Department of Artificial Intelligence and Machine Learning

II B.Tech II Sem

Machine Learning

UNIT-1

Introduction to Machine Learning

By

Mrs.K.Reddemma,

Assistant Professor,

Department of AI&ML.

Machine Learning

UNIT - I

Introduction to Machine learning

① Evolution of ML

② paradigms : learning by Rote, Introduction, Reinforcement, etc.

③ Types of Data, Stages in ML

④ Data Acquisition, Feature Engineering, Representation

⑤ Data Acquisition, Model selection, learning, Evaluation, prediction

⑥ Search & learning, Data sets

UNIT - II

Nearest Neighbor - Based Models

① proximity & Distance Measures

② Non-Metric Similarity Functions

③ Binary pattern proximity

④ classification Algorithms (Distance-Based)

⑤ K-NN classifier, Radius Distance NN Algorithm

⑥ K-NN Regression, Performance of classifiers & Regression Algorithms.

Machine Learning

Unit - 1

Introduction to Machine Learning

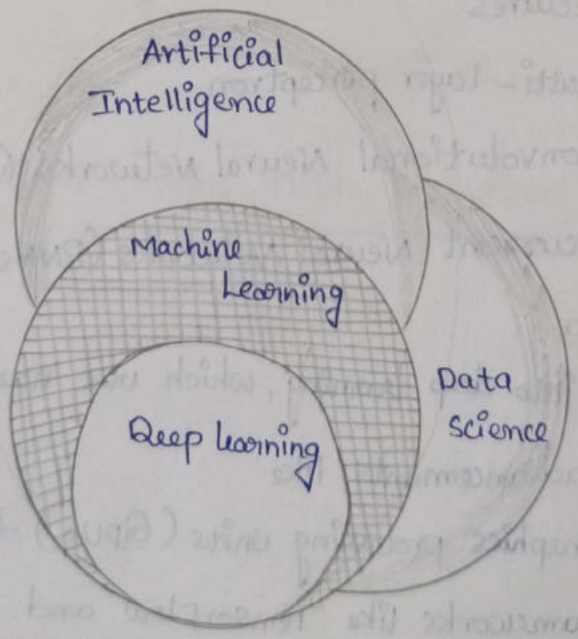
What is Machine Learning :

Machine Learning is a branch of Artificial Intelligence that develops algorithms by learning the hidden patterns of the datasets used it to make predictions on new similar type data, without being explicitly programmed for each task.

Traditional machine learning combines data with statistical tools to predict an output that can be used to make actionable insights.

Machine learning is used in many different applications, From image and speech recognition to natural language processing, recommendation systems, fraud detection, portfolio optimization, automated risk and so on.

Machine learning models are also used to power autonomous vehicles, drones and robots, making them more intelligent and adaptable to changing environments.



Venn diagram for AI, ML, Deep learning and Data Science

Evolution of Machine Learning

Machine learning :

Machine learning is the process of learning models that can predict outcomes based on data.

It involves predicting classes (or) regression values for given input data.

Early AI and symbolic systems :

AI's beginnings focused on symbolic logic

* General problem solver (Gps) and programs using prolog were common.
* Relied on rule-based systems and automated reasoning.
* AI was often seen as "Logic-based"

Shift to Data-Driven learning :

the 1980's saw challenges in symbolic AI due to complex real-world problems.

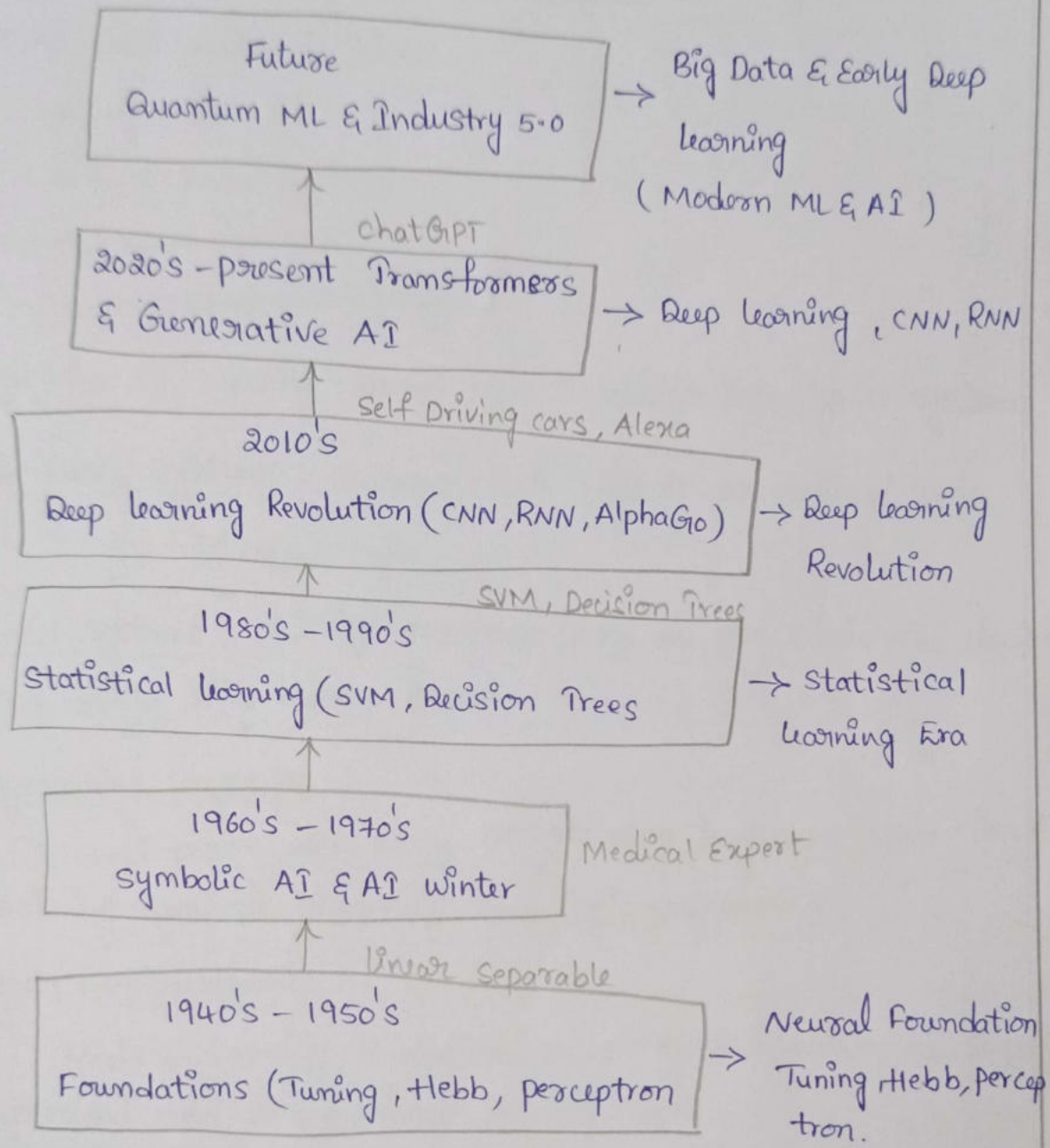
* Focus shifted to data-driven approaches with the rise of neural network architectures
    * Multi-layer perceptron
    * convolutional Neural Networks (CNNs)
    * Recurrent Neural Networks (RNNs)

Deep Learning Era :

* AI Evolved into deep learning, which uses neural network architectures.
* Enabled by advancements like
    1. Graphics processing units (GPUs) for computation
    2. Frameworks like TensorFlow and pytorch

Impact of AI Today : Deep learning, especially convolutional neural networks and generative models, influences various scientific and engineering fields globally.

Evolution of Machine learning



Conventional AI :

Conventional AI is a type of AI that focuses on narrow, task-specific intelligence.

Conventional AI systems are trained on large datasets to learn patterns and make decisions.

* Logic for AI
* Rule-Based systems
* Expert Systems
* programming languages.

* Logic for AI : using formal logic to design intelligent systems.

* Rule-Based systems : systems that rely on predefined rules.

Ex : if-then logic.

* Expert systems : AI systems that ~~rely on~~ mimic the decision-making ability of a human expert.

* programming Languages :

Lisp and prolog were early programming languages used for AI, emphasizing symbolic reasoning and logic programming.

Diagram Explanation :

Machine Learning is decicted as a key area within AI, Supported by Supervised and unsupervised learning tasks.

pattern recognition and data mining are shown as critical applications that rely on machine learning and structured data from databases.

Conventional AI components like rule-based systems are presented as foundational concepts that have evolved into modern AI approaches.

Main components :

1. Machine Learning (ML) :

A subfield of AI focused on enabling systems to learn from data. It is further divided into

* learning from examples - (Supervised learning)

involves tasks like classification and Regression

classification : predicting categorical outcomes

Ex : Mails spam (or) Not spam

Regression : predicting continuous values

Ex : House prices

* learning from observations - (unsupervised Learning)

Learning from observations involves clustering. clustering refers to the Grouping similar data points.

Ex : Customer segmentation

2. pattern Recognition :

Focuses on identifying patterns in data, often linked to tasks in machine learning such as image (or) speech recognition.

3. Data Mining :

The process of efficiently organizing and analyzing large datasets (databases) to uncover useful patterns and insights.

This forms the backbone of AI by providing structured data for analysis.

Databases and Human-Machine Interaction :

Databases are critical for storing and managing data, while human-Machine interaction refers to interfaces (or) systems that allow humans to interact with AI effectively.

The typical background topics of AI are shown Figure 1-2

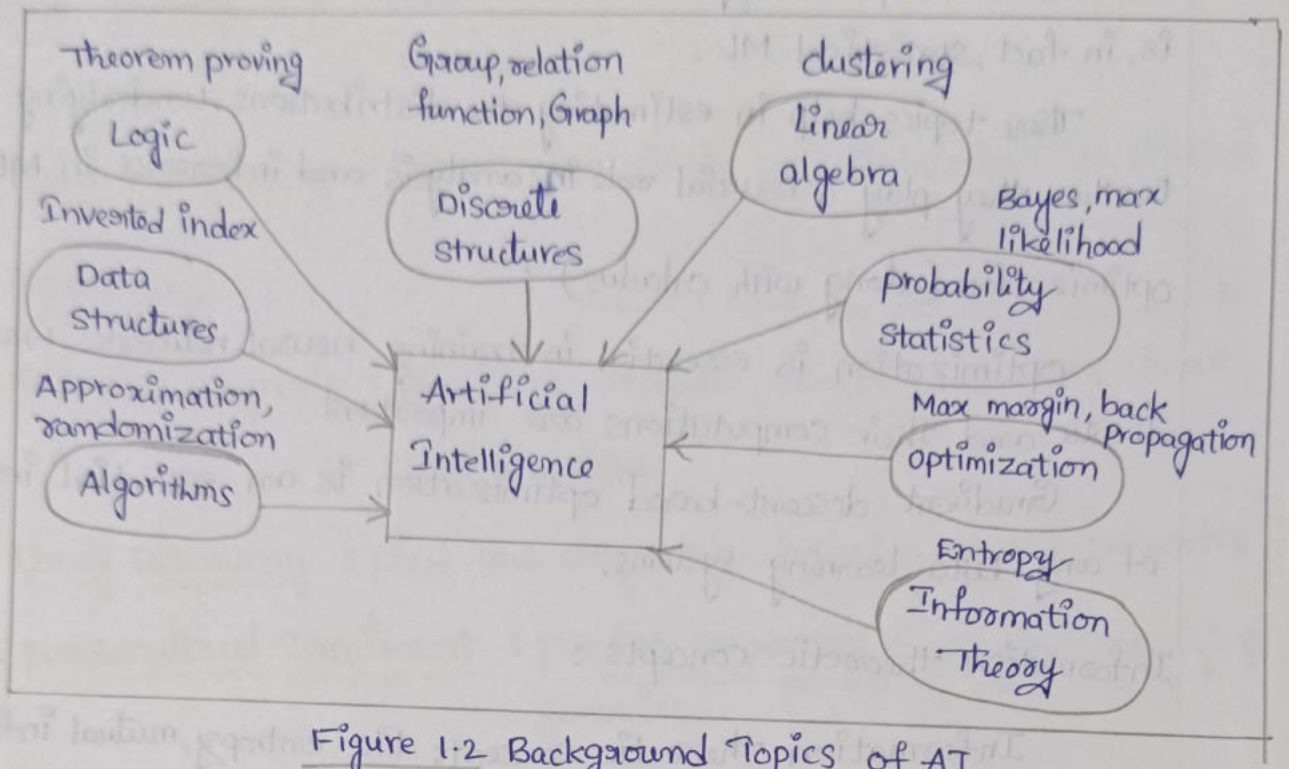


Figure 1.2 Background Topics of AI

Data Structures and algorithms are basic to both conventional and current AI systems. Logic and Discrete structures played an important role in the analysis and synthesis of conventional AI system.

In ML, we deal with vectors and vector spaces and these topics are better appreciated through linear algebra. The data input to an ML system may be viewed as a matrix, popularly called the data matrix.

If there are n data items, each represented as an L-dimensional vector, then the corresponding data matrix A is of size n x L. Linear algebra is useful in analysing the weights associated with the edges in a neural network. Matrix multiplication and eigen analysis are important in initializing the weights of the neural network and in weight updates. It can also help in weights normalization.

The whole activity of clustering may be viewed as data matrix factorization.

* probability and statistics :

The role of probability and statistics need not be explained as ML is, in fact, statistical ML.

These topics help in estimating the distributions underlying the data. Further, they play a crucial role in analysis and inference in ML.

* optimization (along with calculus) :

Optimization is essential in training neural networks where gradients and their computations are important.

Gradient descent-based optimization is an essential ingredient of any Data learning systems.

* Information theoretic concepts :

Information theoretic concepts like entropy, mutual information and Kullback-Leibler divergence are essential to understand topics such as decision tree classifiers, feature selection and deep neural networks.

4. Autonomous vehicles :

* Self-driving Cars : Detect objects, follow lanes, predicts trafic movements.

* Drive Monitoring : Detect drowsiness (or) distraction in drivers

5. Healthcare :

* Diseases Diagnosis : predicts diabetes, Cancer, and more from patient data.

* Drug Discovery : Find new drug molecules using neural networks.

* personalized Treatment : predict responses to medications using genomics.

6. Finance :

* Fraud Detection : Detect unusual transactions (or) Hacking

* Algorithmic Trading : Make Trading decisions using historical data.

* Credit scoring : Assess risk levels for loan applications.

7. E-Commerce & Recommendation systems :

* products Recommendations : Suggest items based on user behavior. Eg : Amazon, Netflix

* Search Ranking : Improve relevance in product search.

8. Gaming & Simulation :

* Game AI : Bots that learn to play games better than humans
Eg : AlphaGo

* Simulation of Real-World Scenarios :
Weather, traffic, robotics

9. Robotics :

* perception : Robots understand their environment using Sensors and vision.

* Motion planning : Navigate around obstacles.

* Human-Robot Interaction :

Use voice, vision, and touch to interact with people.

10. Big Data and Anomaly Detection :

Detect fraud, network attacks, (or) equipment failures using large-scale pattern analysis.

Machine Learning Basics :

① Supervised And Unsupervised learning

① Supervised Learning :

Supervised learning Algorithms are those that learn a mapping from inputs x to outputs y using a labeled dataset. The dataset contains examples of inputs along with the correct outputs (labels), often provided by a human supervisor.

Even when outputs y are generated automatically, the term supervised learning still applies.

(or)

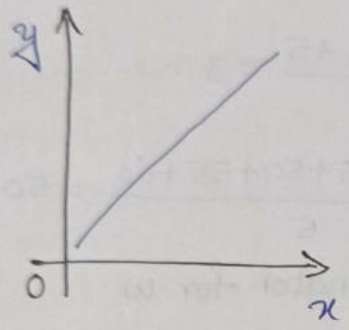
Supervised learning is a type of machine learning where a model learns to map inputs x to outputs y using a labelled dataset. In Deep learning this mapping is learned using deep neural networks that automatically discover useful features from data.

Binary Cross-Entropy (log loss) $= -[y \log(\hat{y}) + (1-y) \log(1-\hat{y})]$

Ex: Spam detection, disease prediction (Yes/No)
Sentiment classification (positive / Negative)

* Linear Regression

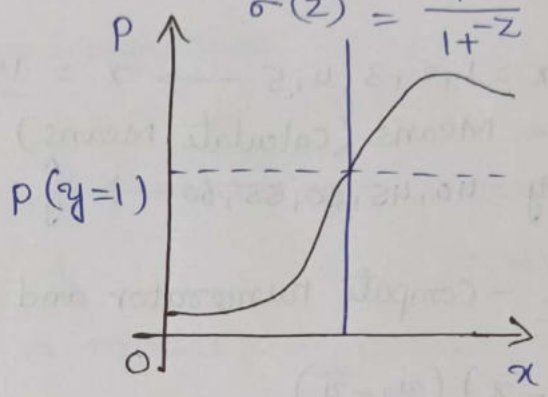
$y = wTx + b$



* Logistic Regression

$P(y=1/x) = \sigma(wT+b)$

$\sigma(z) = \frac{1}{1+e^{-z}}$



Example problem for linear Regression :

① To predict the salary of an employee based on their years of experience. Find a line $y = wx + b$.

Data :

Years of Experience (x)	Salary (y in $1000)
1	40
2	45
3	50
4	55
5	60

Answer :

1. Use least squares (or) gradient descent to fit the best line.
2. After Training data

from input data table $w = 5$,

w = slope of the line

b = y-intercept (values of y when $x = 0$)

$x = 1, 2, 3, 4, 5 \longrightarrow \bar{x} = \frac{1+2+3+4+5}{5} = 3$

step 1 – Means (calculate Means)

$y = 40, 45, 50, 55, 60 \longrightarrow \hat{y} = \frac{40+45+50+55+60}{5} = 50$

step 2 – Compute numerator and denominator for w

$$\Sigma (x_i - \bar{x})(y_i - \bar{y})$$

$$= (1-3)(40-50) + (2-3)(45-50) + (3-3)(50-50) + (4-3)(55-50) + (5-3)(60-50)$$

$$= (-2)(-10) + (-1)(-5) + 0 + (1)(5) + (2)(10)$$

$$= 20 + 5 + 0 + 5 + 20$$

$$= 50$$

$$\Sigma (x_i - \bar{x})^2 = (-2)^2 + (-1)^2 + 0^2 + 1^2 + 2^2$$

$$= 4 + 1 + 0 + 1 + 4$$

$$= 10$$

$$\boxed{\Sigma (x_i - \bar{x})^2 = 10}$$

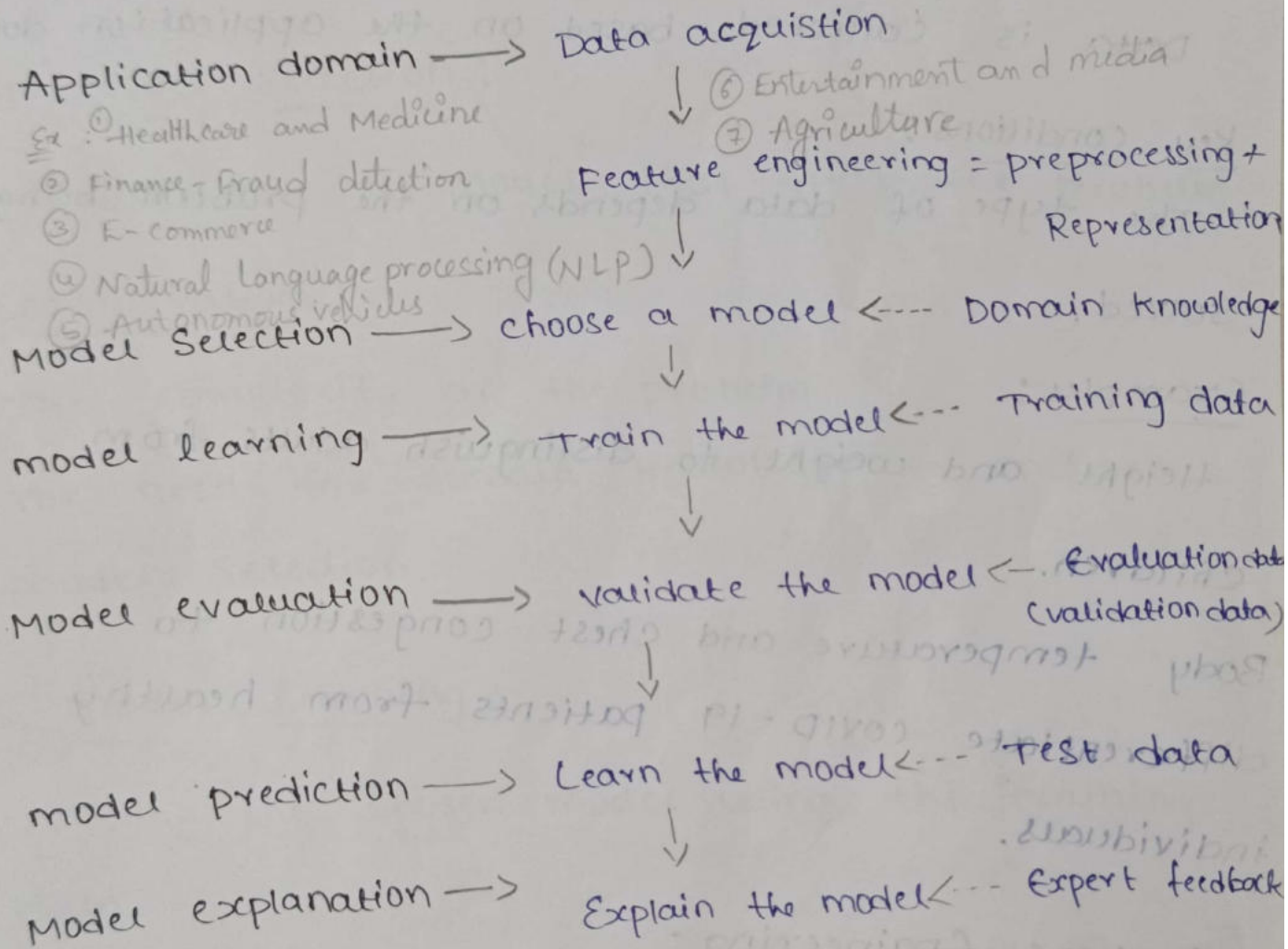
so $y = wx + b$ $\boxed{\because (x = 6 \text{ It is a new input to predict salary})}$

$y = 5x + 35$

$y = 5 \times 6 + 35 \Rightarrow 30 + 35 \rightarrow 65$

Stages in machine learning:

The process of building a machine learning (ML) System involves several critical steps, as highlighted in figure 1.6. The steps emphasize data in the form of training, validation, and test datasets. Below is an explanation of each step.



Stages of a machine learning System:

1. Application :-

understand the probelm or context where the ML System will be applied.

Examples:

Identifying adults vs. children

Diagnosing COVID-19 through specific metrics (eg. temperature, chest congestion)

2. Data Acquistion:

Data is collected based on the application domain

Key conditions:

The type of data depends on the problem being solved.

Examples:

Height and weight to distinguish adults from children.

Body temperature and chest congestion to differentiate COVID-19 patients from healthy individuals.

3. Feature Engineering:

This combines data preprocessing and data representation to ensure the ML Model can use the data effectively.

Steps in Feature Engineering:

Data preprocessing: prepare raw data by handing issues like:

Missing values

Noise or irrelevant information.

4. Model Selection:

Choose an ML model that first the probelm considerations:

the complexity of the probelm.

the need for domain knowledge to enchare model selection.

5. model learning:

Train the chosen model using the training data.

the model learns patterns and relationship in the data.

6. Model Evaluation:

Validate the models performance using validation data.

This step ensures the model generalizes well and avaids overfiting.

7. Model prediction:

The trained model is tested on unseen test data to predict outcomes and evaluate its real-world performance.

8. Model Explanation:

Interpret and explain the model's predictions. Expert feedback can help refines the model.

Data acquistion:

Importance: Data collection is the first step and heavily depends on the probelm domain.

Examples:

For distinguish adults from children.

metrics like height or weight are sufficient.

For diagnosing COVID-19:

Measurements like body temperature and chest congestion are more relevant than height or weight.

Feature Engineering:

Feature Engineering is a two-part process.

1. Data preprocessing:

Handless issues in raw data such as:

Missing values: Gaps in the data that must be filled or imputed.

Outliers: Extreme values that can distort the model.

Noise: unwanted variations or errors in the data.

2. Data Representation:

Transform raw data into a format Suitable for the ML model.

Examples:

Encoding categorical variables (e.g., converting "male / Female" into numerical values)

Normalizing numerical data to standardize scales.

Datasets in ML:

1. Training data:

Used for training the model to learn patterns.

2. Validation Data:

Helps fine-tune the model and validate its performance during training.

3. Test Data:

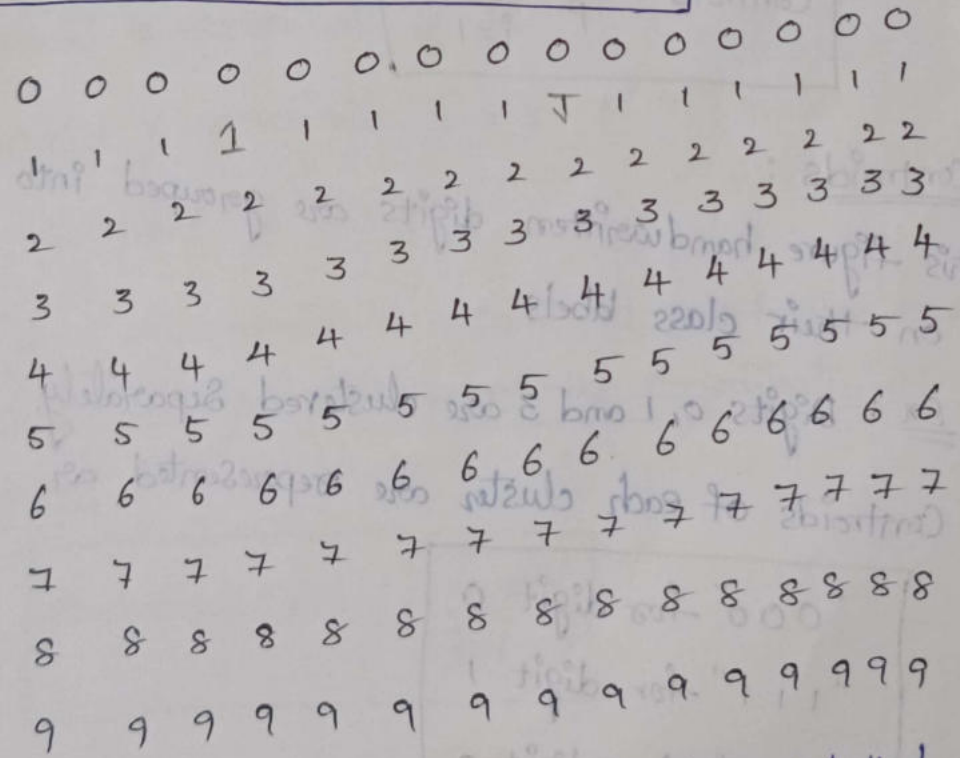
Evaluates the model's final performance on unseen data.

22/1/25 **Learning from observations :**

* observations are also instances like examples but they are different because observations need not be labelled.
* In this case, we cluster (or) group the observations into a smaller number of groups.
* Such grouping is performed with help of a clustering algorithm that assigns similar patterns to the same group / cluster.
* Each cluster could be represented by it's centroid (or) mean
* let $x_1, x_2 \dots x_p$ be p elements of a cluster
* Centroid of the cluster is defined by

$$\frac{1}{p} \sum_{i=1}^{p} x_i$$

> Formula for centroid of a cluster is
>
> $$\text{Centroid} = \frac{1}{p} \sum_{i=1}^{p} x_i$$



Examples of Handwritten digits labelled 0 TO 9

Handwritten Digits :

* This figure shows examples of handwritten digits labeled from 0 To 9.
* Each row contains multiple examples of a single digit.
  For instance, the first row contains several samples of "0"
  The Second row contains "1" and so on "9"
* The labels (0–9) are used as ground truth for supervised learning tasks.

Centroids in clustering :

The text introduces the concept of a centroid is the average (or mean) representation of a cluster of data points.

For given cluster, if there are P elements $x_1, x_2 - - x_p$ the centroid is calculated as.

$$\text{Centroid} = \frac{1}{P} \sum_{i=1}^{P} x_i$$

cluster Centroids :

* In this figure, handwritten digits are grouped into clusters based on their class labels

  Ex : Digits 0, 1 and 3 are clustered Separately.

* The Centroids of each cluster are represented as

"000" for digit 0
"111" for digit 1
"333" for digit 3

These centroids are averages computed from all the Samples within each class.

Sum of coordinates :

Sum of all the x-coordinates of the data points.

Sum of x-coordinates = $\sum_{i=1}^{p} x_i$

* Divided by the Total Number of points.

Average for each dimension by dividing the summed values by the total number points (P).

$$\text{Centroid } x\text{-coordinate} = \frac{\sum_{i=1}^{p} x_i}{P}$$

1. Compute centroid in 2D :

(2,4) (4,5) (6,7)

Sum of x-coordinates $2+4+6 = 12$, $3+5+7 = 15$

2. Divide by the Number of points :

Total no. of points (P) = 3

Centroid x-coordinate $\frac{12}{3} = 4$

Centroid y-coordinate $\frac{15}{3} = 5$

Centroid is (4,5)

Centroid : A centroid is a point in the feature space where the coordinates are the average of the coordinates of all points in the cluster.

$$\text{Centroid } (C) = \frac{1}{P} \sum_{i=1}^{p} x_i$$

Cosine Similarity :

Cosine similarity is the measure of similarity between two non-zero vectors widely applied in many machine learning and data analysis applications.

The Formula for cosine similarity is $\cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|}$

* $A \cdot B$ is the dot product of vectors A and B
* $\|A\|$ and $\|B\|$ are the magnitudes of vectors A and B

* Cosine similarity is a mathematical techique that measures the similarity between two vectors in machine learning
* It's often used in text analysis and recommendation system

① How is cosine similarities used in machine learning ?

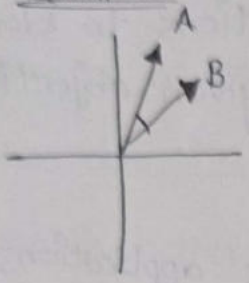
In the realm of machine learning, cosine similarity plays a pivotal role in enhansing text analysis processes. By measuring the similarity between textual data points, this metric enables efficient document clustering and information retrieval systems.

② What is the cosine algorithm in machine learning ?

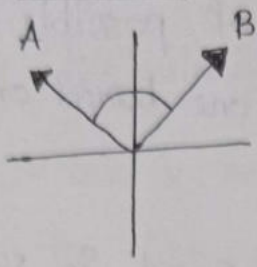
cosine similarity is used as a metric in different machine learning algorithms like the for determing the distance between the neighbors, in recommendation systems, it is used to recommend movies with the same similarities and for texual data, it is used to find the similarity of texts in the document.

The smaller the angle between the two vectors, the more similar they are to each other.

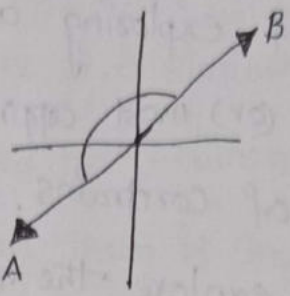
Similar



unrelated



opposite



The cosine similarity is described mathematically as the division between the dot product of vectors and the product of the euclidean norms (or) magnitude of each vector.

$$\cos\theta = \frac{\bar{a} \cdot \bar{b}}{\|\bar{a}\| \, \|\bar{b}\|}$$

$$\|\bar{a}\| = \sqrt{a_1^2 + a_2^2 + a_3^2 + \cdots + a_n^2}$$

$$\|\bar{b}\| = \sqrt{b_1^2 + b_2^2 + b_3^2 + \cdots + b_n^2}$$

Where, a and b are vectors in a multidimensional space.

Since the $\cos(\theta)$ value is in the range $[-1, 1]$

* -1 value is indicates strongly opposite vectors no similarity
* 0 indicates independent vectors (or orthogonal).
* 1 value is indicates high similarity between the vectors.

Search and learning in AI & ML :

Search is a fundamental operation in AI & ML. It refers to the process of exploring a space of possible solutions to identify the best (or) most appropriate one based on a given objective (or) set of contrains.

We explore the role of search in various applications of computer science and it's importance in ML and AI along with relevant examples.

Ex : 1. problem-solving - Sudoku (or) finding shortest path in a graph
Ex. Algorithm

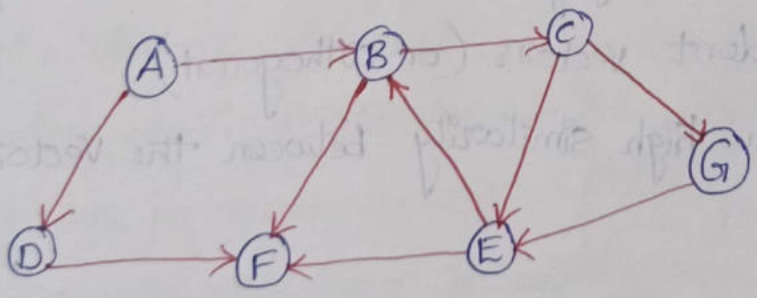
2. Theorem-proving - Automated systems like prolog use search to verify logical proofs.

3. planning - Robots use search algorithms like (BFS) (or) (DFS) to plan paths in navigation.

(BFS) → Breadth-First-Search

(DFS) → depth-first-search

Examples of BFS algorithm :



Adjacency lists

A - B, D
B - C, F
C - E, G
D - F
E - B, F
F - A
G - E

Matching : Matching is an important activity in ML. It is used in both supervised learning and in learning from observations. Matching is carried out by using a proximity measure which can be a distance / dissimilarity measure or a similarity measure. Two data items, u and v, represented as L-dimensional vectors, match better when the distance between them is smaller (or) when the similarity between them is larger.

A popular distance measure is the Euclidean distance and a popular similarity measure is the cosine of the angle between vectors.

The Euclidean distance is given by $d(u,v) = \sqrt{\sum_{i=1}^{L} (u(i) - v(i))^2}$

Euclidean Distance :

Euclidean Distance is the straight-line distance between two points in a multidimensional space. It is calculated using the formula

$$\text{Euclidean Distance} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

using this formula to measure the absolute distance between two vectors (or) points.

Example : Let's consider two points in a 2D space

$A = (1,2)$ and $B = (4,6)$

$$\text{Euclidean Distance} = \sqrt{(4-1)^2 + (6-2)^2}$$

$$= \sqrt{(3)^2 + (4)^2} = \sqrt{9+16} = \sqrt{25}$$

$$= 5$$

If the Euclidean distance is small, the points are closer together.

2. Cosine Similarity :

Cosine similarity measures the cosine of the angle between two vectors. It indicates how similar two vectors are in terms of direction, regardless of magnitude.

> Formula for cosine similarity $= \cos(u,v) = \frac{u^t v}{\|u\| \|v\|}$,

Range : $[-1, 0, 1]$

* $+1$: Completely similar - (same direction)
* 0 : Completely unrelated - (orthogonal)
* -1 : Completely dissimilar - (opposite directions)

Example : let's consider two vectors

$A = [1, 2, 3]$ and $B = [4, 5, 6]$

1. Dot product $(A \cdot B)$:

$$(1 \cdot 4) + (2 \cdot 5) + (3 \cdot 6) = 4 + 10 + 18 = 32$$

1. Magnitude of A and B :

$$|A| = \sqrt{1^2 + 2^2 + 3^2} = \sqrt{1+4+9} = \sqrt{14}$$

$$|B| = \sqrt{4^2 + 5^2 + 6^2} = \sqrt{16+25+36} = \sqrt{77}$$

cosine similarity : $\frac{32}{\sqrt{14} \cdot \sqrt{77}} = \approx 0.974$

cosine similarity close to 1 indicates the vectors are very similar in directions.

Norm : A norm is a function that measures the size of the vector

$|u|$ - vector space

$\|u\| = \sqrt{u_1^2 + u_2^2 + \cdots + u_n^2}$

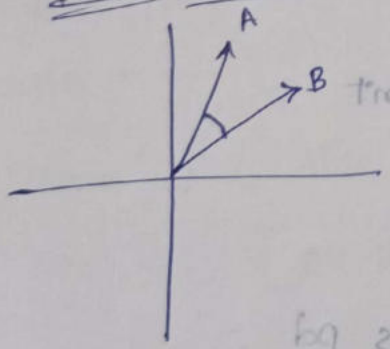
$\|u\| = \sqrt{3^2+4^2} = \sqrt{9+16} = \sqrt{25} = 5$

$\|u\|$ - Double vertical bars

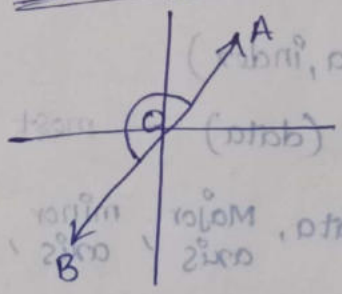
$|v|$ (single vertical bars) → absolute value of a scalar (or) (Given number)

vector – addition of the 2 given values sum value

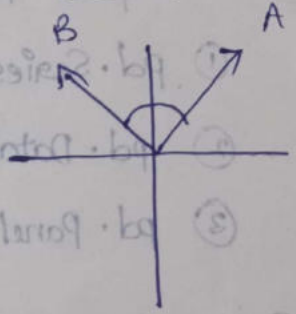
Same direction



opposite direction



orthogonal



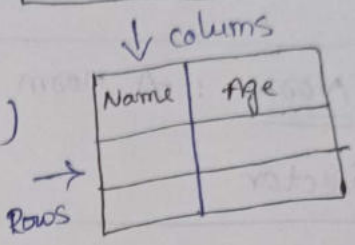
77, 82, 86, 94, A_1, A_2, B_2, B_4, CO, 11,

What is pandas ?

Anakonda Navigator

Pandas is a python library that provides data structures and tools for data analysis and data manipulation

1. High performance Data analysis tool
2. working with large Data sets
3. Supports (or) load files with different formats
4. More flexible
5. Represents in Tabular way (Rows & Columns)
6. Working on missing data
7. Indexing - slicing - Subsetting the large data sets.
8. Merge and join two different datasets easily
9. Reshape data sets.



3 - Data Structures

① Series - One dimensional → list

② Dataframe - Two dimensional → list, Dictionary, Series / Another Dataframe

③ Panel - Multi Dimensional

(data, Major axis, Minor axis)

Syntax : import pandas as pd

① pd.Series(data, index)

② pd.DataFrame(data) - most efficient

③ pd.Panel(data, Major axis, minor axis, dtype)

Series :

① Empty Series

② Series using Array

③ Series using lists

④ Dictionary

① import pandas as pd
S = pd.Series()
Series()

② S = pd.Series(a, index[])

```
import pandas as pd
data = [10, 20, 30, 40]
labels = ['A', 'B', 'C', 'D']
Series = pd.Series (data, index = labels)
print ("Pandas Series with labels :")
print (Series)
```

o/p :

Pandas Series with labels

A	10
B	20
C	30
D	40

Dictionary

```
import pandas as pd
data-dict = {['A':10, 'B':20, 'C':30, 'D':40],
Series = pd.Series (data_dict)
print ("Pandas Series from Dictionary :")
print (Series)
```

o/p

Pandas Series from Dictionary

A	10
B	20
C	30
D	40

dtype : int 64

Pandas dataframe

```
import pandas as pd
data = {
'Name' : ['Alice', 'Bob', 'charlie', 'David'],
'Age' : [24, 27, 32, 29],
'city' : ['Newyork', 'los-Angeles', 'chicago', 'Houston', 'phonix']
}
df = pd.DataFrame (data)
print ("pandas DataFrame :")
print (df)
```

o/p : Name Age city

0	
1	
2	
3	

Learning by Rote :

* Learning by Rote involves memorization in an effective manner
* It is a form of learning.

Ex : That is a form of learning numbers in elementary schools. where the alphabets and numbers are memorized.

* Memorizing the simple additions and multiplications, tables are also examples for the Rote learning.

*

31/1/25

11, 64, Absentees

In Machine learning (ML), learning by Rote refers to a model memorizing ~~training~~ training data instead of generalizing from it. This means the model recalls specific examples rather than identifying, underlying patterns, leading to poor performance on unseen data.

* learning by memorization
* We store the computed values so that we do not have to recompute them later.

Ex : tables, square and cube, Additions, multiplication

* It is a simplest type of learning without any modifications is simply copied into the knowledge base.

Data Sets :

Data sets in ML is a collection of data used to train and test machine learning models.

Inductive - learning by Inductive :

Inductive teaching and learning activities prompt students to generate knowledge through inquiry, reasoning, observation (or) experience, rather than receive it through direct instruction

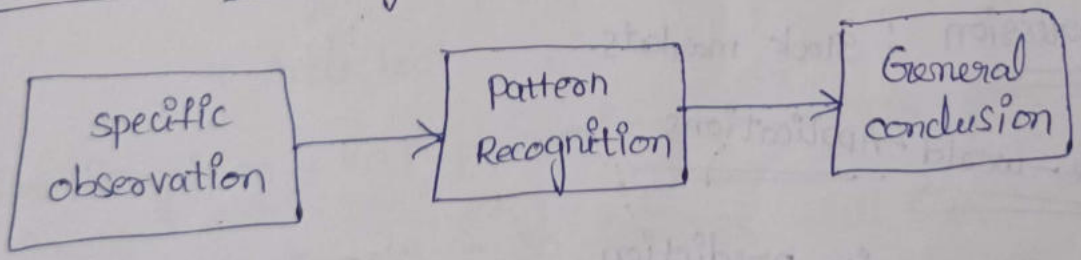
Ex : images, objects, data

There are 4 learning styles - 1. visual
2. auditory
3. Reading
4. Writing

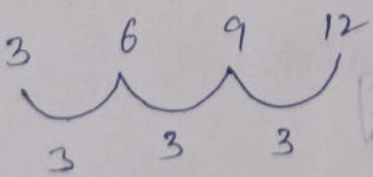
There are two types of Induction process

1. Mutual 2. self Introduction.

Inductive Reasoning :



Ex



To identify the difference b/w 2 values and products the next value number.

Learning by Induction :

The process of learning general patterns (or) rules from a set of specific examples.

Someone is formally introduced into a place of work (or) an organization.

There are 4 type of data - Reading
Writing
Visuals
Audios

Classification : Hand written digits - (0, 9)

Real-world Applications

1. Medical Diagnosis
2. Spam Detection
3. Face Recognition
4. object Detection

Regression : Stock markets

Real-world Applications

1. House price prediction
2. Weather Forecasting
3. Saler prediction
4. Energy consumption Estimation

Data Sets Used :

Data sets for classification

Benchmark dataset for testing machine learning model.

1. MNIST - Handwritten Digits Data set — used for Handwritten digit Recognition (0-9)
   - $28 \times 28 = 784$ (pixels per image)
   - Each pixel value, 0 to 255 / black white
   - Training sets - 60,000 images
   - Test sets = 10,000 images
2. Fashion MNIST Data set
   - ↓
   - A modern alternative to MNIST used for clothing item classification.
   - provided by zalando, an online fashion retailer. [Rang 0 to 255]

[Modified National Institute of standards and Technology MNIST [MNIST]]

classes :
1. T-shirt/top
2. Trouser
3. pullover
4. Dress
5. Coat
6. Sandal
7. Shirt
8. Sneaker
9. Bag
10. Ankle boot

Grayscale - 3 (Red, green, Blue)
0-255
* memory usage less
* processing speed - faster

83, 86, LE 11 Absent

Roll. 88

Liver

(2) face Recognition - 40 people x10 images - 64 x 64 gray scale
(3) Breast Cancer - 569 samples - 30 features - cell size.
(4) Boston Housing - 506 - 13 features
(5) Airline passengers - time series forecasting - monthly record.
(6) Australian weather - weather prediction

Exercise-2 Designing and creating easy-to-communicate

pandas library : visualization using graphs and plots.

(e) Write a program which use pandas inbuilt visualization to plot following graphs :

(i) Bar plots (ii) Histograms (iii) line plots (iv) Scatter plots.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# creating a Sample Dataframe
data = {
'A' : np.random.randint (1, 100, 10),
'B' : np.random.randint (1, 100, 10),
'C' : np.random.randint (1, 100, 10),
}
df = pd.Dataframe (data)
# Bar plot
df.plot (kind = 'bar', figsize = (8,5),
title = 'Bar plot')
plt.show()
# Histogram
df.plot (kind = 'hist', bins = 10, alpha = 0.7, figsize = (8,5), title
    = "Histogram")
plot.show()
# Line plot
df.plot (kind = 'line', figsize = (8,5), title = "Line plot")
plt.show()
# Scatter plot
df.plot (kind = 'scatter', X = 'A',
Y = 'B', figsize = (8,5), title =
"Scatter plot")
plt.show()
```

A, B, C 10 random integers blw 1 TO 100

plot : sequence of the dataframing

figsize → plot size

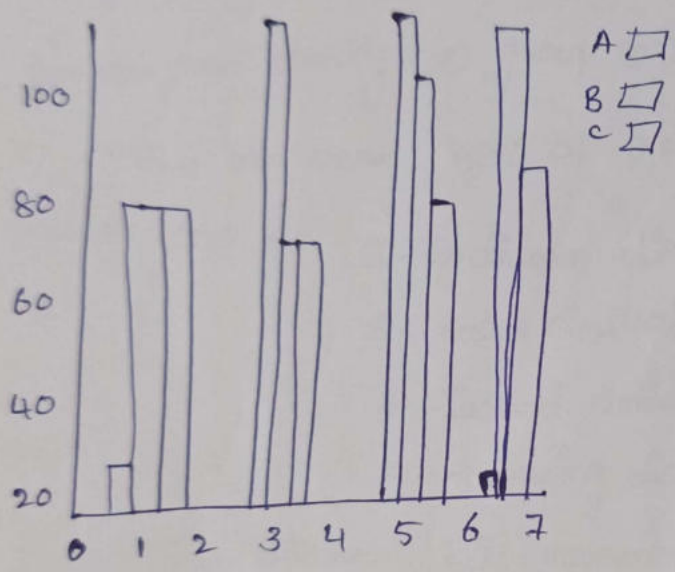
(i) Bar plots : compare categorical data Ex : sales per month

(ii) Histogram : show distribution of numerical data [customer waiting time in Restaurant]

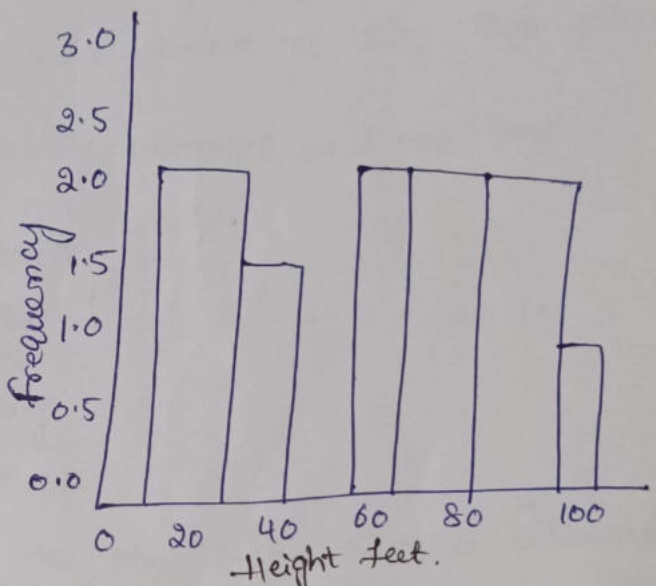
(iii) Line plot : show trends (or) changes over time [Seasonal sales] [Temperature changes in week]

(iv) Scatter plot : show relationship b/w two variables: (temperature)

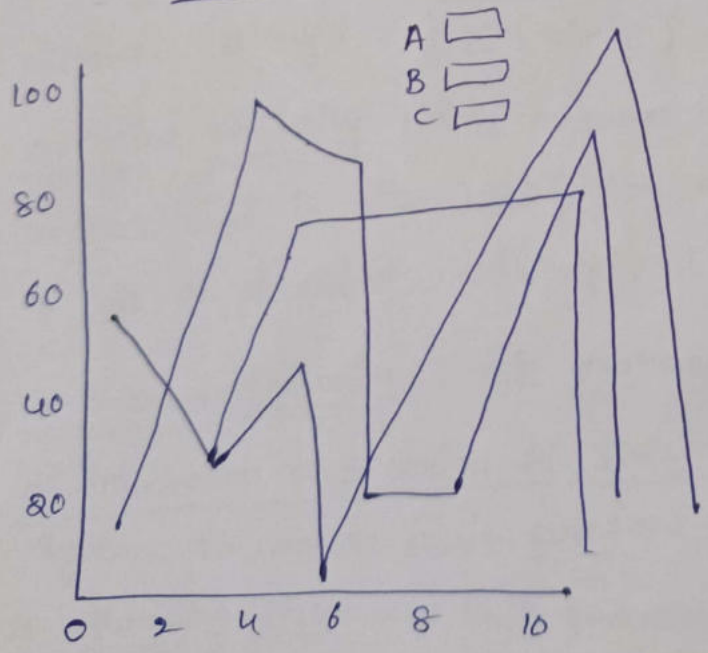
Each plot is generated dynamically from random data, so running the code multiple times will produce different result.



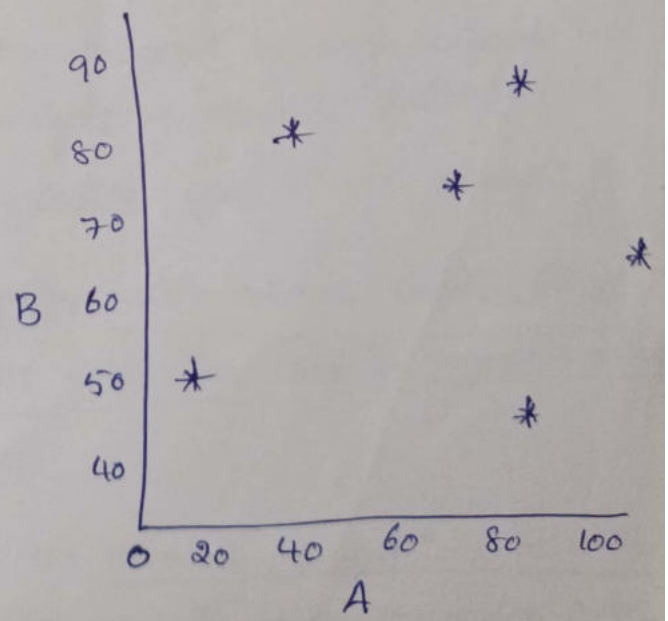
Bar plot



Histogram



Line plot

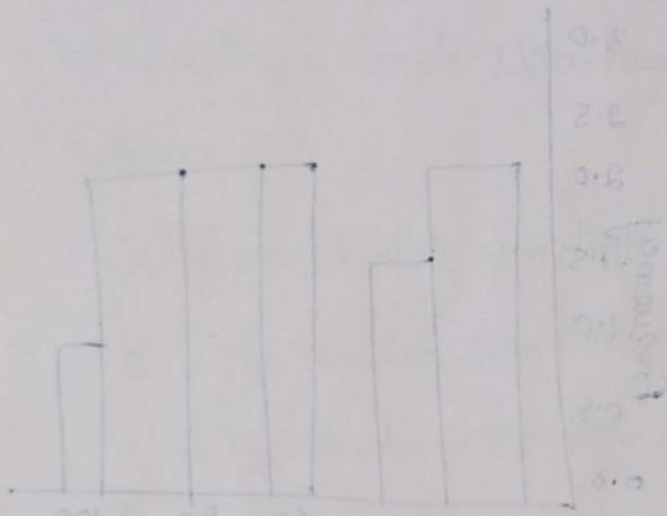


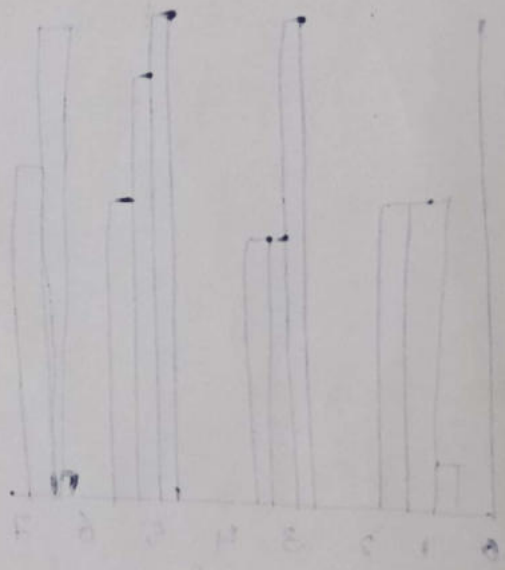
Scatter plot

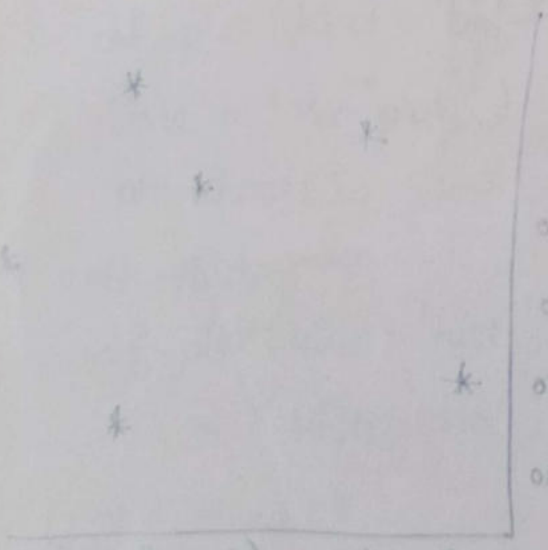
* pandas : used for handling and analyzing structured data (Data Frames)

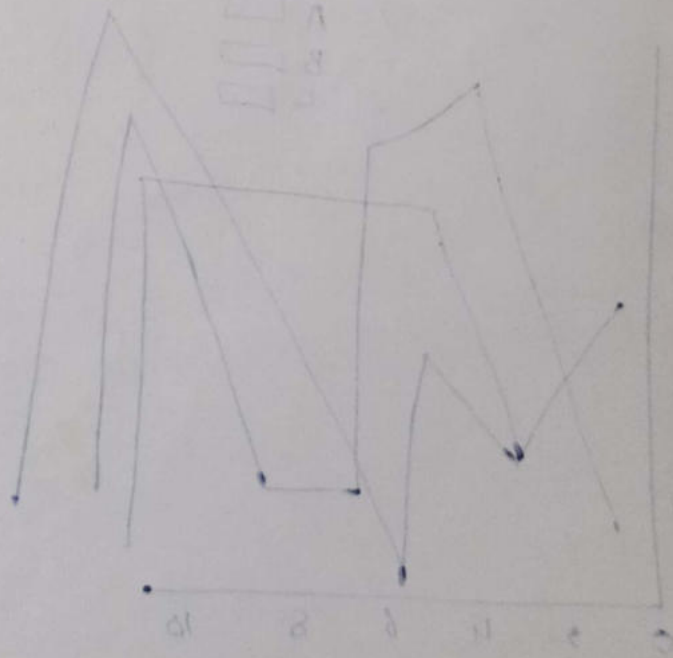
* numpy : used for numerical operations and generating random numbers.

* matplotlib.pyplot : used for plotting and visualization.









Department of Artificial Intelligence and Machine Learning

II B.Tech II Sem

Machine Learning

UNIT-2

Nearest Neighbor-Based Models:

By

Mrs.K.Reddemma,

Assistant Professor,

Department of AI&ML.

7/2/25
108, 110, 113
Absentees
CSE (AI) = B

UNIT - 2

07/02/25
CSE (AI) - A
34, 37, 59, 103

Nearest Neighbor - Based Models :

2.1 Introduction to proximity Measures

proximity measures are used to quantify the degree of similarity or dissimilarity b/w two (or) more pattern vectors.

There pattern vectors represents documents, images (or) even audio (or) video files.

proximity measures are often used by ML algorithms to compare and classify (or) group (or) make predictions using there patterns.

There are many types of proximity measures and the some popular ones are
1. Euclidean distance
2. Cosine similarity
3. Jaccard similarity
4. Hamming distance

1. Euclidean distance : It measures the straight-line distance between two points in multi-dimensional space. so it is also called as the crow-flies distance. $d(x,y) = \sqrt{\sum (x_i - y_i)^2}$

2. Cosine similarity : this measures the cosine of the angle between two vectors and is often used in text analysis to compute similarity b/w a pair of documents. $d(x,y) = 1 - \cos(\theta)$, used in text analysis.

3. Jaccard similarity : this measures the ratio of the cardinalities of intersection over union of two sets and often used in recommendation systems to compare user's preferences.

4. Hamming Distance : This measures the number of positions at which two binary strings differ and is often used in error correction codes.

3. Jaccard similarity formula

The Jaccard similarity (or Jaccard Index) measures the similarity b/w two sets. It is defined as $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$

* $A \cap B$ = Number of elements common in both sets.
* $A \cup B$ = Number of unique elements in both sets.

Example : A = 1, 2, 3, 4

B = 3, 4, 5, 6

* $A \cap B$ = 3, 4 (common elements) → size = 2
* $A \cup B$ = 1, 2, 3, 4, 5, 6 (all unique elements) → size = 6

Formula $J(A,B) = \frac{2}{6} = 0.33$

4. Hamming Distance :

The Hamming distance measures the number of positions at which two strings (or binary sequence) of equal length differ.

It is mainly used in error detection, error correction, and text similarity.

Formula : $H(A,B) = \sum_{i=1}^{n} [A_i \neq B_i]$

* A And B are two strings (or) bit sequences of the same length.
* $[A_i \neq B_i]$ is 1 if the characters at position i are different, otherwise 0.
* The sum gives the number of differing positions.

Example 1

Binary String

Let's say we have two 8-bit binary numbers :

A : 1011101

B : 1001001

comparing each bit :

1	0	1	1	1	0	1	
1	0	0	1	0	0	1	

Hamming distance = 2 (since two bits differ)

Example : 2

For two words of equal length

A : "karolin"

B : "kathrin"

k	a	r	o	l	i	n		
k	a	t	h	r	i	n		

Comparing characters

karolin

kathrin

↑ ↑

Hamming distance = 3 (difference at positions 2, 4 and 6)

2.2 Distance Measures

A distance measures is used to find the dissimilarity between patterns represented as vectors.

Patterns which are more similar should be closer.

The distance function (d) could be a metric (or) a non-metric.

The most popularly used family of distance metrics is called the Minkowski metric.

metric A metric is a type of measure that possesses three key attributes

(Measurements used to evaluate the performance of a model. Error identified improve process)

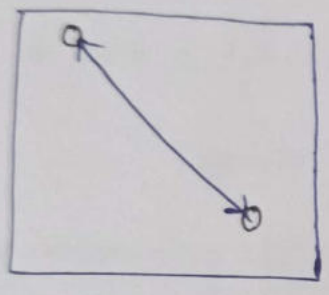
positive reflexivity, Symmentry and triangular inequality

* positive reflexivity : $d(x,x) = 0$
* Symmentry : $d(x,y) = d(y,x)$
* Triangular inequality : $d(x,y) \leq d(x,z) + d(z,y)$

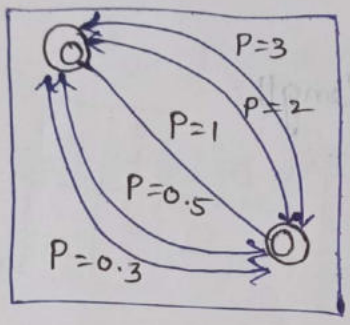
Where x, y, z are any three patterns

The following sections describe different types of dissimilarity /similarity (proximity) measures between patterns vectors.

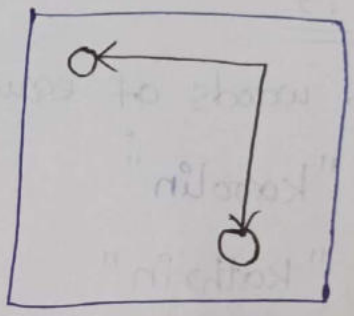
Euclidean

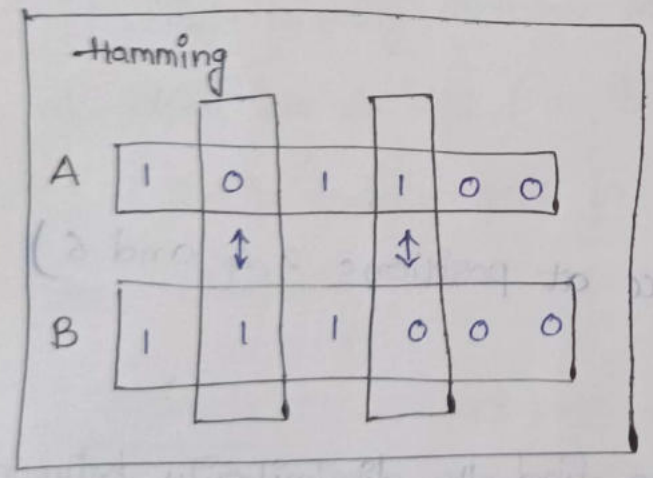


Minkowski



Manhattan





Minkowski Distance :

The Minkowski distance is a generalized metric used to calculate the distance blw two vectors. it is given by

$$d^r(P,Q) = \left(\sum_{K=1}^{L} |P_k - Q_k|^r\right)^{1/r}$$

$|P_k - Q_k|$ → Absolute difference blw two co-ordinates.

Where : P, Q are vectors of length L.

r is a parameter that determines the type of metric

Weighted distance Measures :

Weighted distance Measures are used in Machine learning to calculate the distance between data points while assigning different importance (Weights) to each features (or) dimension.

This is useful in scenarios where certain features are more relevant than others.

There are three types : 1. Weighted Manhattan Distance
2. Weighted Euclidean Distance
3. Minkowski Distance (Generalized Distance)

① Weighted Manhattan Distance :

Manhattan distance is a distance metric used to calculate the distance between two points in a grid-like path.

It is also called L1-distance (or) taxicab distance because it represents the total sum of absolute difference between co-ordinates.

For two points x and y in an L-dimensional space.

$$d(x,y) = \sum_{K=1}^{L} |x_K - y_K|$$

Where

* x_K and y_K are the values of the k-th feature for points x & y
* Absolute difference $|x_K - y_K|$ ensures no negative distance.

Example Calculation :

Suppose we have two points in a 2D space

$x = 3,5$

$y = 1,9$

using Manhattan distance formula : $d(x,y) = |x_1 - y_1| + |x_2 - y_2|$

$d(x,y) = |3-1| + |5-9| = 2+4 = 6$

distance b/w these points is 6

Use Cases

1. K-Nearest Neighbors (KNN) :

Used to measure similarity b/w points

2. K-Means clustering :

When using L1 norm for cluster assignments.

3. Recommendation Systems : To find the closest matches.

4. Anomaly Detection : Helps in identifying outliers based on feature differences.

Example program :

```
from scipy.spatial.distance import cityblock

x = [3, 5]
y = [1, 9]
distance = cityblock(x, y)
print("Manhattan Distance :", distance)
```

output :

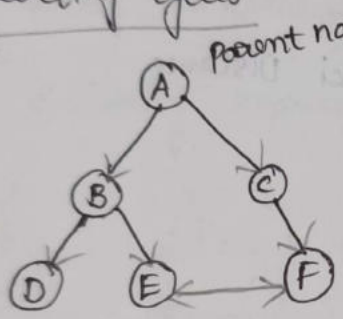
```
Manhattan Distance : 6
```

Lab 11/2/25

Bfs algorithm :

Breadth-First Search (BFS) Algorithm is a fundamental graph traversal algorithm used to explore nodes and edges of a graph systematically. It is particularly useful for finding the shortest path in an unweighted graph.

* Finding shortest path in an unweighted graph
* Network broadcasting (finding all reachable nodes)
* Solving puzzles
* Web crawling
* Detecting cycles



A = B, C
B = E, D
C = (A, F)

D = B
E = B, F
F = C, E

* deque – Double-ended queue (deque) is faster than a list for appending and popping elements
  - add to left side
  - Remove from Right
* FIFO - Initialize
  Loop until queue is Empty

Implementation of BFS Algorithm

1. collections – extend the functionality - build-in type, list, tupple dict, set
2. deque – appending And popping elements from both sides.
3. Graph – Direct graphs, undirect graphs, Weighted graph, unweighted graphs, cyclic graph, Acyclic graph
4. defaultdict ML - knowledge graphs

① BFS algorithm - def

② collections - extends graph functionality - using with build in functionality

Ex : Tupple, list, dict, sets

③ deque - appending and popping element

④ Graphs - Direct, undirect, weighted, unweighted graphs

⑤ default

① Distance Measures - ① Weighted Euclidean
② Weighted Manhattan
③ Minkowski Distance

② Weighted Distance measures - Minkowski Distance

$$d(x,y) = \left[\sum_{k=1}^{L} w_k |p_k - q_k|^r \right]^{1/r}$$

r = parameter value.

P, Q = vector elements

L - diamension

$x = (5,2,4) \quad y = (3,4,2)$

$w_1 = 0.3 \;, w_2 = 0.5 \;, w_3 = 0.2$

$= 0.3 \times (5-3)^2 + 0.5 \times (2-4)^2 + (4-2)^2 \times 0.2$

$= 0.3 \times (2)^2 + 0.5 \times (2)^2 + 0.2 \times (2)^2$

$= 0.3 \times 4 + 0.5 \times 4 + 0.2 \times 4$

$= 1.2 + 2 + 0.8$

$= 4$

Non-Metric Similarity Measures in Machine Learning :

Non-Metric similarity Measures do not satisfy the properties of a Metric distance function.

~~These Measures~~ They are commonly used for images (or) string data and they are resistant to outliers (or) extreamly Noico data one example of a Non-Metric similarity function is the k-median distance between two vectors

Given $x = (x_1, x_2 \cdots x_l)$

$y = (y_1, y_2 \cdots y_l)$

$d(x,y) = k\text{-median} \{|x_1 - y_1| \cdots |x_n - y_n|\}$

similarity measures $s(x,y) = \frac{x^t y}{\|x\| \|y\|}$

$d(x,y) = 1 - s(x,y)$

(Symmetric)

Ex : x, y and z be three vectors in two dimensional space

Angle b/w $x, y = 45$ degrees

$y, z = 45$ degrees

$d(x,y) = 1 - \cos(\pi/2)$

$= 1 - \cos(-1/2)$

$= 1 - \cos(-0.5)$

$= 0.8776$

$d(x,z) + d(z,y) = 1 - \cos(\pi/4) + 1 - \cos(\pi/4)$

$= 2 - \frac{2}{\sqrt{2}}$

$= 0.586$

cos(4) value = -0.6536

Levenshtein Distance :

The levenshtein Distance (Edit distance) is a metric used in Natural language processing (NLP) and string matching tasks to Measure the difference between two sequences (Usually words (or) sentences) It counts the minimum number of operations required to convert one string into another string.

1. Insertion (adding a character)
2. Delition (Removing a character)
3. Substitution (Replacing one character with another)

usages of levenshtein Distance :

1. Text Similarity & Spell checking :

used in search engines and autocorrect to find the closest matching words.

2. plagiarism Detection :

Compares document similarity by checking word variations.

3. DNA Sequence Analysis :

used in bioinformatics to compare genetic sequences.

4. Speech Recognition & chatbots :

Helps recognize variations in words (or) phrases.

5. Data Deduplication :

Identifies similar entries in databases.

Example for levenshtein Distance :

Command for levenshtein - pip install python-Levenshtein

```
import levenshtein
word1 = "machine"
word2 = "maching"   } Define two words
distance = levenshtein.distance(word1, word2)
# calculate levenshtein Distance
print(f "levenshtein Distance between '{word1}' and '{word2}' is:
                "{distance}")
```

output :

levenshtein Distance between 'machine' and 'machine' is : 1

Explaination :

this means only one edit operation (substituting "e" with "g") is needed to convert "machine" into "maching.

Levenshtein Distance : levenshtein distance between two words is the minimum number of single character edits required to change one word into the other.

* Edit Distance (levenshtein Distance)
* Edit distance is a way to measure the similarity of two strings.
* Insertion, deletion, replace.

Ex : FLOMAX } F → V - Substitution
VDLMAX } L → O - Substitution
O → L - Substitution

GILY
GEELY } E → Substraction
E → addition.

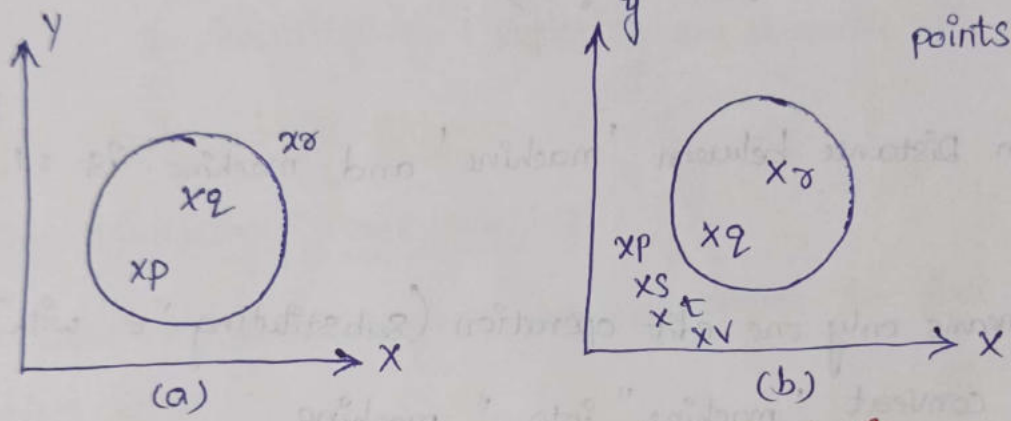
Mutual Neighborhood Distance (MND) :

Mutual Neighborhood Distance (MND) is a function that measures the similarity between two points (or) patterns.

x and y based on their neighborhood relationships. Instead of relying on traditional distance metrics (like Euclidean distance)

It considers how the two points rank each other in terms of proximity to other points in the dataset.

How is MND calculated :



Ranking of the points p, q and r

Mutual Neighbor Distance illustration

* Each point has a neighborhood ranking based on it's distance from other points.
* $N_i N(x, y)$ represents the rank of y when ordered by it's proximity to x.
* $N_i N(y, x)$ represents the rank of x when ordered by it's proximity to y.
* The Mutual Neighborhood Distance (MND) is then computed as

$$MND(x, y) = NN(x, y) + NN(y, x)$$

Relative positional ranking :

	1	2
P	q	r
q	p	r
r	q	p

Mutual Distance :

MND (p,q) = 2

MND (q,r) = 3

MND (p,r) = 4

	1	2	3	4	5
p	s	t	v	q	r
q	p	r	s	t	v
r	q	p	s	t	v

Mutual Distance

MND (p,q) = 5

MND (q,r) = 3

MND (p,r) = 7

properties of MND :

1. Symmetry — MND (x,y) = MND (y,x)
2. Reflexivity - If a point is compared with itself (x=x) then MND (x,x) = 0,
3. Not a Metric — while MND is usefull for measuring similarity It does't satisfy the triangle inequality, meaning it cannot be used as a strict mathematical distance metric.

Formula for MND :

$$MND(A,B) = \frac{1}{|N(A) \cap N(B)|}$$

where * A and B are the two items you are comparing

* N(A) And N(B) are the sets of neighbours for items A and B, respectively.

* $|N(A) \cap N(B)|$ is the number of mutual neighbours between A and B.

If there are no mutual neighbours, the distance is infinite, indicating that the two items are distant or unrelated.

Consider three individuals A, B and C (or) P, Q and R

* P is connected to Q, R and S
* Q is connected to P, R and T
* R is connected to P, Q and U

Steps to calculate Mutual Neighbour Distance (MND)

1. Identify the neighbours of P and Q [N(P), N(Q)]
2. Find the mutual neighbours.
   ↓
   This done by finding the intersection of the two sets of neighbours. $N(P) \cap N(Q)$
   * the number of mutual neighbours is represented as $|N(P) \cap N(Q)|$ - the size of the intersection.
3. Calculate the mutual Neighbour Distance.

$$MND(P, Q) = \frac{1}{|N(P) \cap N(Q)|}$$

if $|N(P) \cap N(Q)| = 0$ there is no mutual neighbours.

Example: Let's say we have 4 items A, B, C and D

* A is connected to B, C and D
* B is connected to A, C and D
* C is connected to A and B
* D is connected to A and B.

$N(A) = \{B, C, D\}$

$N(B) = \{A, C, C, D\}$ $\Rightarrow$ $MND(A,B) = \frac{1}{|N(A) \cap N(B)|} = 1/2$

$N(A) \cap N(B) = C, D$

$|N(A) \cap N(B)| = 2$ $\doteq 0.5$

Mutual distance between A And B is 0.5

$N(C) = \{A, B\}$

$N(D) = \{A, B\}$

MND for each pair :

MND (A,B) :

$N(A) \cap N(B) = \{C, D\} \Rightarrow |N(A) \cap N(B)| = 2$

$MND = 1/2 = 0.5$

MND (A,C) :

$N(A) \cap N(C) = \{B\} \Rightarrow |N(A) \cap N(C)| = 1$

$MND = 1/1 = 1$

MND (A,D) :

$N(A) \cap N(D) = \{B\} \Rightarrow |N(A) \cap N(D)| = 1$

$MND = 1/1 = 1$

MND (B,C) :

$N(B) \cap N(D) = \{A\} \Rightarrow |N(B) \cap N(D)| = 1$

$MND = 1/1 = 1$

MND (C,D) :

$N(C) \cap N(D) = \{A, B\} \Rightarrow |N(C) \cap N(D)| = 2$

$MND = 1/2 = 0.5$

Determine the Relative position of Each Item:

1. Calculate the sum of MNDs for each item.
2. Compare sum of MNDs

Sum of MNDs for A, B, C, D

$A = MND(A,B) + MND(A,C) + MND(A,D)$

$= 0.5 + 1 + 1$

$= 2.5$

$B = MND(B,A) + MND(B,C) + MND(B,D)$

$= 0.5 + 1 + 1$

$= 2.5$

$C = MND(C,A) + MND(C,B) + MND(C,D)$

$= 1 + 1 + 0.5$

$= 2.5$

$D = MND(D,A) + MND(D,B) + MND(D,C)$

$= 1 + 1 + 0.5$

$= 2.5$

Mutual Neighbour Distance is 2.5

Rank the items sequential order dependance on the Distance

1st place : A

2nd place : B

3rd place : C

4th place : D

Proximity Between Binary Patterns :

This section describes different ways to measure similarity (or) distance between binary patterns (binary strings).

We can view L-dimensional binary patterns as binary strings of length l. let p and q be two l-bit binary strings.

Some of the popular proximity measures on such binary patterns are : 1. Hamming Distance

2. Simple Matching coefficient (SMC)

3. Jaccard Coefficient (Jc)

1. Hamming Distance (HD) :

Hamming Distance measures how different two binary strings are by counting the number of bit positions where they differ.

* $p(i) = q(i) \rightarrow$ p and q match on their i^{th} bit
* $p(i) \neq q(i) \rightarrow$ p and q mismatch on their i^{th} bit

(Hamming distance is the number of mismatching bits of the l-bit locations)

Example : consider the two 10-bit patterns p and q given by.

$p = [1]\,0\,0\,0\,0\,0\,1\,0\,0\,[0]$

$q = [0]\,0\,0\,0\,0\,0\,1\,0\,0\,[1]$

The Hamming distance is 2 as they mismatch in bit positions 1 and 10.

2. Simple Matching Coefficient (SMC) :

SMC Measures the proportion of Matching bits (both 0 and 1) out of the total bits.

We define bits where

* M_{00} = number of times both p and q have 0
* M_{11} = " " " " p and q have 1
* M_{01} = " " " p is 0 and q is 1
* M_{10} = " " " p is 1 and q is 0

Formula :

$$SMC(P,Q) = \frac{M_{11}+M_{00}}{M_{00}+M_{01}+M_{10}+M_{11}}$$

* $M_{11} = 1$ (Bit position 7)
* $M_{00} = 7$ (Bit positions : 2, 3, 4, 5, 6, 8, 9)
* $M_{01} = 1$ (Bit position 10)
* $M_{10} = 1$ (Bit position 1)

$$SMC(P,Q) = \frac{M_{11}+M_{00}}{10} = \frac{1+7}{10} = \frac{8}{10} = 0.8$$

3. Jaccard Coefficient (Jc) :

Jaccard Coefficient only considers positions where at least one string has a 1. It measures the ratio of positions where both are 1 to the total positions where at least one is 1.

Formula : $Jc(P,Q) = \frac{M_{11}}{M_{01}+M_{10}+M_{11}}$

17/02/25

Best First Search Algorithm :

Best first search is an instance of the general Tree search (or) graph search algorithm is which

Breadth first search And Best first search

face book suggestions

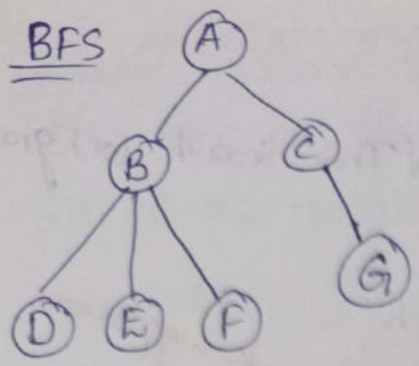
* Both graph traversal / search algorithms
* Different their approach and efficiency.
* Explores all neighbors of a node before moving deeper into the graph (level-wise exploration)
* A queue to store nodes (FIFO) T.C = O(V+E)
* Unweighted graph, find shortest path.
* Uniformly
* Unweighted graph
* Time complexity O(V+E) → Edges.
  ↓
  Vertices
* Social network analysis.

Time com- O(E log V)

Best-First Search

* prioritizing nodes that seem closer to the goal
* A priority queue to store nodes, sorted by heuristic values.
  ↓
  Min-heap
* Time complexity O(E log V)
  (priority queue operations take logarithmatic time)
  ↓
  Google maps, AI pathfinding.
* priority Based.
* depends on priority destination point.

BFS



⇒ Start at A — A

Expand A - B, C

B = C, D, E, F

C — D, E, F, G

D — E, F, G

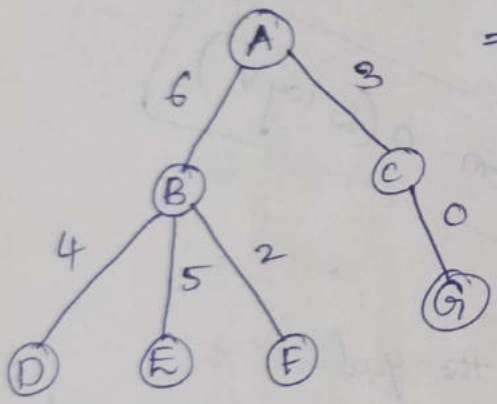
E - F, G

F - G

Expand G — Goal Reached

A B C D E F

Best First Search



⇒ Start At A → A

Expand A → choose low value path

c (h=3)

Expand c → G. (h=0 goal reached)

path → A, C, G

proximity between Binary patterns

1. Hamming Distance (HD)
2. Simple matching coefficient (SMC)
3. Jaccard coefficient (JC)

For the given example :

* $M_{11} = 1$
* $M_{01} = 1$
* $M_{10} = 1$

$$J_C(P,Q) = \frac{1}{1+1+1} = \frac{1}{3} \approx 0.33$$

Consider again :

$P = 1000001000$

$Q = 0000001001$

Here $M_{11} = 1$, $M_{00} = 7$, $M_{01} = M_{10} = 1$

$$SMC(P,Q) = \frac{1+7}{7+1+1+1} = \frac{8}{10} = 0.8$$

$$J_C(P,Q) = \frac{1}{1+1+1} = 1/3 \approx 0.33$$

Different classification Algorithms Based on the Distance Measures

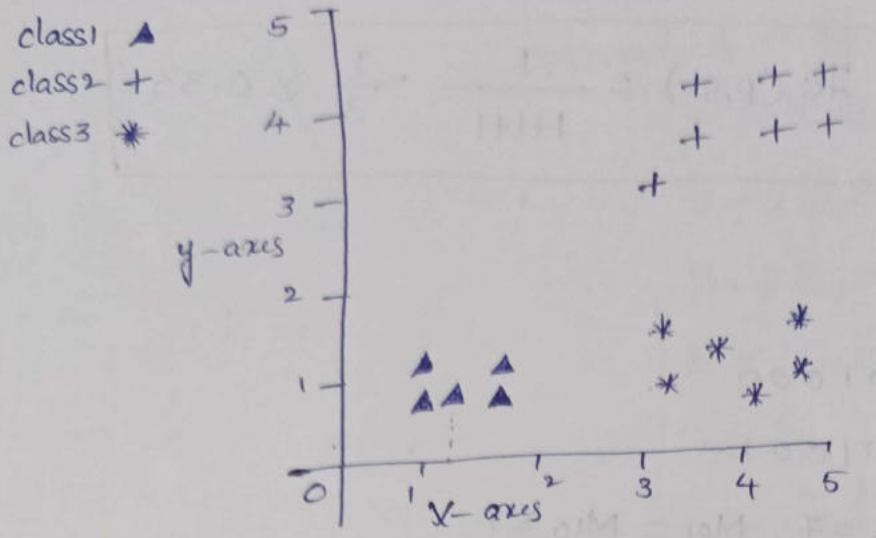
Many classification algorithms rely on distance measure to determine the similarity (or) dissimilarity between data points. this algorithms classify new data points based on their proximity to existing labeled data.

Below are some key classification algorithms that utilize distance Measures :

1. K-Nearest Neighbors (k-NN) classifier
2. Nearest Neighbor classifier (NNC)
3. Weighted K-Nearest Neighbors Algorithm (WKNN)
4. Radius Distance Nearest Neighbor Algorithm
5. Tree-Based Nearest Neighbor Algorithm
6. Mahalanobis Distance-Based classifiers.

21/02/25
(SE(A2)-B / -A → 11, 47
U3, 3 members
B6, C1, 112
3 members

K-Nearest Neighbor classifier :



class 1 A → 5 points
class 2 (+) → 4 points
class 3 (*) → Multiple points

K-Nearest Neighbours (KNN) is a simple, non-parametric and lazy learning algorithm used for classification and regression tasks in Machine learning. It is based on the principal that similar data points exist in close proximity.

$$d = \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$$

point	(x, y)	Distance to (4,4)
A	(1,2)	$\sqrt{(4-1)^2+(4-2)^2} = 3^2+2^2 = 9+4 = \sqrt{13} = 3.6$
B	(2,3)	$\sqrt{(4-2)^2+(4-3)^2} = \sqrt{2^2+1^2} = \sqrt{5} = 2.2$
C	(3,3)	1.4
D	(5,5)	1.4
E	(6,7)	3.6
F	(8,8)	3.7

Different classification Algorithms based on the distance Measures.

22/02/25

KNN - Regression :

K-Nearest Neighbor Regression is a Supervised learning algorithm used to predict continuous values instead of catagorical labels. It works by finding the K-Nearest data points to a given input and averaging their corresponding output values to make a predict

KNN - Regression works :

1. Store all training data - No explicit model is built
2. Find the k-Nearest Neighbors of a given input data point based on a distance metric

   Ex : Euclidean distance.
3. Extract the output values (Y-values) of there nearest neighbours.
4. Comput the predicted output by taking the average of there Y-values.

Mathematical formula :

For a given input x, the predicted value $\hat{y}$ is

$$\hat{y} = \frac{1}{K} \sum_{i=1}^{k} y_i$$

$$\hat{y} = \frac{1}{K} (y^1 + y^2 + \cdots + y^K)$$

$$X = \{(x_1, y_1), (x_2, y_2) \cdots (x_n, y_n)\}$$

1. Find the K-nearest neighbors of x from n data vectors.
   Let them be $x^1, x^2, \cdots x^K$
2. Consider the K-y values associated with there x^is.
   Let them be $y^1, y^2, \cdots, y^K$.

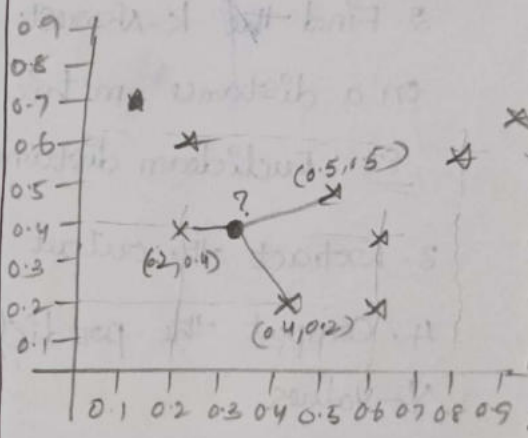
3. Take the average of there y's and declare this average values to be the predicted value of y associated with x.

So the predicted values of Y, call it $\hat{y}$ is

$$\hat{y} = \frac{1}{k}(y^1 + y^2 + \cdots + y^k)$$

Ex:

Number (i)	pattern (x_i)	Target (y_i)
1	(0.2, 0.4) ✓	8
2	(0.4, 0.2) ✓	8
3	(0.6, 0.4)	12
4.	(0.8, 0.6)	16
5.	(1.0, 0.7)	19
6.	(0.8, 0.4)	14
7.	(0.6, 0.2)	10
8.	(0.5, 0.5) ✓	12
9.	(0.2, 0.6)	10



New pattern be $x = (0.3, 0.4)$

$k = 3$

* The 3 Nearest Neighbours of x from the patterns in the table are $(0.2, 0.4)$, $(0.4, 0.2)$, and $(0.5, 0.5)$
* The corresponding target values observed in the table 8, 8 and 12
* The average of there values is $\frac{8+8+12}{3} = 9.33$
* So predicted target value for $x = (0.3, 0.4)$ is 9.33.

Comparison with a Linear Model :

The data can also be modeled using a linear equations.

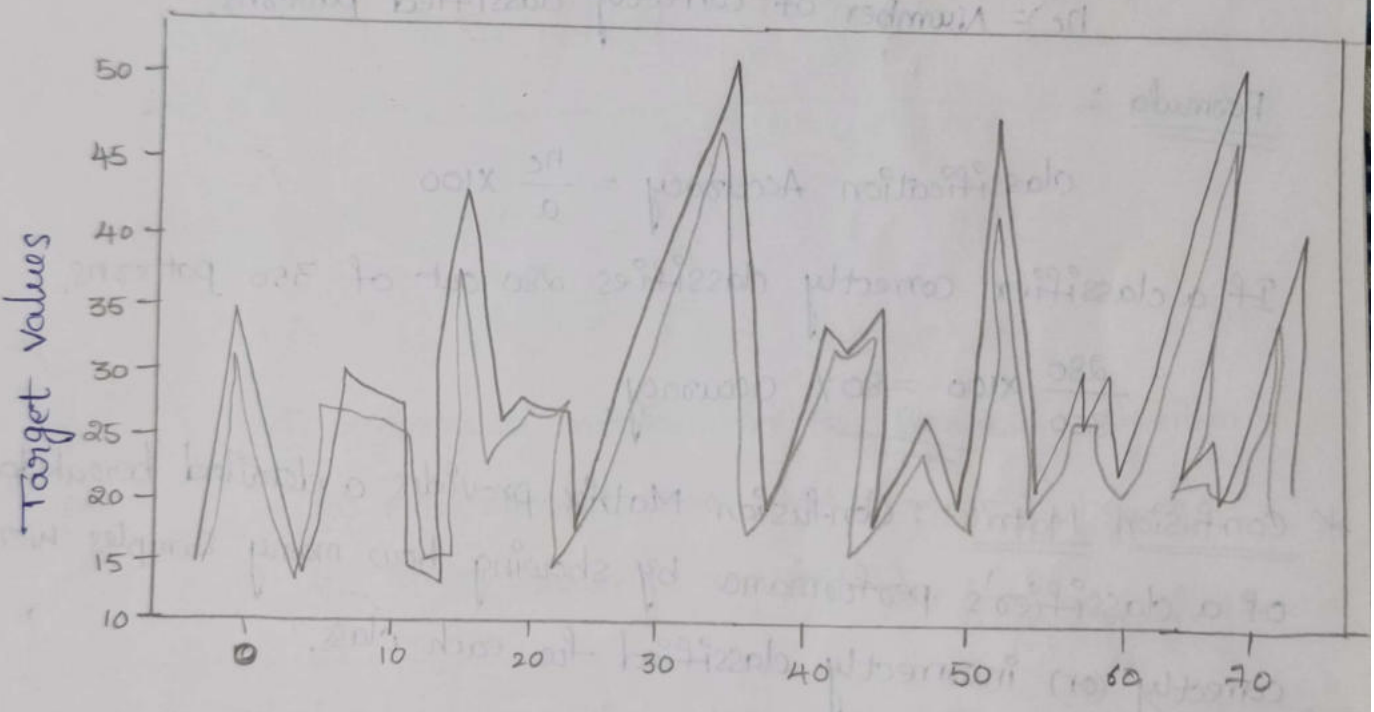
$$y_i = a x_i(1) + b x_i(2) + c$$

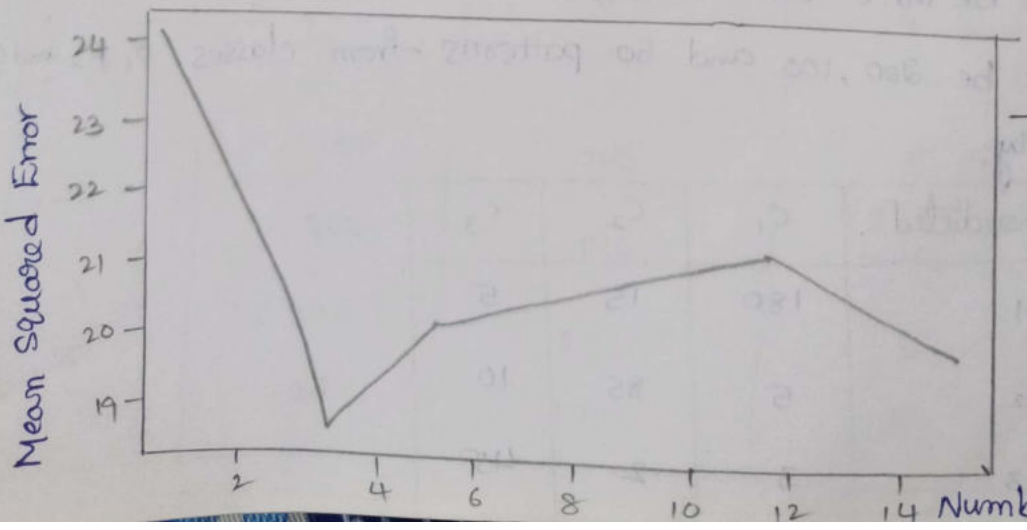
1. Boston Housing Dataset Example

* A real-world dataset containing 13-features

  Ex : Crime rate

* 200 training samples and 50 test samples
* The goal is to predict house prices using k-NN regression.
* Mean Squared Error (MSE) is used to measures prediction accuracy.





→ MSE values on the Boston Housing data KNN Regression.

Performance Measures :

There are several measures used to evaluate the performance of ML models. We will consider some of the popular measures in this sections.

performance of classifiers :

* classification Accuracy :

classification accuracy is a basic performance metric that calculates how many predictions were correct out of the total predictions made.

Given

a = Total number of patterns classified.

n_c = Number of correctly classified patterns.

Formula :

$$\text{classification Accuracy} = \frac{n_c}{a} \times 100$$

If a classifier correctly classifies 280 out of 350 patterns.

$$\frac{280}{350} \times 100 = 80\% \text{ accuracy.}$$

* Confusion Matrix : Confusion Matrix provides a detailed breakdown of a classifier's performance by showing how many samples were correctly (or) incorrectly classified for each class.

* let there be three classes c_1, c_2 and c_3
* let there be 200, 100 and 50 patterns from classes c_1, c_2 and c_3 respectively.

True/predicted	c_1	c_2	c_3
c_1	180	15	5
c_2	5	85	10
c_3	3	2	45

KNN- Regression → classification accuracy - Ex - House Price Predict based on the sq feets →

Confusion Matrix Accuracy $= \frac{TP+TN}{TP+TN+FP+FN} = \frac{35+50}{35+50+10+5}$

$= \frac{85}{100} = 85\% = 0.85$

positive values production $= \frac{TP}{TP+FP} = \frac{35}{35+10} = \frac{35}{45} = 0.78$

Negative values predictions $= \frac{TN}{TP+FN} = \frac{35}{35+5} = \frac{35}{45} = 0.875$

KNN Reg $X = (x_1, y_1) + (x_2+y_2) \cdots (x_n, y_n)$

Actual values = X =

$\hat{y} = \frac{1}{K} \sum_{i=1}^{n}$

22/02/25- CSE[A2]-A- 15, 44 & numbers

① performance matrics for the Regression Algorithm

ⓐ Mean Absolute Error (MAE) $= \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$

ⓑ Mean Squared Error (MSE) $= \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

House	Actual value y_i	predicted values $\hat{y}_i$	Absolute values $y_i - \hat{y}_i$	Squared values
1	200	210	-10	10
2	250	245	5	5
3	180	190	-10	10
4	220	215	5	5
5	275	270	5	5

Performance Metrics of the Regression Algorithms :

Performance Metric : Performance Metric are used to evaluate how well a model is performing with respect to the task it is designed for. These metrics give you insight in to the models accuracy, realiability, and effectiveness in making predictions.

The choice of metric depends on the type of machine learning task and the specific goals of the model.

The most common metrics include : 1. Mean Absolute Error (MAE)
2. Mean Squared Error (MSE)

1. Mean Absolute Error (MAE) : MAE measures the average magnitude of Errors between predicted and actual values, without considering their directions. It is easy to interpret and is less sensitive to outliers.

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

Where : y_i is the actual value

$\hat{y}_i$ is the predicted values

n is the number of data points

Example problem / Real world entity usage Example :

House price prediction : predict the price of a house based on it's features

Ex : Size - square feets
Location
Number of Rooms

After training your model, you want to evaluate it's performance using MSE.

MSE calculations :

Actual price	predicted price
250,000	260,000
300,000	310,000
350,000	340,000
400,000	390,000
450,000	460,000

25/02/25

CSE(AI)-A -10.15 TO 11.15

1, [redacted], 10, 11, [redacted], [redacted], 14, 15, [redacted]

[redacted], [redacted], 26, [redacted], [redacted], [redacted], [redacted], [redacted], 41

[redacted], [redacted], [redacted], [redacted], 48, 49, [redacted], [redacted], [redacted]

42, [redacted], [redacted]

[redacted], [redacted]

12, 20, 29, 32, 36, 46.54

59, 60

The MSE would be calculated as

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{y}_i)^2$$

MSE =

Mean Squared Error (MSE) :

MSE measures the average squared difference b/w actual and predicted values.

$$MSE = \frac{1}{n}\sum(y_i - \hat{y}_i)^2$$

Houses	Actual price	predicted price	Error $(y_i - \hat{y}_i)$	Absolute Error	squared Error
1	200	210	-10	10	100
2	250	245	5	5	25
3	180	190	-10	10	100
4	220	215	5	5	25
5	275	280	-5	5	25

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad \text{- To find Average Error}$$

$$MAE = \left|\frac{10+5+10+5+5}{5}\right| = \frac{35}{5} = 7$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad \text{outliers}$$

$$MSE = \frac{100+25+100+25+25}{5}$$

$$MSE = \frac{275}{5} = 55 \rightarrow \text{To find large errors}$$

Department of Artificial Intelligence and Machine Learning

II B.Tech II Sem

Machine Learning

UNIT-3

Models Based on Decision Trees:

By

Mrs.K.Reddemma,

Assistant Professor,

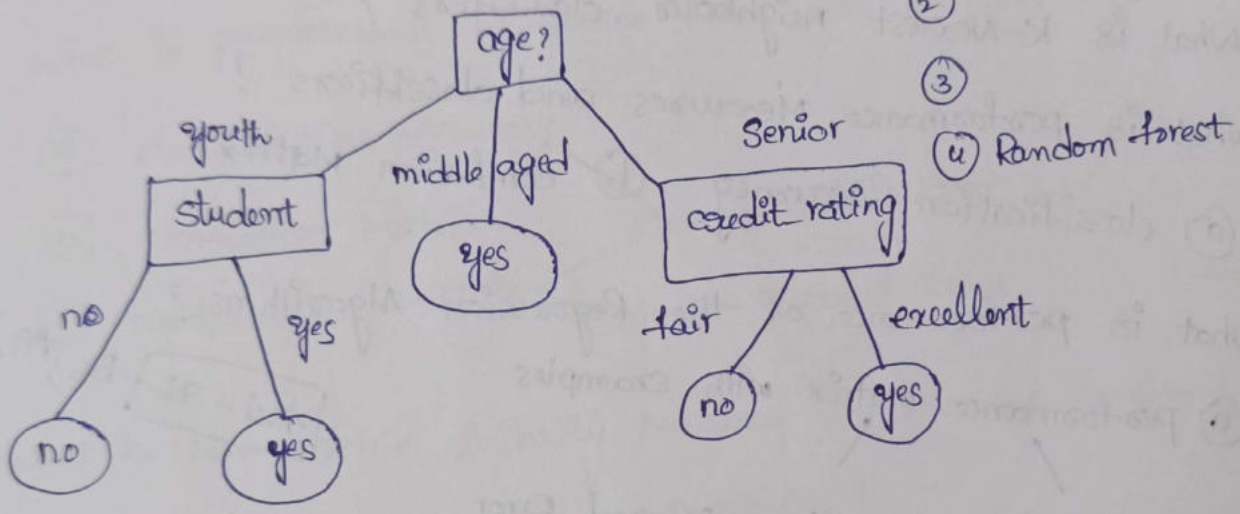
Department of AI&ML.

UNIT-3

Model Based on Decision Trees

Introduction to Decision Trees :

A Decision Tree is a Supervised learning model used for classification and Regression. It splits data into smaller subsets based on feature values, creating a tree-like structure, where each internal node represents a decision.



RID	age	income	student	credit_rating	class : buys_computer

① greedy
②
③
④ Random forest

① greedy – purity in splitting – each child node dependes on the parent node.

② It can be expensive – no. of features depends on the no. of patterns.

③ It can overfit – DT

path is better update the path, To identify the shortest path of the given edges. f_score, g_score.

chapter-3

Models Based on Decision Trees

Decision Trees : Decision Tree is a popular structure in ML. It is built by splitting the data set associated with a node into subsets that typically have less entropy (or) better purity. Splitting is carried out using the values of a selected features this feature-based decision-making at each of the internal nodes offers transparency and is ideally suited for explanation and for easier understanding by application domain experts.

* **properties of Decision Tree building algorithms :**

1. It is greedy
2. It can be expensive
3. It can overfit
4. Random Forest
5. Ada Boost (Adaptive Boosting)
6. Gradient Boost

} predictive models using Decision Trees.

1. It is greedy : ~~the best feature~~

A Decision Tree is greedy because it selects the best features (in terms of purity) at each step, without considering the overall best tree structure

* So, tests feature used at each of the child nodes will depend on the feature selected at the parent node.
* Such a sequential greedy process may fail to be optimal on the whole.

Example :

Imagine you are classifying students based on their grades and study habits.

* Split the study time because it provides the highest purity at that moment.
* But a better long-term strategy could have been spliting based on previous exam scores.

(2) It can be expensive :

Selection of a feature depends both on the number of features and the number of patterns.
So Tests conducted on the feature values at each node need to be simple.

Even simple tests can become computationally demanding when the dimensionality is large.

Example :

Imagine a dataset with 10,0000 records and 100 features

* Each node has to check 100 different splits.
* If the tree grows deep (Ex depth = 10)
  there are 1000 of computations.

Solution : To reduce computation expenses

Calculation of the Entropy values : Decision Tree for Classification

Entropy : Entropy measures the impurity (or) uncertainty in a data set. It is used in decision trees to determine the best feature for splitting data.

Entropy Formula : $H(S) = -\sum p_i \log_2 p_i$

Entropy $H(S)$ for a dataset S with classes $C_1, C_2 \dots C_n$ is given by

$$H(S) = -\sum p_i \log_2 p_i$$

Where :

* p_i = proportion of samples belonging to class i
* $\log_2$ is the base-2 logarithm.

Example Calculation :

Case 1 : pure Dataset (All in one class)

Ex : 10 positive, 0 negative

probabilities : $P_+ = 1$, $P_- = 0$

Entropy : $H = -(1 \cdot \log_2 1 + 0 \cdot \log_2 0)$

$H = 0$ high purity

Case 2 : perfectly impure Dataset (50-50) split

Ex : 5 positive, 5 negative

probabilities = $P_+ = 0.5$, $P_- = 0.5$

Entropy :

$H = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5)$

$= -0.5 - 0.5$

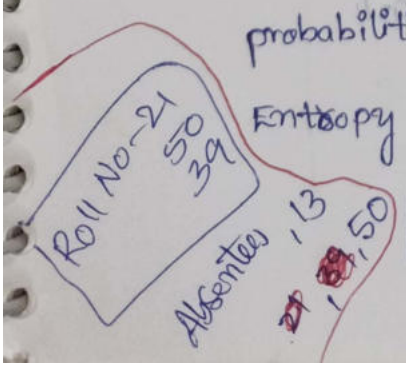
$H = 1$ Low purity

Case 3 : partially mixed Dataset (7, positive, 3 Negative)

probabilities : $P_+ = 0.7$, $P_- = 0.3$

Entropy $(H) = -(0.7 \log_2 0.7 + 0.3 \log_2 0.3)$

$H = 0.88$



Entropy Before splitting :

Let's say we have a dataset with 10 samples.

* 6 belong to class A ($P_A = 6/10 = 0.6$)
* 4 belong to class B ($P_B = \frac{4}{10} = 0.4$)

$$H_{parent} = -(\text{class A } \log_2 \text{ class A} + \text{class B } \log_2 \text{ class B})$$

$$= -(0.6 \log_2 0.6 + 0.4 \log_2 0.4)$$

$$= 0.97$$

Entropy After splitting :

Now, Suppose we split this dataset into two subsets :

* left node — 4 Samples (3A, 1B)

Entropy (H) = 0.81

* Right node : 6 Samples (3A, 3B)

Entropy = 1

$\Rightarrow$ Total Entropy after splitting

$$H_{children} = \frac{4}{10}(0.81) + \frac{6}{10}(1)$$

$$\boxed{H = 0.924}$$

Information Gain calculation $IG_1 = H_{parent} - H_{children}$

$$= 0.97 - 0.924$$

$$\boxed{IG_1 = 0.046}$$

Example : Five different probability distributions and their entropy values.

class	class 1 (P_1)	class 2 (P_2)	Entropy
1	100 ($P_1 = 1.0$)	0 ($P_2 = 0.0$)	0.0
2.	75 ($P_1 = 0.75$)	25 ($P_2 = 0.25$)	0.811
3.	50 ($P_1 = 0.5$)	50 ($P_2 = 0.5$)	1.0
4.	25 ($P_1 = 0.25$)	75 ($P_2 = 0.75$)	0.811
5.	0 ($P_1 = 0.0$)	100 ($P_2 = 1.0$)	0.0

Entropy in Two classes scenario

Entropy is symetric meaning $\boxed{\text{entropy}(P_2, P_1) = \text{entropy}(P_1, P_2)}$

this is because

$$\text{entropy}(P_1 P_2) = -P_1(\log P_1) - P_2(\log P_2) - P_2 \log P_2 - P_1 \log P_1$$

$$\boxed{\text{entropy}(P_1 P_2) = \text{entropy}(P_2, P_1)}$$

In a two-class problem, entropy is maximum when $P_1 = P_2 = 0.5$

partialy derivatives of entropy and equating them to zero.

$$\frac{\partial \text{entropy}(P_1, P_2)}{\partial P_1} = -1 - \log(P_1)$$

$$\frac{\partial \text{entropy}(P_1, P_2)}{\partial P_2} = -1 - \log(P_2)$$

A dataset used to illustrate splitting :

Pattern	Feature 1	Feature 2	class
1	1	1	1
2	1	3	1
3	2	1	1
4	2	3	1
5	4	1	1
6	4	3	2
7	5	1	2
8	5	3	2

Entropy Calculation:

Subset 1

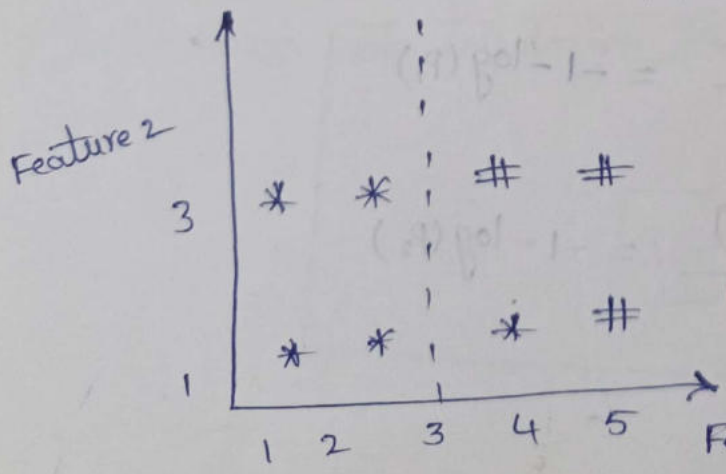
$$4|4 \log(1) + 0 \log(0) = 0$$

Subset 2

$$= -\frac{1}{4} \log\left(\frac{1}{4}\right) - \frac{3}{4} \log\left(\frac{3}{4}\right)$$

$$= \frac{1}{4} \log(4) + \frac{3}{4} \log\left(\frac{4}{3}\right)$$

$$\approx 0.244219$$



Two-dimensional pattern of the data set given in Table 2

3. Decision Trees Can Overfit

Overfitting happens when a decision tree learns too many details from training data, making it less generalizable to new da

* A deep tree can memorize training data rather than finding general patterns.
* Small variations and noise in the training data are treated as important, reducing performance on test data.

Pruning : pruning is a technique that removes unnecessary bran chess after training.

* pre-pruning — Early stopping
* post-pruning — Grow the full tree than remove branches that do not improve accuracy on a validation set

* DT - Accuracy on data set - low
* pruned - " " " " - High

Decision Tree

(1) Greedy Nature - Makes fast decisions but may not be optimal
(2) Expensive - large data sets and high-dimensional featur increase processing time.
(3) overfit - Deep trees memorize data but fail on new exa

Here each of subsets has 4 elements out of the total of 8 elements present.

So the weight are equal to $\frac{4}{8} = \frac{1}{2}$

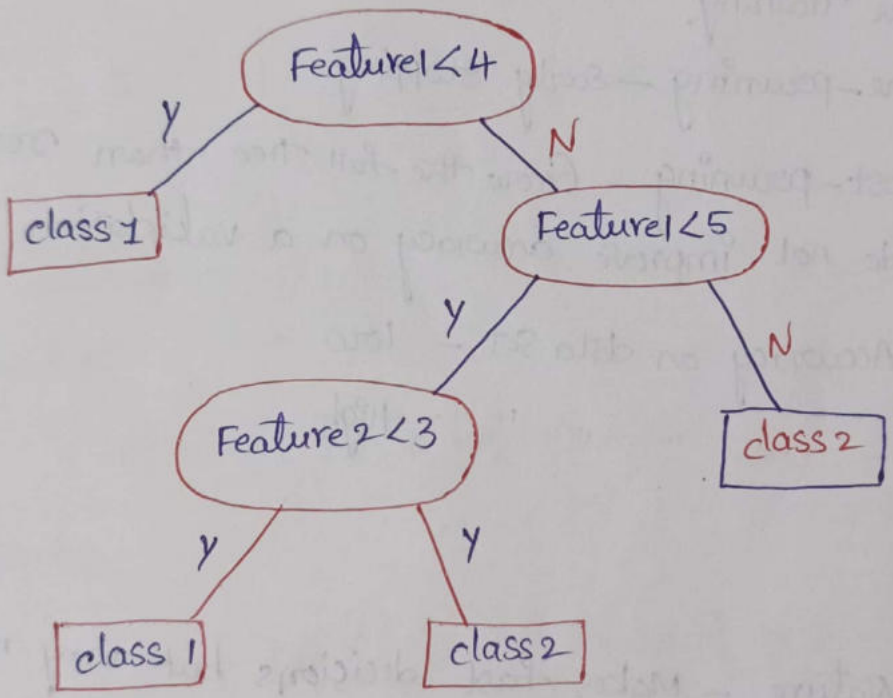
So the weighted value of entropy impurity is

$$\frac{1}{2} \times 0 + \frac{1}{2} \times 0.244 = 0.122$$

So total weighted entropy associated with the split is

0.1221

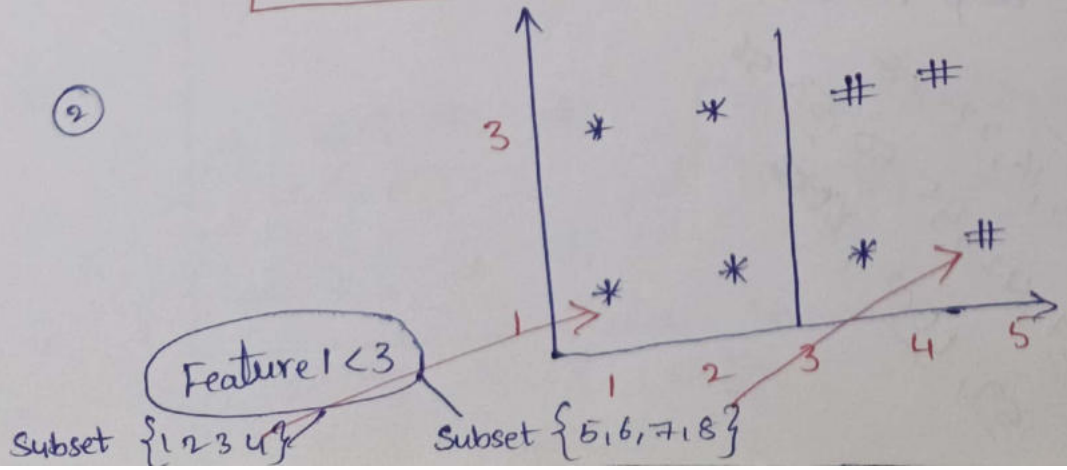
(1)



Decision Tree for the data set

$$\frac{1}{2}(0.244 + 0.301) = 0.2726$$

(2)



Subset : {1, 2, 3, 4}

Entropy : 0, Weight = 0.5

Subset : {5, 6, 7, 8}

Entropy : 0.244, Weight = 0.5

3.3 Impurity Measures For Decision Tree Construction

① Gini Index (impurity at a node for a C-class problems)

The Gini index is a metric used in ML to measure the impurity (or) disorder of a data set. It is commonly used in Decision Trees to determine the best feature to split the data.

Formula : $Gini = 1 - \sum_{i=1}^{C} P_i^2$

Where

* C = Total no. of classes
* P_i = probability of a particular class in a node.

* Range - 0 To 1

⇒ Gini = 0 → The node is pure (all samples belong to the same class)

⇒ Gini = 1 → The node is maximally impure (Samples are evenly distributed among classes)

$$Gini = 1 - \sum_{i=1}^{C} P_i^2$$

$$= 1 - (P_1^2 + P_2^2)$$

$$= 1 - ((0.8)^2 + (0.2)^2)$$

$$= 1 - (0.64 + 0.04)$$

$$= 1 - 0.68$$

$$= 0.32$$

(If a node contains 80% of class P_1 and 20% of class P_2 the Gini Index value is 0.32)

② Misclassification impurity :

$$1 - \max_j(P_j)$$

* P_j is the probability of a class at that node.

Ex : Suppose a node contains the following class distribution

class A = 80% - (0.8)

class B = 15% - (0.15)

class C = 5% - (0.5)

Then the misclassification impurity is

$= 1 - \max(P_j)$ - | $1 - \max(0.7)$

$= 1 - \max(0.8)$ | $1 - 0.7$

$= 1 - 0.8$ | $= 0.3$

$= 0.2$ |

(Misclassification impurity is a metric used in decision trees to measure how often a randomly chosen element would be incorrectly classified if we used the majority class in a given node.)

Example : To find the outlook, Temperature, Humidity (properties of decision Tree properties are) : 1. outlook
2. Temperature
3. Humidity.

To determine the data set properties based on the given data set.

18/03/25 - 10 TO 11
CSE (AI) - A

Decision Tree

Data set Decision Tree properties

Independent features

Pattern	outlook	Temperature (F)	Humidity (%)	class
1	Sunny	65	91	NP
2	Sunny	85	92	NP
3	overcast	85	86	P
4	overcast	70	65	P
5	Rainy	70	65	NP
6	Rainy	79	62	P

For class columns P and NP are represents as

* P - for play
* NP - for No play

① To calculate outlook (F) Feature selection of splitting :

Using outlook

* Subsets → (1,2) (3,4) (5,6)
* subset (5,6) is pure $P_1 = 1$, $P_2 = 0$ [Gini index Rang (0,1)]
  (2,3,4,5,6)
* For (1,2) and (3,4) = $P_1 = P_2 = 0.5$

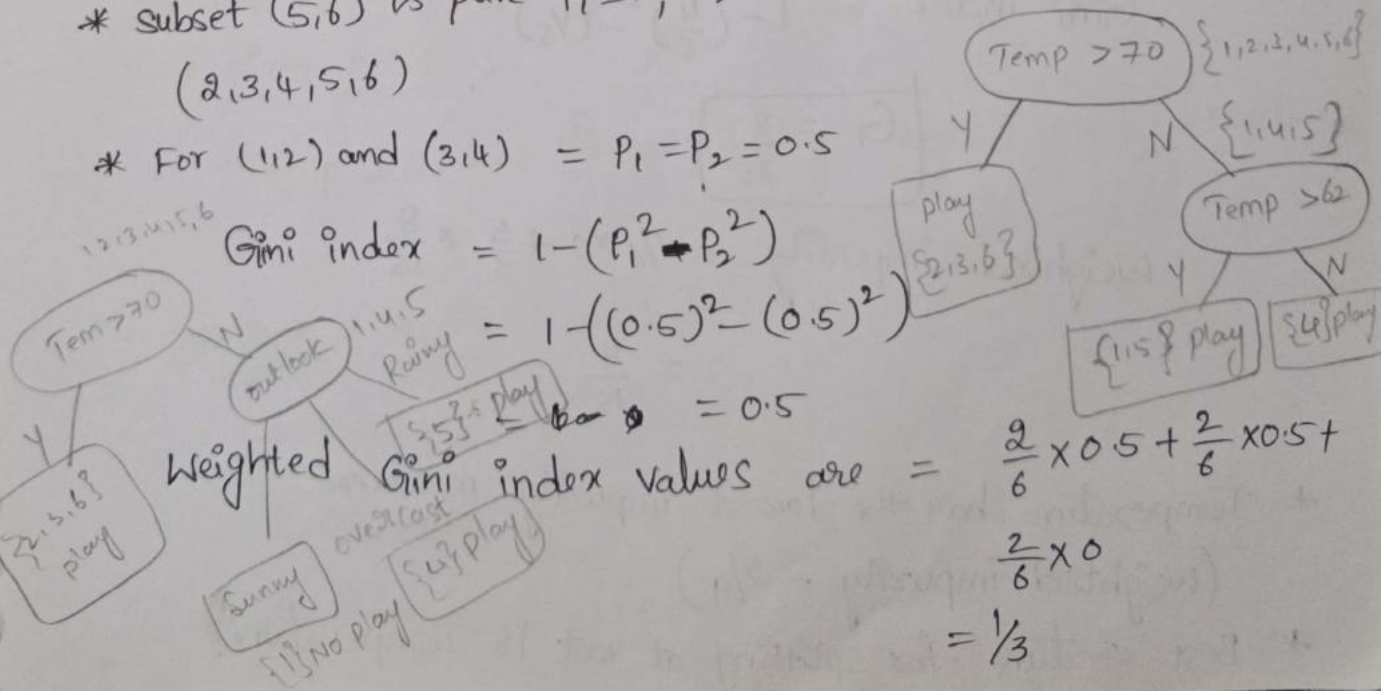
Gini index = $1 - (P_1^2 + P_2^2)$

= $1 - ((0.5)^2 - (0.5)^2)$

= 0.5

Weighted Gini index values are = $\frac{2}{6} \times 0.5 + \frac{2}{6} \times 0.5 + \frac{2}{6} \times 0$

= 1/3



② using Temperature (F°) ≥ 70

Subsets :

Subset 1 – (2, 3, 6)

subset 2 – (1, 4, 5)

Subset (1) is pure – Gini index = 0

for (2, 3, 6) → $P_1 = \frac{2}{3}$, $P_2 = 1/3$

Gini index = $1 - (P_1^2 - P_2^2)$

= $1 - (\frac{2}{3})^2 - (1/3)^2$

$G_1 = \frac{4}{9}$

③ using Humidity < 91%

Split at Humidity < 91

* Two subsets : Subset {1} is pure – Gini index = 0

Subset {2, 3, 4, 5, 6}

$P_1 = \frac{4}{5}$, $P_2 = 1/5$

Gini index = $1 - (\frac{4}{5})^2 - (1/5)^2$

$G_1 = \frac{8}{25}$

Weighted impurity = $1/6 \times 0 + \frac{5}{6} \times \frac{8}{25}$

= $\frac{4}{15}$

* Temperature has the lowest impurity measure (Weighted impurity = 2/9)
* Best feature for splitting at root is Temperature.

By using Temperature > 70 at the root

We obtain :

* pure Subset {2,3,6} → classified as play
* Second Subset {1,4,5}, which needs further splitting

Two options for further splitting :

1. Temperature > 62
2. outlook-based split

Decision Tree for the Data Set

(a) using Temperature-based split

* First split :

* Temp > 70 → {2,3,6} (play) and {1,4,5} (needs further split)

* Second split :

Temp > 62 → {4} (play), {1,5} (No play)

(b) using outlook-based split

* First split :

Temp > 70 → {2,3,6} (play) and {1,4,5} needs further split

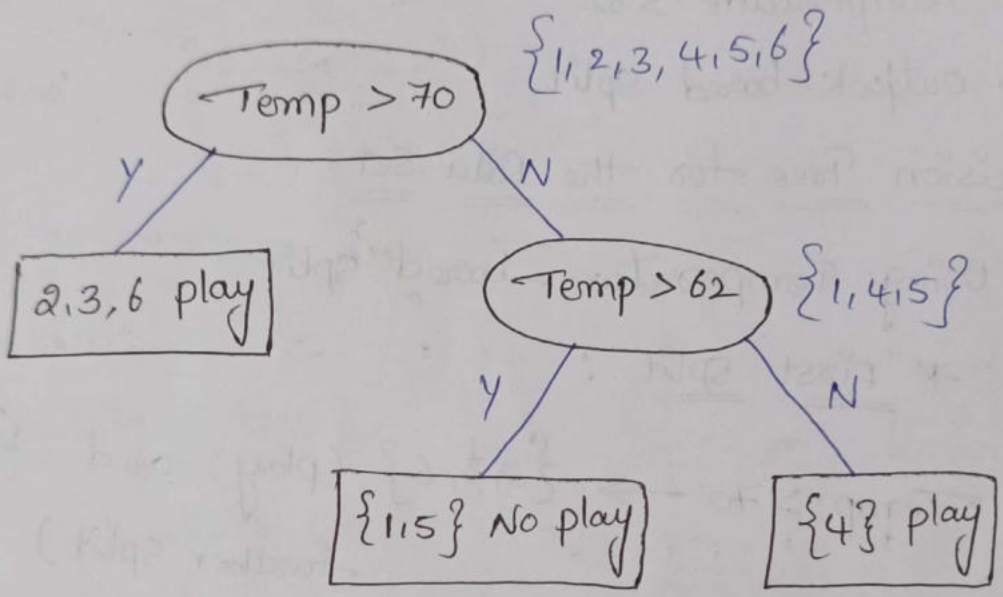
* Second split : outlook

* Sunny → {1} (No play)
* Over cast → {4} (play)
* Rainy → {5} (play)

Impurity Measures :

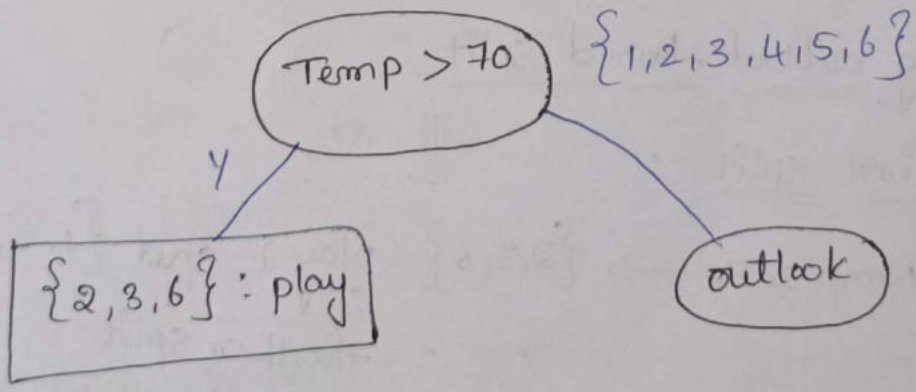
In Machine learning an impurity measure is a metric used to quantify the disorder (or) randomness in a data set. particularly in classification problems. It helps in decision tree algorithms (like CART – classification and Regression Trees) to determine the best feature for splitting nodes.

(a)



Decision Tree for the data set

(b)

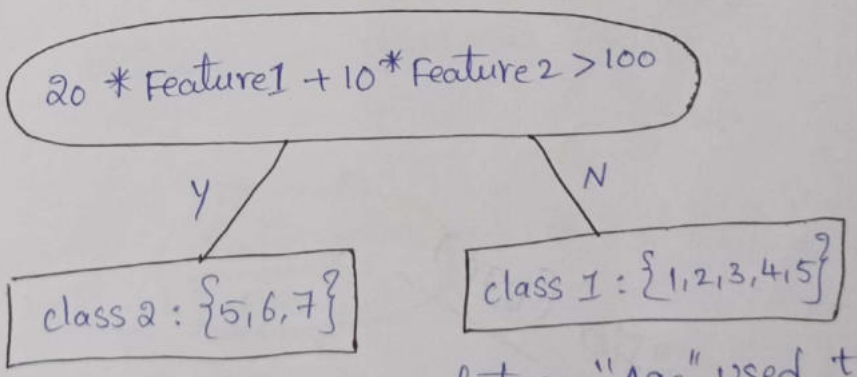


3.4 properties of the Decision Tree classifier (DTC)

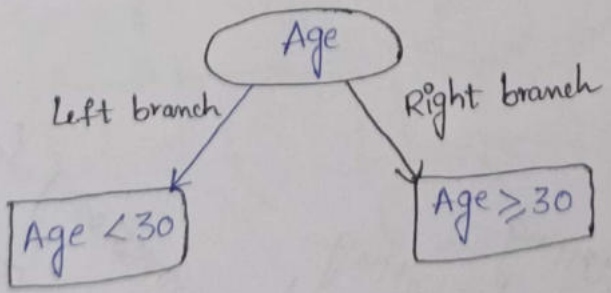
1. Splitting Rule
2. Criterion for splitting Rule
3. Binary (or) Non-Binary
4. Termination condition
5. class labels
6. classification
7. Transparancy
8. Handling mixed data types
9. Eleminating irrelevent features
10. pouning a decision tree.

1. Splitting Rule :

The process of dividing the data set into subsets at each node in the tree based on a selected feature.



Example : Consider a dataset with a feature "Age" used to split data



2. Criterion for splitting Rule :

A mathematical measure used to determine the best way to split data at a node. common criteria include : 1. Gini impurity (used in CART)

2. Entropy (information Gain) (used in ID3)

Ex : For splitting based on Gini impurity

If a node contains 80% class A and class B 20%

$$\text{Gini impurity} = 1 - (P_1^2 - P_2^2)$$

$$= 1 - ((0.8)^2 - (0.2)^2)$$

$$GI = 0.32$$

3. Binary (or) Non-Binary :

* Binary Splitting : Each node splits into exactly two branches.

* Non-Binary Splitting : A node may split into multiple branches based on categorical values.

Example : A no

Binary : salary < 50K → yes (left) / No (Right)

Non-Binary : A node may split

10.15 | ME(AI)-A 12:15 PM LEO5,

Regression Based on Decision Trees :

In decision tree-based regression, the goal is to predict a continuous target variable by recursively splitting the dataset based on the feature values. The decision tree chooses the best split by minimizing the weighted squared Error (SE).

step 1 : Understanding the given Data :

The dataset consists of two features (f_1 and f_2) and a target value. The table represents 9 data points with different values of f_1, f_2 and their respective target values.

Number	f_1	f_2	Target (y)
1	0.2	0.4	8
2	0.4	0.2	8
3.	0.6	0.4	12
4.	0.8	0.6	16
5	1.0	0.7	19
6	0.8	0.4	14
7.	0.6	0.2	10
8.	0.5	0.5	12
9.	0.2	0.6	10

step 2 : Decision Tree Splitting Criterion :

The decision Tree determines splits by evaluating different values of f_1 and choosing the split the minimizes the squared error.

The squared error for a split is calculated as :

$$SE_{split} = \frac{n_1}{n} \sum_{x_i \in S_1} (y_i - \mu_1)^2 + \frac{n_2}{n} \sum_{x_i \in S_2} (y_i - \mu_2)^2$$

Where :

* S_1 and S_2 are the two subsets created after the split
* n_1 and n_2 are the number of elements in each subset
* u_1 and u_2 are the means of the target values in each subset

The mean of each subset is calculated as :

$$u_1 = \frac{1}{n_1} \sum_{x_i \in S_1} y_i \quad , \quad u_2 = \frac{1}{n_2} \sum_{x_i \in S_2} y_i$$

* First we have to find the u_1 and u_2 values and split the data sets as a S_1 and S_2.

$f_1 < 0.8 \longrightarrow \{1,2,3,7,8,9\}$ (S_1) $\{4,5,6\}$ (S_2)

$u_1 \longrightarrow \frac{8+12+10+12+10}{6} = \frac{60}{6} = 10$

$u_2 \longrightarrow \frac{16+19+14}{3} = \frac{49}{3} = 16.33$

$SE(S_1) \Rightarrow (8-10)^2 + (8-10)^2 + (12-10)^2 + (10-10)^2 + (12-10)^2 + (10-10)^2$

$\Rightarrow (2)^2 + (2)^2 + (2)^2 + (0)^2 + (2)^2 + (0)^2$

$\Rightarrow 4+4+4+0+4+0$

$\Rightarrow 16$

$SE(S_1) = 16$

$SE(S_2) \Rightarrow (16-16.33)^2 + (19-16.33)^2 + (14-16.33)^2$

$\Rightarrow (0.33)^2 + (2.67)^2 + (2.33)^2$

$\Rightarrow 0.1089 + 7.1289 + 5.428$

$SE(S_2) \Rightarrow 12.6658$

Weighted (SE) $= \frac{6}{9} \times 16 + \frac{3}{9} \times 12.66$

$= \frac{2}{3} \times 16 + \frac{1}{3} \times 12.66$

$= 10.66 + 4.22$

$= 14.88$

$W(SE) = 14.88$

Step-3 : possible splits Based on f_1

To determine the best split, the decision tree considers different threshold values for f_1

$f_1 < 0.4$
$f_2 < 0.5$
$f_3 < 0.6$
$f_4 < 0.8$
$f_5 < 1.0$

For each split, the dataset is divid into two subsets, and the squared error is computed. The split with the lowest squared error is choosen.

Step-4 : Evaluating Each split :

For each split

1. Divide the dataset into S_1 (values satisfying the condition) and S_2 (Remaining values)
2. Compute the means u_1 and u_2
3. Compute the squared error contribution from both subsets.
4. Find the total squared errors.

The split with the smallest squared error is the optimal choice for the first decision node in the tree.

Squared Error (SE) values for possible splits

Test	SE of S_1	SE of S_2	Weighted SE
$f_1 < 0.4$	2	82	64.22
$f_1 < 0.5$	2.67	52.93	36.11
$f_1 < 0.6$	11	48.8	32.04
$f_1 < 0.8$	16	12.66	14.9
$f_1 < 1.0$	55.5	0	49.33
$f_1 < 1.0$	108	0	108

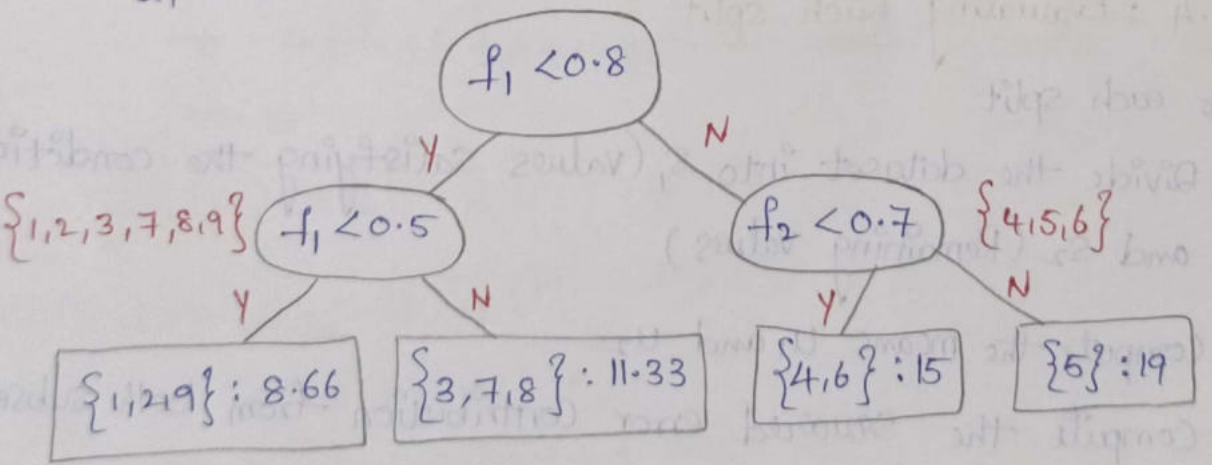
The minimum value of weighted SE, that is 14.9 is obtained for the split based on the test $f_1 < 0.8$.

Subsets S_1 and S_2 for this optimal split are

$S_1 = \{1, 2, 3, 7, 8, 9\}$

$S_2 = \{4, 5, 6\}$

$u_1 = 10$ and $u_2 = 16.33$



Decision tree for the regression data set

Step 1 : Decision Tree Construction :

1. Start at the Root Node :

* The entire dataset is considered at the root
* The model selects the best feature (f_1 (or) f_2) an the best split based on minimizing the squared error.

2. Splitting the data :

* Suppose we choose $f_1 < 0.8$ as the first split.
* If $f_1 < 0.8$, the data goes to the left child.
* If $f_1 \geq 0.8$, the data goes to the right child.

3. Further Splitting :

* For the left child ($f_1 < 0.8$) another split occurs based on $f_1 < 0.5$
* If $f_1 < 0.5$, another split based on f_2 is made.

4. Leaf node :

* When no further meaningful splits are found, the leaf nodes are formed.
* Each leaf node is represented by the mean of the target values in that region.

Step 2 : predicting a new Data point :

Let's say we have a new test point (0.3, 0.4)

* Start at the root
  Since $f_1 = 0.3 < 0.8$ → Move to the left child.
* Next, Since $f_1 = 0.3 < 0.5$ → Move to the left child.
* Finally, the instance lands in a leaf node with an average target value of 8.66
* So, the decision tree predicts 8.66 for (0.3, 0.4)

Step-3 : Computing the squared Error :

* The true target value for (0.3, 0.4) is 9
* The squared error is calculated as

$$(9-8.66)^2 = (0.34)^2 = 0.111$$

Comparison with KNN Regression :

* In KNN Regression, a different approach is used.

  Ex: Averaging k-nearest neighbors.

* For the same test point (0.3, 0.4), KNN Regression predicted 9.33.
* The squared error for KNN was

$$(9-9.33)^2 = (-0.33)^2$$

$$= 0.111$$

The squared error for KNN was 0.111

General Steps to Build a Decision Tree for Regression :

1. Start with the entire ~~root~~ node dataset at the root node.
2. Find the best feature and split point by minimizing the weighted squared error.
3. Divide the dataset into two subsets based on the choosen split.
4. Repeat the process recursively on each subset until a stopping condition is met.

   Ex : max depth, minimum samples per node.
5. Assign the mean target values to each leaf node.
6. Use the tree to make predictions by following decision rules from the root to a leaf.

3.7 Bias - Variance Trade-OFF :

Bias - Variance Trade-OFF :

The bias-variance trade-off is a fundamental concept in machine learning. It explains the balance between a model's simplicity (bias) and complexity (variance).

* **Bias** : Error due to overly simplistic assumptions in the learning model.
* **Variance** : Error due to excessive sensitivity to the training data.

This example illustrates bias and variance through polynomial fitting.

Step 1 : Understanding the Data in Table.

The table consists of four examples with features z and target values y :

Example	z	Target (y)	y_{g1}	y_{p1}	y_{p2}
1	0	1	2/3	1	? 1
2	1	3	11/3	3	? 3
3	2	7	20/3	7	? 7
4	1/2	$(1.75)^{7/4}$	2.16	1.84	1.87

y : Actual Target value

y_{g1} : predictions using a linear model

y_{p1}, y_{p2} : predictions using polynomial models of different degrees

Step 2 : Fitting a linear Model

A simple linear regression model is of the form

$$y = mx + c$$

Given the first three data points, we set up the least-squares equations.

$$\sum_{i=1}^{3} y_i = 3c + m \sum_{i=1}^{3} x_i$$

$$\sum_{i=1}^{3} x_i y_i = c \sum_{i=1}^{3} x_i + m \sum_{i=1}^{3} x_i^2$$

$y = mx + c$

Substituting values :

$$11 = 3c + 3m$$

$$17 = 3c + 5m$$

solving for m and c :

$$m = 3, \quad c = 2/3$$

Thus, the linear model is :

$$y = 3x + \frac{2}{3}$$

Step : 3 : Making predictions :

Bias - Variance Trade-Off

Example : There are four examples with their respective target values consider only for the first three columns in the table. let us select the first 3 rows to train the model and the 4th row for testing let us start with a simple linear model.

It is a straight-line fit of the form $y = mx+c$. where m and c are the unknowns. The least-square fit for the first three points gives us the following two equations based on the two unknowns.

polynomial fits to illustrate bias and variance :

Example	x	Target (y)	y_{S1}	y_{P1}	y_{P2}
1	0	1	2/3 = 0.67	1	1
2	1	3	11/3 3.67	3	3
3	2	7	20/3 6.67	7	7
4	1/2 Low variance	7/4	2.16 underfit	1.84 overfit	1.87 overfit

Step 1 : Simple Linear Model

The first approach is to fit a linear model of the form :

$y = mx+c$ → where - m is the slope, c is the intercept

using the least squares method, two equations are derived.

The summation of target values for training is taken from the first three rows of table 3.7, where the target values (y) are given as :

Example	x	Target (y)
1	0	1
2	1	3
3	2	7

The total sum of the target values is : $\Sigma y_i = 1+3+7$

$= 11$

$y = 11$

⇒ The given linear model is $y = mx + c$

⇒ The least squares method gives the first summation equation

$$\Sigma y_i = c\Sigma 1 + m\Sigma x_i$$

⇒ Since there are three ~~traing~~ training data points : $\Sigma 1 = 3$

And the sum of the x-values is $\Sigma x_i = 0+1+2 = 3$

⇒ So, Substituting these into the equation : $11 = 3c + 3m$

This is the first equation used to solve for m and c.

$y = mx + c$

$y = 11, \quad m = c, \quad c = 3$

$11 = 3x + 3m \rightarrow (1)$

Step 2 :

Least squares Equations :

Sum of y values :

$$\sum_{i=1}^{c} y_i = 3c + m \sum_{i=1}^{3} x_i$$

$$\Sigma y_i = c\Sigma 1 + m\Sigma x_i$$

$y_i = y = 1+3+7 = 11$ (Target values Sum)

$1+3+7 = 3c + m(0+1+2)$ (sum of x values)

$11 = 3c + 3m$

This gives us the first equation :

$11 = 3c + m3$

Second Equation : Sum of $x_i y$ values :

$$\Sigma x_i y_i = c\Sigma x_i + m\Sigma x_i^2$$

x values X target values

$(0x1) + (1x3) + (2x7) = c(0+1+2) + m(0^2+1^2+2^2)$

$0+3+14 = c(3) + m(1+4)$

$17 = 3c + 5m$

This second Equation $17 = 3c + 5m$

Step 3 : Solving for c and m

Now we have the system of equations

① $11 = 3c + 3m$

② $17 = 3c + 5m$

Substract the first equation from the second equation.

$(3c+5m) - (3c+3m) = 17-11$

$2m = 6$

$m = 6/2$

$\boxed{m = 3}$

Substituting $m=3$ into the first equation

$11 = 3c + 3(3)$

$11 = 3c + 9$

$3c = 11-9$

$3c = 2$

$\boxed{c = \frac{2}{3}}$

So, the final linear model is : $y = 3x + \frac{2}{3}$

For this data of three points, we can also fit (overfit) a degree-3 polynomial. In general we can have infinite polynomials of degree 3 that can fit the three training patterns. We consider two such polynomials that capture all the three training examples, that is, the first 3 rows, perfectly.

$y_p = 1 + ax + bx^2 + cx^3$

The polynomials are $y_{p_1} = 1 + \frac{3}{2}x + \frac{1}{4}x^2 + \frac{1}{4}x^3$

$y_{p_1} = 1 + \frac{3}{2}x + \frac{1}{4}x^2 + \frac{1}{4}x^3$

$y_{p_2} = 1 + \frac{5}{3}x + \frac{1}{3}x^3$

Degree-3 polynomial :

$(y_{P_1} = \frac{3}{2}x + \frac{1}{4}x^2 + \frac{1}{4}x^3)$

$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3$

We use the three training points

① $(x_1, y_1) = (0, 1)$

② $(x_2, y_2) = (1, 3)$

③ $(x_3, y_3) = (2, 7)$

Substuting there values into the polynomial equation

① $(x_1, y_1) = (0, 1)$

$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3$

$x = 0, \; y = 1$

$1 = a_0 + a_1(0) + a_2(0)^2 + a_3(0)^3$

$1 = a_0 + 0 + 0 + 0$

$1 = a_0$

$\boxed{a_0 = 1}$

② $(x_2, y_2) = (1, 3)$

$x = 1, \; y = 3$

$3 = a_0 + a_1(1) + a_2(1)^2 + a_3(1)^3$

$3 = a_0 + a_1(1) + a_2 + a_3$

$3 = a_0 + 1(a_1) + a_2 + a_3$

$a_1 + a_2 + a_3 = 3 - 1$

$\boxed{a_1 + a_2 + a_3 = 2}$

③ $(x_3, y_3) = (2, 7)$

$x = 2, \; y = 7$

$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3$

$7 = a_0 + a_1(2) + a_2 (2)^2 + a_3 (2)^3$

$7 = a_0 + 2(a_1) + 4(a_2) + 8(a_3)$

$2(a_1) + 4a_2 + 8a_3 = 1 - 7$

$$\boxed{2a_1 + 4a_2 + 8a_3 = 6}$$

Now solving the system :

1. $a_1 + a_2 + a_3 = 2$
2. $2a_1 + 4a_2 + 8a_3 = 6$

Divide the second equation by 2 :

$$\frac{2}{2} a_1 + \frac{4}{2} a_2 + \frac{8}{2} a_3 = \frac{6}{2}$$

$$1a_1 + 2a_2 + 4a_3 = 3$$

$$\boxed{a_1 + 2a_2 + 4a_3 = 3}$$

Now Substract the first equation from this

$$(a_1 + 2a_2 + 4a_3) - (a_1 + a_2 + a_3) = 3 - 2$$

$$\boxed{a_2 + 3a_3 = 1}$$

choosing reasonable values that satisfy the system, we get.

$$a_1 = \frac{3}{2}, \; a_2 = \frac{1}{4}, \; a_3 = \frac{1}{4}$$

Thus, the final equation for y_{P_1} is $y_{P_1} = 1 + \frac{3}{2}x + \frac{1}{4}x^2 + \frac{1}{4}x$

this polynomial perfectly fits the three training points.

Degree-2-Polynomials

$\left(y_{P_2} = 1 + \frac{5}{3}x + \frac{1}{3}x^3\right)$

Example	x	Target(y)
1	0	1
2	1	3
3	2	7

polynomial Equation $y = a_0 + a_1 x + a_2 x^2 + a_3 x^3$

$(x_1, y_1) = (0,1)$, $(x_2, y_2) = (1,3)$, $(x_3, y_3) = (2,7)$

$(x_1, y_1) = (0,1)$

$x = 0, y = 1$

$\boxed{a_0 = 1}$

$(x_2, y_2) = (1,3)$

$x = 1, y = 3$

$\boxed{a_1 + a_2 + a_3 = 2}$

$(x_3, y_3) = (2,7)$

$x = 2, y = 7$

$\boxed{2a_1 + 4a_2 + 8a_3 = 6}$

Solving for a_1, a_2, a_3

Now, solving the system

① $a_1 + a_2 + a_3 = 2$

② $2a_1 + 4a_2 + 8a_3 = 6$

Dividing equation 2 by 2 : $a_1 + 2a_2 + 4a_3 = 3$

Now subtract equation 1 from this.

$(a_1 + 2a_2 + 4a_3) - (a_1 + a_2 + a_3) = 3 - 2$

$\boxed{a_2 + 3a_3 = 1}$

choosing the value that satisfy the system.

$\boxed{a_1 = \frac{5}{3},\ a_2 = 0,\ a_3 = 1/3}$

Constructing the Final Equation :

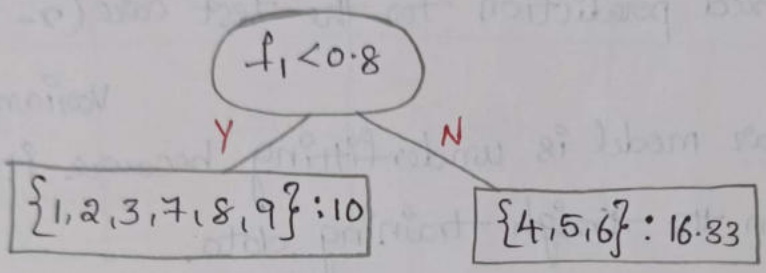
Substituting these values into the polynomial equation

$$y_{P_2} = 1 + \frac{5}{3}x + 0 \cdot x^2 + \frac{1}{3}x^3$$

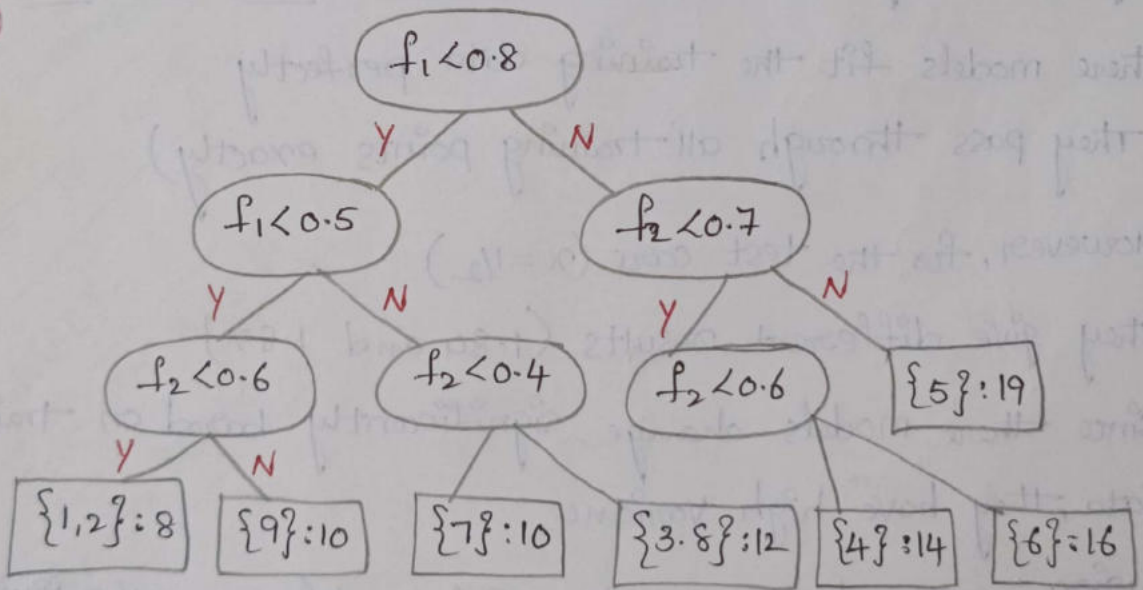
$$y_{P_2} = 1 + \frac{5}{3}x + \frac{1}{3}x^3$$

Decision trees for the regression dataset :

(a)



(b)



Understanding Underfitting and Overfitting from the table (Bias-Variance Trade-off)

* The table provides different models (linear and polynomial) and their predictions for different values of x. Let's analyze the result based on bias and variance.

3.8 Random Forests For classification And Regression :

1. Linear Model (Straight-Line fit) - column y_{s_1}

* this models follows a simple straight-line equation $y = mx+c$
* It does not fit the training data exactly (errors exist in the first three rows)
* this means it has high bias (it simplifies the data too much)
* It gives a fixed prediction for the test case ($x = 1/2$) low Variance.

Conclusion : the linear model is underfitting because it fails to capture patterns in the ~~traigh~~ training data.

2. Degree-3 polynomial Models - columns y_{P_1} and y_{P_2} :

* there models fit the training data perfectly (they pass through all training points exactly)
* However, for the test case ($x = 1/2$) they give different results (1.84 and 1.87)
* since there models change significantly based on training data, they have high variance.

Conclusion : the degree-3 polynomial models are overfitting because they capture noise instead of actual patterns.

Final Result for Bias-Variance Trade off :

* The linear model (y_{s_1}) under-fits because it is too simple
* The degree-3 polynomial model (y_{P_1}, y_{P_2}) over-fit because they memorize training data.
* The best balance would be a degree-2 polynomial (quadratic equation ($y = ax^2+bx+c$) which captures trends without excessive complexity.

Model comparison :

Model	Bias	Variance	Overfitting ?
y_{s1} Linear Model	High	Low	No
y_{p1} 1st polynomial	Medium	Medium	Slightly
y_{p2} 2nd polynomial	Low	High	Yes

26/03/25

F(AI)-B

her

Random Forests For classification And Regression :

Random Forests :

Random Forest (RF) is an ensemble learning technique used for both classification and regression tasks. It builds multiple Decision Trees (DTs) during training and merges their outputs to improve accuracy and reduce overfitting.

Features of Random Forest (RF) :

1. Bootstrapping (Bagging)
2. Random Feature Selection
3. Traing Multiple Decision Trees
4. Aggregating predictions.

1. Bootstrapping (Bagging) :

Bootstrapping means Sampling multiple sets of training data. In the case of RF, it works as follows.

Given n training data patterns in some L-diamensional space, m samples, each of size n data points, are selected with replacement, it is possible for the same data points to occur more than once in a sample.

* Multiple subsets of training data are randomly selected with replacement.

* Each Subset is used to train an individual Decision Tree

2. Random Feature selection :

* Instead of using all features, RF selects a random subset of features for each tree.
* This prevents trees from being overly similar and reduces overfiting.

3. Training Multiple Decision Trees :

* Each tree learns patterns from it's subset.
* Each DT gives a prediction independently.

4. Aggregating predictions :

* For classification : the class with the majority votes from all trees is choosen.
* For Regression : The final prediction is the average of all tree outputs.

Advantages of Random Forest :

1. Reduces overfitting
2. Handles High-Dimensional Data
3. Handles Missing Data
4. Robust to Noisy Data.

1. Reduces overfitting :

The averaging effect of multiple trees prevents overfitting.

2. Handles High-Dimensional Data :

Can work with datasets that have a large number of features

3. Handles Missing Data :

Can maintain accuracy even with some missing values.

4. Robust to Noisy Data : Since it aggregates results, noisy data hoes less impact.

Differences between classification and Regression in RF :

Feature	classification (RF)	Regression (RF)
output Type	categorical labels	continuous values
prediction Method	Majority voting	Averaging predictions
Loss Function	Entropy, Gini Index	Mean squared Errors (MSE) Mean Absolute Error (MAE)

Unit - III - 2nd Part

The Bayes classifier :

1. Introduction of Bayes classifier
2. Baye's Rule and Inference
3. the Baye's classifier and it's optimality
4. Multi class classification
5. class conditional Independence and Naive Bayes classifier (NBC)

Introduction of Bayes classifier :

A Bayes classifier is a simple but powerful classification algorithm that applies Bayes Theorem to predict the class label of a given instance. It is based on the assumption that the probability of a given instance. class given the input data can be completed using prior knowledge about the distribution of data.

Used applications : 1. Spam detection
2. Medical diagnosis
3. Sentiment analysis
4. Document classification

2. Bayes Rule and inference :

Example : let us assume that 10.000 people in a community had undergone the covid-19 text and 50 of them tested positive, while the remaining 9950 tested negative. In this two-calm (binary class) problem, a simple frequency estimate will give us the following values for the probabilities for the two classes.

Understanding prior probabilities :

In the example 10,000 people are tested, and the results are

* 50 tested positive ⟶ probability.

$$P(\text{positive}) = \frac{50}{10000} = 0.005$$

$$P(\text{Negative}) = \frac{9950}{10000} = 0.995$$

=

classification Based on prior probabilities :

If a new person from the community needs classification without any additional information.

* Since P(negative) = 0.995 is much greater than P(positive) = 0.005, the person will be classified as COVID-19 negative.
* This means that every new person would be labeled as negative because it's the most probable outcome.
* However, if the person is actually positive, this classification is incorrect with a probability of 0.005.

This method works well only when the prior probability of a class is significantly different, but it fails when dealing with rare events, like Covid-19 positive cases in this example.

Relation to k-Nearest Neighbors (KNN) classifier :

In KNN classification, a new sample is classified based on the majority class among it's k-nearest neighbors.

If k=n (all training data is used), then KNN simply estimates probabilities as : $P(\text{positive}) = \frac{k_1}{n}$, $P(\text{Negative}) = \frac{k_2}{n}$

This approach gives the same result as classification based on prior probabilities.

Additional information Such as : 1. Symptoms
2. Contact history

Probability Rule : $P(A) \in [0,1]$

Addition Rule for Disjoint Events :

If two events A and B cannot happen at the same time. (i.e $A \cap B = \phi$).

then $P(A \cup B) = P(A) + P(B)$.

Where $A \cup B$ is the union of the sets A and B. This property holds for a countable union of events if they are pairwise disjoint.

2. Baye's Rule and Inference :

Formula for Baye's theorem is

$$P(c|A) = \frac{P(A|c)\,P(c)}{P(A)}$$

$$P(c_1) = P(c_2) = 0.5$$

$$P(wB|c_1) = \frac{20}{30} = \frac{2}{3}$$

$$P(wB|c_2) = P(RB|c_2) = \frac{15}{30} = \frac{1}{2}$$

$$P(c_1|wB) = \frac{P(wB|c_1)\,P(c_1)}{P(wB)}$$

$$P(wB) = P(wB|c_1)\,P(c_1) + P(wB|c_2)\,P(c_2)$$

$$P(wB) = \left(\frac{2}{3} \times 0.5\right) + \left(\frac{1}{2} \times 0.5\right)$$

$$= \frac{1}{3} + \frac{1}{4} = \frac{4}{12} + \frac{3}{12} = \frac{7}{12}$$

applying Baye's Rule $= P(c_1|wB) = \frac{\frac{2}{3} \times 0.5}{\frac{7}{12}}$

$= 4/7$

Example 2 : Covid-19 Test Accuracy

We are given

* prevalence of covid-19

$$P(A) = 0.001$$

not having

* probability of covid-19 :

$$P(A^c) = 1 - 0.001$$

$$P(A^c) = 0.999$$

* True positive Rate (Sensitivity)

$$P(B^c/A^c) = 0.98$$

* False positive Rate $= P(B/A^c) = 1 - 0.98 = 0.02$

We want to find $P(A/B)$, the probability that a patient who tested positive actually has COVID-19.

By using Baye's theorem

$$P(A/B) = \frac{P(B/A)\ P(A)}{P(B)}$$

To compute $P(B)$, we use the law of total probability.

$$P(B) = P(B/A)\ P(A) + P(B/A^c)\ P(A^c)$$

$$P(B) = (0.9 \times 0.001) + (0.02 \times 0.999)$$

$$= 0.0009 + 0.01998$$

$$\boxed{P(B) = 0.02088}$$

Now applying Baye's Rule

$$P(A|B) = \frac{P(B/A)\ P(A)}{P(B)}$$

$$= \frac{(0.9 \times 0.001)}{0.02088}$$

$$= \frac{0.0009}{0.02088}$$

$$= 0.0431$$

$$\boxed{P(A/B) = 0.0431}$$

=

01/04/2025
CSE (AI) – A
To 11:15 Am

The Baye's classifier and it's optimality

The Baye's Rule classifier and it's optimality :

The Baye's Rule classifier is probabilistic model that assigns a test pattern z to a class based on the posterior probabilities of the classes given z. It is optimal because it minimizes the probability of classification error.

Baye's Theorem in classification :

From Baye's theorem

$$p(c_1|x) = \frac{p(x|c_1)\, p(c_1)}{p(x)}$$

$$p(c_2|x) = \frac{p(x|c_2)\, p(c_2)}{p(x)}$$

Where :

* $p(c_1|x)$ and $p(c_2|x)$ are posterior probabilities (probability of a class given the data)
* $p(c_1)$ and $p(c_2)$ are prior probabilities (probability of a class occurring before observing the data)
* $p(x|c_1)$ and $p(x|c_2)$

Department of Artificial Intelligence and Machine Learning

II B.Tech II Sem

Machine Learning

UNIT-4

Linear Discriminants for Machine Learning

By

Mrs.K.Reddemma,

Assistant Professor,

Department of AI&ML.

UNIT-4 – Linear Discriminants for Machine Learning

1. Linear Discriminants for Machine learning Introduction
2. Linear Discriminants for classification
3. perceptron classifier
4. perceptron learning Algorithm
5. Support Vector Machines
6. Linearly Non-separable case.
7. Non-linear SVM
8. Kernel Trick
9. Logistic Regression
10. Linear Regression
11. Multi-layer perceptrons (MLPs)
12. Back propagation for Training an MLP.

① Linear Discriminants Introduction

Linear Discriminants for classification are functions used to separate datapoints into different classes using a straight decision boundary. Ex: (like a line in 2D (or) hyperplane in higher dimensions). It works best when the data is linearly separable.

The goal is to find a function

$$g(x) = Wx + b$$

where x – Input feature vector
W – Weight vector
b – bias (threshold)

3. classification Rule

If $g(x) > 0 \longrightarrow$ class 1

If $g(x) < 0 \longrightarrow$ class 2

$g(x) = 0 \longrightarrow$ point lies on the decision boundary

purpose : To separate data points based on their features used in binary classification problems.

Applications of linear Discriminants

1. perceptron
2. Logistic Regression
3. support vector Machines (svm)
4. Linear Discriminant Analysis (LDA)

visualization : 1. In 2D : It's a line
2. In 3D : It's plane
3. In higher dimensions : It's a hyperplane.

Example problem : Find a linear discriminant function $g(x) = wx+b$ that can separate the two classes.

Two classes of points in 2D

class 1 (Label = +1) :

$x_1 = (2,3)$, $x_2 = (3,3)$

class_2 (Label = -1) :

$x_3 = (0,0)$

$x_4 = (1,1)$

Data points	label 1	label 2
x_1	2	3
x_2	3	3
x_3	0	1
x_4	1	1

Step-by-step Solution:

We want a line that separates the two classes. Assume a discriminant function.

$$g(x) = w_1 x_1 + w_2 x_2 + b$$

Let's try $w = (1,1)$, $b = -4$

So, $g(x) = x_1 + x_2 - 4$

For class 1 : $g(2,3) = 2+3-4 = 1 > 0 \rightarrow$ class +1

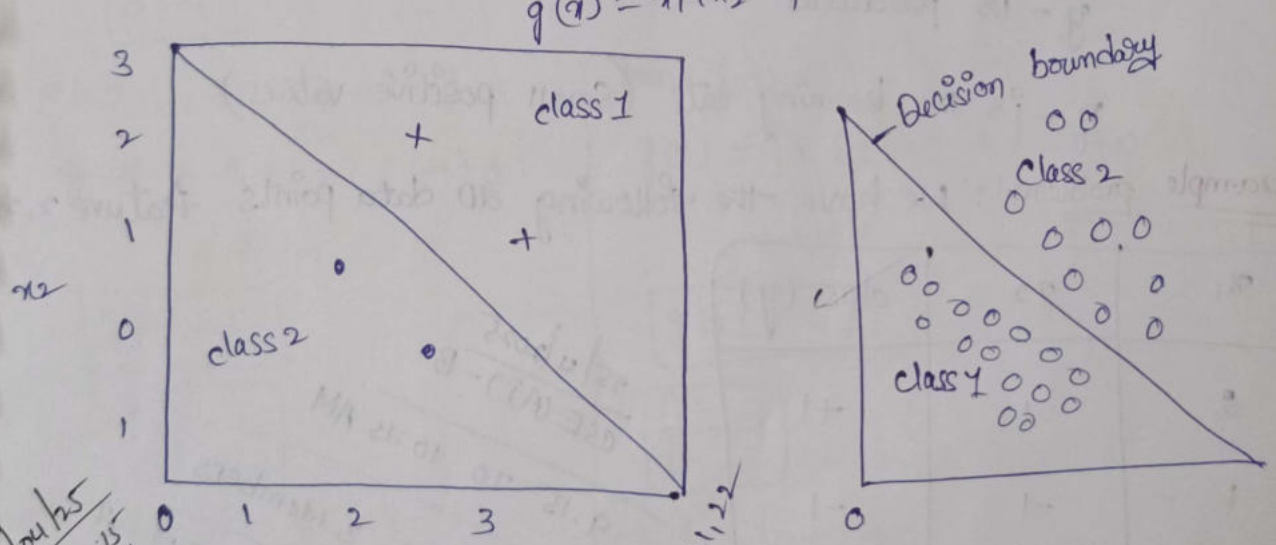
$g(3,3) = 3+3-4 = 2 > 0 \rightarrow$ class +1

For class 2 : $g(0,1) = 0+1-4 = -3 < 0 \rightarrow$ class -1

$g(1,1) = 1+1-4 = -2 < 0 \rightarrow$ class -1

A valid linear discriminant function is :

$$g(x) = x_1 + x_2 - 4$$



23/04/25
11:15 TO 12:15
CSE (AI) - B
64, 82, 185
3 members

23/4/2025
CSE(AI)-A
2:15 TO 3:15
1, 12, 13, 16, 18, 19, 20, 21, 22
23, 24, 25, 26, 28, 31
30 members

③ perceptron classifier

The perceptron is a linear classifier that tries to find a straight line to separate two classes of data.

Mathematical formula : $g(x) = wx + b$

if $g(x) \leq 0$: classify as class −1

steps for perceptron :

1. Initialize weights w and bias b
2. For each data point predict class using current weights, if prediction is wrong, update the weights.

$$w \leftarrow w + \eta (y - \hat{y}) x$$

Where :

y - is actual class

$\hat{y}$ - is predicted class

η - is the learning rate (small positive values)

Example problem : We have the following 20 data points feature x_1, x_2

x_1	x_2	class (y)
2	1	+1
1	−1	−1
2	3	+1
−1	−2	−1

25/4/2025
CSE (AI) - B
9:15 TO 10:15 AM
62, 93, 82, → 3 Members
11:15 TO 12:15 - CSE (AI) - A
2, 3, 12, 37, 1E5,

① Initial Setup :

start with $w=[0,0]$, $b=0$, learning rate $\eta=1$

② Training : we update weights only if the points is misclassified.

point $(2,1)$, $y=+1$

prediction → $g(x) = w_1 x_1 + w_2 x_2 + b$

$= 0*2 + 0*1 + 0$

$= 0$

$\hat{y} = -1$

The perceptron learning Algorithm is a simple supervised learning algorithm used for binary classification. It helps find a linear decision boundary that separates data into two classes

Ex : class +1 and class -1

$w = w + \eta y x$

$= [0,0] + 1*1*[2,1]$

$w = [2,1]$

$b = b + 1*1$

$b = 1$

point $(1,-1)$, $y=-1$

$g(x) = 2*1 + 1*(-1) + 1$

$= 2-1+1$

$g(x) = 2$

$\hat{y} = +1$

point $(2,3)$, $y=+1$

$g(x) = 1*2 + 2*3 + 0$

$= 2+6$

$= 8$

$\hat{y} = +1$,

point $(-1,-2)$, $y=-1$

$g(x) = 1*(-1) + 2*(-2) + 0$

$= -1-4$

$g(x) = -5$

$\hat{y} = -1$

Final output :

Final weights $w = [1,2]$

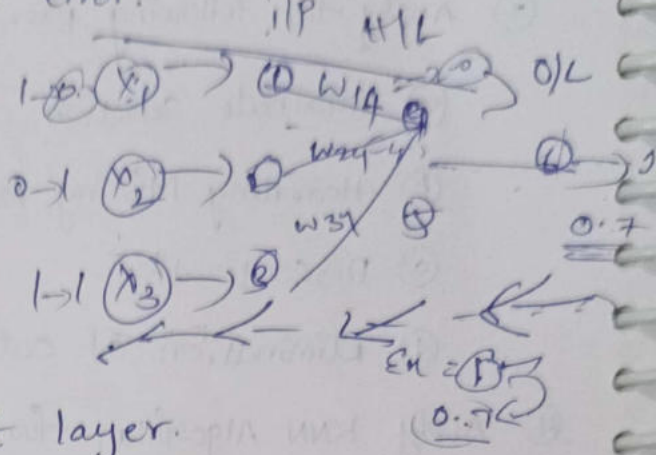
Final bias $b = 0$

Decision Boundary :

$x_1 + 2x_2 = 0$ (or) $x_2 = -0.5 x_1$

Backpropagation for Training an MLP (Multilayer perceptron):

Backpropagation is used to train a neural network by adjusting the weights to minimize the output error.



It works in two main steps:

1. Forward Pass:

* You start with random weights.
* You apply the input at the input layer.
* Each layer processes the inputs and passes the output to next layer
* At the end, you get the network's output y_obt (obtained output).

2 Backward pass (Backpropagation)

* Calculate the error: difference between y_tar and y_obt
* propagate the error backward through the network.
* Use gradient descent to update the weights to reduce the error.

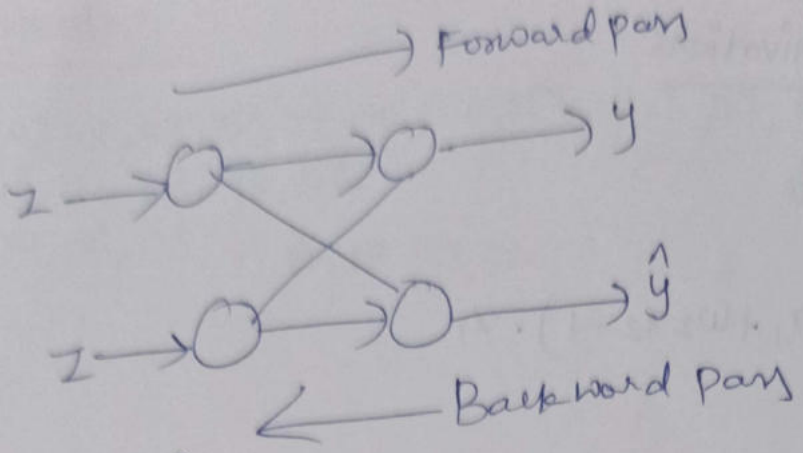
The basic weight update rule is:

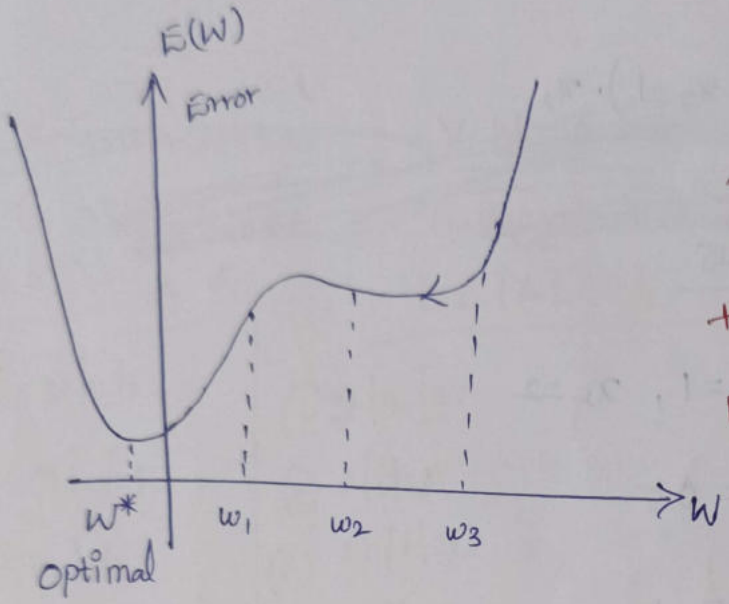
$$W_{updated} = W_{current} - \eta \times \nabla E(W_{current})$$

where:

* η = learning rate (how big a step you take)
* $\nabla E(W_{current})$ = gradient of the error with respect to weights.

Gradient descent tries to move the weights towards a local minimum where the error is small.





problem :

let $w_{current} = (0.5, 0)$ and the pattern $x = (1, 2)$. let $E = (w_1 x_1 + w_2 x_2 - 1)^2$.

Here $\nabla E(w_{current})$ is obtained as follows :

Gradient descent for minimizing error

Given current weights

$w_{current} = (w(1), w(2)) = (0.5, 0)$

$x = (x_1, x_2) = (1, 2)$

$E = (w_1 x_1 + w_2 \cdot x_2 - 1)^2$

square error loss $y = w_1 x_1 + w_2 x_2$

with target output = 1

$$\frac{\partial E}{\partial w_1} = 2(w_1 x_1 + w_2 x_2 - 1) x_1$$

$$\frac{\partial E}{\partial w_1} = 2(w_1 x_1 + w_2 x_2) x_2$$

Goal : compute the gradient of the error with respect to each weight.

$$\nabla E(w) = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2} \right)$$

Step-by-step Derivations:

① Apply chain Rule

$$\frac{\partial E}{\partial w_1} = 2(w_1 x_1 + w_2 x_2 - 1) \cdot x_1$$

$$\frac{\partial E}{\partial w_2} = 2(w_1 x_1 + w_2 x_2 - 1) \cdot x_2$$

① plug in the given values

$w_1 = 0.5$, $w_2 = 0$, $x_1 = 1$, $x_2 = 2$

$y = 0.5 \times 1 + 0 \times 2 = 0.5$

Error term : $y - 1 = 0.5 - 1$

$$\boxed{y = -0.5}$$

Now compute gradients

$$\frac{\partial E}{\partial w_1} = 2(-0.5)(1)$$

$$= -1$$

$$\boxed{\frac{\partial E}{\partial w_1} = -1}$$

$$\frac{\partial E}{\partial w_2} = +2(-0.5)2.$$

$$= -2$$

$$\boxed{\frac{\partial E}{\partial w_2} = -2}$$

Gradient vector :

$$\nabla E(w_{current}) = (-1, -2)$$

CSE(AI)-A
29/4/2025

Kernel Trick :

Kernal Trick an important concept in machine learning especially in Support Vector Machines.

1. Data is not linearly separable in it's original space (input)

2. We might want to map data to a higher-dimensional space to make it separable:

3. But computing this mapping explicitly can be hard (or) inefficient.

Instead of mapping vectors explicitly using a function $\phi(z)$, we compute the dot product in the feature space directly using a kernel function.

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

This functions return the dot product in higher-dimensional space without doing the transformation.

: polynomial Kernel (degree p) :

$$K(x_i, x_j) = (x_i^T x_j)^p$$

$$K(x_i, x_j) = (1 + x_i^T x_j)^p$$

$$K(x_i, x_j) = e^{-\|x_i - x_j\|^2}$$

$$K(x_i, x_j) = \tanh(\alpha x_i^T x_j + b)$$

polynomial Kernel of Degree 2

To show how the kernel trick works, the book shows.

$$K(x_i, x_j) = (x_i^T x_j)^2$$

$$\phi(x) = \begin{bmatrix} x(1)^2 \\ x(2)^2 \\ \sqrt{2}\, x(1)\, x(2) \end{bmatrix}$$

Then

$$K(x, x) = \phi(x)^T \phi(x)$$

$$= x(1)^4 + x(2)^4 + 2x(1)^2 x(2)^2$$

$$= (x(1)^2 + x(2)^2)^2$$

This matches $(x^T x)^2$, so the kernel function gives the same result as explicitly computing in higher-dimensional space.

Multi-layer perceptrons (MLPs)

MLPs are a class of feed-forward artificial neural networks that consist of multiple layers of neurons.

Structure :

Input Layer : Accepts input

Department of Artificial Intelligence and Machine Learning

II B.Tech II Sem

Machine Learning

UNIT-5

Clustering

By

Mrs.K.Reddemma,

Assistant Professor,

Department of AI&ML.

UNIT-5 chapter-7

clustering

Raw Data → Algorithm → Apples, oranges, strawbe (output array)

Introduction to clustering :

clustering is a unsupervised learning technique used to group similar data points together based on their features. It is commonly used for pattern recognition, customer segmentation, anomaly detection, and data compression.

clustering refers to the process of arranging (or) organizing objects according to specific criteria. clustering involves dividing or grouping the data into smaller data sets based on similarities / dissimilarities. Depending on the requirements, this grouping can lead to various outcomes, such as particularing of data, data re-organization compression of data and data summarization.

1. Market Segmentation
2. statistical data analysis
3. Social network analysis
4. Image segmentation
5. Anamoly detection

Ex : Shopping malls, Super markets — provigence

Kids Wear, Ladies Feamale Wear, Gents Wear, Fruits, vegitables

partitioning of Data : (clustering Method)

partitioning of data in ML refers to the process of dividing a dataset into smaller subsets for efficient processing, training and evaluation.

This helps improve model performance, reduce overfitting, and optimize computational efficiency.

Ex : Employee_Table

1. there is an Employee data table containing 30,000 records.
2. the table includes a column Dept_code, which has 1,000 records distinct values.
3. the records are evenly distributed among there 1,000 departments.

So each department has $\frac{30,000 \text{ Records}}{1,000 \text{ departments}}$

= 30 records per department

$30 \times \frac{30,000}{2} = 450,000$ accesses

(Without clustering)

With clustering :

Dept_code = '15'

1000 departments the search uses binary search $\log_2(1000)$ accesses.

Approximate value = $\log_2(1000) \approx 10$ accesses.

* Without clustering = 45,000 accesses
* With clustering = $\log_2(1000) + 30$

02/04/2025

CSE(AI) - A

2:15 pm TO 3:15 pm

(3) Matrix Factorization :

Matrix Factorization is a technique that can be used to represent clustering mathematically. It allows us to express a dataset as the product of two matrices. $X \approx BC$

Where :

* X - is the original data matrix
* B - is the cluster assignment matrix (Indicates which data points belong to which clusters)
* C - is the representative matrix (contains the cluster centroids or leaders)

This approach helps in understanding clustering as a factorization problem, where we break down a large data matrix into meaningful components.

Example : Consider the dataset $X = \begin{bmatrix} 6 & 6 & 6 \\ 6 & 6 & 8 \\ 2 & 4 & 2 \\ 2 & 2 & 2 \end{bmatrix}$ → cluster 1

Here, there are 4 data points in a 3-dimensional space, if we use the leader Algorithm with a threshold of 3-units, we obtain two clusters.

cluster 1 : (6,6,6) and (6,6,8) with (6,6,6) as the leader.

cluster 2 : (2,4,2) and (2,2,2) with (2,4,2) as the leader.

Thus, the cluster Assignment Matrix B is :

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

Each row in B represents a data point, and the 1's indicates the assigned cluster. The cluster Representative Matrix C (leader of cluster) is :

$$C = \begin{bmatrix} 6 & 6 & 6 \\ 2 & 4 & 2 \end{bmatrix}$$

Thus, the matrix factorization : $X \approx B \times C$

Hard clustering (Vs) Soft clustering :

Hard clustering : (Leader Algorithm)

Each data point belongs to only one cluster (values in B are either 0 or 1)

Soft clustering : (Fuzzy clustering)

A data point can belong to multiple clusters with different probabilities, (values in B are between 0 and 1, and each row sums to 1)

Example of Soft clustering :

$$B = \begin{bmatrix} 0.8 & 0.2 \\ 0.7 & 0.3 \\ 0.1 & 0.9 \\ 0.2 & 0.8 \end{bmatrix}$$

Using centroids Instead of leaders

Instead of using a single leader for a cluster we can compute the centroid (mean) of points in each cluster. The updated matrix is

Here

* (6,6,7) is the centroid of cluster 1 (mean of (6,6,6) and (6,6,8)).
* (2,3,2) is the centroid of cluster 2 (mean of (2,4,2) and (2,2,2)).

clustering of patterns : 1. Intra-cluster distance
2. Inter-cluster distance

1. Intra-cluster distance :

Intra-cluster distance should be small, (distance within a cluster), meaning data points in a cluster are close together.

2. Inter-cluster distance :

Inter-cluster distance should be large, ensuring well-separated clusters. (distance blw clusters)

This ensure that the clustering structure is meaningful and useful for applications like image processing, customer segmentation and recommendation system.

04/04/2025
CSE (AI) - B
9:15 AM

04/04/2025

④ Clustering of patterns :

Clustering is an unsupervised learning technique used to group similar patterns (or data points) based on the certain features. It helps discover underlying structure in data without pre-defined labels.

* patterns - Data points characterized by a set of features
* similarity - Measured using distance metrics (Ex - Euclidean, Manhattan, Cosine)
* clusters - Groups of similar patterns.

Break down :

1. Matrix c with Hard clustering (Leader Algorithm)

The representative value of are directly picked from existing data points.

Given that $c = (6,6,6)$ and $(2,4,2)$ this means one cluster is represented by $(6,6,6)$ and the other by $(2,4,2)$

2. Matrix C with Centroid-Based clustering :

Instead of choosing a specific data points; the representative of a cluster is computed or the centroid (mean of points in the cluster)

The Centroid is calculated as follows :

⇒ First cluster : $\{(6,6,6), (6,6,8)\}$

Centroid $\left(\frac{6+6}{2}, \frac{6+6}{2}, \frac{6+8}{2}\right) \Rightarrow \left(\frac{12}{2}, \frac{12}{2}, \frac{14}{2}\right) = (6,6,7)$

⇒ Second cluster : $\{(2,4,2), (2,2,2)\}$

Centroid $\left(\frac{2+2}{2}, \frac{4+2}{2}, \frac{2+2}{2}\right) = \left(\frac{4}{2}, \frac{6}{2}, \frac{4}{2}\right) = (2,3,2)$

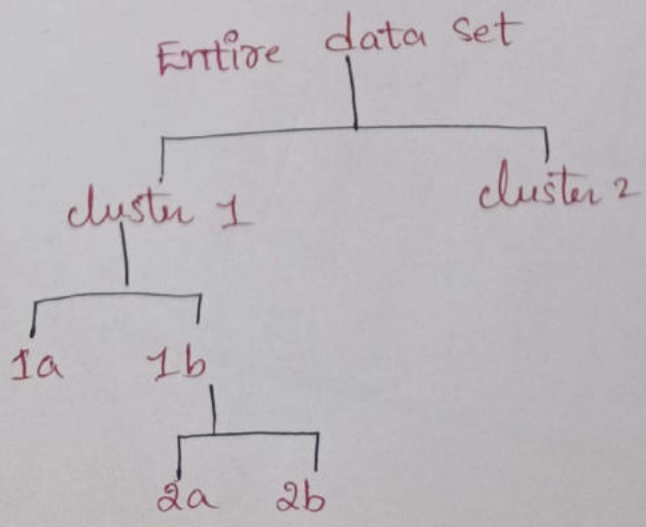
=

Divisive clustering in Machine Learning:

Divisive clustering is a top-down hierarchical clustering approach. It starts with all data points in one large cluster and then recursively splits the clusters into smaller ones, until each point is in its own cluster or a stopping condition is met.

How Divisive Clustering Works:

1. Start with all data in one cluster.
2. At each step:
   * Choose the cluster to split (often the one with the highest dissimilarity or largest size).
   * Split it into two clusters based on a certain criterion (eg.. distance, variance)
3. Continue Splitting until:
   * A predefined number of ~~cluster~~ clusters is reached.
   * A threshold distance or dissimilarity is met.



Step 1 : Top-Down approach

Start with all in one cluster → {A, B, C, D}

Step 2 : split into 2 groups (or) clusters

Based on distance, a good split is

{A, B} and {C, D}

A = 5, B = 2, C = 4, D = 2

A - B = 5 - 2 = 3

C - D = 4 - 2 = 2

Example: Imagine you have 10 data points

① Initially

cluster - 1 = $\{x_1, x_2, x_3 \dots x_{10}\}$

② After 1st split :

cluster_1 = $\{x_1\ x_2\ x_3\ x_4\ x_5\}$

cluster_2 = $\{x_6\ x_7\ x_8\ x_9\ x_{10}\}$

③ After next split :

cluster_1a = $\{x_1, x_2\}$

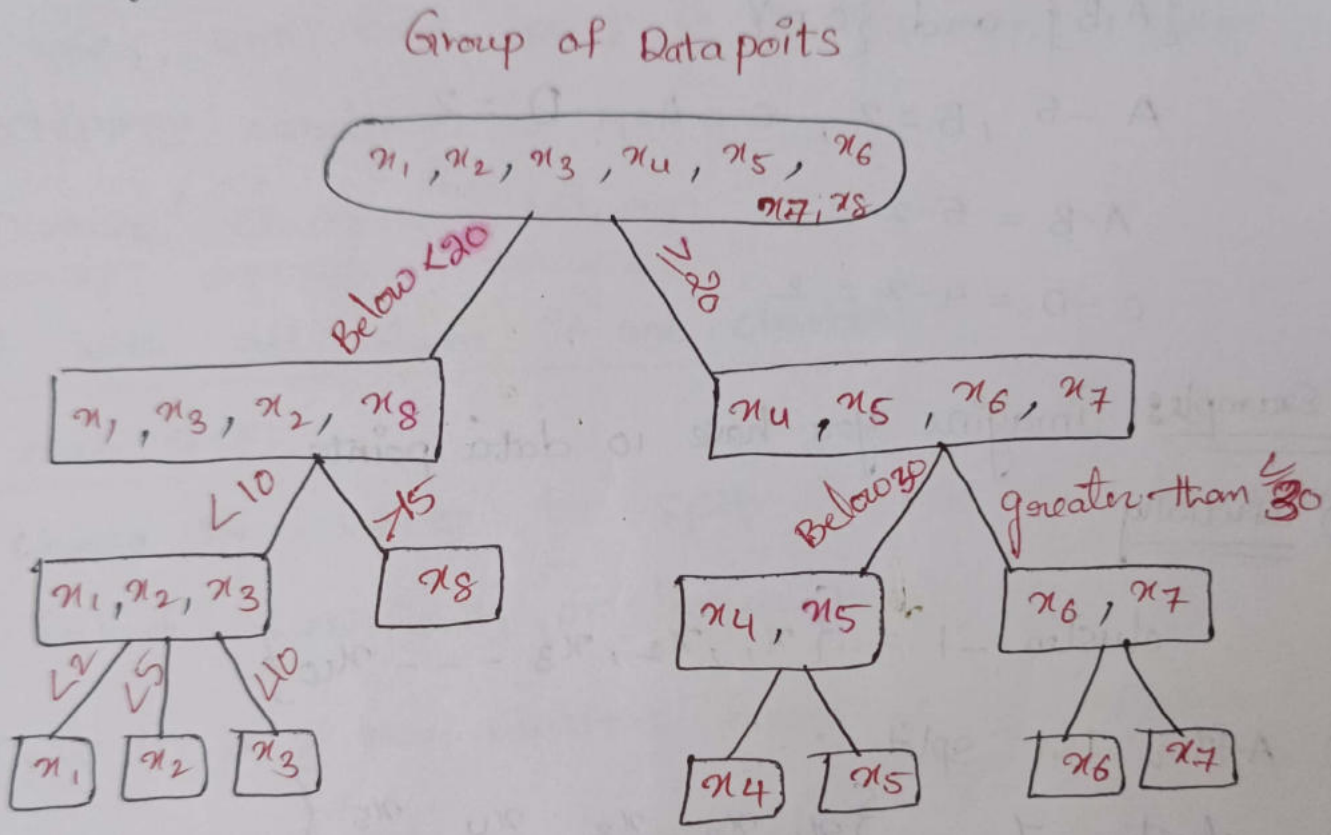
cluster_1b = $\{x_3, x_4, x_5\}$

Advantages of Divisive clustering

1. May produce better results if the global structure is more important than local patterns.
2. Good when a natural top-down structure exists in the data

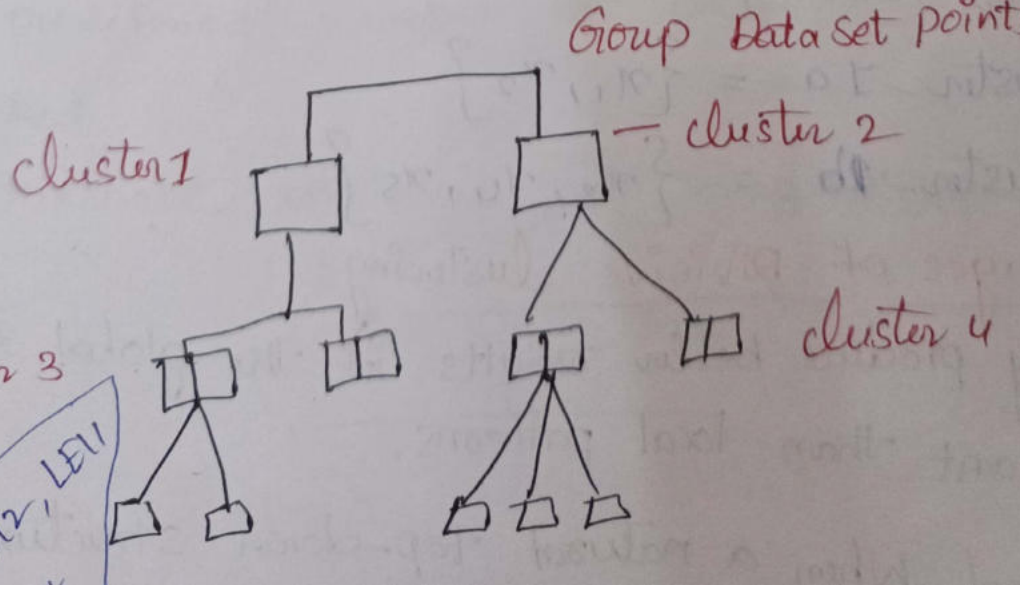
DisAdvantages:

1. computationally expensive (needs to evaluate many possible splits)
2. less commonly used in practice compared to agglomerative clustering.



$x_1 = 1$, $x_2 = 2$, $x_3 = 5$, $x_4 = 20$, $x_5 = 25$, $x_6 = 30$, $x_7 = 35$

$x_8 = 19$



Agglomerative clustering :

Agglomerative clustering is a bottom-up hierarchical clustering method. It is one of the most commonly used hierarchical clustering techniques.

Steps for Agglomerative clustering :

step 1 1. Start with each data point as it's own cluster.

step 2 2. Compute distances (or) similarity between all pairs of clusters.

step 3 3. Merge the two closest clusters.

step 4 4. Repeat steps 2 and 3 until :

* All points are in one single cluster (or)
* A desired number of clusters is reached.

Example :

Step 1 : {A}, {B}, {C}, {D}

Step 2 : Distances

A = 1, B = 2, C = 8, D = 10

$|A-B| = |2-1| = 1$

$|B-C| = |2-8| = 6$

$|C-D| = |8-10| = 2$

$A-C = |1-8| = 7$

step 3 : Merge A and B $\longrightarrow$ {A, B}, {C}

step 4 : Next closest is {A, B} And {C}

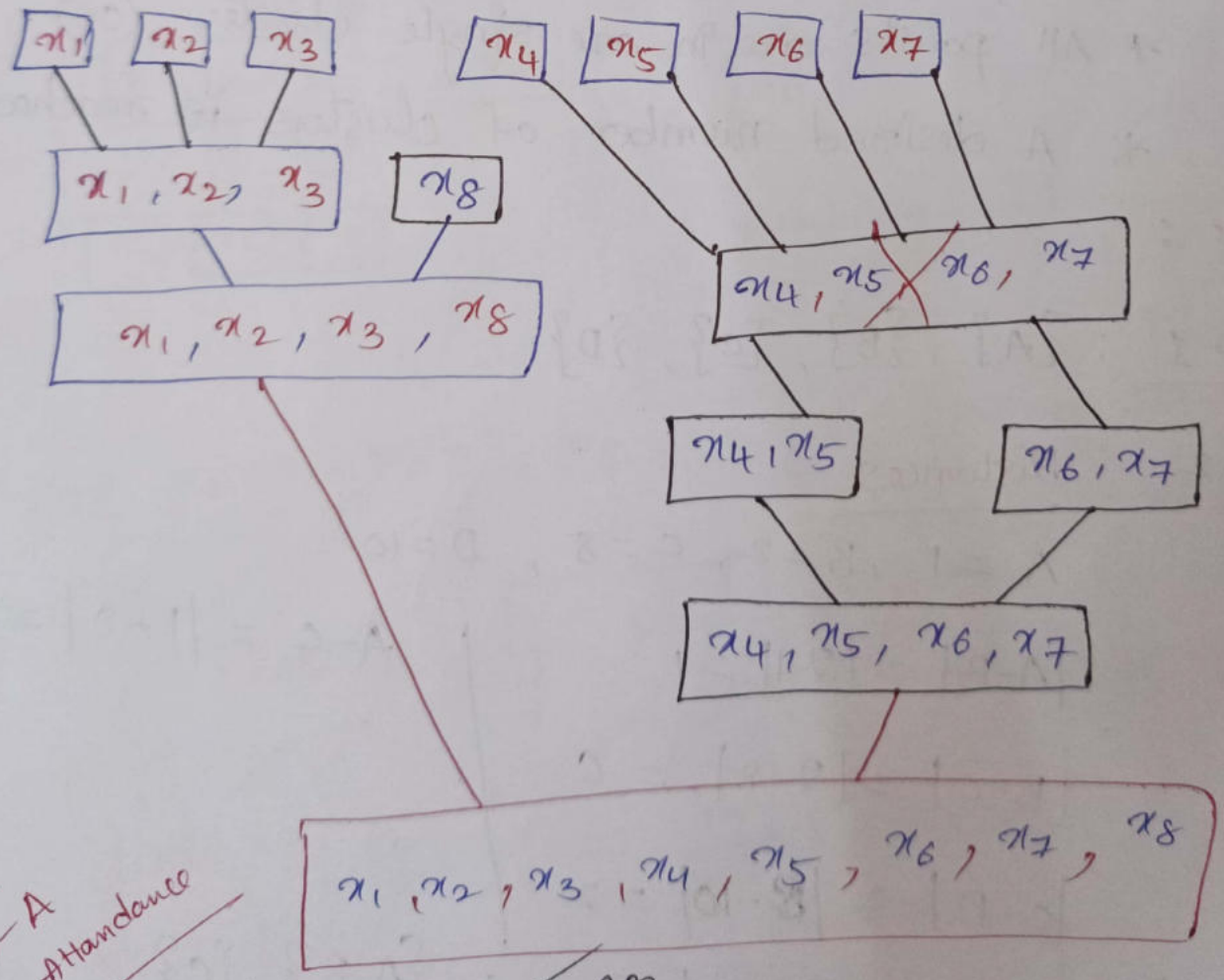
merge $\longrightarrow$ {A, B, C}

Advantages :

1. Works well for small datasets
2. Doesn't require the number of clusters in advance (But you can set one if desired)

DisAdvantages :

1. Computationally expensive for large data sets ($O(n^2 \log n)$)
2. Sensitive to noise and outliers
3. once merged, clusters can't be split (no BackTracking)



8/4/25
CSE(AI)-A
morning Attendance

Partitional clustering :

Data points	f_1	f_2
x_1	1	1
x_2	1	3
x_3	3	3
x_4	7	2
x_5	8	2
x_6	9	3
x_7	7	9
x_8	8	9

(C1: 1, 1 · C2: 7, 2 · C3: 7, 9)

Example of data patterns with two features

Euclidean distance formula

$$\sqrt{(x_1-y_1)^2 + (x_2-y_2)^2}$$

where

(x_1, x_2) is the data point

(y_1, y_2) is the cluster center

Initial clusters :

$c_1 - (1, 1)$

$c_2 - (7, 2)$

$c_3 - (7, 9)$.

Example : ① $x_1 = (1, 1)$ to $c_1 = (1, 1)$

Distance from $\sqrt{(1-1)^2 + (1-1)^2} = \sqrt{0+0} = 0$

② Distance from $x_2 = (1, 3)$ to $c_2 = (7, 2)$

$(1-7)^2 + (3-2)^2 = \sqrt{36+1} = \sqrt{37}$

③ Distance from $x_3 = (3, 3)$ to $c_3 = (7, 9)$

$\sqrt{(3-7)^2 + (3-9)^2} = \sqrt{16+36} = \sqrt{52}$

Euclidean distances with cluster assignments.

Data points	Euclidean distance from c_1	c_2	c_3	Assigned cluster
x_1	0	$\sqrt{37}$	10	c_1
x_2	2	$\sqrt{37}$	$\sqrt{72}$	c_1
x_3	$\sqrt{8}$	$\sqrt{17}$	$\sqrt{52}$	c_1
x_4	$\sqrt{37}$	0	7	c_2
x_5	$\sqrt{50}$	1	$\sqrt{50}$	c_2
x_6	$\sqrt{68}$	$\sqrt{5}$	$\sqrt{40}$	c_2
x_7	10	7	0	c_3
x_8	$\sqrt{113}$	[illegible]	[illegible]	[illegible]

Scanned with OKEN Scanner

Euclidean distance Formula

$$d = \sqrt{(f_{1x} - f_{1c})^2 + (f_{2x} - f_{2c})^2}$$

$x_1 = (1,1)$ TO C_1

$$\sqrt{(1-1.67)^2 + (1-2.33)^2}$$

$$= \sqrt{0.4489 + 1.7689}$$

$$= \sqrt{2.2178}$$

$$\approx 1.49$$

TO $-C_2$

$$\sqrt{(1-8)^2 + (1-2.33)^2} = \sqrt{49 + 1.7689}$$

$$= \sqrt{50.7689}$$

$$\approx 7.13$$

TO $-C_3$

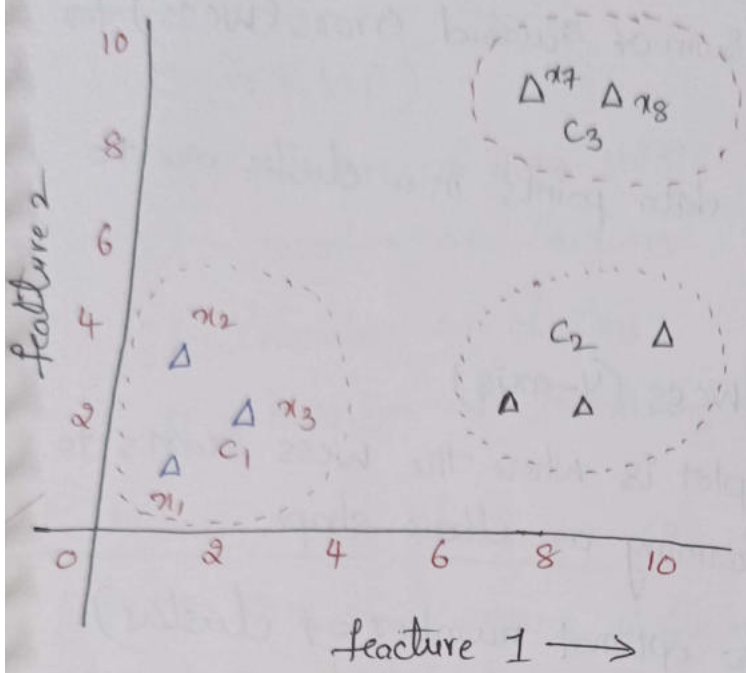
$$\sqrt{(1-7.5)^2 + (1-9)^2} = \sqrt{42.25 + 64} = \sqrt{106.25}$$

$$\approx 10.3$$

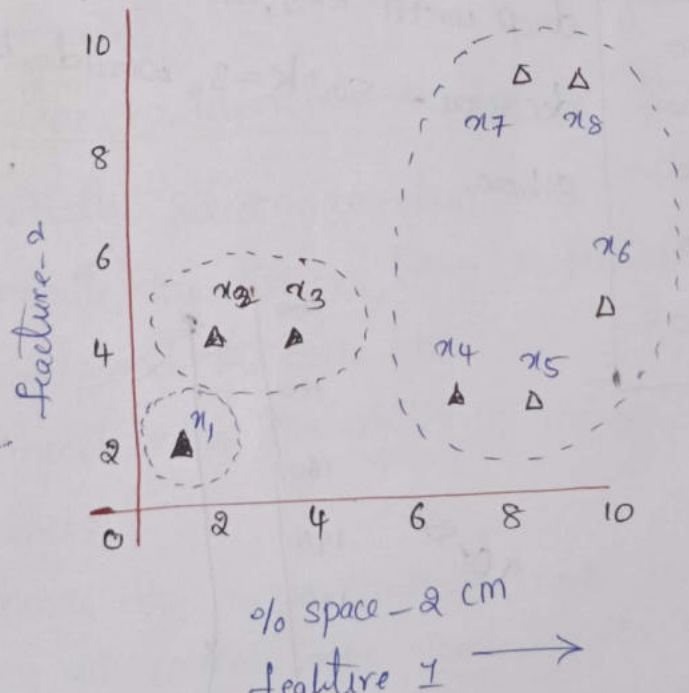
Assigned to C_1 (shortest distance)

Data points	C_1	C_2	C_3	Assigned cluster
x_1	1.49	7.13	10.3	C_1
x_2	0.95	7	8.85	C_1
x_3	1.49	5.24	7.5	C_1
x_4	5.34	1.05	7.02	C_2
x_5	6.34	0.33	7.02	C_2
x_6	7.36	1.2	6.18	C_2
x_7	8.54	6.74	0.5	C_3
x_8	9.2	6.67	0.5	C_3

K-means clustering (with k=3) for the data points listed in above Table :



K-means clustering (with k=3) for the data points with initial cluster centre ; x_1 x_2 and x_3 .



Elbow Method :

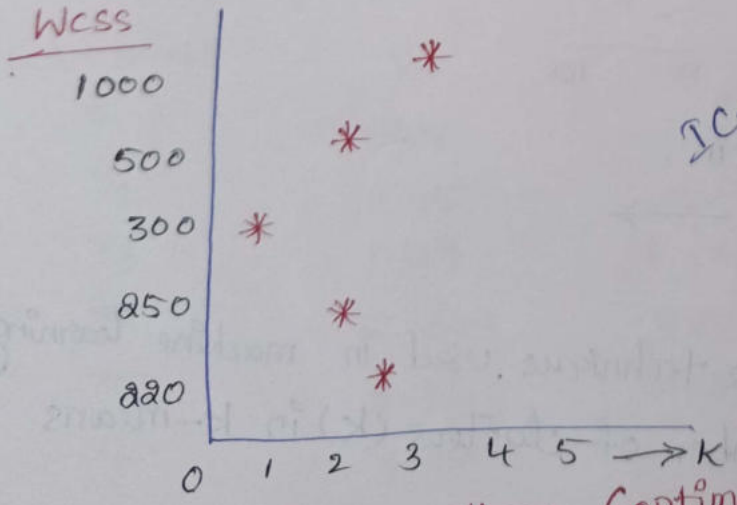
The Elbow method is a popular technique used in machine learning to determine the optimal number of clusters (k) in k-means clustering.

1. Run k-means clustering on your data for different values of k. (Ex from 1 to 10)
2. calculate the within-cluster sum of squared errors (WCSS) for each k.
   (WCSS measures how close data points in a cluster are to their centroids.)
3. plot k (X-axis) Vs WCSS (Y-axis)
4. The "elbow point" in the plot is where the WCSS starts to decrease more slowly forming an elbow shape.
   (This point suggests the optimal number of clusters)

Example :

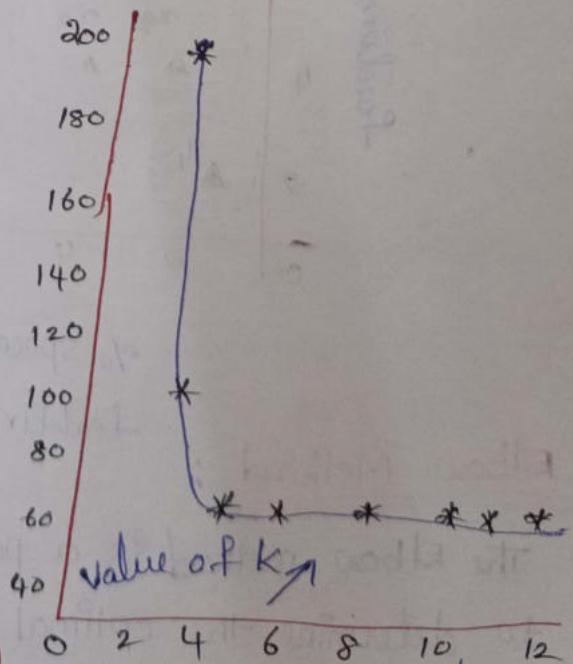
k	WCSS
1	1000
2	500
3	300
4	250
5	220

plotting this would show a sharp drop until k=3, and then a slow decrease — so k=3 would be the elbow.



Elbow (optimal k)

Elbow method for Elbow the k-value.



IC-SSE

Elbow method for approximately k

K-Means ++ clustering :

Time & space complexity

$O(n \cdot l \cdot k \cdot p)$

n = number of data points
l = number of features
k = number of clusters
p = number of iterations

space complexity

$O(kn)$

K-Means ++ clustering :

In the case of k-means clustering, we randomly select the initial k-cluster centres (centroids). The k-means++ clustering algorithm is mainly used for identifying the initial cluster centroids centres. The idea here is to good representatives of the clusters formed.

The steps to identify the initial k-centroids are as follows

1. Initially select a random data point's as the first centroid
2. Compute the distance (say Euclidean distance) between any centroid and the data point.
3. Compute the probabilities among the computed minimum distances.
4. choose the next centroid with maximum probability among all probabilities computed in step3.
5. Repeat steps 2 to 4 till we get k-number of centroids.

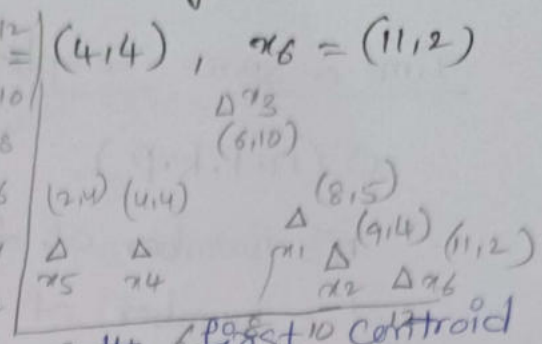
A-Lab
U2, U3, 50

Example : Consider the data set with following data points

$x_1 = (8,15)$, $x_3 = (6,10)$, $x_4 = (4,4)$, $x_6 = (11,2)$

First Randomly chosen centroid

$x_2 = (9,4)$, $x_5 = (2,4)$



Visualization of the data set

Distance from x_1 :

We calculate the Eudidean distance from the first centroid $x_1 = (8,15)$ to all other points.

Formula : $d(x_1, x_2) = \sqrt{(x_1 - x_i)^2 + (y_1 - y_i)^2}$

step-by-step :

Data point	coordinates	Distance from x_1	Distance (approx)	Distance^2
x_1	—	—	—	—
x_2	(9,4)	$\sqrt{(8-9)^2 + (15-4)^2}$ $\sqrt{(1)^2 + (-1)^2}$	1.41 11.05	2 122
x_3	(6,10)	$\sqrt{(8-6)^2 + (15-10)^2}$ $\sqrt{(-2)^2 + 5^2}$	5.39	29
x_4	(4,4)	$\sqrt{(8-4)^2 + (15-4)^2}$ $\sqrt{(-4)^2 + (-11)^2}$	4.12 11.70	137
x_5	(2,4)	$\sqrt{(8-2)^2 + (15-4)^2}$ $\sqrt{(-6)^2 + (-11)^2}$	6.08 12.53	37 157
x_6	(11,2)	$\sqrt{(8-11)^2 + (15-2)^2}$ $\sqrt{3^2 + (-13)^2}$	4.24 13.34	18 178

Sum of distance square $= 122 + 29 + 137 + 37 + 18 = 103$ $= 623$

Now compute probability for each point :

$$\text{probability}(i) = \frac{\text{distance}_i^2}{\text{Total Sum of distance}^2}$$

Data point	Distance^2	probability	
x_2	122	$\frac{2}{103} \approx 0.019$	$\frac{122}{623} = 0.195$
x_3	29	$\frac{29}{103} \approx 0.282$	
x_4	137	$\frac{17}{103} \approx 0.165$	$\frac{137}{623} = 0.219$
x_5	157	$\frac{37}{103} \approx 0.359$	$\frac{157}{623} = 0.25$
x_6	178	$\frac{18}{103} \approx 0.175$	$\frac{178}{623} = 0.28$

Example : $\frac{x_5 \text{ Distance (approx)}}{\text{Total Sum of Distance (approx)}}$

$$= \frac{6.08\ (12.53)}{1.41+5.39+4.12+6.08+4.24} = \frac{6.08}{21.24} = \frac{12.53}{623} = 0.020$$

$$\approx 0.286$$

After Selecting x_5 as Second Centroid :

Now you have two Centroids:

$x_1 = (8,5)$

$x_5 = (2,4)$

Now for each remaining data point, you:

1. Compute distances from both centroids.
2. For each point, keep the minimum distance to any centroid.
3. Use those minimum distances (or their squares) to compute probabilities.

Let's do this:

Data Point	$d(x_1, x_i)$	$d(x_5, x_i)$	Min. Distance	Probability (∝ Min dist)
x_2	1.41	7.0	1.41	$\frac{1.41}{1.41+5.39+2.0+4.24} \approx 0.09$
x_3	5.39	7.2	5.39	$\frac{5.39}{15.16} \approx 0.36$
x_4	4.12	2.0	2.0	$\frac{2.0}{15.16} \approx 0.13$ (but table says 0.27, maybe typo or miscal c
x_6	4.24	9.22	4.24	$\frac{4.24}{15.16} \approx 0.28$

Highest probability = x_3 → Chosen as third centroid

Key point:

* K-means++ usually uses squared distances for probability.

* But your example uses raw distances, so the probabilities are calculated accordingly.

15/04/2025

Soft partitioning :

In Machine learning soft partitioning refers to assigning data points to multiple clusters (or) categories with degree of membership rather than assigning each point to just one cluster (as in hard partitioning)

① Multiple membership approach :

This approach allows data points to belong to multiple clusters. rather than just one (as in traditional hard clustering like K-means)

Methods : 1. Fuzzy clustering
2. Rough clustering
3. Neural network-based clustering

② Stochastic Approach : 1. simulated annealing
2. Evolutionary Algorithms

1. Fuzzy clustering : Each data point is assigned to Multiple clusters, typically more than one based on a membership value. The value is computed using the data point and the corresponding cluster centroid.

2. Rough clustering : Each cluster is assumed to have both a non-overlapping part and an overlaping part. Data points in the non-overlaping portion belong exclusively to that cluster, while data points in the overlaping part may belong to multiple clusters.

3. Neural Network-based clustering : In this method, varying weights associated with the data points are based to obtain a soft partition.

16/04/25
CSE(AI)-A
2.15 PM TO 3.15 PM
02, 10, 15, 22, 24, 28
31, 34, 39, 41, 42
45, 48, 56, 62, 67

② Stochastic Approach :

In this category, the focus is on achieving the global optimal solution for cluster formation through probabilistic methods. some existing methods for this approach include.

1. Simulated annealing
2. Tabu Search
3. Evolutionary algorithms.

1. Simulted Annealing : In this case, the current solution is randomly updated, and the resulting solution is accepted with a certain probability. If the resulting solution is better than the current formation, it is accepted; otherwise, it is accepted with a probability ranging from 0 to 1.

2. Tabu Search : unlike simulated annealing, multiple solutions are stored, and the current solution is modified in various ways to determine the next configuration.

3. Evolutionary Algorithms:

This method maintains population of solutions. In Addition to the fitness values of individuals, a random search based on the interaction among solutions which mutation is employed to generate the next population.

Fuzzy C-Means clustering

Fuzzy c-means clustering (FCM) is an advanced version of k-means clustering used in ML, especially when clusters overlap and a data points can belong to more than one cluster.

Steps for Fuzzy c-Means clustering :

1. **Initialize** : choose number of clusters c.
Randomly initialize the membership matrix (values indicating how much a point belongs to each cluster)

2. **Update cluster centers** :
Each cluster center is calculated as a weighted average of all points, weighted by their membership values.

3. **Update Membership values** :
Membership of a point a cluster depends on it's distance to the cluster center. closer = higher membership.

4. Repeat steps 2 and 3 until convergence (small changes in values)

Mathematical formulation :

The objective function to minimize is

$$J = \sum_{i=1}^{n} \sum_{j=1}^{c} u_{ij}^{m} \cdot \|x_i - c_j\|^2$$

Where : u_{ij} – Membership of data point x_i in cluster j

m – Fuzziness coefficient (usually 2)

c_j – cluster center of cluster j

$|x_i - c_j|$ – Distance between point and cluster center.

Example : Consider the data points with two features.

Example of a data set with two features.

Data points	f_1	f_2
x_1	1	1
x_2	2	2
x_3	4	3
x_4	5	3

Let the number of clusters $c=2$ and $M=2$. To start with assume the following membership values between the data point and the clusters $(\mu^{(0)})$

	x_1	x_2	x_3	x_4
c_1	1	1	0	0
c_2	0	0	1	1

The cluster centres are calculated as follows

$$c_1 f_1 = \frac{\mu_{11}^2 * x_1(f_1) + \mu_{12}^2 * x_2(f_1) + \mu_{13}^3 * x_3(f_1) + \mu_{14}^4 * x_4(f_1)}{\mu_{11}^2 + \mu_{12}^2 + \mu_{13}^2 + \mu_{14}^2}$$

$$c_1(f_1) = \frac{1^2 * 1 + 1^2 * 2 + 0^2 * 4 + 0^2 * 5}{1^2 + 1^2 + 0^2 + 0^2}$$

$$c_1(f_1) = \frac{3}{2} = 1.5$$

$$\boxed{c_1(f_1) = 1.5}$$

$$c_1(f_2) = \frac{u_{11}^2 * x_1 f_2 + u_{12}^2 * x_2 f_2 + u_{13}^2 * x_3 f_2 + u_{14}^2 * x_4(f_2)}{u_{11}^2 + u_{12}^2 + u_{13}^2 + u_{14}^2}$$

$$c_1(f_2) = \frac{1^2 * 1 + 1^2 * 2 + 0^2 + 3 * 0^2 * 3}{1^2 + 1^2 + 0^2 + 0^2}$$

$$c_1(f_2) = \frac{3}{2}$$

$$\boxed{c_1(f_2) = 1.5}$$

So, the updated cluster centre for $c_1 = (1.5, 1.5)$

Similarly

$$c_2(f_1) = \frac{u_{21}^2 * x_1(f_1) + u_{22}^2 * x_2(f_1) + u_{23}^2 * x_3(f_1) + u_{24}^2 * x_4(f_1)}{u_{21}^2 + u_{22}^2 + u_{23}^2 + u_{24}^2}$$

$$c_2(f_1) = \frac{0^2 * 1 + 0^2 * 2 + 1^2 * 4 + 1^2 * 5}{0^2 + 0^2 + 1^2 + 1^2}$$

$$c_2(f_1) = \frac{9}{2}$$

$$\boxed{c_2(f_1) = 4.5}$$

$$c_2(f_2) = \frac{u_{21}^2 * x_1 f_2 + u_{22}^2 * x_2 f_2 + u_{23}^2 * x_3(f_2) + u_{24}^2 * x_4(f_2)}{u_{21}^2 + u_{22}^2 + u_{23}^2 + u_{24}^2}$$

$$c_2(f_2) = \frac{0^2 * 1 + 0^2 * 2 + 1^2 * 3 + 1^2 * 3}{0^2 + 0^2 + 1^2 + 1^2}$$

$$\boxed{c_2(f_2) = 6/2 = 3}$$

So, the updated cluster centre for $C_2 = (4.5, 3)$

Recalculation of membership values :

$d(c_1, x_1) = d((1.5, 1.5), (1,1))$

$= \sqrt{(1.5-1)^2 + (1.5-1)^2}$

$= 0.71$

$d(c_1, x_2) = d(1.5, 1.5)\ (2,2)$

$= \sqrt{(1.5-2)^2 + (1.5-2)^2} = 0.71$

$d(c_1, x_3 = d((1.5, 1.5), (4,3))$

$= \sqrt{(1.5-4)^2 + (1.5-3)^2} = 2.92$

$d(c_1, x_4) = d(1.5, 1.5)\ (5,3)$

$= \sqrt{(1.5-5)^2 + (1.5-3)^2} = 3.81$

$d(c_2, x_1) = d((4.5, 3)\ (1,1))$

$= \sqrt{(4.5-1)^2 + (3-1)^2} = 4.03$

$d(c_2, x_2) = d(4.5, 3)\ (2,2)$

$= \sqrt{(4.5-2)^2 + (3-2)^2} = 2.69$

$d(c_2, x_3) = d(4.5, 3)\ (4,3)$

$= \sqrt{(4.5-4)^2 + (3-3)^2}$

$= 0.25$

$d(c_2, x_4) = d(4.5, 3)\ (5,3)$

$= \sqrt{(4.5-5)^2 + (3-3)^2} = 0.25$

member

nember ship values are :

$$\mu_{ij} = \frac{d(x_j, c_i)^{-1/(M-1)}}{\sum_{l=1}^{c} d(x_j, c_l)^{-1/(M-1)}}$$

$$\mu_{11} = \frac{1/0.71}{\frac{1}{0.71} + \frac{1}{4.03}} = 0.85$$

$$\mu_{12} = \frac{\frac{1}{d(c_1, x_2)}}{\frac{1}{d(c_1, x_2)} + \frac{1}{d(c_2, x_2)}}$$

$$\mu_{12} = \frac{\frac{1}{0.71}}{\frac{1}{0.71} + \frac{1}{2.69}} = 0.79$$

$$\mu_{13} = \frac{\frac{1}{d(c_1, x_3)}}{\frac{1}{d(c_1, x_3)} + \frac{1}{d(c_2, x_3)}}$$

$$\mu_{13} = \frac{\frac{1}{2.92}}{\frac{1}{2.92} + \frac{1}{0.25}} = 0.08$$

$$\mu_{14} = \frac{\frac{1}{d(c_1, x_4)}}{\frac{1}{d(c_1, x_4)} + \frac{1}{d(c_2, x_4)}}$$

$$\mu_{14} = \frac{1/3.81}{\frac{1}{3.81} + \frac{1}{0.25}} = 0.06$$

similarly $u_{21}, u_{22}, u_{23}, u_{24}$ can be computed and their values are 0.15, 0.21, 0.92 and 0.94 respectively.

	x_1	x_2	x_3	x_4
c_1	0.85	0.79	0.08	0.06
c_2	0.15	0.21	0.92	0.94

$$c_1(f_1) = \frac{0.85^2 * 1 + 0.79^2 * 2 + 0.08^2 * 4 + 0.06^2 * 5}{0.85^2 + 0.79^2 + 0.08^2 + 0.06^2}$$

$$c_1(f_1) = \frac{2.04}{1.36} = 1.5$$

$$c_1(f_2) = \frac{0.85^2 * 1 + 0.79^2 * 2 + 0.08^2 * 3 + 0.06^2 * 3}{0.85^2 + 0.79^2 + 0.08^2 + 0.06^2}$$

$$c_1(f_2) = \frac{2.01}{1.36} = 1.48.$$

$c_1 = (1.5, 1.48)$

$$c_2(f_1) = \frac{0.15^2 * 1 + 0.21^2 * 2 + 0.92^2 * 4 + 0.94^2 * 5}{0.15^2 + 0.2^2 + 0.92^2 + 0.94^2}$$

$$= 4.4$$

$$c_2(f_2) = \frac{0.15^2 * 1 + 0.21^2 * 2 + 0.92^2 * 3 + 0.94^2 * 3}{0.15^2 + 0.21^2 + 0.92^2 + 0.94}$$

$$c_2(f_2) = 2.95.$$

updated c_2 cluster value $c_2 = (4.4, 2.95)$

$$d(c_1, x_1) = d(1.5, 1.48)\ (1,1)$$

$$= \sqrt{(1.5-1)^2 + (1.48-1)^2}$$

$$\boxed{d(c_1, x_1) = 0.693}$$

$$d(c_1, x_2) = d(1.5, 1.48)\ (2,2)$$

$$= \sqrt{(1.5-2)^2 + (1.48-2)^2}$$

$$= 0.721$$

$$d(c_1, x_3) = d(1.5, 1.48)\ (4,3)$$

$$= \sqrt{(1.5-4)^2 + (1.48-3)^2}$$

$$= 2.925$$

$$d(c_1, x_4) = d(1.5, 1.48)\ (5,3)$$

$$= \sqrt{(1.5-5)^2 + (1.48-3)^2}$$

$$= 3.815$$

$$d(c_2, x_1) = d(4.4, 2.95)\ (1,1)$$

$$= \sqrt{(4.4-1)^2 + (2.95-1)^2}$$

$$= 3.919$$

$$d(c_2, x_2) = d(4.4, 2.95)\ (2,2)$$

$$= \sqrt{(4.4-2)^2 + (2.95-2)^2}$$

$$= 2.58$$

$$d(c_2, x_3) = d(4.4, 2.95)\ (4,3)$$

$$= \sqrt{(4.4-4)^2 + (2.95-3)^2}$$

$$= 0.40$$

The updated membership values are :

$$\mu_{11} = \frac{\frac{1}{0.693}}{\frac{1}{0.693} + \frac{1}{3.919}} = 0.85$$

$$\mu_{12} = \frac{\frac{1}{0.721}}{\frac{1}{0.721} + \frac{1}{2.58}} = 0.79$$

$$\mu_{13} = \frac{\frac{1}{2.925}}{\frac{1}{2.925} + \frac{1}{0.40}} = 0.12$$

$$\mu_{14} = \frac{\frac{1}{3.815}}{\frac{1}{3.815} + \frac{1}{0.60}} = 0.14$$

Similarly $\mu_{21}, \mu_{22}, \mu_{23}, \mu_{24}$ can be computed and their values are 0.15, 0.21, 0.88 and 0.86 respectively.

Now the updated membership values $\mu^{(2)}$ are given below :

	x_1	x_2	x_3	x_4
C_1	0.85	0.79	0.12	0.14
C_2	0.15	0.21	0.88	0.86

Since we do not end up with the same cluster point (or) same membership values, we iterate for the next step.

The updated cluster centers for the updated membership values ($\mu^{(2)}$) are as follows :

$$c_1(f_1) = \frac{(0.85)^2 * 1 + (0.79)^2 * 2 + (0.12)^2 * 4 + (0.14)^2 * 5}{(0.85)^2 + (0.79)^2 + (0.12)^2 + (0.14)^2}$$

$$c_1(f_1) = \frac{2.04}{1.355} = 1.5$$

$$\boxed{c_1(f_1) = 1.5}$$

$$c_1(f_2) = \frac{(0.85)^2 * 1 + (0.79)^2 * 2 + (0.12)^2 * 3 + (0.14)^2 * 3}{(0.85)^2 + (0.79)^2 + (0.12)^2 + (0.14)^2}$$

$$= \frac{2.01}{1.355} = 1.48$$

$$\boxed{c_1(f_2) = 1.48}$$

and their

$$\boxed{\text{The updated cluster centre for } c_1 = (1.5, 1.48)}$$

below :

$$c_2(f_1) = \frac{(0.15)^2 * 1 + (0.21)^2 * 2 + (0.92)^2 * 4 + (0.94)^2 * 5}{(0.15)^2 + (0.21)^2 + (0.92)^2 + (0.94)^2}$$

$$c_2(f_1) = \frac{7.9143}{1.797} = 4.4$$

$$\boxed{c_2(f_1) = 4.4}$$

t (or)

$$c_2(f_2) = \frac{(0.15)^2 * 1 + (0.21)^2 * 2 + (0.92)^2 * 3 + (0.94)^2 * 3}{(0.15)^2 + (0.21)^2 + (0.92)^2 + (0.94)^2}$$

ship

$$c_2(f_2) = \frac{5.307}{1.797} = 2.95$$

*4

$$\boxed{c_2(f_2) = 2.95}$$

$$\boxed{\text{The updated cluster centre for } c_2 = (4.4, 2.95)}$$

Since there is no change in the cluster centre, the algorithm stops. The final membership values and cluster allocation for the data points is given below Table.

cluster formation using fuzzy c-Means algorithm

Data points	μC_1	μC_2	Allocated clusters
x_1	0.85	0.15	C_1
x_2	0.79	0.21	C_1
x_3	0.12	0.88	C_2
x_4	0.14	0.86	C_2

Rough clustering :

In traditional clustering, each data point belongs to exactly one cluster. But in real-world data, this may not be ideal - some data points clearly belong to one cluster, while others are ambiguous and could belong to multiple clusters. This is where rough set theory helps.

This is abstracted by a rough set which is represented using two sets, these sets are called lower approximation and upper approximation.

1. Lower Approximation :

Set of data points that definitely belong to cluster S.

2. Upper Approximation :

Set of data points that possibly belong to cluster S.

Ex : Overlap with S but not fully contained.

In Essence :

* Lower Approx - Certain members
* Upper Approx - possible members
* The difference $\overline{R}(S) - R(S)$ captures the "roughness"

Ex : Uncertainity

$\underline{R}S = \cup_i G_i$ Where $G_i \subseteq S$

$\overline{R}(S) = \cup_i G_i$ Where $G_i \cap S \neq \emptyset$

Rough K-Means Clustering Algorithm :

This is a Variant of K-Means that integrates the rough set idea. It assigns data points to clusters with certainty (lower) (or) possibility (upper), and update clusters accordingly.

Inputs :

* n : Number of data points
* $\mathcal{X}$: $x_1, x_2 \cdots x_n$ Data points
* k : Number of clusters.
* w_l, w_u : Weights for lower and upper Approximations.
* ϵ : Convergence threshold.

Step-by-step Algorithm :

1. Initialization :

* Randomly assign each data point to the lower approximation of exactly one cluster
* Each point also belongs to that cluster's upper approximation by definition.

2. Centroid calculation :

The cluster center c_j depends on the nature of points in the cluster.

Case 1 : if only lower approx points exist (clean cluster)

* $c_j = \frac{1}{|R(k)|} \sum x_i \in R(k)\ x_i$
* $c_j = \frac{1}{|R(k)|} \sum x_i \in \overline{R}(k)\ x_i$
* $c_j = w_l \cdot \frac{1}{|R(k)|} \sum_{x_i \in R(k)} x_i + w_u \cdot \frac{1}{|\overline{R}(k) - R(k)|} \sum x_i \in \overline{R}(k) - R(k)\ x_i$

1. Distance calculation :

For each point x_i, calculate the Euclidean distance to all cluster centroids c_j.

2. Reassignment :

Assign each data point to the lower approximation of the cluster where it is closest (if it's clearly part of that cluster). Or to the upper approximation of multiple clusters if it's ambiguous.

3. Repeat :

Iterate steps 2-4 until changes in assignments are less than the threshold ϵ.

Intuition :

Rough K-means is more flexible than regular K-Means.

* It handles uncertainty
* prevents forcing ambiguous points into just one cluster
* Gives more realistic clustering, especially in noisy (or) overlapping data.

Example : Consider the data set shown in Table with $k=2$, $w_l = 0.7$ $w_u = 0.3$ and $\epsilon = 2$

Example of data set with two feature

Data points	f_1	f_2
x_1	1	1
x_2	2	2
x_3	4	1
x_4	5	2
x_5	7	0
x_6	8	0

1. Randomly assign each data points to exactly one lower approximation (two clusters)

$\underline{R}(k_1) = \{(1,1), (2,2), (4,1)\}$

$\underline{R}(k_2) = \{(5,2), (7,0), (8,0)\}$

2. Since (i) $\underline{R}(k_1) \neq \emptyset$ and $\overline{R}(k_1) - \underline{R}(k_1) = \emptyset$.

(ii) $\underline{R}(k_2) \neq \emptyset$ and $\overline{R}(k_2) - \underline{R}(k_2) = \emptyset$

the centroid is calculated using

$$c_j = \sum x_i \in \underline{R}(k) \frac{x_i}{|\underline{R}(k)|}.$$

$$c_1 = \left(\frac{1+2+4}{3}, \frac{1+2+1}{3}\right) = (2.33, 1.33)$$

$$c_2 = \left(\frac{5+7+8}{3}, \frac{2+0+0}{3}\right) = (6.67, 0.67)$$

3. Find the Euclidean distance b/w each data point and the cluster centre

With reference to c_1

① $d(1,1), (2.33, 1.33) = \sqrt{(1-2.33)^2 + (1-1.33)^2} = 1.37$ ✓

② $d(2,2), (2.33, 1.33) = \sqrt{(2-2.33)^2 + (2-1.33)^2} = 0.75$

③ $d(4,1), (2.33, 1.33) = \sqrt{(4-2.33)^2 + (1-1.33)^2} = 1.70$

④ $d(5,2), (2.33, 1.33) = \sqrt{(5-2.33)^2 + (2-1.33)^2} = 2.75$

⑤ $d(7,0), (2.33, 1.33) = \sqrt{(7-2.33)^2 + (0-1.33)^2} = 4.86$

⑥ $d(8,0), (2.33, 1.33) = \sqrt{(8-2.33)^2 + (0-1.33)^2} = 5.82$

With Reference to c_2

$d(1,1), (6.67, 0.67) = \sqrt{(1-6.67)^2 + (1-0.67)^2} = 5.68$ ✓

$d(2,2), (6.67), (0.67) = \sqrt{(2-6.67)^2 + (2-0.67)^2} = 4.86$

$d(4,1), (6.67), (0.67) = \sqrt{(4-6.67)^2 + (1-0.67)^2} = 2.69$

$d(5,2), (6.67), (0.67) = \sqrt{(5-6.67)^2 + (2-0.67)^2} = 2.14$

$d(7,0), (6.67), (0.67) = \sqrt{(7-6.67)^2 + (0-0.67)^2} = 0.75$

$d(8,0), (6.67), (0.67) = \sqrt{(8-6.67)^2 + (0-0.67)^2} = 1.49$

4. Use the ratio $\frac{d(x, c_j)}{d(x, c_p)}$, where $1 \leq i, P \leq k$, to determine the membership.

$(1,1) \rightarrow \dfrac{d(1,1),(6.67,0.67)}{d(1,1),(2.33,1.33)} = \dfrac{5.68}{1.37} = 4.14 \not< 2$. So x_1 will be part of $\underline{R}(k_1)$

$(2,2) \rightarrow \dfrac{d(2,2),(6.67,0.67)}{d(2,2),(2.33,1.33)} = \dfrac{4.86}{0.75} = 6.48 \not< 2$

So x_2 will be part of $\underline{R}(k_1)$

$(4,1) \Rightarrow \dfrac{d(4,1),(6.67,0.67)}{d(4,1),(2.33,1.33)} = \dfrac{2.69}{1.7} = 1.58 < 2$

So x_3 will not be part of $\underline{R}(k_1)$ and $\underline{R}(k_2)$

$(5,2) \Rightarrow \dfrac{d(5,2),(2.33,1.33)}{d(5,2),(6.67,0.67)} = \dfrac{2.75}{2.14} = 1.28 < 2$

So x_4 will not be part of $\underline{R}(k_1)$ and $\underline{R}(k_2)$

$(7,0) \Rightarrow \dfrac{d(7,0),(2.33,1.33)}{d(7,0),(6.67,0.67)} = \dfrac{4.86}{0.75} = 6.48 \not< 2$

So x_5 will be part of $\underline{R}(k_2)$

$(8,0) \Rightarrow \dfrac{d(8,0),(2.33,1.33)}{d(8,0),(6.67,0.67)} = \dfrac{5.82}{1.49} = 3.91 \not< 2$

So x_4 will be part of $\underline{R}(k_2)$

Now we have clusters :

$\underline{R}(k_1) = \{(1,1),(2,2)\}$ | $\overline{R}(k_1) = \{(1,1),(2,2),(4,1),(5,2)\}$

$\underline{R}(k_2) = \{(7,0),(8,0)\}$ | $\overline{R}(k_2) = \{(4,1),(5,2),(7,0),(8,0)\}$

Here (i) $\underline{R}(k_1) \neq \theta$ and $\overline{R}(k_1) - \underline{R}(k_1) \neq \theta$

(ii) $\underline{R}(k_2) \neq \theta$ and $\overline{R}(k_2) - \underline{R}(k_2) \neq \theta$

So, the centroid is calculated using.

$$c_j = w_l \times \sum_{x \in \underline{R}(k)} \frac{x_i}{|\underline{R}(k)|} + w_u \times \sum_{x_i \in \bar{R}(k) - \underline{R}(k)} \frac{x_i}{|\bar{R}(k) - \underline{R}(k)|}$$

$$c_1 = 0.7 \times \left(\frac{1+2}{2}, \frac{1+2}{2}\right) + 0.3 \times \left(\frac{4+5}{2}, \frac{1+2}{2}\right)$$

$$\boxed{c_1 = (2.4, 1.5)}$$

$$c_2 = 0.7 \times \left(\frac{7+8}{2}, \frac{0+0}{2}\right) + 0.3 \times \left(\frac{4+5}{2}, \frac{1+2}{2}\right)$$

$$\boxed{c_2 = (6.6, 0.45)}$$

5. Repeat from step 3 until Convergence.

19/04/2025
CSE(AI)-A
11:15 AM
CSE(AI)-B
3:15 TO 4:15
B5

Expectation Maximization - Based clustering

1) computing the probability that a data point belongs to cluster

$$\text{prob}(x_i \in c_j) = z_{ij} \quad \frac{\exp\left[-\frac{1}{2\sigma^2}(x_i - \mu_j)^2\right]}{\sum_{l=1}^{k} \exp\left[-\frac{1}{2\sigma^2}(x_i - \mu_l)^2\right]}$$

Data points $x_1 = 1, x_2 = 6, x_3 = 6, x_4 = 7$

Initial cluster means

cluster 1 $(mu\_1) = 1$

cluster 2 $(mu\_1) = 7$

Distance to $mu\_1 = 1 \quad = (1-1)^2 = 0$

$= (1-7)^2 = 36 \rightarrow \exp(-18) \approx 0$

$$z_{11} = \frac{1}{1 + e^{-18}} \approx 0.9999$$

$z_{12} = 0.0001$

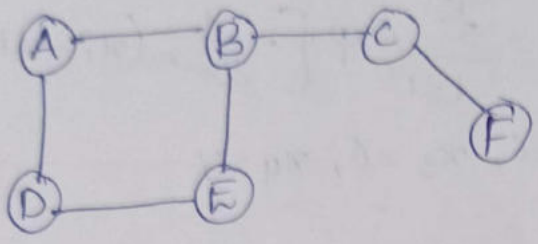
Similarly $x_2 = 6$, $mu\_2 = 7$

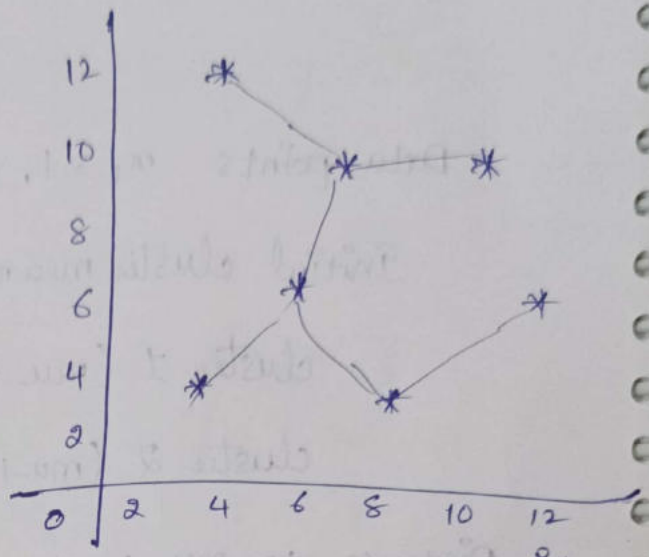
$$\mu_1 = \frac{z_{11} \cdot x_1 + z_{21} \cdot x_2 + z_{31} \cdot x_3 + z_{41} \cdot x_4}{z_{11} + z_{21} + z_{31} + z_{41}}$$

Data point (i)	z_{i1}	z_{i2}
$x_1 = 1$	0.9999	0.0001
$x_2 = 2$	0.9999	0.0001
$x_3 = 6$	0.0001	0.9999
$x_4 = 7$	0.0001	0.999

Spectral clustering :

It is a Graphical Representation of the vertices and edges weighted vertices distance is called the spectral clustering





Vertices (V) and Edges

① V-cluster 1 weights

② V-cluster 2 weights.

one vertices connected to the another vertices edge with edge weight. $(V-C_1)$, $(V-C_2)$ → Divided into 2 clusters.

with weighted values difference.

Data points (i)	feature (f_1)	feature (f_2)
x_1	1	1
x_2	2	2
x_3	4	3
x_4	5	3

P& $(4-0)^2 = (3^2) = 9$

Q.& $(2-2)^2 = 0$

R& $(4-3)^2 = 1^2 = 1$

① SVM, Lin → M
8-topics → 4 classes