierarchical structure:** The dendrogram gives a clear visual representation of how clusters are formed.

Disadvantages of Agglomerative Clustering

- **Computationally expensive:** Calculating distances and merging clusters can be computationally expensive for large datasets, especially as it requires computing pairwise distances between clusters at each step.
- **Sensitive to noisy data and outliers:** Some linkage methods, such as single linkage, can be heavily influenced by outliers or noise in the data.
- **No clear "cut-off" point:** While the dendrogram provides a hierarchical structure, there may be no clear point where the clustering should stop, making it difficult to determine the optimal number of clusters.

■ **Example 4.4** Perform agglomerative clustering using single linkage on the data points $P_1(0.40, 0.53)$, $P_2(0.22, 0.38)$, $P_3(0.35, 0.32)$,$P_4(0.26, 0.19)$, $P_5(0.08, 0.41)$, $P_6(0.45, 0.30)$.

The distance matrix for these data points using euclidean distance is

	P_1	P_2	P_3	P_4	P_5	P_6
P_1	0					
P_2	0.23	0				
P_3	0.22	0.14	0			
P_4	0.37	0.19	0.13	0		
P_5	0.34	0.14	0.28	0.23	0	
P_6	0.24	0.24	0.10	0.22	0.39	0

Merging the two closest clusterrs (P_3 and P_6) will result in the following distance matrix. In single linkage, the distance between two clusters is the minimum distance between any two points in the clusters.

	P_1	P_2	P_3,P_6	P_4	P_5
P_1	0				
P_2	0.23	0			
P_3,P_6	0.22	0.14	0		
P_4	0.37	0.19	0.13	0	
P_5	0.34	0.14	0.28	0.23	0

Merging the two closest clusters (P_4 and P_3, P_6) will result in the following distance matrix.

	P_1	P_2	P_3,P_4,P_6	P_5
P_1	0			
P_2	0.23	0		
P_3,P_4,P_6	0.22	0.14	0	
P_5	0.34	0.14	0.28	0

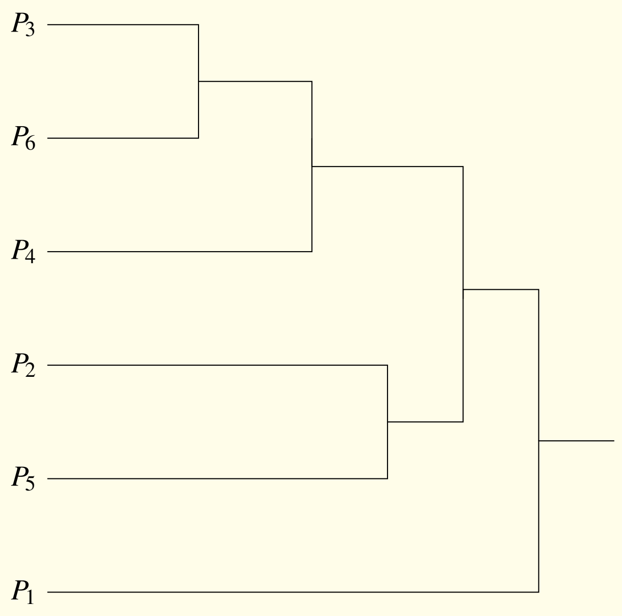
Merging the two closest clusters (P_2 and P_5) will result in the following distance matrix.

	P_1	P_2,P_5	P_3,P_4,P_6
P_1	0		
P_2,P_5	0.23	0	
P_3,P_4,P_6	0.22	0.14	0

Merging the two closest clusters (P_2,P_5 and P_3,P_4,P_6) will result in the following distance matrix.

	P_1	P_2,P_5,P_3,P_4,P_6
P_1	0	
P_2,P_5,P_3,P_4,P_6	0.23	0

The dendrogram is given below.



4.6.2 BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies)

BIRCH is an efficient clustering algorithm that is well-suited for large datasets. BIRCH incrementally clusters data points using a hierarchical structure known as a **CF (Clustering Feature) tree**. Each node in the CF tree is characterized by a **Clustering Feature (CF)**, which is triplet containing following information.

- **N**: The number of data points in the cluster.
- **LS (Linear Sum)**: The sum of all data points, i.e. $\sum_{i=1}^{N} X_i$.
- **SS (Square Sum)**: The sum of squares of all data points, i.e. $\sum_{i=1}^{N} X_i^2$.

These features help in calculating the centroid, radius, and other metrics of a cluster efficiently without needing to store all data points.

$$\text{Radius } R = \sqrt{\frac{SS}{N} - \left(\frac{LS}{N}\right)^2}$$

$$\text{Centroid } = \frac{LS}{N}$$

Parameters

- The **branching factor (B)** controls the maximum number of child nodes that a non-leaf node in the CF tree can have.
- The **threshold (T)** is a crucial parameter that controls the maximum radius (or diameter) of clusters within a leaf node.

Steps in BIRCH

1. Initialize the parameters threshold (T) and branching factor (B)
2. Start with an empty CF tree.
3. For each data point:
   (a) Traverse the tree to find the closest cluster (subcluster) based on the distance to the cluster's centroid.
   (b) If adding the point keeps the radius within the threshold (T), add it to the cluster. If adding the point exceeds the threshold, create a new cluster. If a node exceeds its branching factor (B) (too many clusters), split the node into two.
4. Condense the CF Tree (Optional, Phase 2). Optionally, prune or merge subclusters to reduce the number of clusters if needed.
5. Global Clustering (Optional, Phase 3). Use another clustering algorithm (like K-Means) on the centroids of the subclusters to refine the final clusters.
6. The final clusters are either from the CF tree itself or from a global clustering algorithm applied to the subclusters.

Advantages

- **Scalability:** BIRCH is designed to handle large datasets efficiently. It processes data incrementally, which means that it can handle datasets that do not fit into memory.
- **Speed:** Because it summarizes clusters using CF vectors and constructs the CF tree in a single scan of the data, it is faster than many other clustering algorithms, especially on large datasets.
- **Memory Efficiency:** It compresses data into a tree structure, reducing memory usage compared to methods that store all data points explicitly.
- **Dynamic:** BIRCH builds clusters dynamically as it scans through the data, which makes it well-suited for streaming data.

Limitations

- **Sensitivity to Input Order:** The order in which data points are processed can affect the clustering results.
- **Effectiveness with Spherical Clusters:** BIRCH performs well when clusters are roughly spherical in shape. It may struggle with arbitrarily shaped clusters.
- **Requires Predefined Threshold:** The choice of the threshold parameter (T) is crucial. If it's too large, the algorithm may merge too many points into a single cluster. If it is too small, the tree may become too large, leading to inefficient memory use.

■ **Example 4.5** Perform BIRCH clustering on the data points $x_1 = (3,4)$, $x_2 = (2,6)$, $x_3 = (4,5)$, $x_4 = (4,7)$, $x_5 = (3,8)$, $x_6 = (6,2)$, $x_7 = (7,2)$, $x_8 = (7,4)$, $x_9 = (8,4)$, $x_{10} = (7,9)$ with Threshold $T = 1.5$ and Branch Factor $B = 2$.

- Consider the data point $(3,4)$
  - Since this is the only one point in the cluster now, radius $R = 0$.

- Cluster features are calculated as

$$\begin{aligned} N &= 1 \\ LS &= (3,4) \\ SS &= (3^2,4^2) = (9,16) \end{aligned}$$

  Hence **CF1 = <1,(3,4),(9,16)>**
- Construct a leaf with data point x_1 and as a branch of **CF1**.

- Consider the data point $(2,6)$
  - Cluster features are calculated as

$$\begin{aligned} N &= 2 \\ LS &= (3,4)+(2,6) = (5,10) \\ SS &= (9+2^2,16+6^2) = (13,52) \end{aligned}$$

  - Calculate the radius as follows

$$\begin{aligned} \text{Radius } R &= \sqrt{\frac{SS}{N} - \left(\frac{LS}{N}\right)^2} = \sqrt{\frac{(13,52)}{2} - \left(\frac{(5,10)}{2}\right)^2} \\ &= (0.5,1) < T \end{aligned}$$

  - Since the radius is less than the threshold, add it to the current leaf node and the CF1 gets updated as follows.

$$CF1 =< 2,(5,10),(13,52) >$$

- Consider the data point $(4,5)$
  - Cluster features are calculated as

$$\begin{aligned} N &= 3 \\ LS &= (5,10)+(4,5) = (9,15) \\ SS &= (13+4^2,52+5^2) = (29,77) \end{aligned}$$

  - Calculate the radius as follows

$$\begin{aligned} \text{Radius } R &= \sqrt{\frac{SS}{N} - \left(\frac{LS}{N}\right)^2} = \sqrt{\frac{(29,77)}{3} - \left(\frac{(9,15)}{3}\right)^2} \\ &= (0.47,0.4714) < T \end{aligned}$$

  - Since the radius is less than the threshold, add it to the current leaf node and the CF1 gets updated as follows.

$$CF1 =< 3,(9,15),(29,77) >$$

- Consider the data point $(4,7)$

- Cluster features are calculated as

$$\begin{aligned}N &= 4\\ LS &= (9,15)+(4,7) = (13,22)\\ SS &= (29+4^2, 77+7^2) = (45,126)\end{aligned}$$

- Calculate the radius as follows

$$\text{Radius } R = \sqrt{\frac{SS}{N} - \left(\frac{LS}{N}\right)^2} = \sqrt{\frac{(45,126)}{4} - \left(\frac{(13,22)}{4}\right)^2}$$
$$= (0.41, 0.55) < T$$

- Since the radius is less than the threshold, add it to the current leaf node and the CF1 gets updated as follows.

$$CF1 =< 4,(13,22),(45,126) >$$

- Consider the data point $(3,8)$
  - Cluster features are calculated as

$$\begin{aligned}N &= 5\\ LS &= (13,22)+(3,8) = (16,30)\\ SS &= (45+3^2, 126+8^2) = (54,190)\end{aligned}$$

  - Calculate the radius as follows

$$\text{Radius } R = \sqrt{\frac{SS}{N} - \left(\frac{LS}{N}\right)^2} = \sqrt{\frac{(54,190)}{5} - \left(\frac{(16,30)}{5}\right)^2}$$
$$= (0.33, 0.63) < T$$

  - Since the radius is less than the threshold, add it to the current leaf node and the CF1 gets updated as follows.

$$CF1 =< 5,(16,30),(54,190) >$$

- Consider the data point $(6,2)$
  - Cluster features are calculated as

$$\begin{aligned}N &= 6\\ LS &= (16,30)+(6,2) = (22,32)\\ SS &= (54+6^2, 190+2^2) = (90,194)\end{aligned}$$

  - Calculate the radius as follows

$$\text{Radius } R = \sqrt{\frac{SS}{N} - \left(\frac{LS}{N}\right)^2} = \sqrt{\frac{(90,194)}{6} - \left(\frac{(22,32)}{6}\right)^2}$$
$$= (1.24, 1.97) > T$$

- Since the radius is NOT less than the threshold, this data point cannot be added to the existing cluster with cluster feature CF1. Hence the data point as added to new leaf node and the cluster feature for this cluster is

$$CF2 = < 1, (6,2), (36,4) >$$

- Consider the data point $(7,2)$
  - Since there are two branches, the distance between (7,2) and the centroids of the two leaf nodes have to be calculated.

$$\begin{aligned} \text{Centroid } &= \frac{LS}{N} \\ \text{Centroid of CF1} &= \frac{(16,30)}{5} = (3.2,6) \\ \text{Centroid of CF2} &= \frac{(6,2)}{1} = (6,2) \end{aligned}$$

  Therefore (7,2) is closer to CF2.
  - Cluster features are calculated as

$$\begin{aligned} N &= 2 \\ LS &= (6,2) + (7,2) = (13,4) \\ SS &= (36+7^2, 4+2^2) = (85,8) \end{aligned}$$

  - Calculate the radius as follows

$$\begin{aligned} \text{Radius } R &= \sqrt{\frac{SS}{N} - \left(\frac{LS}{N}\right)^2} = \sqrt{\frac{(85,8)}{2} - \left(\frac{(13,4)}{2}\right)^2} \\ &= (0.5,0) < T \end{aligned}$$

  - Since the radius is less than the threshold, this data point is added to the existing cluster with cluster feature CF2 and it gets updated as

$$CF2 = < 2, (13,4), (85,8) >$$

- Consider the data point $(7,4)$
  - Since there are two branches, the distance between (7,4) and the centroids of the two leaf nodes have to be calculated.

$$\begin{aligned} \text{Centroid } &= \frac{LS}{N} \\ \text{Centroid of CF1} &= \frac{(16,30)}{5} = (3.2,6) \\ \text{Centroid of CF2} &= \frac{(13,4)}{2} = (6.5,2) \end{aligned}$$

  Therefore (7,4) is closer to CF2.

- Cluster features are calculated as

$$N = 3$$
$$LS = (13,4) + (7,4) = (20,8)$$
$$SS = (85+7^2, 8+4^2) = (134,24)$$

- Calculate the radius as follows

$$\text{Radius } R = \sqrt{\frac{SS}{N} - \left(\frac{LS}{N}\right)^2} = \sqrt{\frac{(134,24)}{3} - \left(\frac{(20,8)}{3}\right)^2}$$
$$= (0.47, 0.94) < T$$

- Since the radius is less than the threshold, this data point is added to the existing cluster with cluster feature CF2 and it gets updated as

$$CF2 = < 3, (20,8), (134,24) >$$

• Consider the data point $(8,4)$
  - Since there are two branches, the distance between (8,4) and the centroids of the two leaf nodes have to be calculated.

$$\text{Centroid } = \frac{LS}{N}$$
$$\text{Centroid of CF1} = \frac{(16,30)}{5} = (3.2,6)$$
$$\text{Centroid of CF2} = \frac{(20,8)}{3} = (6.6,2.6)$$

Therefore (8,4) is closer to CF2.

- Cluster features are calculated as

$$N = 4$$
$$LS = (20,8) + (8,4) = (28,12)$$
$$SS = (134+8^2, 24+4^2) = (198,40)$$

- Calculate the radius as follows

$$\text{Radius } R = \sqrt{\frac{SS}{N} - \left(\frac{LS}{N}\right)^2} = \sqrt{\frac{(198,40)}{4} - \left(\frac{(28,12)}{4}\right)^2}$$
$$= (0.7, 1) < T$$

- Since the radius is less than the threshold, this data point is added to the existing cluster with cluster feature CF2 and it gets updated as

$$CF2 = < 4, (28,12), (198,40) >$$

• Consider the data point $(7,9)$

– Since there are two branches, the distance between (7,9) and the centroids of the two leaf nodes have to be calculated.

$$\text{Centroid of CF1} = \frac{(16,30)}{5} = (3.2,6)$$
$$\text{Centroid of CF2} = \frac{(28,12)}{4} = (7,3)$$

Therefore (7,9) is closer to CF2.

– Cluster features are calculated as

$$N = 5$$
$$LS = (28,12) + (7,9) = (35,21)$$
$$SS = (198+7^2, 40+9^2) = (247,121)$$

– Calculate the radius as follows

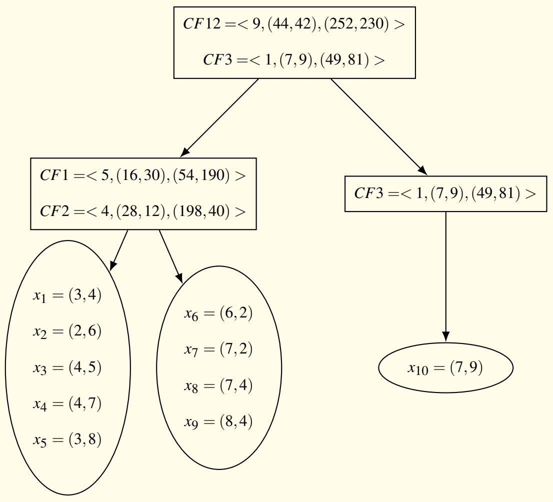
$$\text{Radius } R = \sqrt{\frac{SS}{N} - \left(\frac{LS}{N}\right)^2} = \sqrt{\frac{(247,121)}{5} - \left(\frac{(35,21)}{5}\right)^2}$$
$$= (0.63, 2.56) > T$$

– Since the radius is NOT less than the threshold, this data point cannot be added to the existing cluster with cluster feature CF2.
– Since the branch factor is 2, a new leaf not cannot be attached. Hence, the two cluster features CF1 and CF2 can be merged to form a new cluster feature CF12

$$CF12 =< 9, (44,42), (252,230) >$$

– The new cluster feature CF3 is

$$CF3 =< 1, (7,9), (49,81) >$$


$CF12 =< 9,(44,42),(252,230) >$
$CF3 =< 1,(7,9),(49,81) >$
$CF1 =< 5,(16,30),(54,190) >$
$CF2 =< 4,(28,12),(198,40) >$
$CF3 =< 1,(7,9),(49,81) >$
$x_1 = (3,4)$
$x_2 = (2,6)$
$x_3 = (4,5)$
$x_4 = (4,7)$
$x_5 = (3,8)$
$x_6 = (6,2)$
$x_7 = (7,2)$
$x_8 = (7,4)$
$x_9 = (8,4)$
$x_{10} = (7,9)$

4.7 Density Based Clustering

Density-based methods identify clusters based on the density of data points in a given region. They can discover clusters of arbitrary shapes and sizes. In density based clustering, clusters are formed by connecting regions of high density separated by regions of low density. These methods can identify noise points that do not belong to any cluster. Common algorithms are DBSCAN (Density-Based Spatial Clustering of Applications with Noise) and OPTICS (Ordering Points To Identify the Clustering Structure).

Limitations of density based clustering are

- Requires careful selection of parameters (**epsilon** and **minPts**).
- Performance may degrade with high-dimensional data.

4.7.1 DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a powerful clustering algorithm that identifies clusters in a dataset based on the density of points. DBSCAN does not require the number of clusters to be specified beforehand and can find clusters of arbitrary shape. Key characteristics of DBSCAN are

- **Epsilon** (ε): This is the maximum distance between two points for them to be considered as part of the same cluster.
- **MinPts**: This is the minimum number of points required to form a cluster.
- **Core Point**: A point is called a core point if it has enough neighbors close to it. Specifically, if there are at least a certain number of points (**MinPts**) within a certain distance (ε), then that point is a core point.
- **Border Point**: A point that has fewer than **MinPts** neighbors within ε but lies within the ε-radius of a core point.
- **Noise Point**: A point that does not belong to any cluster.

Steps for DBSCAN

1. Decide on two parameters
   (a) ε: The maximum distance to consider points as neighbors.
   (b) $MinPts$: The minimum number of points needed to form a dense region (a cluster).
2. Go through each point in the dataset. If the point has not been visited yet, check how many points are within its ε distance (its neighbors).
3. If the number of neighbors is less than $MinPts$, label this point as **noise**. If the number of neighbors is at least $MinPts$, label this point as a **core** point and create a new cluster that includes this point.
4. Expand the Cluster:
   (a) For each neighbor of the core point, check
      - If the neighbor has not been visited, mark it as visited and find its neighbors.
      - If this neighbor is also a core point (has enough neighbors), add its neighbors to the list of neighbors for the current cluster.
      - Add the neighbor to the cluster if it is not already included.

   (b) Continue expanding the cluster until no new points can be added. Go back to the list of points and repeat the process for any unvisited points.

■ **Example 4.6** Apply DBSCAN algorithm for the data points $P_1(3,7)$, $P_2(4,6)$, $P_3(5,5)$, $P_4(6,4)$, $P_5(7,3)$, $P_6(6,2)$, $P_7(7,2)$, $P_8(8,4)$, $P_9(3,3)$, $P_{10}(2,6)$, $P_{11}(3,5)$, $P_{12}(2,4)$.

Let $minPts = 4$ and $\varepsilon = 1.9$. Euclidean distance is used to calculate the distance between data points. The distance matrix is given below.

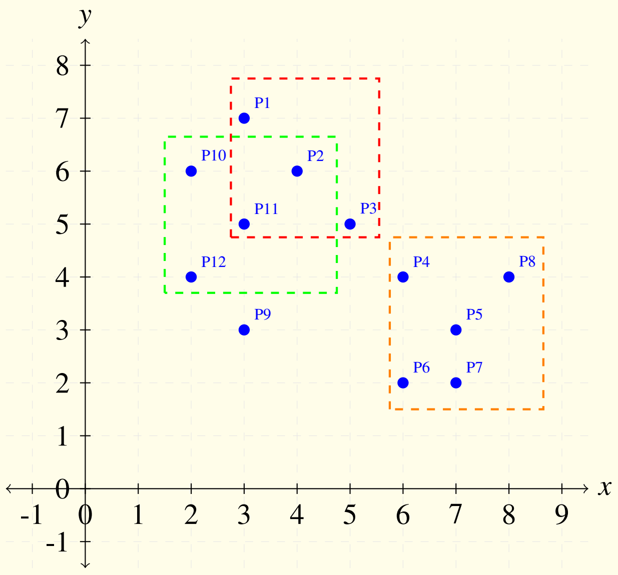
	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}
P_1	0											
P_2	1.41	0										
P_3	2.83	1.41	0									
P_4	4.24	2.83	1.41	0								
P_5	5.66	4.24	2.83	1.41	0							
P_6	5.83	4.47	3.16	2.00	1.41	0						
P_7	6.40	5.00	3.61	2.24	1.00	1.00	0					
P_8	5.83	4.47	3.16	2.00	1.41	2.83	2.24	0				
P_9	4.00	3.16	2.83	3.16	4.00	3.16	4.12	5.10	0			
P_{10}	1.41	2.00	3.16	4.47	5.83	5.66	6.40	6.32	3.16	0		
P_{11}	2.00	1.41	2.00	3.16	4.47	4.24	5.00	5.10	2.00	1.41	0	
P_{12}	3.16	2.83	3.16	4.00	5.10	4.47	5.39	6.00	1.41	2.00	1.41	0

The neighbors of each data points which are at a distance less than ε are given below

Data point	**Neighbors**
P_1	P_2, P_{10}
P_2	P_1, P_3, P_{11}
P_3	P_2, P_4
P_4	P_3, P_5
P_5	P_4, P_6, P_7, P_8
P_6	P_5, P_7
P_7	P_5, P_6
P_8	P_5
P_9	P_{12}
P_{10}	P_1, P_{11}
P_{11}	P_2, P_{10}, P_{12}
P_{12}	P_9, P_{11}

Based on the number of neighbors, the status of each node is given below.

Data point	**Neighbors**	
P_1	Noise	Border
P_2	Core	
P_3	Noise	Border
P_4	Noise	Border
P_5	Core	
P_6	Noise	Border
P_7	Noise	Border
P_8	Noise	Border
P_9	Noise	
P_{10}	Noise	Border
P_{11}	Core	
P_{12}	Noise	Border


y
x
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
P11
P12
-1
0
1
2
3
4
5
6
7
8
9

References